

Panos M. Pardalos
Ding-Zhu Du
Ronald L. Graham
Editors

Handbook of Combinatorial Optimization

Second Edition



Springer Reference

Handbook of Combinatorial Optimization

Panos M. Pardalos • Ding-Zhu Du
Ronald L. Graham
Editors

Handbook of Combinatorial Optimization

Second Edition

With 683 Figures and 171 Tables



Springer Reference

Editors

Panos M. Pardalos
Department of Industrial and
Systems Engineering
University of Florida
Gainesville, FL, USA

Ronald L. Graham
Department of Computer Science and
Engineering
University of California, San Diego
La Jolla, CA, USA

Ding-Zhu Du
Department of Computer Science
The University of Texas at Dallas
Richardson, TX, USA

ISBN 978-1-4419-7996-4 ISBN 978-1-4419-7997-1 (eBook)

ISBN Bundle 978-1-4614-3975-2 (print and electronic bundle)

DOI 10.1007/978-1-4419-7997-1

Springer New York Heidelberg Dordrecht London

Library of Congress Control Number: 2013942873

1st edition: © Springer Science+Business Media New York 1999 (Volumes 1–3 and Supplement Volume A), 2005 (Supplement Volume B)

2nd edition: © Springer Science+Business Media New York 2013

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Many practical applications that include machines, people, or different resources that are indivisible, give rise to optimization models with discrete decision variables. On the other hand, nonconvexities (e.g., complementarity) in optimization models can be formulated using discrete variables.

In past years, we witnessed several developments in combinatorial optimization. These developments came along with progress in related fields such as complexity theory, approximation algorithms, as well as with advances in software and hardware technology.

The first edition of the *Handbook of Combinatorial Optimization* received a warm welcome from the scientific community. We are happy to introduce the second edition of the *Handbook of Combinatorial Optimization*. Some material in the first edition has been revised and several new chapters have been added.

We are grateful to all authors and referees for their valuable time. Moreover, we thank the publisher and the production team for helping with editing the handbook. We hope that this book will continue to be a helpful tool for individuals working in the field of combinatorial optimization.

Acknowledgements. We would like to acknowledge the help of our students during the years of editing the handbook. We have also been partially supported by Air Force, NSF, and DTRA grants.

Panos M. Pardalos
Ding-Zhu Du
Ronald L. Graham

Contents

Volume 1

A Unified Approach for Domination Problems on Different Network Topologies	1
Thang N. Dinh, D. T. Nguyen and My T. Thai	
Advanced Techniques for Dynamic Programming	41
Wolfgang Bein	
Advances in Group Testing	93
Yongxi Cheng	
Advances in Scheduling Problems	145
Ming Liu and Feifeng Zheng	
Algebrization and Randomization Methods	171
Liang Ding and Bin Fu	
Algorithmic Aspects of Domination in Graphs	221
Gerard Jennhwa Chang	
Algorithms and Metaheuristics for Combinatorial Matrices	283
Ilias S. Kotsireas	
Algorithms for the Satisfiability Problem	311
John Franco and Sean Weaver	
Bin Packing Approximation Algorithms: Survey and Classification	455
Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello and Daniele Vigo	
Binary Unconstrained Quadratic Optimization Problem	533
Gary A. Kochenberger, Fred Glover and Haibo Wang	

Combinatorial Optimization Algorithms	559
Elisa Pappalardo, Beyza Ahlatcioglu Ozkok and Panos M. Pardalos	
Combinatorial Optimization in Data Mining	595
Samira Saedi and O. Erhun Kundakcioglu	
Combinatorial Optimization Techniques for Network-Based Data Mining	631
Oleg Shirokikh, Vladimir Stozhkov and Vladimir Boginski	

Volume 2

Combinatorial Optimization in Transportation and Logistics Networks	673
Chrysafis Vogiatzis and Panos M. Pardalos	
Complexity Issues on PTAS	723
Jianer Chen and Ge Xia	
Computing Distances Between Evolutionary Trees	747
Bhaskar DasGupta, Xin He, Tao Jiang, Ming Li, John Tromp, Lusheng Wang and Louxin Zhang	
Connected Dominating Set in Wireless Networks	783
Hongjie Du, Ling Ding, Weili Wu, Donghyun Kim, Panos M. Pardalos and James Willson	
Connections Between Continuous and Discrete Extremum Problems, Generalized Systems, and Variational Inequalities	835
Carla Antoni, Franco Giannessi and Fabio Tardella	
Coverage Problems in Sensor Networks	899
Mihaela Cardei	
Data Correcting Approach for Routing and Location in Networks	929
Boris Goldengorin	
Dual Integrality in Combinatorial Optimization	995
Xujin Chen, Xiaodong Hu and Wenan Zang	
Dynamical System Approaches to Combinatorial Optimization	1065
Jens Starke	
Efficient Algorithms for Geometric Shortest Path Query Problems	1125
Danny Z. Chen	

Energy Efficiency in Wireless Networks	1155
Hongwei Du, Xiuzhen Cheng and Deying Li	
Equitable Coloring of Graphs	1199
Ko-Wei Lih	
Faster and Space Efficient Exact Exponential Algorithms: Combinatorial and Algebraic Approaches	1249
Dongxiao Yu, Yuexuan Wang, Qiang-Sheng Hua and Francis C. M. Lau	
Fault-Tolerant Facility Allocation	1293
Hong Shen and Shihong Xu	
Fractional Combinatorial Optimization	1311
Tomasz Radzik	

Volume 3

Fuzzy Combinatorial Optimization Problems	1357
Panos M. Pardalos, Emel Kizilkaya Aydogan, Feyza Gurbuz, Ozgur Demirtas and Birce Boga Bakirli	
Geometric Optimization in Wireless Networks	1415
Weili Wu and Zhao Zhang	
Gradient-Constrained Minimum Interconnection Networks	1459
Marcus Brazil and Marcus G. Volz	
Graph Searching and Related Problems	1511
Anthony Bonato and Boting Yang	
Graph Theoretic Clique Relaxations and Applications	1559
Balabhaskar Balasundaram and Foad Mahdavi Pajouh	
Greedy Approximation Algorithms	1599
Peng-Jun Wan	
Hardness and Approximation of Network Vulnerability	1631
My T. Thai, Thang N. Dinh and Yilin Shen	
Job Shop Scheduling with Petri Nets	1667
Hejiao Huang, Hongwei Du and Farooq Ahmad	
Key Tree Optimization	1713
Minming Li	
Linear Programming Analysis of Switching Networks	1755
Hung Q. Ngo and Thanh-Nhan Nguyen	

Map of Geometric Minimal Cuts with Applications	1815
Evanthia Papadopoulou, Jinhui Xu and Lei Xu	
Max-Coloring	1871
Julián Mestre and Rajiv Raman	
Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs	1913
Donatella Granata, Raffaele Cerulli, Maria Grazia Scutellà and Andrea Raiconi	
Modern Network Interdiction Problems and Algorithms	1949
J. Cole Smith, Mike Prince and Joseph Geunes	
Network Optimization	1989
Samir Khuller and Balaji Raghavachari	

Volume 4

Neural Network Models in Combinatorial Optimization.....	2027
Mujahid N. Syed and Panos M. Pardalos	
On Coloring Problems	2095
Weifan Wang and Yuehua Bu	
Online and Semi-online Scheduling	2191
Zhiyi Tan and An Zhang	
Online Frequency Allocation and Mechanism Design for Cognitive Radio Wireless Networks	2253
Xiang-Yang Li and Ping Xu	
Optimal Partitions	2301
Frank K. Hwang and Uriel G. Rothblum	
Optimization in Multi-Channel Wireless Networks	2359
Hongwei Du, Weili Wu, Lidong Wu, Kai Xing, Xuefei Zhang and Deying Li	
Optimization Problems in Data Broadcasting	2391
Yan Shi, Weili Wu, Jiaofei Zhong, Zaixin Lu and Xiaofeng Gao	
Optimization Problems in Online Social Networks	2455
Jie Wang, You Li, Jun-Hong Cui, Benyuan Liu and Guanling Chen	
Optimizing Data Collection Capacity in Wireless Networks	2503
Shouling Ji, Jing (Selena) He and Yingshu Li	

Packing Circles in Circles and Applications	2549
Zhao Zhang, Weili Wu, Ling Ding, Qinghai Liu and Lidong Wu	
Partition in High Dimensional Spaces	2585
Zhao Zhang and Weili Wu	
Probabilistic Verification and Non-approximability	2625
Mario Szegedy and Kashyap Kolipaka	
Protein Docking Problem as Combinatorial Optimization Using Beta-Complex	2685
Deok-Soo Kim	

Volume 5

Quadratic Assignment Problems	2741
Rainer E. Burkard	
Reactive Business Intelligence: Combining the Power of Optimization with Machine Learning	2815
Roberto Battiti and Mauro Brunato	
Reformulation–Linearization Techniques for Discrete Optimization Problems	2849
Hanif D. Sherali and Warren P. Adams	
Resource Allocation Problems	2897
Naoki Katoh, Akiyoshi Shioura and Toshihide Ibaraki	
Rollout Algorithms for Discrete Optimization: A Survey	2989
Dimitri P. Bertsekas	
Simplicial Methods for Approximating Fixed Point with Applications in Combinatorial Optimization	3015
Chuangyin Dang	
Small World Networks in Computational Neuroscience.....	3057
Dmytro Korenkevych, Jui-Hong Chien, Jicong Zhang, Deng-Shan Shiau, Chris Sackellares and Panos M. Pardalos	
Social Structure Detection	3089
Kai Yang, Weili Wu, Wei Zhang, Tzuwei Hsu and Lidan Fan	
Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work	3133
Frederick C. Harris Jr. and Rakhi Motwani	
Steiner Minimum Trees in E^3: Theory, Algorithms, and Applications.....	3179
J. MacGregor Smith	

Tabu Search	3261
Fred Glover and Manuel Laguna	
Variations of Dominating Set Problem	3363
Liying Kang	
Index	3395

Contributors

Warren P. Adams Department of Mathematical Sciences, Clemson University, Clemson, SC, USA

Beyza Ahlatcioglu Ozkok Department of Mathematics, Faculty of Arts and Science, Yildiz Technical University, Istanbul, Turkey

Farooq Ahmad Information Technology, University of Central Punjab, Lahore, Pakistan

Carla Antoni Naval Academy, Livorno, Italy

Emel Kizilkaya Aydogan Faculty of Engineering, Department of Industrial Engineering, Erciyes University, Kayseri, Turkey

Birce Boga Bakirli Faculty of Engineering, Department of Industrial Engineering, Gazi University, Ankara, Turkey

Balabhaskar Balasundaram School of Industrial Engineering and Management, Oklahoma State University, Stillwater, OK, USA

Roberto Battiti LION Lab, Università di Trento, and Lionsolver Inc., Trento, Italy

Wolfgang Bein Department of Computer Science, University of Nevada, Las Vegas, Las Vegas, NV, USA

Dimitri P. Bertsekas Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA

Vladimir Boginski Industrial and Systems Engineering Department, University of Florida, Shalimar, FL, USA

Anthony Bonato Department of Mathematics, Ryerson University, Toronto, ON, Canada

Marcus Brazil Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville, VIC, Australia

Mauro Brunato LION Lab, Università di Trento, and Lionsolver Inc., Trento, Italy

Yuehua Bu Department of Mathematics, Zhejiang Normal University, Jinhua, People's Republic of China

Rainer E. Burkard Institute of Optimization and Discrete Mathematics, Graz University of Technology, Graz, Austria

Mihaela Cardei Computer and Electrical Engineering and Computer Science Department, Florida Atlantic University, Boca Raton, FL, USA

Raffaele Cerulli Department of Mathematics, University of Salerno, Fisciano (SA), Italy

Gerard Jennhwa Chang Department of Mathematics, National Taiwan University, Taipei, Taiwan

Danny Z. Chen Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA

Guanling Chen Department of Computer Science, University of Massachusetts, Lowell, MA, USA

Jianer Chen Department of Computer Science and Engineering, Texas A&M University, College Station, TX, USA

Xujin Chen Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing, People's Republic of China

Xiuzhen Cheng Department of Computer Science, George Washington University, Washington, D.C., USA

Yongxi Cheng School of Management, Xi'an Jiaotong University, Xi'an, Shaanxi, People's Republic of China

Jui-Hong Chien Biomedical Engineering Department, University of Florida, Gainesville, FL, USA

Edward G. Coffman Department of Computer Science, Columbia University, New York, NY, USA

János Csirik Department of Computer Algorithms and Artificial Intelligence, University of Szeged, Szeged, Hungary

Jun-Hong Cui Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA

Chuangyin Dang Department of Systems Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong

Bhaskar DasGupta University of Illinois, Chicago, IL, USA

Ozgur Demirtas Department of Business Administration, Erciyes University, Kayseri, Turkey

Liang Ding Department of Computer Science, The University of Texas-Pan American, Edinburg, TX, USA

Ling Ding Institute of Technology, University of Washington Tacoma, Tacoma, WA, USA

Thang N. Dinh Department of Computer Science, Information Science and Engineering, University of Florida, Gainesville, FL, USA

Hongjie Du Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Hongwei Du Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, People's Republic of China

Lidan Fan Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

John Franco School of Electronic and Computing Systems, University of Cincinnati, Cincinnati, OH, USA

Bin Fu Department of Computer Science, The University of Texas-Pan American, Edinburg, TX, USA

Gábor Galambos Faculty of Education, Department of Computer Science, University of Szeged, Szeged, Hungary

Xiaofeng Gao Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, People's Republic of China

Joseph Geunes Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

Franco Giannessi University of Pisa, Pisa, Italy

Fred Glover OptTek Systems, Inc, Boulder, CO, USA

Boris Goldengorin Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

Donatella Granata Department of Statistics, Probability and Applied Statistics, University of Rome "La Sapienza", Rome, Italy

Feyza Gurbuz Faculty of Engineering, Department of Industrial Engineering, Erciyes University, Kayseri, Turkey

Frederick C. Harris Jr. Department of Computer Science & Engineering, University of Nevada, Reno, NV, USA

Jing (Selena) He Department of Computer Science, Kennesaw State University, Kennesaw, GA, USA

Xin He Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

Tzuwei Hsu Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Xiaodong Hu Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing, People's Republic of China

Qiang-Sheng Hua Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, People's Republic of China

Hejiao Huang Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, People's Republic of China

Frank K. Hwang National Chiao Tung University (Retired), Hsinchu, People's Republic of China

Toshihide Ibaraki The Kyoto College of Graduate Studies for Informatics, Kyoto, Japan

Shouling Ji Department of Computer Science, Georgia State University, Atlanta, GA, USA

Tao Jiang Department of Computer Science and Engineering, University of California, Riverside, CA, USA

Liying Kang Department of Mathematics, Shanghai University, Shanghai, People's Republic of China

Naoki Katoh Department of Architecture and Architectural Engineering, Graduate School of Engineering, Kyoto University, Kyoto, Japan

Samir Khuller Department of Computer Science, University of Maryland, College Park, MD, USA

Deok-Soo Kim Department of Industrial Engineering, Hanyang University, Seongdong-ku, Seoul, South Korea

Donghyun Kim Department of Mathematics and Computer Science, North Carolina Central University, Durham, NC, USA

Gary A. Kochenberger School of Business, University of Colorado, Denver, CO, USA

Kashyap Kolipaka Department of Computer Science, Rutgers, The State University of New Jersey, Piscataway, NJ, USA

Dmytro Korenkevych Industrial and System Engineering Department, University of Florida, Gainesville, FL, USA

Ilias S. Kotsireas Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, ON, Canada

O. Erhun Kundakcioglu Department of Industrial Engineering, University of Houston, Houston, TX, USA

Manuel Laguna Leeds School of Business, University of Colorado, Boulder, CO, USA

Francis C. M. Lau Department of Computer Science, The University of Hong Kong, Hong Kong, People's Republic of China

Lei Lei Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

Deying Li Department of Computer Science, Renmin University, Beijing, People's Republic of China

Ming Li Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada

Minming Li Department of Computer Science, City University of Hong Kong, Hong Kong, People's Republic of China

Xiang-Yang Li Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

Yingshu Li Department of Computer Science, Georgia State University, Atlanta, GA, USA

You Li Department of Computer Science, University of Massachusetts, Lowell, MA, USA

Ko-Wei Lih Institute of Mathematics, Academia Sinica, Taipei, Taiwan

Benyuan Liu Department of Computer Science, University of Massachusetts, Lowell, MA, USA

Ming Liu School of Economics & Management, Tongji University, Shanghai, People's Republic of China

Qinghai Liu College of Mathematics and System Sciences, Xinjiang University, Urumqi, Xinjiang, People's Republic of China

Zaixin Lu Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Silvano Martello DEI “Guglielmo Marconi”, University of Bologna, Bologna, Italy

Julián Mestre School of Information Technologies, The University of Sydney, Sydney, NSW, Australia

Rakhi Motwani Department of Computer Science & Engineering, University of Nevada, Reno, NV, USA

Hung Q. Ngo Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

D. T. Nguyen Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL, USA

Thanh-Nhan Nguyen Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

Foad Mahdavi Pajouh School of Industrial Engineering and Management, Oklahoma State University, Stillwater, OK, USA

Evanthia Papadopoulou Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

Elisa Pappalardo Dipartimento di Matematica e Informatica, Universita di Catania, Catania, Italy

Panos M. Pardalos Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

Mike Prince Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

Tomasz Radzik Department of Informatics, Kings College London, London, UK

Andrea Raiconi Department of Mathematics, University of Salerno, Fisciano (SA), Italy

Balaji Raghavachari Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Rajiv Raman Indraprasta Institute of Information Technology, Delhi, India

Uriel G. Rothblum Faculty of Industrial Engineering and Management Science, Technion - Israel Institute of Technology, Haifa, Israel

Chris Sackellares Research Department, Optima Neuroscience, Gainesville, FL, USA

Samira Saedi Department of Industrial Engineering, University of Houston, Houston, TX, USA

Maria Grazia Scutellà Department of Computer Science, University of Pisa, Pisa, Italy

Hong Shen School of Computer Science, The University of Adelaide, Adelaide, SA, Australia

School of Information Science and Technology, Sun Yat-sen University, Guangzhou, People's Republic of China

Yilin Shen Department of Computer Science, Information Science and Engineering, University of Florida, Gainesville, FL, USA

Hanif D. Sherali Grado Department of Industrial and Systems Engineering (0118), Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

Yan Shi Department of Computer Science and Software Engineering, University of Wisconsin - Platteville, Platteville, WI, USA

Deng-Shan Shiau Research Department, Optima Neuroscience, Alachua, FL, USA

Akiyoshi Shioura Department of System Information Sciences, Graduate School of Information Sciences, Tohoku University, Sendai, Japan

Oleg Shirokikh Industrial and Systems Engineering Department, University of Florida, Gainesville, FL, USA

J. Cole Smith Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

J. MacGregor Smith Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst, MA, USA

Jens Starke Department of Mathematics & Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark

Vladimir Stozhkov Industrial and Systems Engineering Department, University of Florida, Gainesville, FL, USA

Mujahid N. Syed Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

Mario Szegedy Department of Computer Science, Rutgers, The State University of New Jersey, Piscataway, NJ, USA

Zhiyi Tan Department of Mathematics, Zhejiang University, Hangzhou, People's Republic of China

Fabio Tardella Sapienza University of Rome, Rome, Italy

My T. Thai Department of Computer Science, Information Science and Engineering, University of Florida, Gainesville, FL, USA

John Tromp CWI, Amsterdam, The Netherlands

Daniele Vigo DEI "Guglielmo Marconi", University of Bologna, Bologna, Italy

Chrysafis Vogiatzis Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

Marcus G. Volz TSG Consulting, Melbourne, VIC, Australia

Peng-Jun Wan Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

Haibo Wang College of Business Administration, Texas A&M International University, Laredo, TX, USA

Jie Wang Department of Computer Science, University of Massachusetts, Lowell, MA, USA

Lusheng Wang Department of Computer Science, City University of Hong Kong, Hong Kong

Weifan Wang Department of Mathematics, Zhejiang Normal University, Jinhua, People's Republic of China

Yuexuan Wang Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, People's Republic of China

Sean Weaver U.S. Department of Defense, Ft. George G. Meade, MD, USA

James Willson Department of Computer Science, The University of New Mexico, Albuquerque, NM, USA

Lidong Wu Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Weili Wu Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Ge Xia Department of Computer Science, Lafayette College, Easton, PA, USA

Kai Xing Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Jinhui Xu Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

Lei Xu Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

Ping Xu Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

Shihong Xu School of Computer Science, The University of Adelaide, Adelaide, SA, Australia

Boting Yang Department of Computer Science, University of Regina, Regina, SK, Canada

Kai Yang Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Dongxiao Yu Department of Computer Science, The University of Hong Kong, Hong Kong, People's Republic of China

Wenan Zang Department of Mathematics, The University of Hong Kong, Hong Kong

An Zhang Institute of Operational Research and Cybernetics, Hangzhou Dianzi University, Hangzhou, People's Republic of China

Jicong Zhang Industrial and System Engineering Department, University of Florida, Gainesville, FL, USA

Louxin Zhang Department of Mathematics, National University of Singapore, Singapore

Wei Zhang Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Xuefei Zhang Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Zhao Zhang College of Mathematics and System Sciences, Xinjiang University, Urumqi, Xinjiang, People's Republic of China

Feifeng Zheng Glorious Sun School of Business and Management, Donghua University, Shanghai, People's Republic of China

Jiaofei Zhong Department of Mathematics and Computer Science, University of Central Missouri, Warrensburg, MO, USA

A Unified Approach for Domination Problems on Different Network Topologies

Thang N. Dinh, D. T. Nguyen and My T. Thai

Contents

1	Introduction	2
2	Dominating Set Variants and Generalizations.....	3
2.1	Positive Influence Dominating Set.....	3
2.2	Multiple-Hop PIDS Problem.....	5
2.3	Generalized Dominating Set Problems.....	6
3	Approximability of Domination Problems in General Graphs.....	7
3.1	Hardness of Approximation.....	7
3.2	Approximation Algorithm.....	18
4	Domination Problems in Special Graph Classes.....	20
4.1	Power-Law Networks.....	21
4.2	Dense Graphs.....	27
4.3	Finding Optimal Solutions in Trees.....	27
5	Scalable Algorithm for Multiple-Hop PIDS.....	29
5.1	VirAds: Multiple-Hop PIDS Greedy Algorithm.....	30
5.2	Experimental Study.....	32
6	Conclusion.....	37
	Recommended Reading	38

Abstract

This chapter studies approximability and inapproximability for a class of important domination problems in different network topologies. In addition to the well-known *connected dominating set*, *total dominating set*, more general forms

T.N. Dinh (✉) • M.T. Thai

Department of Computer Science, Information Science and Engineering, University of Florida,
Gainesville, FL, USA

e-mail: tdinh@cise.ufl.edu; mythai@cise.ufl.edu

D.T. Nguyen

Department of Computer & Information Science & Engineering, University of Florida,
Gainesville, FL, USA

e-mail: dtnguyen@cise.ufl.edu

of the problems, in which each node is required to be dominated by more than one of its neighbors, are also considered. An example is the *positive influence dominating set* (PIDS) problem, originated from the context of influence propagation in social networks. The PIDS problem seeks for a minimal set of nodes P such that all other nodes in the network have at least a fraction $\rho > 0$ of their neighbors in P . Furthermore, domination problems can be hybridized to form new problems such as T-PIDS and C-PIDS, the *total* version and the *connected* version of PIDS. The goal of the chapter is to narrow the gaps between the approximability and inapproximability of those domination problems. Going beyond the classic $O(\log n)$ results, the chapter presents the explicit constants in the approximation factors to obtain tighter approximation bounds. While the first part of the chapter focuses on the general topology, the second part presents improved approximation results in different network topologies including power-law networks, social networks, and treelike networks.

1 Introduction

The dominating set problem in graph asks for a minimum size subset of vertices in which each vertex is either in the dominating set or adjacent to some vertex in the dominating set. It is one of the most widely studied topics in graph theory. In a survey by Hanes et al. [1], over 1,200 papers on the subject were listed in the bibliography. Domination problems have found many practical applications in building routing virtual backbones, power management, topology control, broadcast scheduling in sensor networks, and many others [2–9]. Recently, a variation of dominating set, called *positive influence dominating set* (PIDS), has been applied to identify influential users in social networks for viral marketing [10–12].

The approximability of the dominating set problem is the same with that of the *set cover* problem [13]. The work of Slavik [14] states that both problems can be approximated within a factor $\ln n - \ln \ln n + \theta(1)$, where n is the number of vertices in the dominating set instance (or the number of elements in the set cover instance). The lower bound on the approximability is given by Feige [13] in which set cover cannot be approximated within a factor $(1 - o(1)) \ln n$, unless NP has slightly superpolynomial time algorithms.¹ Hence, the approximation factor for the DS problem is tight up to a small sub-logarithm factor. However, this is not true for many variants of the DS problem, in which the gaps between lower bounds and upper bounds on the approximability are rather large.

In many domination problems, only the asymptotic approximation factor $O(\log n)$ is given without stating the explicit constant in the factor. In the case of the PIDS problem, the best hardness result is only APX-hard [11], while the best approximation factor is $O(\log n)$ [16]. This chapter is an effort to narrow down the approximability gaps for variants of the DS problem in some important graph classes, revealing the hidden constant in the approximation factors. In addition,

¹Under the typical assumption that $P \neq NP$, the best known inapproximability result is $c \cdot \ln n$ [15] for some constant $c > 0$.

inapproximability of numerous domination problems can be achieved under the same framework in which parameters in the Feige's construction for set cover [13] are fine-tuned. For special graph classes, better approximation factors can be obtained by exploiting the graph topologies. A common theme in this chapter is that approximability results are first proved for the PIDS problem, and then necessary modification to achieve the same results for other variants is presented.

Organization. The chapter continues with a brief summary on variants of the DS problem and their current approximability states. After that, the approximability of the considered domination problems in general graphs is presented. The rest of the chapter studies the domination problems in special graph classes including power-law graphs (social networks), dense graphs, and trees. Finally, a scalable algorithm for multiple-hop PIDS problem is introduced. Experiments on real-world social networks are presented to demonstrate the application of the dominating problem in finding influential users in social networks.

2 Dominating Set Variants and Generalizations

The most famous variant of the DS problem is the *connected dominating set* problem. The problem seeks for a minimum size DS which induces a connected subgraph. The problem has the same inapproximability $(1 - o(1)) \ln n$ of the DS problem. Guha and Khuller [17] give an $H(\Delta) + 2 = \ln n + O(1)$ approximation algorithm for finding minimum CDS, where $H(n) = 1 + 1/2 + \dots + 1/n$ is the harmonic function.

Another well-studied variant of DS is *total dominating set*. The total dominating set has an addition condition on the subgraph induced by the dominating set that the subgraph has no isolated vertices. The problem has the same hardness of the DS problem and can be approximated within a factor $\ln n + O(1)$ [18].

In addition to the well-known *connected dominating set* and *total dominating set*, more generalization forms of the problems, in which each nodes is required to be dominated by more than one of its neighbors, are also considered. The *t-tuple* DS problem seeks for a subset $S \subset V$ in graph $G = (V, E)$ so that each vertex is either in S or adjacent to at least t vertices in S . The problem admits a hardness of approximation factor $(1 - \varepsilon) \ln n$ and an $\ln(\Delta + 1) + 1$ approximation [19–21]. A generalization of the *t-tuple* DS problem is the k -connected *t-tuple* dominating set problem [22] in which the subgraph is induced by the *t-tuple* DS. There is a $6 + \ln 5/2(k-1) + 5/k$ approximation algorithm for the problem on UDG. No hardness results other than NP-completeness for the problem were presented in literature.

2.1 Positive Influence Dominating Set

The positive influence dominating set problem emerges in the context of social networks, in which a set of influential users are sought for to propagate the influence to other users. Regularly, individuals tend to be influenced by the opinions/behaviors

of their relatives, friends, and colleagues. For example, children whose parents smoked are twice as likely to begin smoking between 13 and 21 [23], and peer pressure accounts for 65 % reasons for binge drinking, a major health issue, by children and adolescents [24]. Moreover, the tendency of a user to adopt a behavior increases together with the number of his neighbors that follow that behavior.

The motivation behind the PIDS problem is to exploit the relationships and influences among individuals in social networks might offer considerable benefit to both the economy and society. As an example, positive impacts of intervention and education programs on a properly selected set of initial individuals can diffuse widely into society via various social contacts: face to face, phone calls, email, social networks, and so on.

Formally, let $G = (V, E)$ be a graph that represents the social network and an influence factor $0 < \rho < 1$, in the PIDS problem one wishes to find a minimum set of vertices P , so that every other vertices in G has at least a fraction ρ of their neighbors in P . Denote by $N(v)$ the set of neighbors of a vertex $v \in V$ and $d(v) = |N(v)|$ the degree of v . The problem is defined in graph theory language as follows.

POSITIVE INFLUENCE DOMINATING SET (PIDS)

Input: An undirected graph $G = (V, E)$ with influence factor $0 < \rho < 1$.

Problem: Find a subset $P \subset V$ such that $\forall u \in V \setminus P : |N(u) \cap P| \geq \rho d(u)$.

Vertices in P are said to dominate or influence their neighbors in $V \setminus P$. The constant ρ is called the *influence factor*, since it determines for each vertex the minimum number of neighbors to include in the PIDS. The studied problem in [10–12] is a special case of the above problem with $\rho = 1/2$.

It is rather difficult to show the inapproximability for the PIDS problem. The best hardness result is only APX-hard [11], i.e., there is a certain constant $\varepsilon > 0$ that approximating the problem within a factor $1 + \varepsilon$ is NP-hard. This lower bound on the approximability is far from the typical lower bounds $O(\ln n)$ in many domination problems. In contrast to t -tuple DS and any DS problems with fixed domination requirements, adding edges to a PIDS instance during a reduction will change the domination requirements. The “dynamic” threshold $\rho d(v)$, rather than one fixed t , makes the PIDS problem resist to proof approaches for known domination problems.

The literature on identifying influential users begins with Domingos and Richardson [25] who were the first to study the propagation of influence in networks. Kempe et al. [26, 27] formulated the influence maximization problem as an optimization problem. Later, Leskovec et al. [28] study the influence propagation in a different perspective in which they aim to find a set of nodes in networks to detect the spread of virus as soon as possible.

Similarly, the *connected* and *total* versions of the PIDS problem can be defined as follows.

TOTAL POSITIVE INFLUENCE DOMINATING SET (T-PIDS)

Input: An undirected graph $G = (V, E)$ with influence factor $0 < \rho < 1$.

Problem: Find a subset $P_T \subset V$ such that $\forall u \in V : |N(u) \cap P_T| \geq \rho d(u)$.

CONNECTED POSITIVE INFLUENCE DOMINATING SET (C-PIDS)

Input: An undirected connected graph $G = (V, E)$ with influence factor $0 < \rho < 1$.

Problem: Find a subset $P_C \subset V$ such that $\forall u \in V \setminus P_C : |N(u) \cap P_C| \geq \rho d(u)$ and P_C induces a *connected* subgraph of G .

Similarly, *total connected positive influence dominating set* (TC-PIDS) problem asks for a connected and total PIDS P_{TC} of G .

2.2 Multiple-Hop PIDS Problem

The PIDS problem can be extended to allow multiple-hop influence. The multiple-hop PIDS problem [29] seeks for a minimal cost seeding, measured as the number of nodes, to massively and quickly spread the influence to the whole network (or a large segment of the network). The influence is limited to the nodes that are within d hops from the seeding for some constant $d \geq 1$. In other words, the influence is expected to spread to the whole networks within d propagation rounds.

Locally Bounded Diffusion Model. Let $R_0 \subset V$ be the subset of vertices selected to initiate the influence propagation, called the *seeding*. Also a vertex $v \in R_0$ is said to be a seed. The propagation process happens in round, with all vertices in R_0 are influenced (thus active in adopting the behavior) at round $t = 0$. At a particular round $t \geq 0$, each vertex is either active (adopted the behavior) or inactive, and each vertex's tendency to become active increases when more of its neighbors become active. If an inactive vertex u has more than $\lceil \rho d(u) \rceil$ active neighbors at round t , then it becomes active at round $t + 1$. The process goes on for a maximum number of d rounds, and a vertex once becomes active will remain active until the end. An initial set R_0 of vertices is a *d -seeding* if R_0 can make all vertices in the networks active within at most d rounds.

The influence factor $0 < \rho < 1$ decides how widely and quickly the influence propagates through the network. Influence factor ρ reflects real-world factors such as how easy to share the content with others or some intrinsic benefit for those who initially adopt the behavior. In case $\rho = 1/2$, the model is also known as the *majority* model that has many application in distributed computing, voting system [30], etc.

Problem Definition. The multiple-hop PIDS problem is defined as follows: *Given an undirected graph $G = (V, E)$ modeling a social network and an influence factor $0 < \rho < 1$, find in V a minimum size d -seeding, i.e., a subset of vertices, that can activate all vertices in the network within at most d rounds.*

Influence propagation with a limited number of hops (multiple-hop PIDS) as well as a special case of T-PIDS, when $\rho = 1/2$, were first considered in Wang et al. [10] in which they iteratively find dominating sets until forming a T-PIDS. Feng et al. [16] showed NP-completeness for the PIDS problem, when $\rho = 1/2$.

The APX-hardness and an $O(\log n)$ approximation algorithm for T-PIDS problem were introduced in [11]. Under the condition that the geometric representation of a unit disk graph is given and the maximum degree is bounded by a constant, Zhang et al. [31] devised a polynomial time approximation scheme (PTAS) for the t -latency bounded information propagation.

2.3 Generalized Dominating Set Problems

Another way to generalize the PIDS problem is to replace the domination requirement $\rho d(v)$ for vertex v with a function $r_v(x)$ of $d(v)$, called the *threshold function*. This yields the so-called generalized DS problem.

GENERALIZED DOMINATING SET (GDS)

Input: An undirected graph $G = (V, E)$ and a family of functions $r_v(x)$ for $v \in V$.

Problem: Find a subset $P \subset V$ such that $\forall v \in V \setminus P : |N(v) \cap P| \geq r_v(d(v))$.

Different family of $\{r_v(x)\}_{v \in V}$ corresponds to different domination problems, for example, $r_v(x) = 1$ gives the usual dominating set; $r_v(x) = \rho x$ gives the PIDS problem; $r_v(x) = t$, where t is a positive constant, gives the t -TUPLE DOMINATING SET problem [20, 21]; and $r_v(x) = c_v$, where c_v are positive constants, gives the FIXED THRESHOLD DOMINATING SET (FTDS) problem. Moreover, for scale-free networks that degree sequences follow power-law distribution, the threshold functions can take any of the form \sqrt{x} or $\log x$, etc.

Notice that using a function $r_v(x)$ instead of a simple constant c_v for vertex v (the case of FTDS) allows the domination requirement of v to be adjusted accordingly when the network evolves or is augmented. It is important to treat GDS as a family of problems but not a single problem. Different families of threshold functions might lead to very different approximation algorithm and inapproximability results. For example, setting $r_v(x) = x$ yields the MINIMUM VERTEX COVER that has simple two-approximation algorithm and $2 - \varepsilon$ lower bound under the unique game conjecture [32], while most domination problems achieve only $O(\log n)$ approximation algorithms.

In the same way that T-PIDS, C-PIDS, and TC-PIDS are formulated, the following problems are defined by considering the generalized threshold function:

- TOTAL GENERALIZED DOMINATING SET (T-GDS).
- CONNECTED GENERALIZED DOMINATING SET (C-GDS).
- k -CONNECTED GENERALIZED DOMINATING SET (k C-GDS), in which the subgraph induced by the dominating set is k -vertex connected, i.e., there are at least k -vertex-disjoint paths between any vertices in the dominating set, where k is an integral constant. C-GDS is a special case of k C-GDS with $k = 1$.
- TOTAL CONNECTED GENERALIZED DOMINATING SET (TC-GDS).
- k -CONNECTED TOTAL GENERALIZED DOMINATING SET (k CT-GDS).

Threshold Function. Not all threshold functions yield meaningful domination problems. For example, if $r_v(x) > x$, then there is no way to satisfy the requirement $|N(v) \cap P| > r_v(d(v)) > d(v)$. Only *monotone increasing threshold function* $r_v(x)$ that satisfies the following conditions is considered in the rest of the chapter:

1. $0 < r_v(x) < x \quad \forall x > 0$.
2. $0 \leq r_v(x) - r_v(x-1) \leq 1 \quad \forall x > 0$.
3. $\lim_{x \rightarrow \infty} \frac{r_v(x)}{x} < 1$.

The first condition makes the problem “nontrivial” to solve (to be precise, it makes the problem hard to approximate within $O(\log n)$). The second and third conditions guarantee the growing rate of the function to be linear or sublinear.

3 Approximability of Domination Problems in General Graphs

This section presents inapproximability results (lower bound) for domination problems in families GDS, T-GDS, kC -GDS, and TC-GDS followed by approximation algorithms (upper bound). All the results also hold for PIDS, C-PIDS, and T-PIDS, as they are the special cases of the generalized dominating set problems considered in this section.

3.1 Hardness of Approximation

First, the hardness $\ln \Delta - O(\ln \ln \Delta)$ is proved by studying the problems in B -bounded graphs (Sect. 3.1.2). Then the proofs are extended to obtain the hardness $(1/2 - \varepsilon) \ln n$ in Sect. 3.1.3. Finally, the hardness result for multiple-hop PIDS is derived through a reduction from the one-hop PIDS problem.

The proofs require setting parameters in Feige’s reduction for set cover [13]. The hardness results of set cover cannot be applied in a black-box fashion since it leads to weaker inapproximability $O(\log B)$ and $O(\log n)$ for B -bounded graph and general graph. That is, they cannot provide the explicit constants in the inapproximability factors. The challenge lies on bounding the size of added vertices in our reductions with the size of the optimal set cover. Two important quantities that are not mentioned in the Feige’s reduction are the maximum capacity of a set and the maximum frequency of a point.

The approximation factor and hardness of approximation of domination problems are summarized in Tables 1 and 2. The new results shown in this chapter are the following:

- The proof that domination problems belong to the families GDS, T-GDS, C-GDS, and TC-GDS can be approximated within $\ln \Delta + O(1)$ but cannot be approximated within $(\frac{1}{2} - o(1)) \ln n$, unless $\text{NP} \subset \text{DTIME}(n^{O(\log \log n)})$.
- On B -bounded degree graphs, none of the problems can be approximated within $\ln B - O(\ln \ln B)$, under the standard assumption $P \neq NP$. As a consequence,

Table 1 Hardness of approximation for domination problems in graphs

Problem	Original	T-(*)	$kC\text{-(*)}$	$kCT\text{-(*)}$
t -tuple DS	$(1 - \varepsilon) \ln n$ [20]	$(1 - \varepsilon) \ln n$	$(1 - \varepsilon) \ln n, k = 1$ $k = 1$ [20], $\forall k [\star]$	$(1 - \varepsilon) \ln n, k = 1$ $k = 1$ [20], $\forall k [\star]$
Positive Influence DS	APX-hard, $\rho = \frac{1}{2}$ [12]	–	APX-hard, $k = 1$ [11]	–
	$(1 - \varepsilon) \ln \Delta [\star]$	$(1 - \varepsilon) \ln \Delta [\star]$	$(1 - \varepsilon) \ln \Delta [\star]$	$(1 - \varepsilon) \ln \Delta [\star]$
	$(\frac{1}{2} - \varepsilon) \ln n [\star]$	$(\frac{1}{2} - \varepsilon) \ln n [\star]$	$(\frac{1}{2} - \varepsilon) \ln n [\star]$	$(\frac{1}{2} - \varepsilon) \ln n [\star]$
Fixed Threshold DS	$(1 - \varepsilon) \ln n$	$(1 - \varepsilon) \ln n$	$(1 - \varepsilon) \ln n [\star]$	$(1 - \varepsilon) \ln n [\star]$

Symbol $[\star]$ means results are derived from this chapter. The hardness $(1 - \varepsilon) \ln \Delta$ is proved under the typical assumption $P \neq NP$, while the hardness $(\frac{1}{2} - \varepsilon) \ln n$ is proved with the assumption $NP \not\subseteq DTIME(n^{O(\log \log n)})$

Table 2 Hardness of approximation for domination problems in graph with maximum degree Δ

Problem	Original	T-(*)	$kC\text{-(*)}$	$kCT\text{-(*)}$
Dominating Set	$\ln \Delta - O$ $(\ln \ln \Delta)$ [33]	$\ln \Delta - O$ $(\ln \ln \Delta)$ [33]	$\ln \Delta - O$ $(\ln \ln \Delta)$ $k = 1$ [33], $\forall k [\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)$ $k = 1$ [33], $\forall k [\star]$
t -tuple DS	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$
Positive Influence DS	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$
Fixed Threshold DS	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$	$\ln \Delta - O$ $(\ln \ln \Delta)[\star]$

Symbol $[\star]$ means results is derived from this chapter. The hardness are proved with the assumption $P \neq NP$

the considered problems are not approximated within $\ln \Delta - O(\ln \ln \Delta)$, unless $P = NP$.

- If the network is scale-free, i.e., the degree sequence follows a power-law distribution or the network is dense, the greedy algorithm obtains a constant factor approximation algorithm for “PIDS-like” domination problems.
- In networks with tree topologies, it is possible to find the optimal solution in linear time for all considered domination problems.

3.1.1 Feige’s Reduction for Set Cover

This part briefly summarizes the Feige’s proof for set cover (SC) which is essential to understand the incoming proofs. Feige presented a reduction from a k -prover proof system for a MAX 3SAT-5 instance ϕ that is a *conjunctive normal form* formula consists of n variables and $\frac{5n}{3}$ clauses of exactly three literals. The verifier interacts with k provers and asks provers different questions based on a random string r ; each question involves $l/2$ clauses and $l/2$ variables. If the formula ϕ is

satisfiable, then the provers have a strategy that cause the verifier accepts for all random strings. If only a $(1 - \varepsilon)$ fraction of the clauses in ϕ are simultaneously satisfiable, then for all strategies of the provers, the verifier weakly accepts with a probability at most $k^2 \cdot 2^{-cl}$, where c is a constant that depends only on ε .

The core of the SC gadget is a partition system $B(m, L, k, d)$, where B is a ground set of m points. The partition system is a collection of $L = 2^l$ partitions P_1, \dots, P_L of B , each partition P_i has exactly k disjoint subsets $p_{i,1}, \dots, p_{i,k}$. Any cover of m points in B requires at least $d = (1 - \frac{2}{3})k \ln m$ subsets. The condition to make constructing such a system possible is that $k < \frac{\ln m}{3 \ln \ln m}$.

Let $R = (5n)^l$ denote the number of possible random strings for the verifier. Make R copies of partition system B . Let B_r denote the copy of the partition associated with the random string r and $p_{i,j}^r$ the copy of set $p_{i,j}$ in B_r .

The instance of SC in the Feige's reduction is now presented. The universal set $\mathcal{U} = \bigcup_{r \in R} B_r$ contains $N = |\mathcal{U}| = mR$ points; and the set system is $\mathcal{S} = \{S_{q,a,i}\}_{q,a}$, where i can be deduced from syntax of (q, a) . Each set $S_{q,a,i}$ corresponds to a question-answer pair (q, a) of the i th prover and $S_{q,a,i} = \bigcup_{(q,i) \in r} p_{a_r,i}^r$ where $(q, i) \in r$ means on random string r , the i th prover receives question q , and a_r is the assignment of variables extracted from a .

As long as $k^2 2^{-cl} < \frac{8}{k^3 \ln^2 m}$, the hardness result will be $(1 - \frac{4}{k}) \ln m$, i.e., if formula ϕ is satisfiable, then mR points in \mathcal{U} can be covered by kQ subsets, and if only $(1 - \varepsilon)$ fraction of the clauses are simultaneously satisfiable, the minimum set cover has size at least $(1 - \frac{4}{k}) \ln m / kQ$. Here, Q is the set of all $n^l (5/3)^{l/2}$ possible questions. The condition can be satisfied with $l > \frac{1}{c} (5 \log k + 2 \log \ln m)$.

The hardness ratio $(1 - f(k)) \ln m$ of the set cover is obtained from the following key lemma.

Lemma 1 (Lemma 4.1 [13]) *If ϕ is satisfiable, then the above set of $N = mR$ points can be covered by kQ subsets. If only a $(1 - \varepsilon)$ fraction of the clauses in ϕ are simultaneously satisfiable, the above set requires $(1 - 2f(k))kQ \ln m$ subsets in order to be covered, where $f(k) \rightarrow 0$ as $k \rightarrow \infty$.*

Note that $\ln m = (1 - \varepsilon) \ln N$ by the setting of n, l , and m in the proof. Thus, the final hardness ratio is $(1 - \varepsilon) \ln N$, where $N = |\mathcal{U}|$. However, choosing different settings of n, l , and m yields different hardness ratios.

Upper bounds for quantities that appear in later proofs are presented for convenience.

- The *number of subsets* $|\mathcal{S}| \leq |Q|2^{2l}$. Since, for each question $q \in Q$, there are at most 2^{2l} answers of $2l$ bit length.
- The *maximum size* of a subset $\Delta_{\mathcal{S}} = \max_{S \in \mathcal{S}} |S| \leq m3^{l/2}$. Since each i and $q \in Q$ there are at most $3^{l/2}$ random strings r such that the verifier makes query q to the i th prover and $|p_{a_r,i}^r| \leq m$.

- The *maximum frequency* of a point (element) in \mathcal{U} : $f \leq k2^l$. Because, for a pair (q, i) , each partition $p_{a_r, i}^r$ is included at most 2^l times, plus each point in B_r appears in exactly k partitions.

3.1.2 Hardness of Approximation: $O(\ln \Delta)$

The hardness results $\ln \Delta - O(\ln \ln \Delta)$ for domination problems are shown indirectly by proving the hardness $\ln B - O(\ln \ln B)$ in the graphs with degree bounded by a constant B . In the first part, hardness results for PIDS and its variations are established. In the second part, the hardness of domination problems in the families GDS, T-GDS, kC -GDS, and d BD-GDS is proved with subtle modifications on the reductions in the first part.

Positive Influence Dominating Set. The inapproximability of PIDS and its variants is given in the following theorem.

Theorem 1 *Neither PIDS, T-PIDS, nor kC -PIDS can be approximated within $\ln B - O(\ln \ln B)$ in B -bounded graphs, unless $P=NP$.*

The proof is based on a reduction from an instance of the *bounded set cover* problem (SC_B) to an instance of PIDS problem whose degrees are also bounded by $B' = B \text{ poly log } B$.

BOUNDED SET COVER (SC_B)

Input: A set system $(\mathcal{U}, \mathcal{S})$, where $\mathcal{U} = \{e_1, e_2, \dots, e_n\}$, is a universe and \mathcal{S} is a collection of subsets of \mathcal{U} . Each subset in \mathcal{S} has at most B elements and each point belongs to at most B subsets, for a predefined constant $B > 0$.

Problem: Find a cover that is a subfamily $\mathcal{C} \subseteq \mathcal{S}$ of sets whose union is \mathcal{U} with the minimum number of subsets.

For a sufficient large constant $B_0 > 0$, set cover problem where each set has at most $B > B_0$ elements is hard to approximate to within a factor of $\ln B - O(\ln \ln B)$, unless $P = NP$ [34].

The proof [34] maps an instance of $GAP - SAT_{1,\gamma}$ to an instance $\mathcal{F} = (\mathcal{U}, \mathcal{S})$ of set cover with $\Delta_S \leq B$. Parameters l, m in Feige's construction [13] are fixed to $\theta(\ln \ln B)$ and $\frac{B}{\text{poly log}(B)}$, respectively. Since the parameters can be found in constant time using brute-force search, the assumption for the hardness is $P \neq NP$ instead of $NP \not\subset \text{DTIME}(n^{O(\log \log n)})$. The produced instance has the following properties that will be used later in our proofs:

- $|\mathcal{U}| = mn^l \text{ poly log } B$, $|\mathcal{S}| = n^l \text{ poly log } B$.
- $\Delta_S \leq B$, $F_{\mathcal{U}} \leq \text{poly log } B$ for sufficient large B .

A sufficient large constant B_0 gives us $f \leq \text{poly log}(B) \leq B$ for all $B \geq B_0$. \square

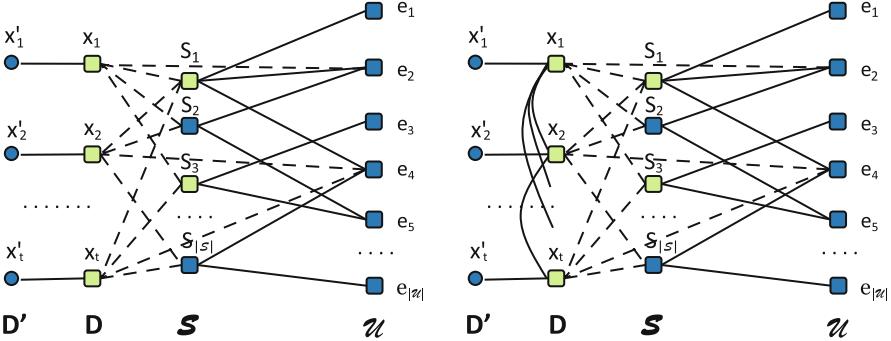


Fig. 1 Reduction from SC_B to PIDS (left) and T-PIDS (right)

SC_B-PIDS Reduction. For each instance $\mathcal{F} = (\mathcal{U}, \mathcal{S})$ of SC_B, construct a graph $\mathcal{H} = (V, E)$ as follows (Fig. 1):

- Construct a bipartite graph with the vertex set $\mathcal{U} \cup \mathcal{S}$ and edges between \mathcal{S} and all elements $x \in \mathcal{S}$, for each $S \in \mathcal{S}$.
- Add a set D consisting of t vertices and a set D' with same number of vertices, say $D = \{x_1, x_2, \dots, x_t\}$ and $D' = \{x'_1, x'_2, \dots, x'_t\}$. The value of t will be determined later.
- Connect x_i to x'_i , $\forall i = 1 \dots t$ to force the selection of x_i in the optimal PIDS.
- Connect each vertex $e_j \in \mathcal{U}$ to $\lceil \frac{\rho}{1-\rho} f(e_j) \rceil - 1$ and each vertex $S_k \in \mathcal{S}$ to $\lceil \frac{\rho}{1-\rho} |S_k| \rceil$ vertices in D , where $f(e_j)$ is the frequency of point e_j . The connection minimizes the degree differences of vertices in D .

Lemma 2 *The size difference between the optimal PIDS of \mathcal{H} and the optimal SC_B of \mathcal{F} is exactly the cardinality of D , i.e., $\text{OPT}_{\text{PIDS}}(\mathcal{H}) = \text{OPT}_{\text{SC}}(\mathcal{F}) + t$.*

Proof Let P be an optimal PIDS of \mathcal{H} . Since either x_i or x'_i must be selected into P , and $x'_i \in P$ can be always replaced with x_i inside P , thus, it is safe to assume that $D' \cap P = \emptyset$ and $D \subset P$.

By the construction, each vertex $S_k \in \mathcal{S}$ has enough required neighbors in P , while each vertex $e_i \in \mathcal{U}$ needs at least one more neighbors in P or it has to be selected. Since all vertices in \mathcal{U} must be adjacent to at least one vertex in \mathcal{S} , it is possible to replace each vertex $e_i \in P$ with one of its neighbors in \mathcal{S} without increasing the size of P . Thus, w.l.o.g. assumes that the optimal solution will contain vertices in \mathcal{S} but not in \mathcal{U} .

Hence, $P \setminus D$ must induce a cover for $\mathcal{F} = (\mathcal{U}, \mathcal{S})$. In other words, $\text{OPT}_{\text{PIDS}}(\mathcal{H}) \geq \text{OPT}_{\text{SC}}(\mathcal{F}) + t$.

Besides, given a cover $\mathcal{C} \subseteq \mathcal{S}$ for $(\mathcal{U}, \mathcal{S})$, it is easy to check that $\mathcal{C} \cup D$ gives a PIDS for \mathcal{H} . Thus, $\text{OPT}_{\text{SC}}(\mathcal{F}) + t \geq \text{OPT}_{\text{PIDS}}(\mathcal{H})$ that completes the proof. \square

The requirements to transfer hardness results of set cover to PIDS problem are to keep the degree of vertices in \mathcal{H} bounded by B' and keep t sufficiently small in comparison with the optimal solution of the SC_B instance in order to derive the ratio.

Lemma 3 *There exists a construction of \mathcal{H} with $t \leq \frac{\text{OPT}_{SC}}{\ln^2 B}$ and $B' = \Delta(\mathcal{H}) = O(B \text{ poly log } B)$.*

Proof First, the total degree of vertices in D , $\text{vol}(D)$, is bounded. For two sets of vertices A and B , define $\phi(A, B)$ the set of edges crossing between them.

$$\begin{aligned} \text{vol}(D) &= |\phi(D, D')| + |\phi(D, \mathcal{U})| + |\phi(D, \mathcal{S})| \\ &= |D| + \sum_{S_k \in \mathcal{S}} \lceil \frac{\rho}{1-\rho} |S_k| \rceil + \sum_{e_j \in \mathcal{U}} \lceil \frac{\rho}{1-\rho} f(e_j) - 1 \rceil \\ &\leq \frac{2\rho}{1-\rho} |\mathcal{S}| B + |\mathcal{S}| + t = \left(\frac{2\rho}{1-\rho} B + 1 \right) |\mathcal{S}| + t. \end{aligned} \quad (1)$$

The above inequalities hold because of $\sum_{S_k \in \mathcal{S}} |S_k| = \sum_{e_j \in \mathcal{U}} f(e_j)$ and $|S_k| \leq B$,

$\forall S_k \in \mathcal{S}$.

Select $t = \frac{|\mathcal{U}|}{B \ln^2 B}$. Since each set in \mathcal{S} can cover at most B elements, it follows that $\text{OPT}_{SC} \geq \frac{|\mathcal{U}|}{B}$; hence, $\frac{\text{OPT}_{SC}}{\ln^2 B} \geq t$.

To have a valid construction of \mathcal{H} , it is sufficient that $t \cdot B' \geq \text{vol}(D)$. Thus, select B' satisfying

$$\begin{aligned} B' &\geq \frac{1}{t} \left(\left(\frac{2\rho}{1-\rho} B + 1 \right) |\mathcal{S}| + t \right) \\ &\approx \left(\frac{2\rho}{1-\rho} B + 1 \right) \frac{B \ln^2 B n^l \text{ poly log } B}{mn^l \text{ poly log } B} \approx B \text{ poly log } B. \end{aligned} \quad (2)$$

Hence, setting $B' = B \text{ poly log } B$ gives us the desired construction of \mathcal{H} . \square

Theorem 2 *There exist constants B_1, c_1 such that for every $B' \geq B_1$, it is NP-hard to approximate the PIDS problem in graphs with degrees bounded by B' within a factor of $\ln B' - c_1 \ln \ln B'$.*

Proof Assume that there exists an algorithm that finds a PIDS of size at most $\ln B' - c_1 \ln \ln B'$ the optimal size in graph with degrees bounded by B' . Then the contradiction is obtained by showing how to approximate the SC_B problem with ratio $\ln B - c_0 \ln \ln B$. Selecting sufficient large B_1 is not difficult and shall be ignored to make the proof simpler.

Let $\mathcal{F} = (\mathcal{U}, \mathcal{S})$ be an instance of SC_B . Construct an instance \mathcal{H} of PIDS problem using the reduction SC_B -PIDS. From (2), there exists constant $\beta > 0$

so that $B' \leq B \ln^\beta B$. The approximation for PIDS returns a solution of size at most $(\ln B' - c_1 \ln \ln B') \text{OPT}_{PIDS}$. Then, that can be converted into a solution of SC_B by excluding vertices in D (see [Lemma 2](#)) and obtain a cover of size at most

$$\begin{aligned} & (\ln B' - c_1 \ln \ln B')(\text{OPT}_{SC} + t) - t \\ & \leq (\ln B' - c_1 \ln \ln B')\text{OPT}_{SC} + (\ln B' - c_1 \ln \ln B') \frac{\text{OPT}_{SC}}{\ln^2 B} \\ & \leq \left(\ln B + \beta \ln \ln B - c_1 \ln(\ln B + \theta(\ln \ln B)) + O\left(\frac{1}{\ln B}\right) \right) \text{OPT}_{SC}. \end{aligned}$$

Select $c_1 = c_0 + \beta + 1$. The solution for SC_B problem is then smaller than $\ln B - c_0 \ln \ln B$ times OPT_{SC} which implies $P = NP$. \square

Theorem 3 *It is NP-hard to approximate T-PIDS and kC-PIDS problems in graphs of bounded degree $B' > B_2$ within a factor of $\ln B' - c_2 \ln \ln B'$ for some constants $B_2, c_2 > 0$.*

Proof The reduction SC_B -PIDS shall be adjusted to achieve the same hardness result. The following adjustments are made to guarantee *total* and *k-connected* properties:

- On top of the subgraph induced by D , construct $(2 + O(1))k$ -regular Ramanujan graphs so that the subgraph has vertex expansion at least k [\[35\]](#). At the same time, connect S to D so that each vertex in S has at least k neighbors in D . It is easy to check that the subgraph induced by the solution to the dominating set problem is k -vertex connected.
- Connect a vertex $x_i \in D$ with $\lceil \frac{\rho}{1-\rho} d(x_i) \rceil$ other nodes in D , balancing nodes' degrees in D , so that D can dominate themselves. Thus, the degree of each node in D is roughly multiplied by a constant. Since $t = \frac{|U|}{B \ln^2 B} \gg \lceil \frac{\rho}{1-\rho} B' \rceil$, there are always enough vertices in D to connect x_i to.

The rest of the proof is straightforward. \square

Generalized Dominating Set Problems. By modifying the construction in the hardness proof for PIDS, the hardness results for GDS and its variants are obtained in the following theorem.

Theorem 4 *Domination problems in families GDS, T-GDS, and kC-GDS with threshold functions $r_v(x)$ cannot be approximated within $\ln B - O(\ln \ln B)$ in B-bounded graphs, unless $P = NP$.*

Proof Given a B -bounded graph $G = (V, E)$ and a functions $r_v(x)$ satisfying the conditions of a threshold function, let $\rho^* = \max_{v \in V} \lim_{x \rightarrow \infty} \frac{r_v(x)}{x}$. By the third condition,

$\rho^* < 1$ and $\rho = \frac{1}{2}(\rho^* + 1) < 1$. Since $\rho > \rho^*$, there exists an absolute constant $x_0 > 0$ such that $r_v(x) < \rho x \quad \forall x > x_0$ and $\forall v \in V$. Assume through the proof that B is sufficiently large so that $l = \theta(\log \log B) > x_0$.

The main idea in the reduction from dominating set with domination requirement $r_v(d(v))$ is to add connections from v to some $a(v)$ more vertices that are guaranteed to be in the optimal solution so that the remaining domination requirement on v becomes one for vertices in \mathcal{U} and zero for vertices in \mathcal{S} . It is necessary to find $x_v \in \mathbb{N}$ so that

$$c_v(x_v) = r_v(d(v) + x_v) - x_v \in \begin{cases} (0, 1] & \text{if } v \in \mathcal{U} \\ (-1, 0] & \text{if } v \in \mathcal{S}. \end{cases} \quad (3)$$

Claim For $\mu_v = \max\{\lceil \frac{\rho}{1-\rho} d(v) \rceil, x_0\}$, $c_v(\mu_v) \leq 0$.

Proof $c_v(\mu_v) = \rho(d(v) + \mu_v) - x_v = \rho(d(v) - \frac{1-\rho}{\rho} \mu_v) < 0$. □

Since

- $c_v(0) = r_v(d(v)) > 0$.
- $\Delta c_v = c_v(x+1) - c_v(x) \in (-1, 0]$.
- $c_v(\mu_v) < 0$.

There exists some $0 \leq x_v \leq \mu_v$ such that $c_v(x_v)$ satisfies Eq. 3. Notice that x_0 is substantially smaller than B . Hence, the same construction in the case of PIDS with influence factor ρ will work for the general cases as well. The only exception is that vertices in $v \in (\mathcal{S} \cup \mathcal{U})$ where x_v is connected to vertices in D . □

The following result is a direct corollary of the hardness result for the bounded degree case.

Theorem 5 Unless $P = NP$, domination problems in families GDS, T-GDS, and kC -GDS cannot be approximated within a factor of $\ln \Delta - O(\ln \ln \Delta)$, where Δ is the maximum degree.

3.1.3 Hardness of Approximation: $O(\ln n)$

Ideally, since the maximum degree Δ in the graph can be as large as the number of vertices n , it is natural to expect that the PIDS problem cannot be approximated within a factor $\ln n$. However, the hardness result for PIDS with the best parameter settings is only $(1/2 - o(1)) \ln n$. It is conjectured that there is a $1/2 \ln n$ approximation for the PIDS and T-PIDS problem.

Generalized Dominating Set Problems

Theorem 6 Domination problems in families GDS, T-GDS, and kC -GDS with threshold functions $r_v(x)$ cannot be approximated within $\frac{1}{2}(1 - o(1)) \ln n$ where n is the number of vertices, unless $NP \subset DTIME(n^{O(\log \log n)})$.

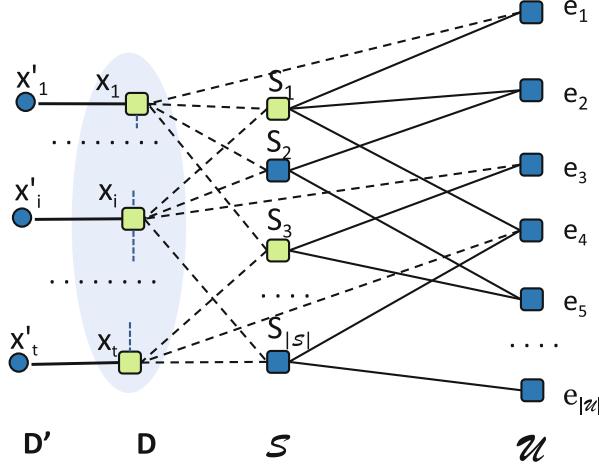


Fig. 2 Reduction from SC_B to GDS, T-GDS, and $k\text{C-GDS}$

Proof The same gadget in Fig. 2 is used to prove for the three general problems. From the gadget in the proof of Theorem 4, it is not necessary to keep degree of vertices in the gadget bounded. Thus, vertices in D are all connected to each other.

Let ρ, x_0, μ_v be the same parameters as in the proof of Theorem 4. The sufficient conditions to make the construction feasible are:

- (GDS): To be able to connect each vertex $v \in (\mathcal{S} \cup \mathcal{U})$ to μ_v vertices in D :

$$|D| = O\left(\max_{v \in (\mathcal{S} \cup \mathcal{U})} \theta\left(\frac{\rho}{1-\rho} \Delta_S\right)\right) = O(\Delta_S) = O(m^{3/2})$$

- ($k\text{C-GDS}$): To add at least k edges from each vertex in \mathcal{S} to vertices in D :

$$|D| = O(k|\mathcal{S}|) = O(|\mathcal{S}|) = O(n^l (5/3)^{l/2} 2^{2l}) = O(n^l 2^{\theta(l)})$$

- (T-GDS): Vertices in D have to dominate themselves. Since $\mu_v = \max\{\lceil \frac{\rho}{1-\rho} d(v) \rceil, x_0\}$, this can be satisfied when

$$|D| - 1 = O\left(\frac{\rho}{1-\rho} \frac{1}{|D|} \sum_{v \in (\mathcal{S} \cup \mathcal{U})} \mu_v\right).$$

Or equivalently

$$|D|^2 = O\left(\sum_{v \in (\mathcal{S} \cup \mathcal{U})} d(v) + x_0(|\mathcal{S}| + |\mathcal{U}|)\right)$$

$$= O\left(2 \sum_{v \in \mathcal{U}} d(v) + |\mathcal{S}| + |\mathcal{U}|\right) = O(m R k 2^l)$$

To summarize, the sufficient condition so that the dominating set of the graph is k -connected and total at the same time is

$$|D| = O(m 2^{\theta(l)} + n^l 2^{\theta(l)} + (m R k 2^l)^{1/2}). \quad (4)$$

Notice that, from the proof of [Theorem 2](#), the hardness ratio is in the form $\frac{(1 - \frac{4}{k})kQ \ln m + |D|}{kQ + |D|}$. In the Feige's reduction, $|D| = O(\Delta_S) = O((5n)^{\frac{2l}{\varepsilon}} 2^{\theta(l)})$ that yields the $(1 - \varepsilon) \ln n$ hardness ratio for set cover but makes the hardness ratio of the domination problem gets arbitrary close to 1. Fortunately, it is possible to reduce the maximum degree by setting $m = (5n)^{cl}$ with a small constant $c > 0$. The consequence is that $\ln m$ is no longer $\ln N + O(1)$; thus, the inapproximability ratio is reduced.

The optimal setting to get the best inapproximability ratio is to set $m = (5n)^{l(1-\varepsilon)}$ for some $\varepsilon > 0$. Then, $N = mR = (5n)^{l(2-\varepsilon)}$, or $m = N^{\frac{1-\varepsilon}{2-\varepsilon}}$. Hence, from (4), it is sufficient that

$$|D| = n^l \frac{2^{\theta(l)}}{n^{l\frac{\varepsilon}{2}}} = o(Q)$$

for sufficiently large l and n . Hence, the hardness ratio will be

$$\frac{(1 - \frac{4}{k})kQ \ln m + o(Q)}{kQ + o(Q)} > \left(1 - \frac{5}{k}\right) \ln m.$$

The number of vertices in the graph is

$$n_H = 2|D| + |\mathcal{S}| + |\mathcal{U}| < \theta(m 3^{l/2}) + n^l 2^{2l} \left(\frac{5}{3}\right)^{l/2} + (5n)^{2l-\varepsilon} < 2|\mathcal{U}| = 2N.$$

Thus, the hardness ratio is at least

$$\left(1 - \frac{5}{k}\right) \ln \left(\frac{n_H}{2}\right)^{1/2 - \frac{\varepsilon}{4-2\varepsilon}} > \left(1 - \frac{5}{k}\right) \frac{1}{2} \left(1 - \frac{\varepsilon}{2-\varepsilon}\right) \ln n_H - \theta(1) > \frac{1}{2}(1-\varepsilon) \ln n_H$$

for sufficiently large k and sufficiently small ε . □

Theorem 7 Domination problems in families GDS, T-GDS, and kC -GDS with threshold functions $r_v(x) = O(\log x)$ cannot be approximated within $(1 - o(1)) \ln n$ where n is the number of vertices, unless $\text{NP} \subset \text{DTIME}(n^{O(\log \log n)})$.

Proof When $r_v(x) = O(\log x)$, the size of $|D|$ is only $O(\log \Delta_S)$. The original settings in the Feige's reduction will give the ratio $(1 - \varepsilon) \ln n_H$. □

Multiple-Hop PIDS Problem. The hardness is shown through a reduction from the one-hop PIDS problem to the multiple-hop PIDS problem with $d \geq 2$. The hardness result follows immediately by the [Theorem 2](#) in the previous section. Given a graph $G = (V, E)$ as an instance of the PIDS problem, construct an instance $G' = (V', E')$ of the PIDS problem as follows (and as illustrated in [Fig. 5](#)).

Replace each edge $(u, v) \in E$ by a gadget called transmitter. The transmitter connecting vertices u and v is a chain of $d - 1$ path, named uv_1 to uv_{d-1} . The vertex u is connected to uv_1 , uv_1 is connected to uv_2 and so on, and vertex uv_{d-1} is connected to v . Each vertex uv_i , $i = 1 \dots d - 1$, is connected to all flash points. An example for transmitter is shown in [Fig. 3](#). The transmitter is designed so that if all flash points and vertex u are selected at the beginning, then vertex uv_{d-1} will be activated after $d - 1$ rounds. Hence, the number of activated neighbors of v after $d - 1$ rounds will equal the number of selected neighbors of v in the original graph ([Fig. 4](#)).

Finally, each edge (w_p, z_p) is replaced with a transmitter. In order to activate all dummy vertices z_p after d rounds, assume, w.l.o.g., that all flash points must be selected in an optimal solution. The following lemma follows directly from the construction.

Lemma 4 *Every solution of size k for the one-hop ($d = 1$) PIDS problem in G induces a solution of size $k + c(\rho)$ for the d -hop PIDS problem in G' .*

On another direction, the following lemma holds.

Lemma 5 *An optimal solution of size k' for the d -hop PIDS problem induces a size $k' - c(\rho)$ solution for the one-hop PIDS problem in G .*

Proof For a transmitter connecting u to v , if the solution of the d -hop PIDS problem contains any of the intermediate vertices uv_1, \dots, uv_{d-1} , that intermediate vertex can be replaced with either u or v to obtain a new solution of same size (or less). Hence, assume, w.l.o.g., that none of the intermediate vertices are selected. Therefore, all flash points must be selected in order to activate the dummy vertices. It is easy to see that the solution of d -hop PIDS excluding the flash points will be a solution of one-hop PIDS in G with size $k' - c(\rho)$. \square

Note that the number of vertices in G' is upper bounded by dn^2 , i.e., $\ln |V'| < 2\ln|V| + \ln d$. Thus, using the same arguments used in the proof of [Theorem 6](#), it can be shown that a $(\frac{1}{4} - \varepsilon) \ln n$ approximation algorithm leads to a $(\frac{1}{2} - \varepsilon) \ln n$ approximation algorithm for the one-hop PIDS problem (contradicts [Theorem 6](#)).

Theorem 8 *The PIDS problem cannot be approximated within $(\frac{1}{4} - \varepsilon) \log n$ for $d \geq 1$, unless $NP \subset DTIME(n^{O(\log \log n)})$.*

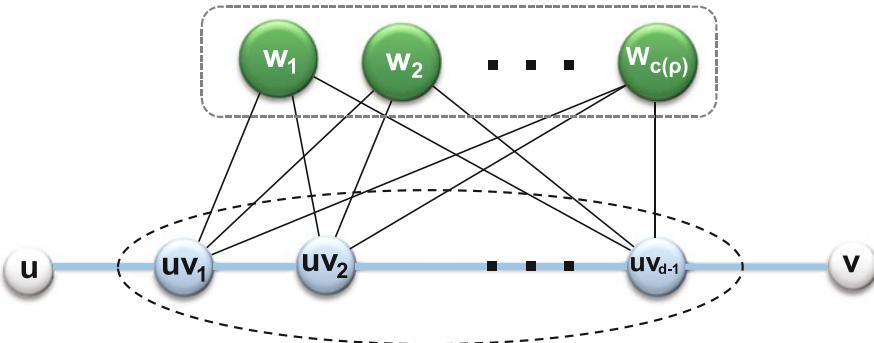


Fig. 3 The transmitter gadget

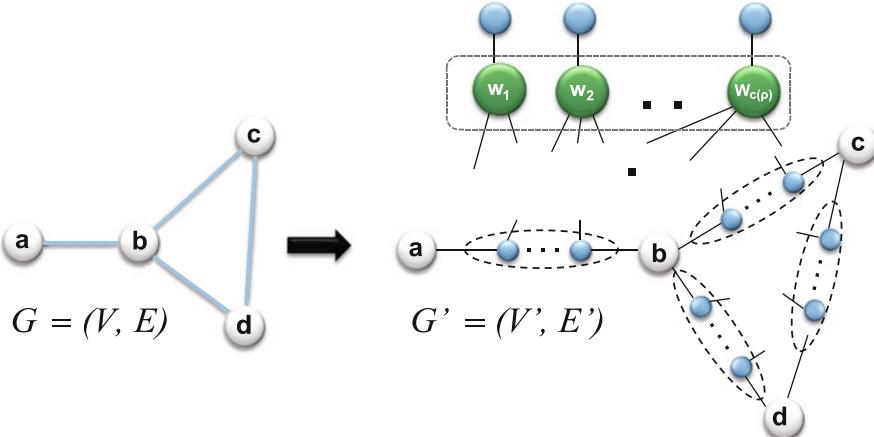


Fig. 4 Gap reduction from one-hop PIDS to d -hop PIDS

3.2 Approximation Algorithm

This part presents briefly how approximation algorithms for studied domination problems can be obtained via a simple reduction to the constrained multiset multicover (CMM) problem. In turn, the MMC problem can be seen as an integer covering problem. Thus, the domination problems benefit directly from advances in integer linear programming and the integer covering problems in particular.

Generalized Dominating Sets. It is easy to show that all domination problems in the generalized class can be approximated within a factor $\ln \Delta + O(1)$.

Theorem 9 Given a graph $G = (V, E)$, there exist $O((|V| + |E|) \log \log |V|)$ algorithms that approximate GDS within $H(2\Delta)$ and T-GDS within $H(\Delta)$.

Proof The proof begins with the definition of the constrained multiset multicover problem (CMM).

CONSTRAINED MULTISET MULTICOVER (CMM)

Input: A set cover instance $(\mathcal{U}, \mathcal{S})$. Each point e has an integer requirement r_e and occurs in a set S with arbitrary multiplicity, denoted $m(S, e)$. Moreover, associate a cost, c_S , with each set $S \in \mathcal{S}$.

Problem: The minimum cost subcollection which fulfils all elements' cover requirements provided each multiset is picked at most once.

Lemma 6 ([36]) There is a natural greedy algorithm that finds a constrained multiset multicover within an H_k factor of the optimal solution, where $k = \max_S \sum_e m(S, e)$.

The GDS problem on the graph $G = (V, E)$ can be reduced to the following instance of CMM:

- $\mathcal{U} = \{e_u : u \in V\}$.
- The cover requirement of e_u is set to $r_u = \lceil r_u(d(u)) \rceil$.
- $\mathcal{S} = \{S_v : v \in V\}$, where S_v contains $\{e_u : u \in N(v)\}$ plus r_v copies of e_v . That is, $m(S_v, e_u) = 1, \forall u \in N(v)$, and $m(S_v, e_v) = \lceil r_v(d(v)) \rceil$.

It follows that the GDS problem can be approximated within

$$H \left(\max_{v \in V} d(v) + r_v(d(v)) \right) \leq H(2\Delta) = H(\Delta) + O(1).$$

In case of T-GDS, the only difference in the reduction is that each multiset S_v contains all the neighbors of v , but not any copies of v . The approximation ratio is, hence, $H(\Delta)$.

Implementation Issue: Straightforward implementations might incur high time complexity. In each step, the greedy algorithms select the node with the highest coverage which is bounded by $O(n)$. Hence, using van Emde Boas priority queue [37] to maintain and update nodes' coverages, the total time complexity can be brought down to $O((|V| + |E|) \log \log |V|)$. Details are presented in the [Algorithm 1](#). \square

Corollary 1 There are polynomial time algorithms that approximate PIDS and T-PIDS problems within ratios $\ln \Delta + \frac{4}{3}$ and $\ln \Delta + 1$, respectively.

Proof Apply [Theorem 9](#) with $r_v(x) = \rho x$ and use the approximation $H(n) \approx \ln n + 0.58$; the approximation ratios for PIDS and T-PIDS can be rewritten as $(\ln \Delta + \frac{4}{3})$ and $(\ln \Delta + 1)$. \square

Connected Generalized Dominating Sets. The connected version of GDS problem requires either submodular theory or a conversion to the Steiner tree problem to obtain the approximation factor as presented in the proof of the following theorems.

Theorem 10 *There is a polynomial time algorithm that finds a weighted C-GDS of size at most $1.55 \ln \Delta + \frac{5}{2}$ time that of the minimum C-GDS.*

Proof The same approach for connected dominating problem in [17] can be applied. In the first stage, a GDS is found using the greedy heuristic described in **Theorem 9**. In the second stage, the vertices in the DS are connected using Steiner tree algorithm.

Since a GDS is also a (normal) dominating set, adding the optimal solution of C-GDS to the found GDS in the first stage gives a Steiner tree of the GDS.

Hence, using a c -approximation algorithm for Steiner tree to connect vertices in a GDS yields a $c(H(\Delta) + \ln 2 + 1)$ approximation algorithm for C-GDS problem. Apply the best known approximation ratio for Steiner tree problem $c \approx 1.55$ by Robins and Zelikovsky [38]; the two-stage algorithm admits a $1.55 \ln \Delta + 2.5$ approximation algorithm. \square

Theorem 11 *There are polynomial time algorithms that approximate unweighted C-GDS, and TC-GDS within ratios $H(3\Delta)$ and $H(2\Delta)$, respectively.*

Proof The greedy algorithm presented in [11] can be extended to work with arbitrary threshold function $r_v(x)$ instead of $r_v(x) = \lceil x/2 \rceil$. The analysis involves a non-submodular potential function that can be overcome with techniques in [39, 40]. Though, it is unclear if the technique can be extended for the weighted cases. \square

4 Domination Problems in Special Graph Classes

This section studies hardness and approximation algorithms for domination problems in special graph classes. In most cases, better approximation algorithms with improved ratio can be obtained. For example, almost all domination problems in power-law networks admit trivial constant factor approximation algorithms as shown later. In planar graphs and disk graphs, there exist *polynomial time approximation scheme* (PTAS) that approximate the optimal solution within a factor arbitrary close to one [41–46]. In the context of wireless sensor networks (WSNs), distributed $(1 + \varepsilon)$ approximation algorithm on unit disk graphs (UDGs) with $O(\log^* |G|)$ rounds is known [47]. However, PTAS is not known for the weighted version of the problem on UDGs. The latest result for that problem is a $5 + \varepsilon$ approximation algorithm [48].

4.1 Power-Law Networks

Many social, biological, and technology networks including OSNs display a non-trivial topological feature: their degree sequences can be well approximated by a power-law distribution [49]. Many optimization problems that are hard on general graphs can be solved much more efficient in power-law graphs [50, 51].

The analysis in this part uses the well-known $P(\alpha, \beta)$ model [52] in which there are y vertices of degree x , where x and y satisfy $\log y = \alpha - \beta \log x$. In other words,

$$|\{v : d(v) = x\}| = y = \frac{e^\alpha}{x^\beta}.$$

Basically, α is the logarithm of the size of the graph and β is the log-log growth rate of the graph. Graphs with the same β value often express the same behaviors and common characteristics. Hence, it is natural to categorize all graphs sharing a β value into a family of graphs and regard to β as a constant (not a part of the input). Without affecting the conclusions, real numbers will be used instead of rounding down to integers. The error terms can be easily bounded and are sufficiently small in the proofs.

The maximum degree in a $P(\alpha, \beta)$ graph is $e^{\frac{\alpha}{\beta}}$. The number of vertices and edges are

$$n = \sum_{x=1}^{e^{\frac{\alpha}{\beta}}} \frac{e^\alpha}{x^\beta} \approx \begin{cases} \zeta(\beta)e^\alpha & \text{if } \beta > 1 \\ \alpha e^\alpha & \text{if } \beta = 1 \\ \frac{e^{\frac{\alpha}{\beta}}}{1-\beta} & \text{if } \beta < 1 \end{cases}, \quad m = \frac{1}{2} \sum_{x=1}^{e^{\frac{\alpha}{\beta}}} x \frac{e^\alpha}{x^\beta} \approx \begin{cases} \frac{1}{2}\zeta(\beta-1)e^\alpha & \text{if } \beta > 2 \\ \frac{1}{4}\alpha e^\alpha & \text{if } \beta = 2 \\ \frac{1}{2} \frac{e^{\frac{2\alpha}{\beta}}}{2-\beta} & \text{if } \beta < 2 \end{cases}$$

where $\zeta(\beta) = \sum_{i=1}^{\infty} \frac{1}{i^\beta}$ is the Riemann zeta function.

Theorem 12 *Minimum PIDS is APX-hard, i.e., there is a positive constant ε depends only on the log-log growth rate β such that approximating PIDS within $1 + \varepsilon$ is NP-hard [53].*

Theorem 13 *In a power-law graph $G \in P(\alpha, \beta)$, the size of the optimal PIDS*

$$\text{OPT}_{\text{PIDS}} = \begin{cases} \Omega(n^\beta) & \text{if } \beta < 1 \\ \Omega(n/\log n) & \text{if } \beta = 2 \\ \Omega(n) & \text{if } \beta > 2. \end{cases}$$

Proof 1. Let k be the size of the optimal PIDS. Note that all nodes of degree more than k/ρ must be selected (otherwise, the number of selected neighbors will exceed k). The number of nodes with degree larger than k will be

$$k > \sum_{x=k/\rho}^{e^{\frac{\alpha}{\beta}}} \frac{e^\alpha}{x^\beta} > \sum_{x=k/\rho}^{e^{\frac{\alpha}{\beta}}} \frac{e^\alpha}{x} = e^\alpha (\ln e^{\frac{\alpha}{\beta}} - \ln k). \quad (5)$$

Solving the above relation gives us $k > \Omega(e^\alpha) = \Omega(n^\beta)$.

2. Similarly, if $\beta = 2$, then $k = \Omega(e^\alpha) = \Omega(n/\log n)$.
3. The lower bound is obtained through a dual-setting approach. Consider the following linear program and its dual of the PIDS problem:

$$\begin{array}{ll} \textbf{LP:} \min & \sum_{v \in V} x_v \\ & \text{s. t. } r_v x_v + \sum_{u \in N(v)} x_u \geq r_v \\ & \quad -x_u \geq -1 \\ & \quad x_u \geq 0 \end{array} \quad \begin{array}{ll} \textbf{DP:} \max & \sum_{u \in V} r_u y_u - \sum_{v \in V} z_v \\ & \text{s. t. } r_u y_u + \sum_{v \in N(u)} y_v - z_u \leq 1 \\ & \quad z_v \geq 0 \\ & \quad y_v \geq 0 \end{array} \quad (6)$$

where $r_u = \rho d_u$. Note that for the integral versions of LP (16) and DP (16), both settings $r_u = \rho d_u$ and $r_u = \lceil \rho d_u \rceil$ yield the same optimal solutions; however, setting $r_u = \rho d_u$ simplifies the approximation ratio analysis.

Set $y_u = \gamma \forall u \in V$. The goal is to look for value of γ that achieves the tightest lower bound on the size of the optimal PIDS.

To satisfy constraints in the dual, set $z_u = \max\{(\rho + 1)d_u\gamma - 1, 0\}$. Then the objective value becomes

$$DP = \rho\gamma \sum_{u \in V} d_u - \sum_{u \in V} \max\{(\rho + 1)d_u\gamma - 1, 0\} \quad (7)$$

$$= \rho\gamma \sum_{u \in V} d_u - \sum_{d_u > \tau(\gamma)} ((\rho + 1)d_u\gamma - 1) \quad (8)$$

where $\tau(\gamma)$ denotes $(\rho + 1)^{-1}\gamma^{-1}$.

Substituting $\gamma = \frac{1}{(\rho+1)\tau}$ into (8), this leads to

$$DP = \frac{\rho}{(\rho + 1)\tau} \sum_{x=1}^{e^{\frac{\alpha}{\beta}}} \frac{e^\alpha}{x^\beta} x - \sum_{x>\tau} \left(\frac{1}{\tau} \frac{e^\alpha}{x^\beta} x - \frac{e^\alpha}{x^\beta} \right) \quad (9)$$

$$= \frac{\rho \zeta(\beta - 1)}{(\rho + 1)\tau} e^\alpha - \sum_{x>\tau} \left(\frac{1}{\tau} \frac{e^\alpha}{x^{\beta-1}} - \frac{e^\alpha}{x^\beta} \right). \quad (10)$$

Except for at most $\lfloor e^{\frac{\alpha}{\beta}} \rfloor$ points $\tau = 1, 2, \dots, \lfloor e^{\frac{\alpha}{\beta}} \rfloor$, the derivative of the objective function, $\frac{dDP}{d\tau}$, is defined. Moreover, at those integral points, both one-sided limits,

$\lim_{\tau \rightarrow i^-} DP$ and $\lim_{\tau \rightarrow i^+} DP$, agree, i.e., DP is a continuous function everywhere with respect to τ .

Lemma 7 For every $\tau \in (i, i + 1), i \in \mathbb{N}^+$, the derivative $\frac{d DP}{d \tau}$ is defined and satisfies

$$\frac{d DP}{d \tau} = -\frac{1}{\tau^2} \left(\frac{\rho \zeta(\beta - 1)}{\rho + 1} e^\alpha - \sum_{x > \tau} \frac{e^\alpha}{x^{\beta-1}} \right). \quad (11)$$

By (11), there exists a fixed dividing point $x_0 \in \mathbb{N}^+$ that depends only on β , satisfying $\frac{d DP}{d \tau}(\tau) \geq 0, \forall \tau < x_0$ and $\frac{d DP}{d \tau}(\tau) < 0, \forall \tau > x_0$. Since DP is continuous everywhere, it obtains the global maximum value at $\tau = x_0$.

It is needed to show that the value of DP at $\tau = x_0$ is $\Omega(n)$, and since the objective of the primal is lower bounded by DP , it follows that the size of the minimum PIDS will be at least $\Omega(n)$.

$$\begin{aligned} DP(x_0) &= \frac{1}{x_0} \left(\frac{\rho \zeta(\beta - 1)}{\rho + 1} e^\alpha - \sum_{x > x_0} \frac{e^\alpha}{x^{\beta-1}} \right) + \sum_{x > x_0} \frac{e^\alpha}{x^\beta} \\ &\geq \sum_{x > x_0} \frac{e^\alpha}{x^\beta} \approx \left(\zeta(\beta) - \sum_{x \leq x_0} \frac{1}{x^\beta} \right) e^\alpha \approx \left(1 - \frac{\sum_{x \leq x_0} \frac{1}{x^\beta}}{\zeta(\beta)} \right) n = \Omega(n). \square \end{aligned}$$

Using the same approach in [Theorem 13](#) gives the similar bounds for T-PIDS in the following theorem.

Theorem 14 In a power-law graph $G \in P(\alpha, \beta)$, the size of the optimal T-PIDS.

$$\text{OPT}_{\text{T-PIDS}} = \begin{cases} \Omega(n) & \text{if } \beta < 1 \text{ or } \beta > 2 \\ \Omega(n/\log n) & \text{if } \beta = 1. \end{cases}$$

Theorem 15 In a power-law graph $G \in P(\alpha, \beta)$, the size of the optimal C-PIDS

$$\text{OPT}_{\text{C-PIDS}} = \begin{cases} \Omega(n) & \text{if } \beta > 2 \\ \Omega(n/\log n) & \text{if } \beta = 1. \end{cases}$$

If networks have optimal PIDS/T-PIDS/C-PIDS of $\Omega(n)$ size, clearly, any algorithms that produce valid PIDS/T-PIDS/C-PIDS will be constant factor approximation algorithms.

Theorem 16 Given a power-law $P(\alpha, \beta)$ graph $G = (V, E)$:

1. $\beta < 1$: PIDS is not in APX (cannot be approximated within a constant factor), while T-PIDS admits a constant factor approximation algorithm.
2. $\beta > 2$: There exist constant factor approximation algorithms for PIDS, T-PIDS, and C-PIDS problems.

Theorem 17 Given a power-law network $G \in P(\alpha, \beta)$, with $\beta > 2$ and constant $0 < \rho < 1$, any d -seeding is of size at least $\Omega(n)$.

Proof The proof consists of two parts. It is shown in the first part that the volume, i.e., the total degree of vertices, of any d -seeding must be $\Omega(m)$. The second part shows that any subset of vertices $S \subset V$ with volume $\text{vol}(S) = \Omega(m)$ in a power-law network with power-law exponent $\beta > 2$ will imply that $|S| = \Omega(n)$. Thus, the theorem follows.

In the first part, there are two following cases.

Case $\rho > \frac{1}{2}$: Let $S = R_0$ be the optimal solution for the PIDS problem on $G = (V, E)$, and $S = R_0, R_1, R_2, \dots, R_d$ are vertices that become active at round $0, 1, 2, \dots, d$, respectively (see Fig. 5). Thus, $\{R_i\}_{i=0}^d$ form a partition of V . Moreover, for each $1 \leq t \leq d$ the following inequality holds:

$$|\phi(R_t, \bigcup_{i=1}^{t-1} R_i)| \geq \frac{\rho}{1-\rho} \left(|\phi(R_t, \bigcup_{j=t+1}^d R_j)| + 2|\phi(R_t, R_t)| \right) \quad (12)$$

where $\phi(A, B)$ denotes the set of edges connecting one vertex in A to one vertex in B . This can be seen as at least a fraction $\frac{\rho}{1-\rho}$ among edges incident with the vertices activated in round t must be incident with active vertices in the previous rounds.

Sum up all inequalities in (12) for $t = 1 \dots d$. This yields

$$\sum_{t=1}^d |\phi(R_t, \bigcup_{i=1}^{t-1} R_i)| \geq \frac{\rho}{1-\rho} \sum_{t=1}^d \left(|\phi(R_t, \bigcup_{j=t+1}^d R_j)| + 2|\phi(R_t, R_t)| \right).$$

After eliminating the common factors in both sides,

$$\begin{aligned} & \sum_{i=0}^{d-1} |\phi(R_i, \bigcup_{t=i+1}^d R_t)| \\ & \geq \frac{\rho}{1-\rho} \sum_{j=1}^{d-1} |\phi(R_j, \bigcup_{t=j+1}^d R_t)| + 2 \sum_{t=1}^{d-1} |\phi(R_t, R_t)|. \end{aligned}$$

And after some algebra,

$$\text{vol}(R_0) \geq |\phi(R_0, \bigcup_{t=1}^d R_t)|$$

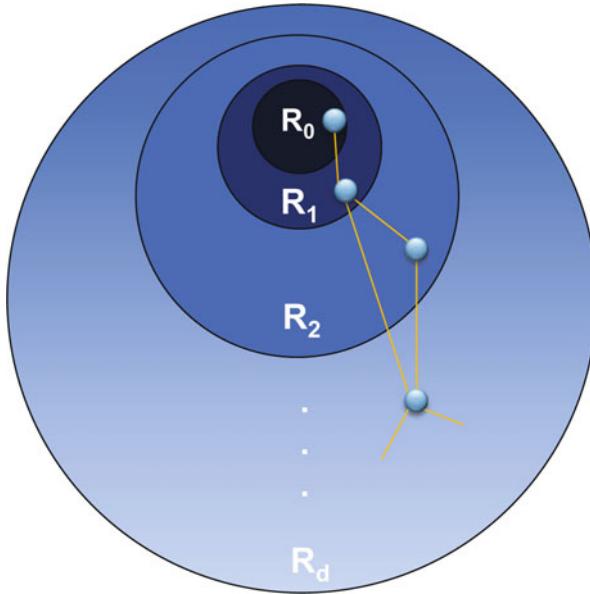


Fig. 5 The influence propagation in the network

$$\begin{aligned}
 &\geq \frac{2\rho - 1}{1 - \rho} \sum_{j=1}^{d-1} |\phi(R_i, \bigcup_{t=j+1}^d R_t)| + 2 \sum_{t=1}^d |\phi(R_t, R_t)| \\
 &\Leftrightarrow \frac{\rho}{1 - \rho} |\phi(R_0, V)| - |\phi(R_0, R_0)| \\
 &\geq \frac{2\rho - 1}{1 - \rho} |E| + \frac{3 - 4\rho}{1 - \rho} \sum_{t=1}^d |\phi(R_t, R_t)|. \tag{13}
 \end{aligned}$$

Hence, when $\rho > 1/2$, $\text{vol}(R_0) \geq \frac{2\rho-1}{1-\rho} |E| = \Omega(m)$ for any d -seeding R_0 .

Case $\rho \leq \frac{1}{2}$: An edge is said to be active if it is incident to at least one active vertex. At round $t = 0$, there are at most $\text{vol}(R_0)$ active edges, those who are incident to R_0 . Equation 12 implies that the number of active edges in each round increases at most ρ^{-1} times. After d rounds, the number of active edges will be bounded by $\text{vol}(R_0) \times \rho^{-d}$. Since all edges are active at the end, the following inequality holds:

$$\text{vol}(R_0) \geq \rho^{-d} |E|.$$

The second part of the proof shows that if a subset $S \subset V$ has $\text{vol}(S) = \Omega(m)$, then $|S| = \Omega(n)$ whenever the power-law exponent $\beta > 2$. Assume that $\text{vol}(S) \geq cm$, for some positive constant c . The size of S is minimum when S contains only the

highest degree vertices of V . Let k_0 be the minimum degree of vertices in S in that extreme case, it follows that

$$cm = \frac{c}{2} \sum_{k=1}^{e^{\frac{\alpha}{\beta}}} k \frac{e^\alpha}{k^\beta} \leq \text{vol}(S) \leq \frac{1}{2} \sum_{k=k_0}^{e^{\frac{\alpha}{\beta}}} k \frac{e^\alpha}{k^\beta}.$$

Simplifying two sides, it follows that

$$\sum_{k=1}^{k_0-1} \frac{1}{k^{\beta-1}} \leq (1-c) \sum_{k=1}^{e^{\frac{\alpha}{\beta}}} \frac{1}{k^{\beta-1}} = (1-c)\zeta(\beta-1).$$

Since, the zeta function $\zeta(\beta-1)$ converges for $\beta > 2$, there exists a constant $k_{\rho,\beta}$ that depends only on ρ and β that satisfies

$$\sum_{k=1}^{k_{\rho,\beta}} \frac{1}{k^{\beta-1}} > (1-c)\zeta(\beta-1).$$

Obviously, $k_0 \leq k_{\rho,\beta}$. Thus, the number of vertices that are in S is at least

$$\sum_{k=k_{\rho,\beta}}^{e^{\frac{\alpha}{\beta}}} \frac{e^\alpha}{k^\beta} = \left(1 - \sum_{k=1}^{k_{\rho,\beta}} \frac{1}{k^\beta}\right)n = \Omega(n).$$

The last step holds because the sum $\sum_{k=1}^{k_{\rho,\beta}} \frac{1}{k^\beta}$ is bounded by a constant since $k_{\rho,\beta}$ is a constant. \square

In both cases $\rho > 1/2$ and $\rho \leq 1/2$, the size of a d -seeding set is at least $\Omega(n)$. However, there is a clear difference in the propagation speed with respect to d between two cases. When $\rho < 1/2$, the number of active edges can increase exponentially (but is still bounded if d is a constant), and it is likely that the number of active vertices also exponentially increases. In contrast, when $\rho > 1/2$, exploding in the number of active edges (and hence active vertices) is impossible as the volume of the d -seeding is tied to the number of edges m by a fixed constant $\frac{2\rho-1}{1-\rho}$, regardless of the value of d .

Theorem 18 *In power-law networks with power exponent $\beta > 1$, there are $O(1)$ approximation algorithms for the PIDS problem for bounded value of $d \geq 1$.*

4.2 Dense Graphs

Lemma 8 *If P_T is a T-PIDS of $G = (V, E)$, then $|P_T| \geq \Omega(\sqrt{|V| + |E|})$.*

Proof Let $k = |P_T|$ be the size of a T-PIDS. All $v \in V \setminus P_T$ must be adjacent to at least one vertex in P_T . Thus, $|V| \leq |P_T| + \sum_{v \in P_T} |N(v) \setminus P_T|$. Moreover, for each vertex $v \in P_T$, $|N(v) \cap P_T| \geq \rho|N(v)| \Rightarrow |N(v) \setminus P_T| \leq \frac{1-\rho}{\rho}|N(v)| \leq \frac{1-\rho}{\rho}(k-1)$. Therefore,

$$n \leq k + k \frac{1-\rho}{\rho}(k-1) = \frac{1-\rho}{\rho}k^2 + \frac{2\rho-1}{\rho}k. \quad (14)$$

Divide edges in E into three categories: (1) edges whose both ends are in P_T , (2) edges whose exact one end is in P_T , and (3) edges whose both ends are not in P_T . There are at most $\binom{k}{2}$ edges of type 1. For a vertex $v \in P_T$, the number of type 2 edges incident to v is at most $\frac{1-\rho}{\rho}(k-1)$ since v is adjacent to at most $k-1$ vertices in P_T . Hence, the number of type 2 edges is upper bounded by $2k \frac{1-\rho}{\rho}(k-1) = \frac{2(1-\rho)}{\rho} \binom{k}{2}$. For each vertex $u \notin P_T$, the number of type 3 edges incident to u is at most $\frac{1-\rho}{\rho}$ times the number of type 2 edges incident to u . Therefore, the number of type 3 edges is at most $\frac{(1-\rho)^2}{\rho^2} \binom{k}{2}$.

Adding all three types of edges together yields the following:

$$|E| \leq \left(1 + \frac{2(1-\rho)}{\rho} + \frac{(1-\rho)^2}{\rho^2}\right) \binom{k}{2} = \frac{1}{\rho^2} \binom{k}{2}. \quad (15)$$

It follows from (14) and (15) that $|P_T| = k = \Omega(\sqrt{|V| + |E|})$. \square

The bound is tight, i.e., there is a graph in which the minimum T-PIDS has size $\Omega(\sqrt{|V| + |E|})$. For example, consider a “hairy” clique of $n = k + k \cdot \lfloor \frac{1-\rho}{\rho}(k-1) \rfloor$ vertices and $m = \binom{k}{2} + k \cdot \lfloor \frac{1-\rho}{\rho}(k-1) \rfloor$ edges by connecting each vertex in a clique of size k to $\lfloor \frac{1-\rho}{\rho}(k-1) \rfloor$ leaf nodes (Fig. 6). The minimum T-PIDS will be the clique itself that is of size $k = \Omega(\sqrt{n+m})$.

Theorem 19 *For a dense graph $G = (V, E)$ with $|E| = \Omega(|V|^2)$, there exist constant factor approximation algorithms for PIDS, T-PIDS, and C-PIDS problems.*

4.3 Finding Optimal Solutions in Trees

In trees, it is possible to find optimal GDS and T-GDS in polynomial time. However, designing such algorithms in a linear-time fashion is not too obvious. This part

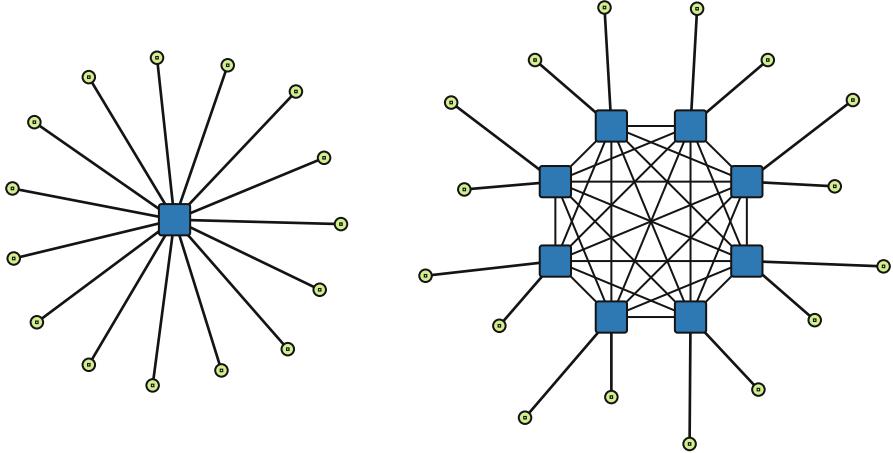


Fig. 6 A PIDS (*left*) may consist of only one node, while a T-PIDS (*right*) must contain at least $O(\sqrt{|V| + |E|})$

```

GDS-TREE( $G$ )
1:  $P = \emptyset$ 
2: PIDS-VISIT( $u$ ), for any  $u \in V$ 
3: return  $P$ 

GDS-VISIT( $u$ )
1: for each unvisited  $v \in N(u)$  do
2:   GDS-VISIT( $v$ )
3:   if  $r_p(v, P) > 0$  then
4:      $P = P \cup \{u\}$ 
5: if  $r_p(u, P) > 1$  then
6:    $P = P \cup \{u\}$ 

```

Fig. 7 GDS-TREE(G)

```

T-GDS-TREE( $G$ )
1:  $\mathcal{T} = \emptyset$ 
2: T-GDS-VISIT( $u, u$ ), for any  $u \in V$ 
3: return  $\mathcal{T}$ 

T-GDS-VISIT( $u, p_u$ )
1: for each unvisited  $v \in N(u)$  do
2:   T-GDS-VISIT( $v, u$ )
3: if  $r_t(u, \mathcal{T}) > 0$  then
4:    $\mathcal{T} = \mathcal{T} \cup \{p_u\}$ 
5: Select arbitrary  $r_t(u, \mathcal{T})$  unselected
neighbors(children) of  $u$  into  $\mathcal{T}$ .

```

Fig. 8 T-GDS-TREE(G)

presents two depth-first search-based (DFS) algorithms in [Figures 7](#) and [8](#) for GDS and T-GDS, respectively. Notice that the solution for kC -GDS and TC -GDS problems on trees is trivial; the optimal solution is simply the set of all non-leaf nodes.

Theorem 20 *Optimal GDS and T-GDS in trees can be found in linear time.*

Proof At a given step, P/\mathcal{T} denotes the current GDS/T-GDS. For each $v \in V$, define the functions $r_p(v, P) = \lceil r_v(d(v)) \rceil (1 - \mathbf{1}_P(v)) - |N(v) \cap P|$ and $r_t(v, \mathcal{T}) = \lceil r_v(d(v)) \rceil - |N(v) \cap \mathcal{T}|$, where $\mathbf{1}_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$

Functions $r_p(v, P), r_t(v, \mathcal{T})$ determine the minimum numbers of v 's neighbors to include into the optimal solutions. A node u with $r_p(u, P) > 0$ or $r_t(u, \mathcal{T}) > 0$ is called *uncovered*; otherwise, u is called *covered*.

Assume that the tree is rooted at some vertex u . For an edge (u, v) , if u is visited before v , then u is the parent of v and v is a child of u .

Correctness. Proof by induction that each selection step is optimal.

GDS: Assume that all the selections made so far are optimal. Assume node u , the parent of v is selected in step 3, Fig. 7. From $r_p(v, P) > 0$, it follows that

1. $v \notin P$ or else $r_p(v, P) < 0$.
2. $r_p(v, P) = 1$ if not v has already been selected by the end of GDS-VISIT(v).

To cover v , it is necessary to either select v, u or some children of v . However, since all nodes in the subtree rooted at v have been covered, there will be no extra benefit in selecting v or its children. Formally, if an optimal solution selects v or its children, it is always possible to replace the selected vertex with u and obtain a new optimal PIDS. In case $r_p(u, P) > 1$, the only choice is to select u .

T-GDS: After T-GDS-VISIT(v, p_v) finishes, v always becomes covered. Assume the selection of vertices into \mathcal{T} is optimal so far. During the visit of a node u , if $r_t(u, \mathcal{T}) > 0$, p_u , the parent of u is selected whenever $p_u \notin \mathcal{T}$. Since p_u might cover other vertices, while selecting, children of u will not affect any uncovered vertices other than u . Finally, children of u might be selected to fully covered u (but only after p_u is selected).

Time Complexity. Values of $r_p(v, P)$ and $r_t(v, \mathcal{T})$ can be maintained in $O(|V|)$. After a new vertex is added, it is necessary to update $r_p(\cdot)$ and/or $r_t(\cdot)$ values of that node and all its neighbors. Each node is added at most once; hence, the total cost has the same order with the total degree of all vertices, i.e., $2|V|$. Adding the time $O(|V|)$ taken by the DFS traversal, the overall time complexities are still $O(|V|)$. \square

5 Scalable Algorithm for Multiple-Hop PIDS

In order to understand the influence propagation when the number of propagation hops is bounded, this section introduces VirAds, an efficient algorithm for the PIDS problem. After that, VirAds's performance is examined through experiments in real-world large-scale social networks.

5.1 VirAds: Multiple-Hop PIDS Greedy Algorithm

With the huge magnitude of OSN users and data available on OSNs, scalability becomes the major problem in designing algorithm for PIDS. VirAds is scalable to network of hundred of millions links and provides high-quality solutions in our experiments.

Before presenting VirAds, let us consider a natural greedy for the PIDS problem in which the vertex that can activate the most number of inactive vertices within d hops is selected in each step. This greedy is unlikely to perform well on practice for the following two reasons. First, at early steps, when not many vertices are selected, every vertex is likely to activate only itself after being chosen as a seed. Thus, the algorithm cannot distinguish between good and bad seeds. Second, the algorithm suffers serious scalability problems. To select a vertex, the algorithm has to evaluate for each vertex v how many vertices will be activated after adding v to the seeding, e.g., by invoking an $O(m + n)$ breadth-first search procedure rooted at v . In the worst-case when $O(n)$ vertices are needed to evaluate, this alone can take $O(n(m + n))$. Moreover, as shown in the previous section, the seeding size can be easily $\Omega(n)$; thus, the worst-case running time of the naive greedy algorithm is $O(n^2(m + n))$, which is prohibitive for large-scale networks.

As shown in [Algorithm 1](#), our VirAds algorithm overcomes the mentioned problems in the naive greedy by favoring the vertex which can activate the most number of *edges* (indeed, it also considers the number of active neighbor around each vertex). This avoids the first problem of the naive greedy algorithm. At early steps, the algorithm behaves similar to the degree-based heuristics that favors vertices with high degree. However, when a certain number of vertices are selected, VirAds will make the selection based on the information within d -hop neighbor around the considered vertices rather than only one-hop neighbor as in the degree-based heuristic.

The scalability problem is tackled in VirAds by efficiently keeping track of the following measures for each vertex v :

- r_v : The round in which v is activated
- $N_v^{(e)}$: The number of new active edges after adding v into the seeding
- $N_v^{(a)}$: The number of extra active neighbors v needs in order to activate v
- $r_v^{(i)}$: The number of activated neighbors of v up to round i where $i = 1 \dots d$

Given those measures, VirAds selects in each step the vertex u with the highest *effectiveness* which is defined as $N_u^{(e)} + N_u^{(a)}$. After that, the algorithm needs to update the measures for all the remaining vertices.

Except for $N_v^{(e)}$, all other measures can be effectively kept track of in only $O((m + n)d)$ during the whole algorithm. When a vertex u is selected, it causes a chain reaction and activates a sequence of vertices or lower the rounds in which vertices are activated. New activated vertices are successively pushed into the queue Q for further updating much like what happens in the Bellman-Ford shortest-paths algorithm. Note that for each node $u \in V$, changing of r_u can cause at most d update for $r_w^{(.)}$ where w is a neighbor of u . For all neighbors of u , the total number of update is, hence, $O(d \cdot d(u))$. Thus, the total time for updating $r_w^{(.)} \forall w \in V$ in VirAds will be at most $O((m + n) \cdot d)$.

Algorithm 1: VirAds – Viral Advertising in OSNs

```

Input: Graph  $G = (V, E)$ ,  $0 < \rho < 1$ ,  $d \in \mathbb{N}^+$ 
Output: A small  $d$ -seeding
 $N_v^{(e)} \leftarrow d(v)$ ,  $N_v^{(a)} \leftarrow \rho \cdot d(v)$ ,  $r_v \leftarrow d + 1$ ,  $v \in V$ ;
 $r_v^{(i)} = 0$ ,  $i = 0..d$ ,  $P \leftarrow \emptyset$ ;
while there exist inactive vertices do
  repeat
     $u \leftarrow \text{argmax}_{v \notin P} \{N_v^{(e)} + N_v^{(a)}\}$ ;
    Recompute  $N_v^{(e)}$  as the number of
      new active edges after adding  $u$ .
  until  $u = \text{argmax}_{v \notin P} \{N_v^{(e)} + N_v^{(a)}\}$ ;
   $P \leftarrow P \cup \{u\}$ ;
  Initialize a queue:  $Q \leftarrow \{(u, r_v)\}$ ;
   $r_u \leftarrow 0$ ;
  foreach neighbor  $x$  of  $u$  do
     $N_x^{(a)} \leftarrow \max\{N_x^{(a)} - 1, 0\}$ ;
  while  $Q \neq \text{empty}$  do
     $(t, oldRound_t) \leftarrow Q.pop()$ ;
    foreach neighbor  $w$  of  $t$  do
      foreach  $i = r_t$  to  $\min\{oldRound_t - 1, r_w - 2\}$  do
         $r_w^{(i)} = r_w^{(i)} + 1$ ;
        if  $(r_w^{(i)} \geq \rho \cdot d_w) \wedge (r_w \geq d) \wedge (i + 1 < d)$  then
          foreach neighbor  $x$  of  $w$  do
             $N_x^{(a)} \leftarrow \max\{N_x^{(a)} - 1, 0\}$ ;
           $r_w = i + 1$ ;
          if  $w \notin Q$  then
             $Q.push((w, r_w))$ ;
  Output  $P$ ;

```

To maintain $N_v^{(e)}$, the easiest approach is to recompute all $N_v^{(e)}$. This approach, called *Exhaustive Update*, is extremely time-consuming as discussed in the naive greedy. Instead, $N_v^{(e)}$ is updated only if “necessary.” In details, vertices are stored in a max priority queue in which the priority is their *effectiveness*. In each step, the vertex u with the highest effectiveness is extracted, and $N_u^{(e)}$ is recomputed. If after updating, u still has the highest effectiveness, u is then selected. Otherwise, u is pushed back to the priority queue, the new vertex with the highest effectiveness is considered, and so on.

The PIDS problem can be easily shown to be NP-hard by a reduction from the set cover problem. Thus, there are only two possible choices: either designing heuristics which have no worst-case performance guarantees or designing approximation algorithms which can guarantee the produced solutions are within a certain factor from the optimal. Formally, a β -approximation algorithm for a minimization

(maximization) problem always returns solutions that are at most β times larger (smaller) than an optimal solution.

Unfortunately, there is unlikely an approximation algorithm with factor less than $O(\log n)$ as shown in next section. However, under the assumption that the network is power-law, our VirAds is an approximation algorithm for PIDS with a constant factor.

Theorem 21 *In power-law networks, VirAds is an $O(1)$ approximation algorithm for the PIDS problem for bounded value of d .*

The theorem follows directly from the result in previous section that the optimal solution has size at least $\Omega(n)$ in power-law networks. Thus, the ratio between the VirAds's solution and the optimal solution is bounded by a constant.

5.2 Experimental Study

This section presents experiments on OSNs to show the efficiency of VirAds algorithm in comparison with simple degree centrality heuristic. In addition, the experiments give insight into the trade-off between the number of times the information is allowed to propagate in the network and the seeding size.

5.2.1 Comparing to Optimal Seeding

One advantage of the discrete diffusion model over probabilistic ones [26, 27] is that the exact solution can be found using mathematical programming. This enables us to study the exact behavior of the seeding size when the number of propagation hop varies.

The PIDS problem can be formulated as a 0–1 integer linear programming (ILP) problem below.

$$\text{minimize} \sum_{v \in V} x_v^0 \quad (16)$$

$$\text{subject to} \sum_{v \in V} x_v^d \geq |V| \quad (17)$$

$$\sum_{w \in N(v)} x_w^{i-1} + \lceil \rho \cdot d(v) \rceil x_v^{i-1} \geq \lceil \rho \cdot d(v) \rceil x_v^i \quad (18)$$

$$x_v^i \geq x_v^{i-1} \quad \forall v \in V, i = 1 \dots d \quad (19)$$

$$x_v^i \in \{0, 1\} \quad \forall v \in V, i = 0 \dots d \quad (20)$$

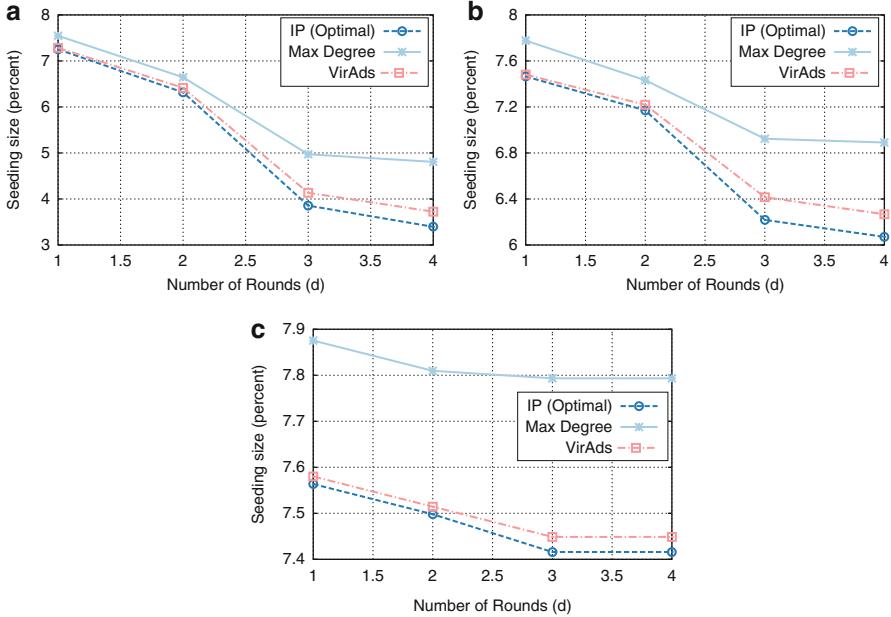


Fig. 9 Seeding size (in percent) on Erdos's collaboration network. VirAds produces close to the optimal seeding in only fractions of a second (in comparison to 2-day running time of the IP(optimal)). (a) $\rho = 0.4$. (b) $\rho = 0.6$. (c) $\rho = 0.8$

$$\text{where } x_v^i = \begin{cases} 0 & \text{if } v \text{ is inactive at round } i \\ 1 & \text{otherwise} \end{cases}.$$

The objective of the ILP is to select a minimum number of seeds at the beginning. The constraint (2) guarantees all nodes are activated at the end, while (3) deals with propagation condition; the constraint (4) is simply to keep vertices active once they are activated.

The ILP problem is solved on Erdos collaboration networks, the social network of the famous mathematician [54]. The network consists of 6,100 vertices and 15,030 edges. The ILP is solved with the optimization package GUROBI 4.5 on Intel Xeon 2.93 Ghz PC and setting the time limit for the solver to be 2 days. The running time of the IP solver increases significantly when d increases. For $d = 1, 2$, and 3 , the solver return the optimal solutions. However, for $d = 4$, the solver cannot find the optimal solutions within the time limit and returns suboptimal solutions with relative errors at most 15 %.

The optimal (or suboptimal) seeding sizes are shown in Fig. 9a–c, for $\rho = 0.4, 0.6$ and 0.8 , respectively. VirAds provides close-to-optimal solutions and performs much better than Max Degree. Especially, when $\rho = 0.8$, the VirAds's

Table 3 Sizes of the investigated networks

	Physics	Facebook	Orkut
Vertices	37,154	90,269	3,072,441
Edges	231,584	3,646,662	223,534,301
Avg. degree	12.5	80.8	145.5

seeding is only different with the optimal solutions by one or two nodes. In addition, VirAds only takes fractions of a second to generate the solutions.

As shown in [Theorem 17](#), the seeding takes a constant fraction of nodes in the network. For Erdos collaboration network, the seeding consists of 3.8–7 % the number of nodes in the networks. Further, the seeding can consist as high as 20–40 % nodes in the network for larger social networks in next section.

Although the mathematical approach can provide accurate measurement on the optimal seeding size, it cannot be applied for larger networks. The rest of our experiments measures the quality and scalability of our proposed algorithm VirAds on a collection of large networks.

5.2.2 Large Social Networks

All the algorithms are tested on networks of various sizes including coauthors network in Physics sections of the e-print arXiv [\[26\]](#), Facebook [\[55\]](#), and Orkut [\[56\]](#), a social networking run by Google. Links in all three networks are undirected and unweighted. The sizes of the networks are presented in [Table 3](#).

Physics: The physics coauthors network shall be referred as Physics network or simply Physics. Each node in the network represents an author, and there is an edge between two authors if they coauthor one or more papers. *Facebook* dataset consists 52 % of the users in the New Orleans [\[55\]](#). *Orkut* dataset is collected by performing crawling in last 2006 [\[56\]](#). It contains about 11.3 % of Orkut’s users.

5.2.3 Solution Quality in Large Social Networks

The VirAds algorithm is compared with the following heuristic *Random* method in which vertices are picked up randomly until forming a d -seeding and *Max Degree* method in which vertices with highest degree are selected until forming a d -hop seeding. Finally, VirAds is compared with its naive implementation, called *Exhaustive Update*, in which after selecting a vertex into the seeding, the effectiveness of all the remaining vertices are recalculated. With more accurate estimation on vertex effectiveness, Exhaustive Search is expected to produce higher-quality solutions than those of VirAds.

The seeding size with different number of propagation hop d when $\rho = 0.3$ is shown in [Fig. 10](#). To our surprise, VirAds even performs equal or better than *Exhaustive Update* despite that it uses significantly less effort to update vertex effectiveness. VirAds has smaller seeding in Physics than *Exhaustive Update*; both of them give similar results for Facebook, while *Exhaustive Update* cannot finish

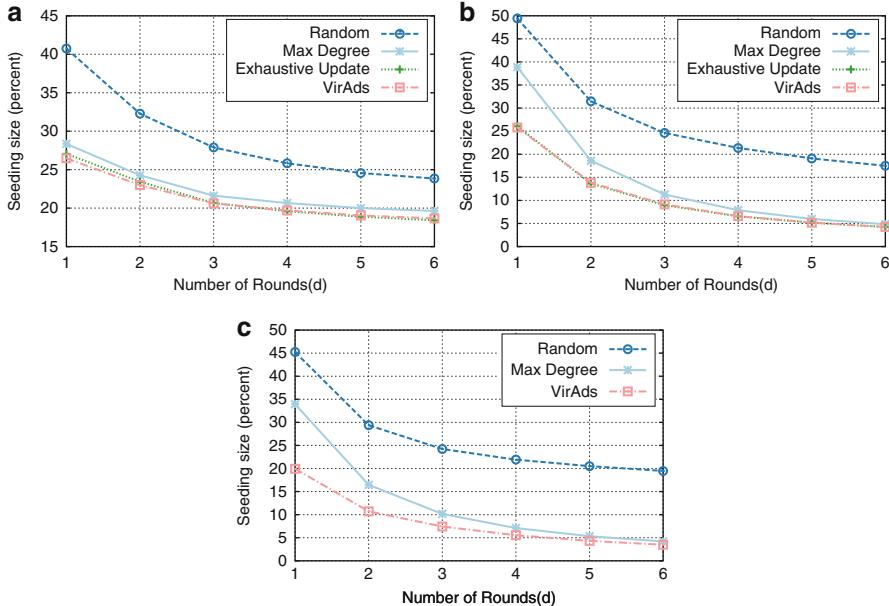


Fig. 10 Seeding size when the number of propagation hop d varies ($\rho = 0.3$). VirAds consistently has the best performance (a) Physics. (b) Facebook. (c) Orkut

on Orkut after 48 h and was forced to terminate. Sparingly update, the vertices' effectiveness turns out to be efficient enough since the influence propagation is locally bounded. In addition, the seeding produced by VirAds is almost two times smaller than those of *Random*.

The gap between VirAds and Max Degree is narrowed when the number of maximum hops increases. Hence, selecting nodes with high degrees as seeding is a good long-term strategy, but might not be efficient for fast propagation when the number of hops is limited. In Facebook and Orkut, when $d = 1$, *Max Degree* has 60–70 % more vertices in the seeding than *VirAds*. In Physics, the gap between *VirAds* and the *Max Degree* is less impressive. Nevertheless, *VirAds* consistently produces the best solutions in all networks.

5.2.4 Scalability

The running time of all methods at different propagation hop d is presented in Fig. 11. The time is measured in second and presented in the log scale. The running times increase slightly together with the number of propagation rounds d and are proportional to the size of the network. The *Exhaustive Update* has the worst running time, taking up to 15 min for Physics and 20 min for Facebook. For Orkut, the algorithm cannot finish within 2 days, as mentioned. The three remaining

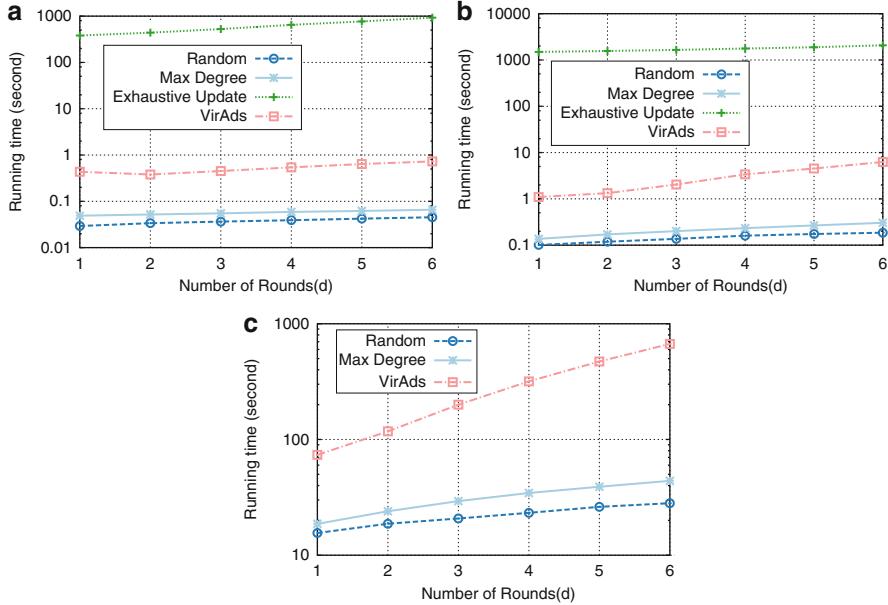


Fig. 11 Running time when the number of propagation hop d varies ($\rho = 0.3$). Even for the largest network of 110 million edges, VirAds takes less than 12 min **(a)** Physics. **(b)** Facebook. **(c)** Orkut

algorithms *VirAds*, *Max Degree*, and *Random* take less than 1 s for Physics and less than 10 s for Facebook. Even on the largest network Orkut with more than 220 million edges, VirAds requires less than 12 min to complete.

5.2.5 Influence Factor

The performance of VirAds and the other method at different influence factor ρ is presented. The number of propagation rounds d is fixed to 4. The size of d -seeding sets are shown in Fig. 12. VirAds is clearly still the best performer. The seeding sizes of VirAds are up to five times smaller than those of Max Degree for small ρ (although it's hard to see this on the charts due to small seeding sizes).

Since all tested networks are social networks with small diameter, the seeding sizes go to zero when ρ is close to zero. The exception is the Physics, in which the seeding sizes do not go below 10 % the number of vertices in the networks even when $\rho = 0.05$. A closer look into the Physics network reveals that the network contains many isolated cliques of small sizes (2, 3, 4, and so on) which correspond to authors that appear in only one paper. In each clique, regardless of the threshold ρ , at least one vertex must be selected; thus the, seeding size cannot

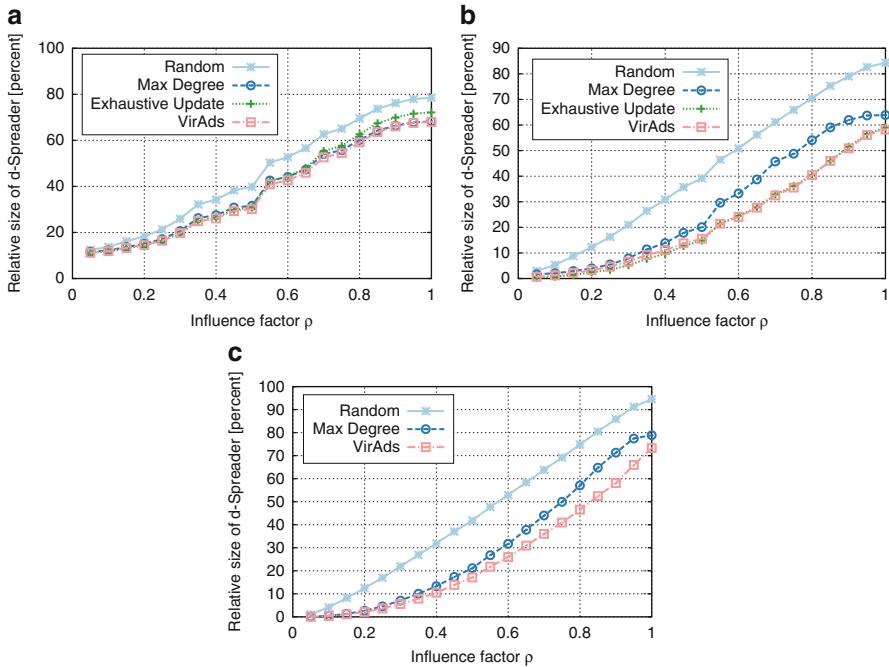


Fig. 12 Seeding size at different influence factors ρ (the maximum number of propagation hops is $d = 4$) **(a)** Physics. **(b)** Facebook. **(c)** Orkut

get below the number of isolated cliques in the networks. To eliminate the effect of isolated cliques, a possible approach is to restrict the problem to the largest component in the network.

6 Conclusion

This chapter studies approximability and inapproximability for important variants of the *dominating set* problems in different network topologies. In general networks, the considered domination problems are hard to approximate within a factor $(1/2 - o(1)) \ln n$, and there exist approximation algorithms with factor $\ln n + O(1)$ for the problems. Thus, there is still a $1/2 \ln n$ gap between the lower bound and upper bound for the approximability of those problems. Note that Srinivasan [57] gives an $\ln \frac{n}{\text{OPT}} + O(\ln \ln \frac{|n|}{\text{OPT}})$ approximation factor for DS which provides a better approximation factor when the optimal solution is relatively large to the input size. In addition, the sizes of PIDS and T-PIDS are at least $\sqrt{|V| + |E|}$; thus, it is conjectured that PIDS and T-PIDS problems can be approximated within a factor $1/2 \ln n + O(1)$.

Recommended Reading

1. T.W. Haynes, S.T. Hedetniemi, P.J. Slater, *Fundamentals of Domination in Graphs*. Monographs and Textbooks in Pure and Applied Mathematics (Marcel Dekker, New York, 1998)
2. S. Ivan, Dominating set based bluetooth scatternet formation with localized maintenance. *Parallel and Distributed Processing Symposium, International* (IEEE Computer Society, Washington, DC, 2002)
3. T. Moscibroda, R. Wattenhofer, Maximizing the lifetime of dominating sets, in *IEEE International Parallel and Distributed Processing Symposium '05*, IPDPS '05, Washington, DC, USA (IEEE Computer Society, Washington, DC, 2005)
4. J. Blum, M. Ding, A. Thaeler, X. Cheng, Connected dominating set in sensor networks and manets, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos (Springer, New York, 2005), pp. 329–369
5. M.T. Thai, F. Wang, D. Liu, S. Zhu, D.-Z. Du, Connected dominating sets in wireless networks with different transmission ranges. *IEEE Trans. Mob. Comput.* **6**(7), 721–730 (2007)
6. M.T. Thai, R. Tiwari, D.-Z. Du, On construction of virtual backbone in wireless ad hoc networks with unidirectional links. *IEEE Trans. Mob. Comput.* **7**(9), 1098–1109 (2008)
7. R. Tiwari, T.N. Dinh, M.T. Thai, On approximation algorithms for interference-aware broadcast scheduling in 2d and 3d wireless sensor networks, in *International Conference on Wireless Algorithms, Systems, and Applications '09*, WASA '09 (Springer, Berlin/Heidelberg, 2009)
8. R. Tiwari, T.N. Dinh, M.T. Thai, On centralized and localized approximation algorithms for interference-aware broadcast scheduling. *IEEE Trans. Mob. Comput.* **12**(2), 233–247 (2013)
9. F. Wang, M.T. Thai, D.-Z. Du, On the construction of 2-connected virtual backbone in wireless networks. *IEEE Trans. Wirel. Commun.* **8**(3), 1230–1237 (2009)
10. F. Wang, E. Camacho, K. Xu, Positive influence dominating set in online social networks, in *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications*, COCOA '09 (Springer, Berlin, Heidelberg, 2009), pp. 313–321
11. X. Zhu, J. Yu, W. Lee, D. Kim, S. Shan, D.-Z. Du, New dominating sets in social networks. *J. Global Optim.* **48**, 633–642 (2010). doi:10.1007/s10898-009-9511-2.
12. F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, S. Shan, On positive influence dominating sets in social networks. *Theor. Comput. Sci.* **412**(3), 265–269 (2011). Combinatorial Optimization and Applications – COCOA 2009
13. U. Feige, A threshold of $\ln n$ for approximating set cover. *J. ACM* **45**(4), 634–652 (1998)
14. P. Slavík, A tight analysis of the greedy algorithm for set cover, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96 (ACM, New York, 1996), pp. 435–441
15. N. Alon, D. Moshkovitz, S. Safra, Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms* **2**(2), 153–177 (2006)
16. Z. Feng, Z. Zhao, W. Weili, Latency-bounded minimum influential node selection in social networks, in *Wireless Algorithms, Systems, and Applications*, ed. by B. Liu, A. Bestavros, D.-Z. Du, J. Wang. Lecture Notes in Computer Science (Springer, Berlin/New York, 2009), pp. 519–526
17. S. Guha, S. Khuller, Approximation algorithms for connected dominating sets. *Algorithmica* **20**, 374–387 (1998)
18. V.V. Vazirani, *Approximation algorithms* (Springer, Berlin/New York, 2001)
19. C. Liao, G.J. Chang, Algorithmic aspect of k-tuple domination in graphs. *Taiwan. J. Math.* **6**, 415–420 (2003)
20. R. Klasing, C. Laforest, Hardness results and approximation algorithms of k-tuple domination in graphs. *Inf. Process. Lett.* **89**, 75–83 (2004)
21. B. Wang, K.-N. Xiang, On k-tuple domination of random graphs. *Appl. Math. Lett.* **22**(10), 1513–1517 (2009)

22. W. Shang, P. Wan, F. Yao, X. Hu, Algorithms for minimum m-connected k-tuple dominating set problem. *Theor. Comput. Sci.* **381**(1–3), 241–247 (2007)
23. K.G. Hill, J.D. Hawkins, R.F. Catalano, R.D. Abbott, J. Guo, Family influences on the risk of daily smoking initiation. *J. Adolesc. Health* **37**(3), 202–210 (2005)
24. J.B. Standridge, R.G. Zylstra, S.M. Adams, Alcohol consumption: an overview of benefits and risks. *South. Med. J.* **97**(7), 664–672 (2004)
25. P. Domingos, M. Richardson, Mining the network value of customers, in *KDD '01: Proceedings of The 7th ACM SIGKDD International Conference on Knowledge Discovery and Data mining* (ACM, New York, 2001), pp. 57–66
26. D. Kempe, J. Kleinberg, É. Tardos, Maximizing the spread of influence through a social network, in *KDD'03: Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, New York, 2003), pp. 137–146
27. D. Kempe, J. Kleinberg, E. Tardos, Influential nodes in a diffusion model for social networks, in *International Colloquium on Automata, Languages and Programming '05* (Springer, Berlin/Heidelberg, 2005), pp. 1127–1138
28. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in *ACM SIGKDD Conference on Knowledge Discovery and Data Mining '07* (ACM, New York, 2007), pp. 420–429
29. T.N. Dinh, N.T. Dung, M.T. Thai, Cheap, easy, and massively effective viral marketing in social networks: Truth or fiction? in *Proceedings of the 23rd ACM conference on Hypertext and Social Media, HT '12* (ACM, Milwaukee, 2012)
30. D. Peleg, Local majority voting, small coalitions and controlling monopolies in graphs: A review, in *SIROCCO'96: Colloquium on Structural Information and Communication Complexity*, Weizmann Science Press of Israe, Jerusalem, Israel, (1996), pp. 152–169
31. W. Zhang, Z. Zhang, W. Wang, F. Zou, W. Lee, Polynomial time approximation scheme for t-latency bounded information propagation problem in wireless networks. *J. Comb. Optim.*, 1–11 (2010). doi:10.1007/s10878-010-9359-x.
32. S. Khot, O. Regev, Vertex cover might be hard to approximate to within 2-[epsilon]. *J. Comput. Syst. Sci.*, **74**(3), 335–349 (2008)
33. M. Chlebík, J. Chlebíková, Approximation hardness of dominating set problems in bounded degree graphs. *Inf. Comput.* **206**(11), 1264–1275 (2008)
34. L. Trevisan, Non-approximability results for optimization problems on bounded degree instances, in *ACM Symposium on Theory of Computing '01* (ACM, New York, 2001), pp. 453–461
35. N. Kahale, Eigenvalues and expansion of regular graphs. *J. ACM*, **42**, 1091–1106 (1995)
36. S. Rajagopalan, V.V. Vazirani, Primal-dual rnc approximation algorithms for (multi)-set (multi)-cover and covering integer programs, in *Annual IEEE Symposium on Foundations of Computer Science '93* (IEEE Computer Society, Washington, DC, 1993), pp. 322–331
37. E.B. van Peter, R. Kaas, E. Zijlstra, Design and implementation of an efficient priority queue. *Math. Syst. Theory* **10**, 99–127 (1977)
38. G. Robins, A. Zelikovsky, Tighter bounds for graph steiner tree approximation. *SIAM J. Discret. Math.* **19**, 122–134 (2005)
39. L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, K. Ko, A greedy approximation for minimum connected dominating sets. *Theor. Comput. Sci.* **329**, 325–330 (2004)
40. D.-Z. Du, R.L. Graham, P.M. Pardalos, P.-J. Wan, W. Wu, W. Zhao, Analysis of greedy approximations with nonsubmodular potential functions, in *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms, SODA '08* (Society for Industrial and Applied Mathematics, Philadelphia, 2008), pp. 167–175
41. B.S. Baker, Approximation algorithms for np-complete problems on planar graphs. *J. ACM* **41**, 153–180 (1994)
42. S. Khanna, R. Motwani, Towards a syntactic characterization of ptas, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, STOC '96* (ACM, New York, 1996), pp. 329–337

43. D. Eppstein, Diameter and treewidth in minor-closed graph families. *Algorithmica* **27**(3), 275–291 (2000)
44. M. Grohe, Local tree-width, excluded minors, and approximation algorithms. *Combinatorica* **23**, 613–632 (2003)
45. E.D. Demaine, M. Hajiaghayi, Bidimensionality: new connections between fpt algorithms and ptas, in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '05 (Society for Industrial and Applied Mathematics, Philadelphia, 2005), pp. 590–601
46. M. Gibson, I. Pirwani, Algorithms for dominating set in disk graphs: Breaking the $\log n$ barrier, in *European Symposia on Algorithms 2010*, ed. by M. de Berg, U. Meyer. Volume 6346 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2010), pp. 243–254
47. A. Czygrinow, M. Hanckowiak, W. Wawrzyniak, Fast distributed approximations in planar graphs, in *Distributed Computing*, ed. by G. Taubenfeld. Volume 5218 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2008), pp. 78–92
48. D. Dai, C. Yu, A $5+[\epsilon]$ -approximation algorithm for minimum weighted dominating set in unit disk graph. *Theor. Comput. Sci.* **410**(8–10), 756–765 (2009)
49. M. Girvan, M.E. Newman, Community structure in social and biological networks. *PNAS* **99**(12), 7821–7826 (2002)
50. C. Gkantsidis, M. Mihail, A. Saberi, Conductance and congestion in power law graphs, in *SIGMETRICS '03: Proceedings of the International Conference on Measurements and Modeling of Computer Systems* (ACM, New York, 2003), pp. 148–159
51. A. Ferrante, G. Pandurangan, K. Park, On the hardness of optimization in power-law graphs. *Theor. Comput. Sci.* **393**(1–3), 220–230 (2008)
52. W. Aiello, F. Chung, L. Lu, A random graph model for power law graphs. *Exp. Math.* **10**, 53–66 (2000)
53. Y. Shen, D.T. Nguyen, Y. Xuan, M.T. Thai, New techniques for approximating optimal substructure problems in power-law graphs. *Theor. Comput. Sci.* (2011)
54. A. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, T. Vicsek, Evolution of the social network of scientific collaborations. *Phys. A* **311**(3–4), 590–614 (2002)
55. B. Viswanath, A. Mislove, M. Cha, K.P. Gummadi, On the evolution of user interaction in facebook, in *WOSN'09* (ACM, New York, 2009)
56. A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in *IMC'07*, San Diego, CA (ACM, New York, 2007)
57. A. Srinivasan, Improved approximations of packing and covering problems, in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, STOC '95 (ACM, New York, 1995), pp. 268–276

Advanced Techniques for Dynamic Programming

Wolfgang Bein

Contents

1	Introduction	42
2	A Few Standard Introductory Examples.....	44
2.1	A Brief Introduction to Dynamic Programming: Fibonacci, Pascal, and Making Change	44
2.2	Chained Matrix Multiplication Problem.....	46
2.3	Shortest Paths.....	47
2.4	The Knapsack Problem.....	50
2.5	Binary Search Trees.....	52
2.6	Pyramidal Tours for the Traveling Salesman Problem.....	54
3	Open-ended Dynamic Programming: Work Functions.....	54
4	Intricate Dynamic Programming: Block Deletion in Quadratic Time.....	58
4.1	Preliminaries.....	58
4.2	A Dynamic Program for Complete Block Deletion.....	59
4.3	Computing Block Deletion.....	62
5	Total Monotonicity and Batch Scheduling.....	63
5.1	The Problem $1 s - \text{batch} \sum w_i C_i$	63
5.2	List Batching.....	64
5.3	The Monge Property and Total Monotonicity.....	66
6	The SMAWK and LARSCH Algorithm.....	68
6.1	The Matrix Searching Problem.....	68
6.2	The Online Matrix Searching Problem.....	72
6.3	Algorithm LARSCH.....	73
6.4	Standard Type Process: P_t for t Even (INTERPOLATE).....	75
6.5	Standard Type Process: P_t for t Odd (REDUCE).....	75
7	The Quadrangle Inequality and Binary Search Trees.....	76
7.1	Background.....	76
7.2	Decomposition Techniques.....	78
7.3	Online Decomposition.....	80

W. Bein

Department of Computer Science, University of Nevada, Las Vegas, Las Vegas, NV, USA
e-mail: beinw@unlv.nevada.edu

8 Conclusion.....	85
Further Reading.....	86
Cross-References.....	88
Recommended Reading.....	88

Abstract

This is an overview over dynamic programming with an emphasis on advanced methods. Problems discussed include path problems, construction of search trees, scheduling problems, applications of dynamic programming for sorting problems, server problems, as well as others. This chapter contains an extensive discussion of dynamic programming speedup. There exist several general techniques in the literature for speeding up naive implementations of dynamic programming. Two of the best known are the Knuth-Yao quadrangle inequality speedup and the SMAWK/LARSCH algorithm for finding the row minima of totally monotone matrices. The chapter includes “ready to implement” descriptions of the SMAWK and LARSCH algorithms. Another focus is on dynamic programming, online algorithms, and work functions.

1 Introduction

Dynamic programming, formally introduced by Richard Bellman (August 26, 1920–March 19, 1984) at the Rand Corporation in Santa Monica in the 1940s, is a versatile method to construct efficient algorithms for a broad range of problems. As with many tools which evolved over time, the original paper sounds antiquated. In his monograph “Dynamic Programming,” Bellman [25] describes the principle of optimality, which is central to dynamic programming: “An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.” Not exactly what is found in textbooks today. But then in those days the focus was more on stochastic processes and not so much on combinatorial optimization. The term “programming” is outdated as well, as it does not refer to programming in a modern programming language. Instead, programming means planning by filling in tables. The term “linear programming” derives in the same way. Over the decades, dynamic programming has evolved and is now one of the “killer” techniques in algorithmic design. And, of course, the laptop computer on which this chapter was written is more powerful than anyone in Bellman’s days could have imagined.

Today, the emphasis is on how to organize dynamic programming in a way that makes it possible to solve massive problems (which – again – would have never been considered in Bellman’s days) in reasonable time. The focus is on dynamic programming speedup. Such speedup comes from carefully observing what values are essential and need to be computed and which might be unnecessary. Keeping

proper look-up tables on the side can accomplish this sometimes. But there are many situations where there are inherent monotonicity properties which can be exploited to only calculate a fraction of what is necessary in a simple-minded approach.

The goal of this chapter is to briefly introduce dynamic programming, show some of the diversity of problems that can be tackled efficiently with dynamic programming, and – centrally – focus on the issue of dynamic programming speedup. Clearly, the chapter does not cover all of dynamic programming, but there is a section on recommended reading, [Sect. 8](#), which covers some ground.

The chapter is organized as follows: [Sect. 2](#) starts with a simple introduction to dynamic programming using a few standard examples, described more tersely than in a textbook and augmented with a general few themes. A reader altogether unfamiliar with dynamic programming might utilize some of the resources given in [Sect. 8](#).

[Section 3](#) contains “open-ended dynamic programming,” where a process is updated continuously. This is closely related to the theory of online optimization. In online computation, an algorithm must make decisions without knowledge of future inputs. Online problems occur naturally across the entire field of discrete optimization, with applications in areas such as robotics, resource allocation in operating systems, and network routing. Dynamic programming plays an important role in this kind of optimization. Notably, work functions replace tables or matrices used in offline optimization.

[Section 4](#) contains an example from sorting. The intent of this section is twofold: First, lesser-known dynamic programming techniques for sorting are highlighted. Second, here is an example where a straightforward simple-minded implementation might give an algorithm of cubic complexity or worse, and a more intricate setup solves the problem in quadratic time, thus achieving substantial dynamic programming speedup.

[Section 5](#) gives an example from scheduling (a batching problem) to illustrate important properties for dynamic programming speedup: the Monge property and total monotonicity. Techniques exploiting these techniques are now standard, and the reader might consult [Sect. 8](#) to see how prolific such techniques are.

Speedup is based on two important and intricate algorithms: SMAWK and LARSCH. Use of these is essential for many fast dynamic programming algorithms. However, these algorithms are currently only accessible through the original publications. [Section 6](#) contains “ready to implement” descriptions of the SMAWK and LARSCH algorithms.

Another type of speedup is based in the Knuth-Yao quadrangle inequality; this dynamic programming speedup works for a large class of problems. Even though both the SMAWK algorithm and the Knuth-Yao (KY) speedup use an implicit quadrangle inequality in their associated matrices, on second glance, they seem quite different from each other. [Section 7](#) discusses the relation between these kinds of speedup in greater detail.

Finally, [Sect. 8](#) is a cross-reference list with other chapters, and [Sect. 8](#) gives concluding remarks.

2 A Few Standard Introductory Examples

As mentioned earlier, this section has a few introductory examples.

2.1 A Brief Introduction to Dynamic Programming: Fibonacci, Pascal, and Making Change

Many problems can be solved recursively by dividing an instance into subinstances. But a direct implementation of a recursion is often inefficient because subproblems overlap and are recomputed numerous times. The calculation of the Fibonacci numbers provides a simple example:

$$f_n = \begin{cases} f_{n-1} + f_{n-2} & \text{if } n > 1 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0. \end{cases} \quad (1)$$

Recursively, the Fibonacci numbers can be calculated in the following way:

```
function fib(n)
    if n = 0 or n = 1 return n
    else return fib(n - 1) + fib(n - 2).
```

This is extremely inefficient; many values will be recalculated; see Fig. 1. Instead, one could use an array and would simply fill in values from “left to right,” starting with F_0 and F_1 and the using Eq. 1 to continue. This way every value is only calculated once.

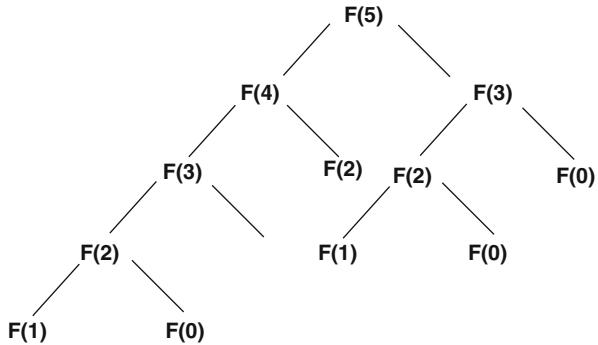
The Pascal triangle for calculating binomial coefficients provides a good example for the use of *tables* in dynamic programming. Recall the definition of the binomial coefficient:

$$\binom{n}{k} = \begin{cases} \binom{n-1}{k-1} + \binom{n-1}{k} & \text{if } 0 < k < n \\ 1 & \text{if } k = 0 \text{ or } k = n \\ 0 & \text{else.} \end{cases} \quad (2)$$

Clearly, the coefficients can be calculated using the following recursive program:

```
function c(n, k)
    if k = 0 or k = n return 1
    else return c(n - 1, k - 1) + c(n - 1, k).
```

Fig. 1 Recursive calculation of the fifth Fibonacci number



But again this is inefficient, as terms are recalculated over and over. Indeed, since the solution is ultimately made up of terms of value 1, the run time of this algorithm is $\Omega(\binom{n}{k})$. This problem consists of *overlapping subproblems*, which often makes it inefficient to use recursions directly. Instead, one can calculate the coefficients *bottom up* using a table to store intermediate results. In the case of the binomial coefficient, this is the Pascal triangle Fig. 2. With it the run time is $\Theta(nk)$ with space requirement $\Theta(k)$ (as only two rows at a time need to be stored).

Most often dynamic programming is used for optimization problems. As an example consider the problem of making change with the minimum number of coins. Given are n denominations of value $\{d_1, \dots, d_n\}$, an unlimited supply of these coins, and the problem is to make change for amount A . For example, if the denominations are 1, 5, 10, 12, and $A = 21$, then the minimum number of coins is 3. Note that the greedy algorithm uses 6 coins. What is important here is that the principle of optimality holds: If optimal change for amount C involves making change into amounts A and B , $C = A + B$, then change for A and B is also optimal. More generally, the principle states that in an optimal sequence of decisions or choices, each subsequence must also be optimal.

Let

$C[i, j] =$ minimum number of coins required to make change for amount j
using only coins of denomination $\{1, \dots, i\}$,

where $i = 1, \dots, n$ and $j = 1, \dots, A$. The principle of optimality implies the following recurrence:

$$C[i, j] = \min\{C[i - 1, j], 1 + C[i, j - d_i]\}, \quad (3)$$

where out-of-bounds values are set to ∞ .

Just as before with Fibonacci and Pascal, one uses a table to calculate values “bottom-up.” The table is initialized by setting all $C[i, 0]$ to 0. Then the table is filled row by row using Eq. 3. An example is in Table 1.

Fig. 2 The Pascal triangle

	0	1	2	3	...	$k-1$	k
0	0						
1		1	1				
2			1	2	1		
3				1	3	3	1
\vdots							
$n-1$						$c(n-1, k-1)$	$c(n-1, k)$
n						$\searrow + \downarrow$	$c(n, k)$

Table 1 An instance of the change-making problem: There are four types of coins with denominations $d_1 = 1, d_2 = 5, d_3 = 10$, and $d_4 = 12$. The amount is $A = 21$. The last entry in the table gives the minimum number of coins, which is 3. Note that the greedy algorithm uses six coins

A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
d_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
d_2	0	1	2	3	4	1	2	3	4	5	2	3	4	5	6	3	4	5	6	7	4	5
d_3	0	1	2	3	4	1	2	3	4	5	1	2	3	4	5	2	3	4	5	6	2	3
d_4	0	1	2	3	4	1	2	3	4	5	1	2	1	2	3	2	3	2	3	4	2	3

2.2 Chained Matrix Multiplication Problem

Given is a sequence of matrices M_1, \dots, M_n to be multiplied. The dimensions are $d_0 \times d_1, \dots, d_{n-1} \times d_n$, denoted by (d_0, d_1, \dots, d_n) , for short. The question is what parenthesization gives the minimum number of multiplications. For example, given three matrices, A, B, C , with dimensions $(12, 5, 90, 2)$, the calculation $A(BC)$ requires $5 \times 90 \times 2$ multiplications to produce BC and then $12 \times 5 \times 2$ to multiply the result by A , for a total of 1,020. The order $(AB)C$ is much worse: 7500 multiplications. For general n , exhaustive search is prohibitive for the following reason: Let

$$T(n) = \text{number of ways to parenthesize a product of } n \text{ matrices},$$

then

$$T(n) = \sum_{i=1}^{n-1} T(i)T(n-i) \quad (4)$$

with $T(2) = T(1) = 1$. The solution to recurrence 4 is

$$T(n) = \frac{1}{n} \binom{2n-2}{n-1}, \quad (5)$$

which is $\Omega(4^n/n^2)$.

Clearly the principle of optimality applies. If the parenthesization is optimal for $M = M_1, \dots, M_n$ and there are two parts $A = M_1, \dots, M_k$ and $B = M_{k+1}, \dots, M_n$ such that $M = AB$, then the parenthesizations for A and B are optimal. Let

$$M[i, j] = \text{optimal number of multiplications to compute } M_i \cdots M_j.$$

Thus, the recurrence is

$$M[i, j] = \min_{i \leq k < j} \{M[i, k] + M[k + 1, j] + d_{i-1}d_kd_j\} \text{ for } 1 \leq i < j \leq n, \quad (6)$$

with $M[i, i] = 0$.

The dynamic program fills a table with entries for $i \leq j$, starting with the main diagonal, which is set to $M[i, i] = 0$. The computation then progresses to $M[i, i+1]$ and so forth until the element in the north-east corner $M[1, n]$ is reached. In order to be able to recover the solution, the index k which gives the minimum in 6 is also stored. When the algorithm reaches $(1, n)$, this index gives the index of the first cut. One then proceeds recursively on both sides to construct the actual parenthesization. [Figure 3](#) gives an example.

2.3 Shortest Paths

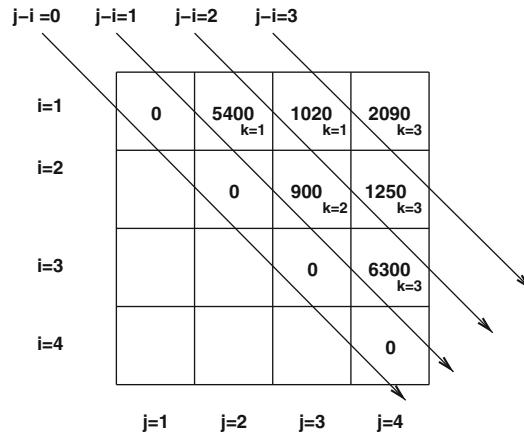
Consider a directed graph on nodes $V = \{1, \dots, n\}$ with distance matrix $D[i, j] \geq 0$, where $D[i, j] = \infty$ if there is no edge between node i and node j . The object is to calculate the shortest path between each pair of nodes. To use dynamic programming, one defines

$$D_k[i, j] = \text{length of a shortest path from } i \text{ to } j \text{ which only uses nodes } \{1, \dots, k\}.$$

Clearly $D_0 = D$. Starting with D_0 one calculates D_1, D_2, \dots, D_n using the following recursion:

$$D_k[i, j] = \min\{D_{k-1}[i, j], D_{k-1}[i, k] + D_{k-1}[k, j]\}. \quad (7)$$

The algorithm, which computes this series of tables, is called the Floyd algorithm. The reader is invited to follow the calculations in [Table 2](#), which gives an example for the graph in [Fig. 4](#). The principle of optimality is at work here: If a path from node i to node j is optimal and passes through node k , then the path from i to k , as well as the path from k to j , is optimal. This is not true for the longest simple path problem. (A simple path is a path without repeated nodes.) In graph of [Fig. 4](#), the longest simple path from node 2 to node 4 is 2, 1, 3, 4, but the path 2, 1 is not a longest simple path from 2 to 1. Indeed the problem “longest simple path” is \mathcal{NP} -hard. This does not contradict the fact that the Floyd algorithm works for



$$M[1, 1] = M[2, 2] = M[3, 3] = 0$$

$$M[1, 2] = \mathbf{M[1, 1]} + \mathbf{M[2, 2]} + 12 \times 5 \times 90 = 5400, \text{ store } k = 1$$

$$M[2, 3] = \mathbf{M[2, 2]} + \mathbf{M[3, 3]} + 5 \times 90 \times 2 = 900, \text{ store } k = 2$$

$$M[3, 4] = M[3, 3] + M[4, 4] + 90 \times 2 \times 35 = 6300, \text{ store } k = 3$$

$$\begin{aligned} M[1, 3] &= \min\{\mathbf{M[1, 1]} + \mathbf{M[2, 3]} + 12 \times 5 \times 2, M[1, 2] + M[3, 3] + 12 \times 90 \times 2\} \\ &= 1020, \text{ store } k = 1 \end{aligned}$$

$$\begin{aligned} M[2, 4] &= \min\{M[2, 2] + M[3, 4] + 5 \times 90 \times 35, \mathbf{M[2, 3]} + \mathbf{M[4, 4]} + 5 \times 2 \times 35\} \\ &= 1250, \text{ store } k = 3 \end{aligned}$$

$$\begin{aligned} M[1, 4] &= \min\{M[1, 1] + M[2, 4] + 12 \times 5 \times 35, M[1, 2] + M[3, 4] + 12 \times 90 \times 35, \\ &\quad \mathbf{M[1, 3]} + \mathbf{M[4, 4]} + 12 \times 2 \times 35\} \\ &= 2090, \text{ store } k = 3 \end{aligned}$$

Optimal Parenthesization: $(M_1(M_2M_3)M_4)$

Fig. 3 An example for the dynamic program for the problem “chained matrix multiplication.” In the example, $n = 4$ and the dimensions are $(12, 5, 90, 2, 35)$. The calculation proceeds by first filling the table for indices with $j - i = 0$ (i.e., initialization of $M[i, i]$) and then continues along the diagonals $j - i = 1, 2, 3$. Calculations which give the minima for each cell in Eq. 6 are shown in bold type, and the corresponding index k is stored. Once the value of $M[1, 4]$ is known, then the solution can be recovered using these indices: $(M_1M_2M_3)M_4$ can be concluded from the index $k = 3$ in cell $(1, 4)$. Next, look up cell $(1, 3)$ to find the parenthesization $M_1(M_2M_3)$.

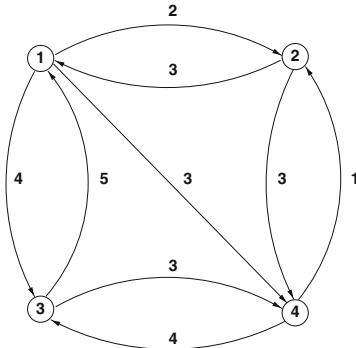
negative distances if there are no negative cycles. (Longest simple path cannot be reduced to shortest path with negative distances because of the cycle restriction.) The Floyd algorithm can be used to detect negative cycles by checking for negative elements in the main diagonal of D_n .

The time complexity of Floyd algorithm is $O(n^3)$. The calculation can be performed by keeping only D_{k-1} and D_k at each iteration, and thus the space

Table 2 The Floyd algorithm for the example of Fig. 4. One proceeds from matrix D_{k-1} to matrix D_k by applying the rule $D_k[i, j] = \min\{D_{k-1}[i, j], D_{k-1}[i, k] + D_{k-1}[k, j]\}$ for all i and j . In the lower right corner of the table, the matrix of pointers P is shown. For example, the length of a shortest path from node 3 to node 2 is 4 since $D_4[3, 2] = 4$. To construct the path look up $P[3, 2] = 4$ to find that the path goes through node 4. Recursively, look up $P[3, 4] = P[4, 2] = 0$. Thus the shortest path is 3, 4, 2

D_0	0	2	4	3	D_1	0	2	4	3
	3	0	∞	3		3	0	7	3
	5	∞	0	3		5	7	0	3
	∞	1	4	0		∞	1	4	0
D_2	0	2	4	3	D_3	0	2	4	3
	3	0	7	3		3	0	7	3
	5	7	0	3		5	7	0	3
	4	1	4	0		4	1	4	0
D_4	0	2	4	3	P	0	0	0	0
	3	0	7	3		0	0	1	0
	5	4	0	3		0	4	0	0
	4	1	4	0		2	0	0	0

Fig. 4 A directed graph with distances



requirement is $O(n^2)$. In order to retrieve the actual path (and not only its length), a matrix P of pointers is used, where $P[i, j]$ contains the number of the last iteration k that causes a change in $D_k[i, j]$ ($P[i, j]$ is initialized to 0). To construct the path between node i and j , look up $P[i, j]$ at the end. If $P[i, j] = 0$, then there was never a change for any D_k and the shortest path is the edge (i, j) , else the shortest path goes through k . Recursively examine $P[i, k]$ and $P[k, j]$.

The resulting algorithm is named after Floyd (It appeared as a one-page note in the Communications of the ACM – together with other algorithms of the time. Therefore, the antiquated title is “Algorithm 97: Shortest Path,” Floyd [55]).

Another algorithm of the time is the matrix algorithm, originally given in the context of the transitive closures by Warshall. The paper is a short two-pager; see Warshall [92].

Define

$D^{(k)}[i, j] = \text{length of a shortest path from } i \text{ to } j \text{ containing at most } k \text{ edges}$

and set

$$D^{(0)} == \begin{cases} 0 & \text{if } i = j \\ \infty & \text{else.} \end{cases}$$

Clearly by the principle of optimality,

$$D^{(k)}[i, j] = \min\{D^{(k-1)}[i, j], \min_{1 \leq l \leq n, l \neq j} \{D^{(k-1)}[i, l] + D[l, j]\}\} \quad (8)$$

$$= \min_{1 \leq l \leq n} \{D^{(k-1)}[i, l] + D[l, j]\}. \quad (9)$$

The previous equation defines a matrix multiplication where the usual multiplication means “+” and the operation “+” means “min.” [Table 3](#) gives an example. In other words,

$$D^{(k)} = D^k.$$

The solution to the shortest path problem is therefore obtained by calculating D^k with the operators properly replaced. The run time of this algorithm is at first glance $O(n^4)$, but it can be improved by “repeated squaring”: Calculate

- $D^2 = D \cdot D$
- then $D^4 = D^2 \cdot D^2$
- then $D^8 = D^4 \cdot D^4$, and so forth

If $n - 1$ is not a power of 2, then D^{n-1} can be obtained by going up to the highest power of 2 smaller than $(n - 1)$ and multiplying the powers of the binary representation of $(n - 1)$. As a result the number of matrix multiplications is only logarithmic, giving a run time of $O(n^3 \log n)$.

2.4 The Knapsack Problem

This is a classical problem in combinatorial optimization: Given are n items $\{1, \dots, n\}$ with weights $w_i > 0$ and profits $p_i > 0$, and there is a knapsack of weight capacity $W > 0$. One is to fill the knapsack in such a way that the profit of the items chosen is maximized while obeying that the total weight be less than W . Let

$P[i, j] = \text{maximum profit, which can be obtained if the}$

$\text{weight limit is } j \text{ and only items from } \{1, \dots, i\} \text{ may be included,}$

Table 3 Example of the progression of the matrix algorithm for the all shortest path problem given in Fig. 4. The operations in the “matrix multiplication” are not the usual “+” and “.” but rather “min” and “+”

$I \cdot D =$	$\begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & \infty & 3 \\ 5 & \infty & 0 & 3 \\ \infty & 1 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & \infty & 3 \\ 5 & \infty & 0 & 3 \\ \infty & 1 & 4 & 0 \end{pmatrix} = D^{(1)}$
$D^{(1)} \cdot D =$	$\begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & \infty & 3 \\ 5 & \infty & 0 & 3 \\ \infty & 1 & 4 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & \infty & 3 \\ 5 & \infty & 0 & 3 \\ \infty & 1 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & 7 & 3 \\ 5 & 4 & 0 & 3 \\ 4 & 1 & 4 & 0 \end{pmatrix} = D^{(2)}$
$D^{(2)} \cdot D =$	$\begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & 7 & 3 \\ 5 & 4 & 0 & 3 \\ 4 & 1 & 4 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & \infty & 3 \\ 5 & \infty & 0 & 3 \\ \infty & 1 & 4 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 4 & 3 \\ 3 & 0 & 7 & 3 \\ 5 & 4 & 0 & 3 \\ 4 & 1 & 4 & 0 \end{pmatrix} = D^{(3)}$

where $1 \leq i \leq n$ and $0 \leq j \leq W$. The following recursion is valid:

$$P[i, j] = \max\{P[i - 1, j], P[i - 1, j - w_i + p_i]\}, \quad (10)$$

where $P[0, j] = 0$ and $P[i, j] = -\infty$ when $j < 0$. Equation 10 says that for a new element i one does not include it (left of max) or one does include it (right of max). If one includes the element, then by the principle of optimality the problem with capacity reduced by the weight of element i on the earlier elements $\{1, \dots, i - 1\}$ must be optimal.

The table $P[i, j]$ can be filled in either row by row or column by column fashion. The run time of this algorithm is $\Theta(nW)$. Note that this is not a polynomial algorithm for the knapsack problem. The input size is $O(n \log \max_{i=1,\dots,n} w_i + \log W)$, which means that $\Theta(nW)$ is exponential in the input size. This is, of course, to be expected as the knapsack problem is \mathcal{NP} -hard. The algorithm’s complexity is called *pseudo-polynomial*. Many dynamic programs give pseudo-polynomial algorithms.

Pseudo-polynomial, Strongly Polynomial. Let s be the input to some decision problem Π . Let $|s|_{\log}$ be the length of the input – that is, the length of the binary encoding – and let $|s|_{\max}$ be the magnitude of the largest number in s . A problem Π is pseudo-polynomially solvable if there is an algorithm for Π with run time bounded by a polynomial function in $|s|_{\max}$ and $|s|_{\log}$. A decision problem is a *number problem* if there exists no polynomial p such that $|s|_{\max}$ is bounded by $p(|s|_{\log})$ for all input s . (For example, the decision version of the knapsack problem is a number problem.) Note that by these definitions it immediately follows that an \mathcal{NP} -complete *non-number* problem (such as HAMILTONIAN CIRCUIT, for

Table 4 An example of the pseudo-polynomial dynamic programming algorithm for the knapsack problem on nine items. The weights are $w[1] = 1, w[2] = 2, w[3] = 3, w[4] = 4, w[5] = 5, w[6] = 6, w[7] = 7, w[8] = 8, w[9] = 9$ and the profits are $p[1] = 1, p[2] = 2, p[3] = 5, p[4] = 10, p[5] = 15, p[6] = 16, p[7] = 21, p[8] = 22, p[9] = 35$. The size of the knapsack is 15. The table shows in position (i, j) the maximum profit, which can be obtained if the weight limit is j and only items from $\{1, \dots, i\}$ may be included. Bold entries indicate that in Eq. 10 the second choice is the maximizer, i.e., the new item is considered for inclusion. From the regular-bold information the actual solution can be reconstructed: The last entry in the table is 50. Because of the bold type face item $i = 9$ is included. Thus one looks $w_9 = 9$ many cells to the left in the previous row. The regular type face of 16 in cell (7, 6) means that item 8 is not included, looking at the cells above neither are items 7 and 6. Item 5 is included, next look-up cell (4, 1) to see that no more items are included. The solution is $\{5, 9\}$ with a profit of 50

W	0	1	2	3	4	5	6	7	8	9	11	12	13	14	15
i=1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
i=2	0	1	2	3	3	3	3	3	3	3	3	3	3	3	3
i=3	0	1	2	5	6	7	8	8	8	8	8	8	8	8	8
i=4	0	1	2	5	10	11	12	15	16	17	18	18	18	18	18
i=5	0	1	2	5	10	15	16	17	20	25	26	27	30	31	32
i=6	0	1	2	5	10	15	16	17	20	25	26	31	32	33	36
i=7	0	1	2	5	10	15	16	21	22	25	26	31	36	37	38
i=8	0	1	2	5	10	15	16	21	22	25	26	31	36	37	38
i=9	0	1	2	5	10	15	16	21	22	35	36	37	40	45	50

example) cannot be solved by a pseudo-polynomial algorithm (unless $\mathcal{P} = \mathcal{NP}$). For polynomial p let Π_p denote the subproblem which is obtained by restricting Π to instances with $|s|_{\max} \leq p(|s|_{\log})$. Problem Π is called *strongly \mathcal{NP} -complete* if Π is in \mathcal{NP} and there exists a polynomial p for which Π_p is \mathcal{NP} -complete. It is easy to see (Table 4):

Theorem 1 A strongly \mathcal{NP} -complete problem cannot have a pseudo-polynomial algorithm unless $\mathcal{P} = \mathcal{NP}$.

2.5 Binary Search Tress

The construction of optimal binary search trees is a classic optimization problem. One is interested in constructing a search tree, in which elements can be looked up as quickly as possible. The first dynamic program for this problem was given by Gilbert and Moore [59] in the 1950s. More formally, given are n search keys with known order $\text{Key}_1 < \text{Key}_2 < \dots < \text{Key}_n$. The input consists of $2n + 1$ probabilities p_1, \dots, p_n and q_0, q_1, \dots, q_n . The value of p_l is the probability that a search is for the value of Key_l ; such a search is called *successful*. The value of q_l is the probability that a search is for a value between Key_l and Key_{l+1} (set $\text{Key}_0 = -\infty$

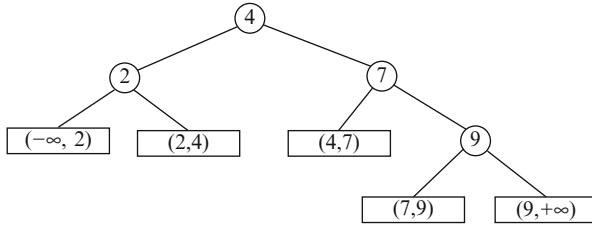


Fig. 5 A binary search tree with successful (round) and unsuccessful (rectangular) nodes

and $\text{Key}_{n+1} = \infty$); such a search is called *unsuccessful*. Note that in the literature the problem is sometimes presented with *weights* instead of *probabilities*, in that case the p_l and q_l are not required to add up to 1.

The binary search tree constructed will have n internal nodes corresponding to the successful searches, and $n+1$ leaves corresponding to the unsuccessful searches. The *depth* of a node is the number of edges from the node to the root. Denote $d(p_l)$ the depth of the internal node corresponding to p_l and $d(q_l)$ the depth of the leaf corresponding to q_l . A successful search requires $1 + d(p_l)$ comparisons, and an unsuccessful search requires $d(q_l)$ comparisons. See Fig. 5. So, the expected number of comparisons is

$$\sum_{1 \leq l \leq n} p_l (1 + d(p_l)) + \sum_{0 \leq l \leq n} q_l d(q_l). \quad (11)$$

The goal is to construct an *optimal binary search tree* that minimizes the expected number of comparisons carried out, which is (11).

Let $B_{i,j}$ be the expected number of comparisons carried out in a optimal subtree containing the keys $\text{Key}_{i+1} < \text{Key}_2 < \dots < \text{Key}_j$. Observing that in a search the probability to search in the region between Key_{i+1} and Key_j is $\sum_{l=i+1}^j p_l + \sum_{l=i}^j q_l$ it is clear that the following recurrence holds:

$$B_{i,j} = \begin{cases} 0, & \text{if } i = j; \\ \sum_{l=i+1}^j p_l + \sum_{l=i}^j q_l + \min_{i < t \leq j} (B_{i,t-1} + B_{t,j}), & \text{if } i < j, \end{cases} \quad (12)$$

where the cost of the optimal binary search tree is $B_{0,n}$. Calculating $B_{i,j}$ requires $O(j - i)$ time, thus calculating all of the $B_{i,j}$ requires $O(n^3)$ time.

2.6 Pyramidal Tours for the Traveling Salesman Problem

In the traveling salesman problem one is given a set of n “cities” $V = \{1, \dots, n\}$ as well as a distance matrix $d[i, j]$. Find a permutation t (“the tour”), such that

$$f(t) = d(t(n), t(1)) + \sum_{i=1}^{n-1} d(t(i), t(i+1))$$

is minimized. This problem is well studied and it is known to be \mathcal{NP} -hard. (A good resource on the Traveling Salesman Problem is the “guided tour book” by Lawler, Lenstra, Rinnoy Kan, and Shmoys [79]). A tour t is said to be *pyramidal* if, t is of the form $1, i_1, i_2, i_3, \dots, n, j_1, \dots, j_{n-r-2}$, where $1 < i_1 < i_2 < i_3 < \dots < n$ and $j_1 > j_2 > j_3 > \dots > j_{n-r-2}$. A pyramidal tour can be found by dynamic programming in $\Theta(n^2)$. To this end, let $H[i, j]$ be the length of a shortest Hamiltonian path from i to j subject to the condition that the path goes from i to 1 in descending order followed by the rest in ascending order from 1 to j . The reader is encouraged to verify that by the principle of optimality the following recursion holds:

$$H[i, j] = \begin{cases} H[i, j-1] + d[j-1, j] & \text{for } i < j-1, \\ \min_{k < i} \{H[i, k] + d[k, j]\} & \text{for } i = j-1, \\ H[i-1, j] + d[i, i-1] & \text{for } i < j+1, \\ \min_{k < j} \{H[k, j] + d[j, k]\} & \text{for } i = j+1. \end{cases} \quad (13)$$

Then the cost of shortest pyramidal tour is

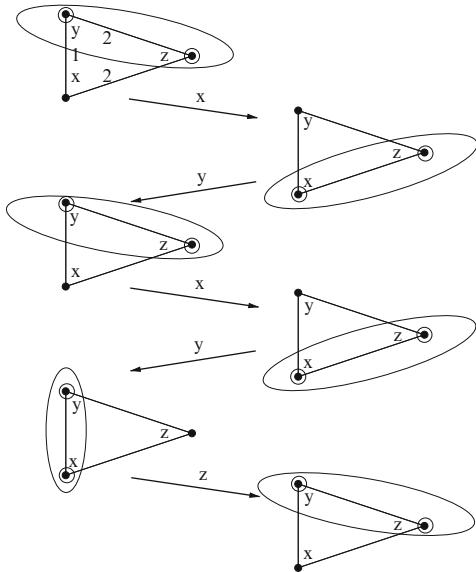
$$\min\{H[n, n-1] + d[n-1, n], H[n-1, n] + d[n, n-1]\}.$$

A matrix d is *Monge* if for all $i < i'$ and $j < j'$, $d[i, j] + d[i', j'] \leq d[i', j] + d[i, j']$. If the matrix d is a Monge matrix then it is easy to see that there exists an optimal tour, which is pyramidal.

3 Open-ended Dynamic Programming: Work Functions

Dynamic programs are useful in decision making for problems where new requests are constantly added, and updates need to be performed. The *k-server problem*, originally given by Manasse, McGeoch, and Sleator [82], is defined as follows: one is given $k \geq 2$ mobile servers which reside in a metric space M . A sequence of

Fig. 6 The two points x and y are at distance 1 and a point z is at distance 2 from x and y . The request sequence is $xyxyz$. The solution, i.e., the positions of the servers for each request, are circled. The solution has a cost of 7, which is not optimal since it is better to move the server at point z for the first and last request



requests is issued, where each request is specified by a point $r \in M$. To “satisfy” this request, one of the servers must be moved to r , at a cost equal to the distance from its current location to r . (If a request is to a point that already has a server the cost is zero.) The goal is to minimize the total service cost. An algorithm \mathcal{A} for the k -server problem computes a solution which determines which server is moved at each step. [Figure 6](#) gives a very simple example for the 2-server problem, i.e., the server problem for $k = 2$ in a metric space with only three points. Interestingly, a very similar metric space is powerful enough to model the noted “ski rental problem”, see Karlin, Manasse, Rudolph, and Sleator [68] for details.

This problem can be solved by dynamic programming calculating *work functions*, when the length of the request sequence and all the requests are known in advance. This standard version of the problem is also called the offline version of the k -server problem. Credit for introducing work functions goes to Larmore and Chrobak [42].

Work functions provide information about the optimal cost of serving the past request sequence. For a request sequence ϱ , by $\omega_\varrho(X)$ one denotes the minimum cost of serving ϱ and ending in configuration X – an unordered k -tuples of points. The function ω_ϱ is called the work function after request sequence ϱ . The notation ω is used to denote any work function ω_ϱ , for some request sequence ϱ . Immediately from the definition of work functions it can be concluded that the optimal cost to service ϱ is $opt(\varrho) = \min_X \omega_\varrho(X)$.

For given ϱ , the work function ω_ϱ can be computed using dynamic programming. Initially, $\omega_\epsilon(X) = S^0 X$, for each configuration X (ϵ is the empty request sequence). For a non-empty request sequence ϱ , if r is the last request in ϱ , write $\varrho = \sigma r$.

Then ω_ϱ can be computed recursively as $\omega_\varrho = \omega_\sigma \wedge r$, where “ \wedge ” is the *update operator* defined as follows:

$$\omega \wedge r(X) = \min_{Y \ni r} \{\omega(Y) + \text{dist}(Y, X)\} \quad (14)$$

Here $\text{dist}(Y, X)$ denotes the minimum-matching distance between X and Y . Note that $|\omega(X) - \omega(Y)| \leq XY$ for any work function ω and any configurations X and Y . This inequality is called the *Lipschitz property*. A set of configurations $S = \{X_1, X_2, \dots\}$ is said to support a work function ω if, for any configuration Y , there exists some $X \in S$ such that $\omega(Y) = \omega(X) + \text{dist}(X, Y)$. If ω is supported by a finite set (which it usually is), then there is a unique minimal set S which supports ω , which is called the work function support of ω . Note the following: If $r \in X$, then $\omega \wedge r(X) = \omega(X)$. If $r \notin X$, let Y be the configuration that contains r and minimizes $\omega(Y) + YX$, and let x be the point in X that is matched to r in the minimum matching between X and Y . Then $\omega(Y) + YX = \omega(Y) + rx + Y(X - x + r) \geq \omega(X - x + r) + rx$. Thus the update formula Eq. 14 can be rewritten as $\omega \wedge r(X) = \min_{x \in X} \{\omega(X - x + r) + rx\}$. Also note that in calculating the functions ω_ϱ one need only keep track of the value of the function at their support. This is important as the number of configurations in the domain of ω_ϱ grows as requests grows. Figure 7 shows how to calculate an optimal solution, the progression of ω_ϱ and support for the example in Fig. 6.

In practice, requests might be given one at a time and the algorithm then has to make a decision about which server to move before future requests are known. An algorithm \mathcal{A} is said to be *online* if its decisions are made without the knowledge of future requests. It is unlikely that such an algorithm would achieve optimality. Similar to approximation algorithms, the quality of the algorithm is measured by comparing against the offline cost: \mathcal{A} is *C-competitive* if the cost incurred by \mathcal{A} to service each request sequence ϱ is at most C times the optimal (offline) service cost for ϱ , plus possibly an additive constant independent of ϱ . The *competitive ratio* of \mathcal{A} is the smallest C for which \mathcal{A} is C -competitive. The competitive ratio is frequently used to study the performance of online algorithms for the k -server Problem, as well as other optimization problems. The reader is referred to the book of Borodin and El-Yaniv [30] for a comprehensive discussion of competitive analysis.

It is interesting to note that the work functions ω_ϱ play a central role in the algorithm with current best competitiveness for the k -server problem: Work Function Algorithm. The Work Function Algorithm chooses its service of the request sequence ϱ as follows: Suppose that WFA is in configuration S , and that the current work function is ω . On request r , WFA chooses some $x \in S$ which minimizes $rx + \omega \wedge r(S - x + r)$, and moves the server from x to r . If there is more than one choice which minimizes that quantity, the choice is arbitrary. WFA can be seen as a “linear combination” of two greedy strategies. The first one, a *short-sighted greedy*, minimizes the cost rx in the given step. The second, a *retrospective greedy*, chooses the optimal configuration after r , that is, the configuration $S - x + r$ that minimizes $\omega \wedge r(S - x + r)$. The short-sighted greedy strategy is not competitive for any k . The retrospective greedy strategy is not competitive for $k \geq 3$, and its

	$\omega_\varrho(\{y, z\})$	$\omega_\varrho(\{x, z\})$	$\omega_\varrho(\{x, y\})$
initial	$\omega_\epsilon:$	<u>0</u>	1
request x	$\omega_x:$	2	<u>1</u>
request y	$\omega_{xy}:$	<u>2</u>	3
request x	$\omega_{xyx}:$	4	<u>3</u>
request y	$\omega_{xyxy}:$	4	4
request z	$\omega_{xyxyz}:$	<u>4</u>	6

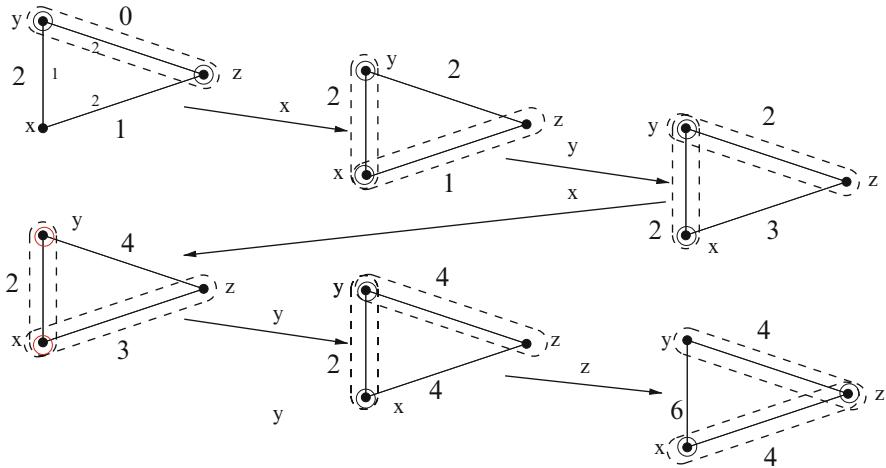


Fig. 7 Calculating the optimal solution using dynamic programming for the example in Fig. 6. The work functions ω_ϱ are shown in the figure as values adjacent to the corresponding pairs of points. Equivalently, the work functions can be written into the traditional dynamic programming table (top). The support is marked by *ovals* in the figure and is *underlined* in the table. The minimum value of the last row is 4 – the optimal value

competitive ratio for $k = 2$ is at least 3. The reader might verify that the service sequence depicted in Fig. 6 shows the steps of WFA for that example.

Manasse, McGeoch, and Sleator [82], have proved the following:

Theorem 2 *No online algorithm for k servers has competitive ratio smaller than k if a metric space has at least $k + 1$ points.*

They also give the *k-server conjecture* which states that, for each k , there exists an online algorithm for k servers which is k -competitive in any metric space. For $k > 2$, this conjecture has been settled only in a number of special cases, including trees and spaces with at most $k + 2$ points. (cf. the work of Chrobak, Karloff, Payne, and Vishwanathan [44], the work of Chrobak and Larmore [41] and the work of Koutsoupias and Papadimitriou [72].) Koutsoupias and Papadimitriou have shown:

Theorem 3 *The Work Function Algorithm is $(2k - 1)$ -competitive for k servers in arbitrary metric spaces.*

Thus a wide gap remains. Even some simple-looking special cases remain open, for example the 3-server problem on the circle, in the plane, or in 6-point spaces. Chrobak and Larmore [42] (see also [43]) prove:

Theorem 4 *The Work Function Algorithm is 2-competitive for $k = 2$.*

Bein, Chrobak, and Larmore [15] show:

Theorem 5 *The Work Function Algorithm is 3-competitive for $k = 3$ if the metric space M is the Manhattan plane.*

4 Intricate Dynamic Programming: Block Deletion in Quadratic Time

Sorting problems under various operations have been studied extensively, including work on sorting with prefix reversals, transpositions and block moves. This section contains an example from this realm and shows that intricate setup of dynamic programming can speed up dynamic programming schemes.

4.1 Preliminaries

Define a *permutation* of length n to be a list $x = (x_1, \dots, x_n)$ consisting of the integers $\{1, \dots, n\}$ where each number occurs exactly once. For any $1 \leq i \leq j \leq n$ denote the sublist of x that starts at position i and ends with position j by $x_{i\dots j}$. A list y is a *subsequence* of x if y is obtained from x by deleting any number of elements. For example, $(2, 3)$ is a subsequence of $(2, 4, 3, 1)$, but not a sublist. Since x has no duplicate symbols, a subsequence of x is uniquely characterized by its set of items. By a slight abuse of notation, one identifies a subsequence with the set of its items. Define the *closure* of a subsequence of x to be the smallest sublist of x which contains it. For example, the closure of the subsequence $(2, 3)$ of $(2, 4, 3, 1)$ is the sublist $(2, 4, 3)$. If A and A' are subsequences of a list x , say that A and A' are *separated* if the closures of A and A' are disjoint.

A block deletion sequence for a subsequence y of x consists of a sequence A_1, \dots, A_m of disjoint non-empty subsequences of y such that

1. for all $i = 2, \dots, m$, A_i is a block in $y - \bigcup_{u=1}^{i-1} A_u$, and
2. $y - \bigcup_{u=1}^m A_u$ is a monotone increasing list.

For example, a minimum length block deletion sequence for the list $(1, 4, 2, 5, 3)$ consists of two steps. First delete the block (2) , obtaining $(1, 4, 5, 3)$, then delete the block $(4, 5)$, obtaining the sorted list $(1, 3)$. Figure 8 shows another example of a block deletion sequence. A complete block deletion sequence for a subsequence y

Fig. 8 A block deletion sequence. There are five steps: A_1, \dots, A_5

0	6	4	7	5	1	8	3	2	9
0	6	7	5	1	8	3	2	9	$\overset{A_2}{}$
0	6	7	5	8	3	2	9	$\overset{A_3}{}$	$\overset{A_1}{}$
0	5	8	3	2	9	$\overset{A_4}{}$	$\overset{A_5}{}$	$\overset{A_2}{}$	$\overset{A_1}{}$
0	5	8	9	$\overset{A_3}{}$	$\overset{A_4}{}$	$\overset{A_5}{}$	$\overset{A_2}{}$	$\overset{A_1}{}$	9

of x consists of a block deletion sequence A_1, \dots, A_m of y such that $y - \bigcup_{u=1}^m A_u$ is the empty list.

4.2 A Dynamic Program for Complete Block Deletion

Consider first the complete block deletion problem for all sublists of x , which will be solved in quadratic time by dynamic programming. Once the $O(n^2)$ answers to this problem are obtained, the original block deletion problem can be solved in quadratic time. The following three lemmas are used:

Lemma 1 *If A_1, \dots, A_m is a block deletion sequence for a sublist y of x , and $1 \leq u < v \leq m$, then either A_u and A_v are separated, or A_u is a subsequence of the closure of A_v .*

Proof The closure of A_u cannot contain any item of A_v , since otherwise A_u could not be deleted before A_v . If all items of A_v are before A_u or all items of A_v are after A_u , then A_u and A_v are separated. If some items of A_v are before A_u and some items are after A_u , then A_u is a subsequence of the closure of A_v . \square

Lemma 2 *If $A_1, \dots, A_t, A_{t+1}, \dots, A_m$ is a block deletion sequence for a sublist y of x , and A_t and A_{t+1} are separated, then A_t and A_{t+1} may be transposed, i.e., $A_1, \dots, A_{t+1}, A_t, \dots, A_m$ is a block deletion sequence for y .*

Proof For any u , let $y_u = x - \bigcup_{v < u} A_v$. By definition, A_t is a block of y_t , and A_{t+1} is a block of $y_{t+1} = y_t - A_t$. Since A_t and A_{t+1} are separated, A_{t+1} is also a block of y_t . Thus, A_{t+1} can be deleted before A_t . \square

Lemma 3 *For any $1 \leq i \leq j \leq n$, if there is a complete block deletion sequence for $x_{i \dots j}$ of length m , then there is a complete block deletion sequence for $x_{i \dots j}$ of length m such that x_i is deleted in the last step.*

Proof Let A_1, \dots, A_m be a complete block deletion sequence of $x_{i \dots j}$. Suppose that $x_i \in A_t$ for some $t < m$. Since x_i is deleted in the t^{th} move of the sequence, all deletions after that must involve blocks whose first symbol occurs to the right of x_i in x . That is, for any $v > t$, x_i cannot be an item of the closure of A_v , hence, by [Lemma 1](#), A_t and A_v must be separated. By [Lemma 2](#), one can transpose A_t with A_v for each $v > t$ in turn, moving A_t to the end of the deletion sequence. \square

Next is given a recurrence to compute the minimum length of a complete block deletion sequence. This recurrence will be used in [Algorithm 1](#).

Theorem 6 *Given a permutation x , let $t_{i,j}$ denote the minimum length of any complete block deletion sequence for the sublist $x_{i \dots j}$. The values of $t_{i,j}$ can be computed inductively by the following: Set $t_{i,i} = 1$, $t_{i,j} = 0$ for $i > j$, and for $i < j$ set*

$$t_{i,j} = \begin{cases} \min\{1 + t_{i+1,j}, t_{i+1,\ell-1} + t_{\ell,j}\}, & \text{if } \exists \ell \in \{i+1, \dots, j\} \text{ such that } x_\ell = x_i + 1, \\ 1 + t_{i+1,j}, & \text{otherwise.} \end{cases} \quad (15)$$

Proof The proof is by induction on $j - i$ to show that $t_{i,j}$ is computed correctly. The base case is trivial, namely $t_{i,i} = 1$, because it takes one block deletion to delete a sublist of length 1.

For the inductive step, assume that $t_{i,j}$ is the length of a minimum block deletion sequence for $x_{i \dots j}$ when $j - i < k$. For $j - i = k$, let $m = t_{i,j}$ and let A_1, \dots, A_m be a corresponding minimum length complete block deletion sequence of $x_{i \dots j}$. By [Lemma 3](#), one can insist that $x_i = a$ is an item in A_m . Consider now two cases based on whether there is an ℓ with $i < \ell \leq j$ such that $x_\ell = a + 1$ (The reader might also consult [Fig. 9](#)).

If the element $a + 1$ does not occur in the interval $\{i + 1, \dots, j\}$, then the element a is not part of a block in this interval and must be deleted by itself. So, A_1, \dots, A_{m-1} is a complete block deletion sequence of $x_{i+1 \dots j}$. Note that it must be of optimum length, for if B_1, \dots, B_r were a shorter complete block deletion sequence of $x_{i+1 \dots j}$, then $B_1, \dots, B_r, \{a\}$ would be shorter than A_1, \dots, A_m . Thus, as $j - (i + 1) = k - 1$, by the induction hypothesis $t_{i+1,j} = m - 1$. So $t_{i,j} = 1 + t_{i+1,j}$, which is optimum.

Now for the case that the element $x_\ell = a + 1$ does occur in the interval $\{i + 1, \dots, j\}$. This means that $a + 1$ and possibly other larger values can be included in A_m when element a is deleted. If element $a + 1$ is not included in A_m then the same argument used in the previous paragraph shows that $m = 1 + t_{i+1,j}$. If on the other hand $a + 1$ is included in A_m , then by [Lemma 1](#), for any t , $(1 \leq t \leq m)$, A_t is either completely before or completely after the element $a + 1$, since A_m is deleted after A_t . By [Lemma 2](#), one can permute the indices so that, for some $u < m$,

1. if $t \leq u$, then A_t is a subsequence of $x_{i+1 \dots \ell-1}$, and
2. if $u < t \leq m$, then A_t is a subsequence of $x_{\ell+1 \dots j}$.

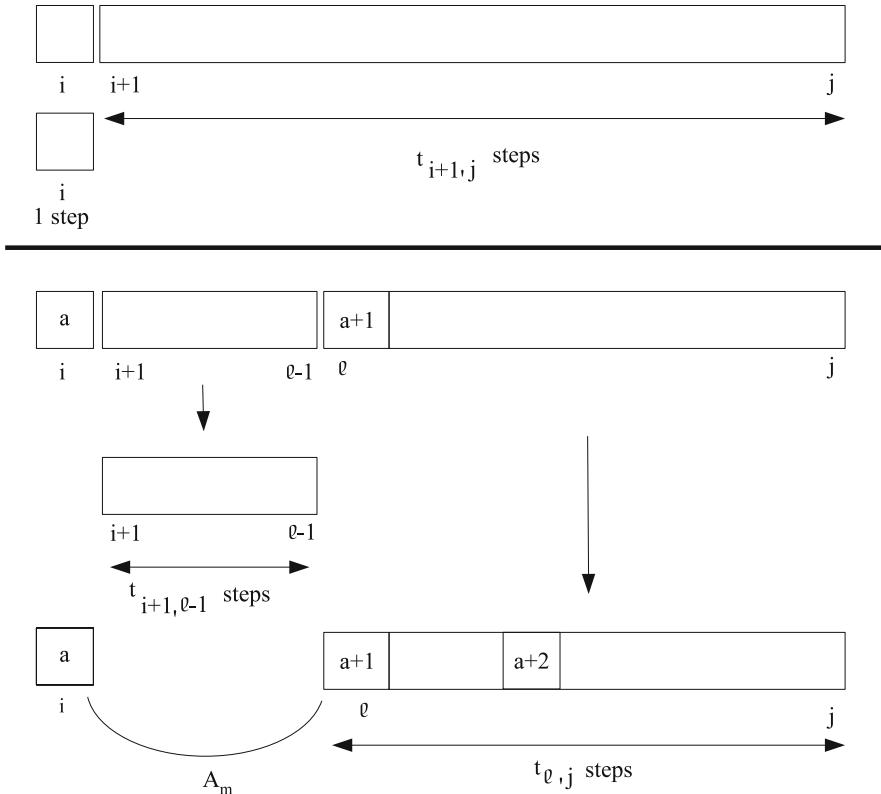


Fig. 9 The recurrence for the $t_{i,j}$ values

Consequently, A_1, \dots, A_u is a block deletion sequence for $x_{i+1 \dots \ell-1}$ and $A_{u+1}, \dots, A_m - \{a\}$ is a block deletion sequence for $x_{\ell+1 \dots j}$. Both of these block deletion sequences must be optimum for their respective intervals. That is, for example, if B_1, \dots, B_r were a shorter complete block deletion sequence for $x_{i+1 \dots \ell-1}$, then $B_1, \dots, B_r, A_{u+1}, \dots, A_m$ would be a complete block deletion sequence for $x_{i \dots j}$, contradicting the minimality of m . By the induction hypothesis, as $\ell - 1 - (i + 1) < j - i$ and $j - \ell < j - i$, it follows that $t_{i+1,\ell-1} = u$ and $t_{\ell,j} = m - u$, so $t_{i,j} = t_{i+1,\ell-1} + t_{\ell,j} = u + (m - u) = m$. \square

The resulting dynamic programming algorithm **BLOCKDELETION**, which is derived from the recurrence of **Theorem 6**, is shown below.

Next the analysis of the run time of algorithm **BLOCKDELETION** is considered. Let $z = (z_1, \dots, z_n)$ be the inverse permutation of x , i.e., $x_i = k$ if and only if $z_k = i$. Note that z can be computed in $O(n)$ preprocessing time.

Theorem 7 Algorithm **BLOCKDELETION** has run time $O(n^2)$.

Algorithm 1 BLOCKDELETION(x)

Let n be the number of elements in x

```

for  $i \leftarrow 1$  to  $n$  do
     $t[i, i] \leftarrow 1$ 
for  $k \leftarrow 2$  to  $n$  do
    for  $i \leftarrow 1$  to  $n - k + 1$  do
        Let  $x_\ell \leftarrow x_i + 1$ ;  $j \leftarrow i + k - 1$ 
        if  $i < \ell \leq j$ 
            then  $t[i, j] \leftarrow \min\{1 + t[i + 1, j], (t[i + 1, \ell - 1] + t[\ell, j])\}$ 
        else  $t[i, j] \leftarrow 1 + t[i + 1, j]$ 
return

```

Proof. To prove the theorem one shows that if $t_{u,v}$ are already known for all $i < u \leq v \leq j$, then $t_{i,j}$ can be computed in $O(1)$ time. Let $m = t_{i,j}$ for $i < j$ and let A_t be the subsequence of $x_{i\dots j}$ that is deleted at step t of the complete block deletion sequence of $x_{i\dots j}$ of length m . By Lemma 3, assume that $x_i \in A_m$. If $|A_m| > 1$, the index $\ell = z_{x_i+1}$, i.e., $x_\ell = 1 + x_i$, can be found in $O(1)$ time, since one has already spent $O(n)$ preprocessing time to compute the array z . The recurrence thus takes $O(1)$ time to execute for each i, j . \square

Note that to obtain the actual sequence of steps in the optimum complete block deletion sequence one can store appropriate pointers as the $t_{i,j}$ are computed.

4.3 Computing Block Deletion

The reader is reminded that for the block deletion problem one needs to find the minimum length sequence of block deletions to transform a permutation into a monotone increasing list. Next it is shown how one obtains a solution to the block deletion problem for x in $O(n^2)$ -time given that all $t_{i,j}$ are known for $1 \leq i \leq j \leq n$. Define a weighted acyclic directed graph G with one node for each $i \in \{0, \dots, n+1\}$. There is an edge from i to j if and only if $i < j$ and $x_i < x_j$, and the weight of that edge is $t_{i+1,j-1}$. Simply observe that there is a path $\langle 0, i_1, i_2, \dots, i_k, n+1 \rangle$ in G exactly when x_{i_1}, \dots, x_{i_k} is a monotone increasing list. Furthermore the weight of the edge $\langle i_\ell, i_{\ell+1} \rangle$ gives the minimum number of block deletions necessary to delete elements between position i_ℓ and $i_{\ell+1}$ in x . Thus if there is a block deletion sequence of x of length m , there must be a path from 0 to $n+1$ in G of weight m , and vice-versa.

Using standard dynamic programming, a minimum weight path from 0 to $n+1$ can be found in $O(n^2)$ time. Let $0 = i_0 < i_1 < \dots < i_\ell = n+1$ be such a minimum weight path, and let $w = \sum_{u=1}^\ell t_{i_{u-1}+1, i_u-1}$ be the weight of that minimum path. Since every deletion is a block deletion, the entire list can be deleted to a monotone list in w block deletions. Thus it follows:

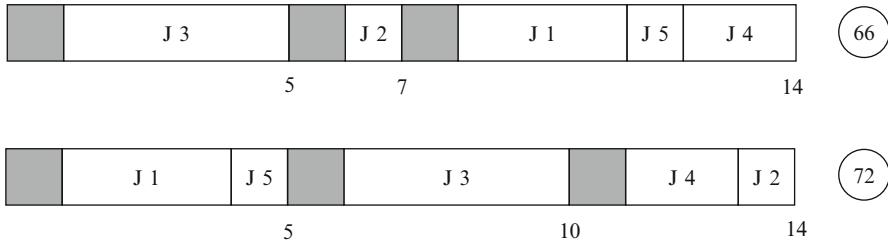


Fig. 10 A batching example. Shown are two feasible schedules for a 5-job problem where processing times are $p_1 = 3, p_2 = 1, p_3 = 4, p_4 = 2, p_5 = 1$ and the weights are $w_1 = w_4 = w_5 = 1$ and $w_2 = w_3 = 2$. The encircled values give the sum of weighted completion times of the depicted schedules

Theorem 8 *The block deletion problem can be solved in time $O(n^2)$.*

5 Total Monotonicity and Batch Scheduling

Although dynamic programming has been around for decades there have been more recent techniques for making dynamic programming more efficient. This section describes an example from scheduling, where such a technique is applied. Before the technique is shown, the simple traditional dynamic programming for the scheduling problem is given and then the dynamic programming speedup technique is described.

5.1 The Problem $1|s - \text{batch}| \sum w_i C_i$

Consider the *batching problem* where a set of jobs $\mathcal{J} = \{J_i\}$ with processing times $p_i > 0$ and weights $w_i \geq 0, i = 1, \dots, n$, must be scheduled on a single machine, and where \mathcal{J} must be partitioned into *batches* $\mathcal{B}_1, \dots, \mathcal{B}_r$. All jobs in the same batch are run jointly and each job's completion time is defined to be the completion time of its batch. One assumes that when a batch is scheduled it requires a setup time $s = 1$. The goal is to find a schedule that minimizes the *sum of completion times* $\sum w_i C_i$, where C_i denotes the completion time of J_i in a given schedule. Given a sequence of jobs, a batching algorithm must assign every job J_i to a batch. More formally, a feasible solution is an assignment of each job J_i to the m_i^{th} batch, $i \in \{1, \dots, n\}$ (Fig. 10).

The problem considered has the jobs executed sequentially, thus the problem is more precisely referred to as the *s-batch* problem. There is a different version of the problem not studied here, where the jobs of a batch are executed in parallel, known as the *p-batch* problem. In that case, the length of a batch is the maximum of the processing times of its jobs. The s-batch is also denoted in $\alpha|\beta|\gamma$ notation as the $1|s\text{-batch}| \sum w_i C_i$ problem. Brucker and Albers [6] showed that the

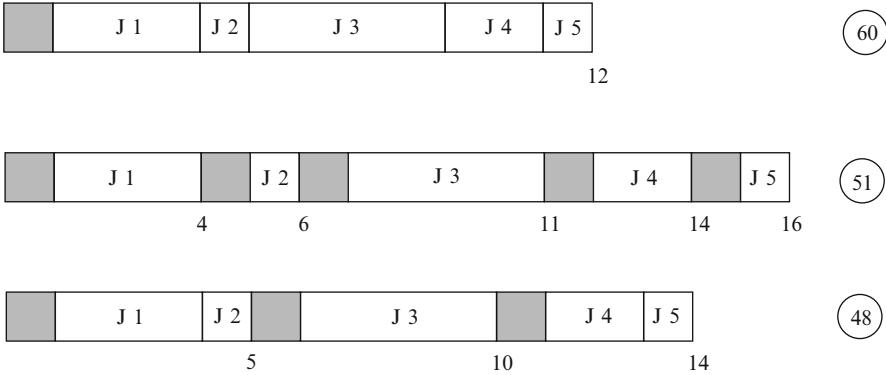


Fig. 11 List batching. Shown are three schedules for a 5-job problem where all weights are 1 and the processing requirements are $p_1 = 3, p_2 = 1, p_3 = 4, p_4 = 2, p_5 = 1$. The encircled values give the sum of weighted completion times of the depicted schedules

$1|s\text{-batch}| \sum w_i C_i$ problem is \mathcal{NP} -hard in the strong sense by giving a reduction from 3-PARTITION.

There is a large body of work on dynamic programming and batching; see the work of Baptiste [12], Baptiste and Jouglet [13], Brucker, Gladky, Hoogeveen, Kovalyov, Potts Tautenhahn, and van de Velde [36], Brucker, Kovalyov, Shafransky, and Werner [37], and Hoogeveen and Vestjens [64], as well as the scheduling text book by Peter Brucker [33]. Batching has wide application in manufacturing (see e.g., [34, 85, 98]), decision management (see, e.g., [74]), and scheduling in information technology (see e.g., [46]). More recent work on online batching is related to the TCP (Transmission Control Protocol) acknowledgment problem (see [20, 49, 67]).

5.2 List Batching

A much easier version of the problem is the *list* version of the problem where the order of the jobs is given, i.e., $m_i \leq m_j$ if $i < j$. An example is given in Fig. 11.

Assume that the jobs are $1, \dots, n$ and are given in this order. One can then reduce the list batching problem to a shortest path problem in the following manner: Construct a weighted directed acyclic graph G with nodes $i = 1, \dots, n$ (i.e., one node for each job) and add a dummy node 0. There is an edge (i, j) if and only if $i < j$. (See Fig. 12 for a schematic.) Let edge costs $c_{i,j}$ for $i < j$ be defined as

$$c_{i,j} = \left(\sum_{\ell=i+1}^n w_\ell \right) \left(s + \sum_{\ell=1}^j p_\ell \right). \quad (16)$$

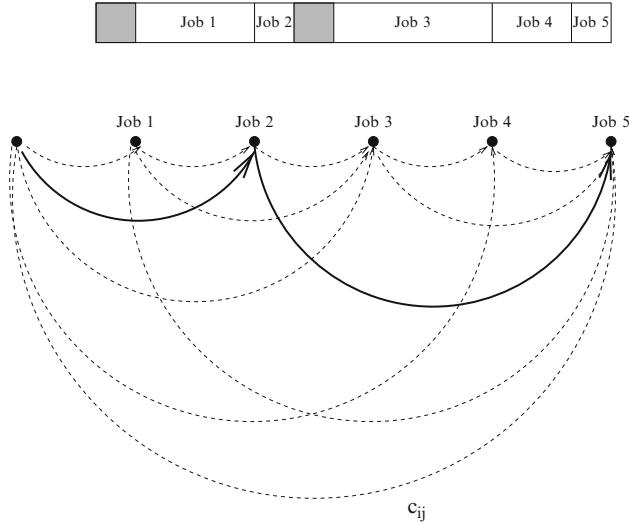


Fig. 12 Reduction of the list batching problem to a path problem

It is easily seen (see [6] for details) that the cost of path $< 0, i_1, i_2, \dots, i_k, n >$ gives the $\sum C_i w_i$ value of the schedule which batches at each job i_1, i_2, \dots, i_k . Conversely, any batching with cost A corresponds to a path in G with path length A .

A shortest path can be computed in time $O(n^2)$ using the following dynamic program:

Let

$$E[\ell] = \text{cost of the shortest path from } 0 \text{ to } \ell,$$

then

$$E[\ell] = \min_{0 \leq k < \ell} \{E[k] + c_{k,\ell}\} \text{ with } E[0] = 0, \quad (17)$$

which results in a table, in which elements can be computed row by row (see Fig. 13).

In other words, the dynamic program computes the row minima of the $n \times n$ matrix E , where

$$E[\ell, k] = \begin{cases} E[k] + c[k, \ell] & \text{if } \ell < k \\ \infty & \text{else} \end{cases} \quad (18)$$

with $\ell = 1, \dots, n$ and $k = 0, \dots, n - 1$.

As it turns out it is not necessary to calculate all entries of E to calculate the optimal solution. Surprisingly, only $O(n)$ have to be looked up throughout the entire calculation. The reason that this is possible is that E is a matrix with special properties discussed next.

Row 1	$c[0,1]+E[0]$	$\min \rightarrow E[1]$
Row 2	$c[0,2]+E[0] \quad c[1,2]+E[1]$	$\min \rightarrow E[2]$
Row 3	$c[0,3]+E[0] \quad c[1,3]+E[1] \quad c[2,3]+E[3]$	$\min \rightarrow E[3]$
Row 4	$c[0,4]+E[0] \quad c[1,4]+E[1] \quad c[2,4]+E[2] \quad c[3,4]+E[4]$	$\min \rightarrow E[4]$

Fig. 13 Dynamic programming tableau

5.3 The Monge Property and Total Monotonicity

Definition 1 A matrix A is Monge if for all $i < i'$ and $j < j'$,

$$A[i, j] + A[i', j'] \leq A[i', j] + A[i, j']. \quad (19)$$

Definition 2 A 2×2 matrix is monotone if the rightmost minimum of the upper row is not to the right of the rightmost minimum of the lower row. More formally, $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ is monotone if $b \leq a$ implies that $d \leq c$.

Definition 3 A matrix A is called totally monotone if all 2×2 dimensional submatrices are monotone.

Observation 1 Every Monge matrix is totally monotone.

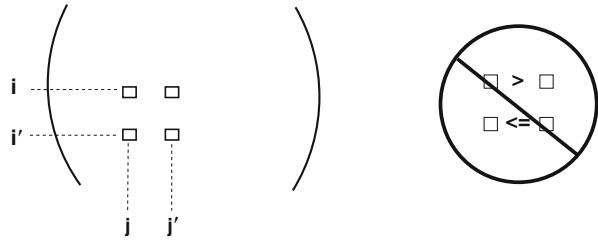
The reader is referred to Fig. 14. Monge matrices occur routinely. For example in the batching to shortest path reduction, the cost matrix C is a Monge matrix:

Lemma 4 The matrix $C = (c_{i,j})$ defined in (16) is Monge for all choices of $p_i, w_i \geq 0$. Furthermore values can be queried in $O(1)$ time after linear preprocessing.

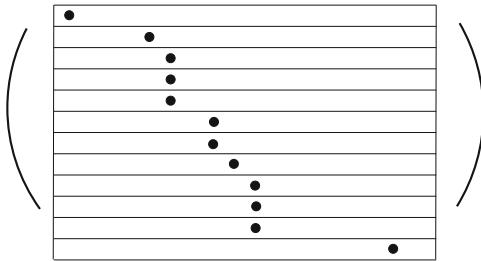
Proof Let $W_i = \sum_{v=1}^i w_v$ and $P_i = \sum_{v=1}^i p_v$ be the partial sum of the p_i and w_i values. Then

$$c[i, j] = c_{i,j} = (W_n - W_i)(s + P_j - P_i)$$

Fig. 14 Monge and monotonicity. Top left the Monge property is shown, which prohibits the situation top right. Thus row minima veer to the right (bottom figure)



$$A_{ii'} + A_{jj'} \leq A_{i'j} + A_{j'i}$$



For $i < i'$ and $j < j'$

$$c[i, j] + c[i', j'] - c[i', j] - c[i, j'] \quad (20)$$

$$= (P_{j'} - P_j)(W_{i'} - W_i) \quad (21)$$

$$\geq 0. \quad (22)$$

Also, notice that these values can be queried in $O(1)$ time after linear preprocessing by setting up arrays of partial sums for W_i and P_i in linear time. \square

Furthermore, the matrix E is also a Monge matrix:

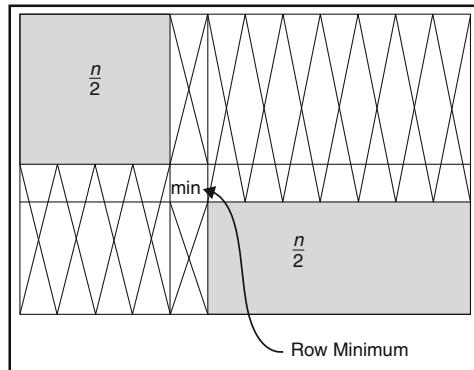
Lemma 5 *The matrix $E = (E_{\ell,k})$ defined in (18) is Monge.*

Proof Monge is preserved under addition and taking the minimum. \square

Computing the row minima of a Monge (or totally monotone) matrix can be done trivially in $O(n \log n)$, cf. Fig. 15. A complex recursive procedure by Agarwal, Klawé, Moran, Shor, and Wilber [3] known as the SMAWK algorithm (the name was derived using the initials of authors of the original paper in which the algorithm was introduced), which has linear run time.

Note that the dynamic program of Fig. 13 is essentially used to calculate the row minima of E – a Monge matrix. However, the trivial $O(n \log n)$ cannot be used here since that algorithm requires all elements of E to be available offline,

Fig. 15 Calculating all row minima of a Monge matrix. The algorithm finds the minimum of the middle row in linear time and then recurses on the upper left and lower right parts of the matrix



i.e., before the computation begins. Instead, there is a protocol by which each element can be queried: Once the minimum of row 1 is known, then any element in column 1 can be generated in constant time; once the minimum of row 2 is known then any element of row 1 and 2 are “knowable”; and so forth. (See also Fig. 16.)

More formally the protocol is as follows:

1. For each row index ℓ of E , there is a column index γ_ℓ such that for $k > \gamma_\ell$, $E_{\ell,k} = \infty$. Furthermore, $\gamma_\ell \leq \gamma_{\ell+1}$.
2. If $k \leq \gamma_\ell$, then $E[\ell, k]$ can be evaluated in $O(1)$ time *provided that the row minima of the first ℓ rows are already known*.
3. E is a totally monotone matrix.

Larmore and Schieber [77] have developed an algorithm that generalizes the SMAWK to run in linear time in this case as well. Their algorithm is also known as the LARSCH algorithm – again, as with the SMAWK algorithm, the name was derived using initials of authors of the original paper in which the algorithm was introduced in Larmore and Schieber [77]. Both SMAWK and LARSCH are important in dynamic programming speedup and are explained in the next section.

6 The SMAWK and LARSCH Algorithm

This section gives “ready to implement” descriptions of the SMAWK and LARSCH algorithms mentioned in the previous section.

6.1 The Matrix Searching Problem

The *matrix searching problem* is the problem of finding all row minima of a given matrix M . If n, m are the number of rows columns, respectively, of M , the problem clearly takes $\Theta(nm)$ time in the worst case. However, if M is *totally monotone* the problem can be solved in $O(n + m)$ time using the SMAWK algorithm [3].

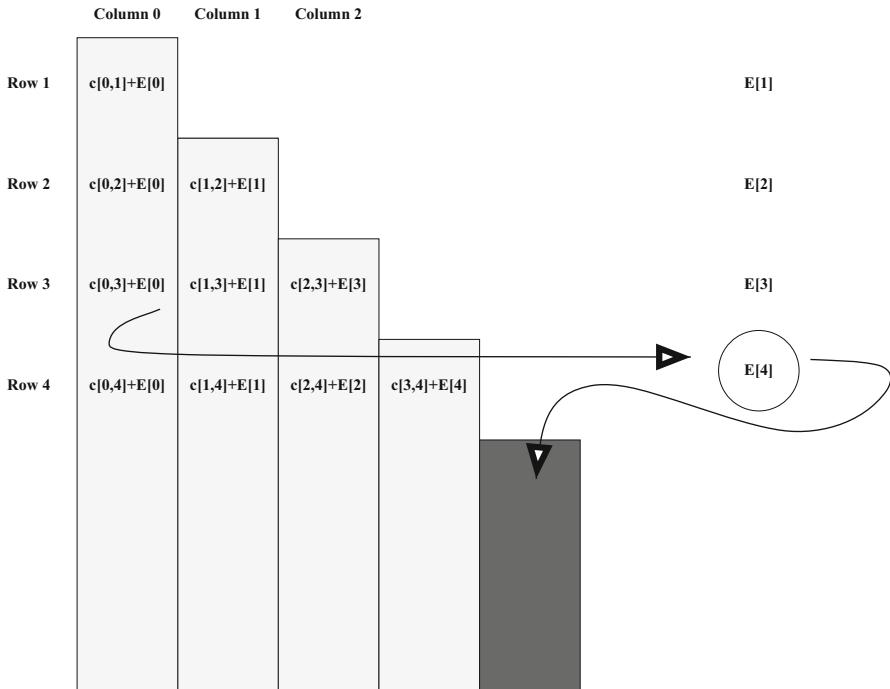


Fig. 16 The “online” protocol of the tableau. Note that once the minimum of row 4 is known, column 4 is “knowable”

SMAWK is recursive, but uses two kinds of recursion, called INTERPOLATE and REDUCE, which are described in the following. Let $1 \leq J(i) \leq m$ be the index of the minimum element of the i^{th} row.¹ Since M is totally monotone, $J(i) \leq J(k)$ for any $i < k$.

1. For small cases, SMAWK uses a trivial algorithm. If $m = 1$, the problem is trivial. If $n = 1$, simply use linear search.
2. If $n \geq 2$, then INTERPOLATE can be used. Simply let M' be the $\lfloor n/2 \rfloor \times m$ submatrix of M consisting of all even indexed rows of M . Recursively, find all row minima of M' , and then use linear search to find the remaining minima of M in $O(n + m)$ time.
3. If $m > n$, then REDUCE can be used. The set $\{J(i)\}$ clearly has cardinality at most n . REDUCE selects a subset of the columns of M which has cardinality at most n , and which includes Column $J(i)$ for all $1 \leq i \leq n$. Let M' be the submatrix of M consisting of all the columns selected by REDUCE. Then, recursively, find all row minima of M' . These will exactly be the row minima of M . The time required to select the columns of M' is $O(n + m)$.

¹Use the leftmost rule to break ties.

SMAWK then operates by alternating the two kinds of recursion. If the initial matrix is $n \times n$, then INTERPOLATE reduces to the problem on a matrix of size roughly $n/2 \times n$, and then REDUCE reduces to the problem on a matrix of size roughly $n/2 \times n/2$, and so forth.

Time Complexity

Let $T(n)$ be the time required by SMAWK to find all row minima of an $n \times n$ totally monotone matrix. Applying both INTERPOLATE and REDUCE, obtain the recurrence

$$T(n) = O(n) + T(n/2)$$

and thus $T(n) = O(n)$. By a slight generalization of the recurrence, one can show that SMAWK solves the problem for an $n \times m$ matrix in $O(n + m)$ time.

INTERPOLATE is explained using the code below.

Code for INTERPOLATE

```

1: { $M$  is an  $n \times m$  totally monotone matrix, where  $m \leq n$ .}
2: if  $n = 1$  then
3:    $J(1) = 1$ 
4: else
5:   Let  $M'$  be the matrix consisting of the even indexed rows of  $M$ .
6:   Obtain  $J(i)$  for all even  $i$  by a recursive call to REDUCE on  $M'$ .
7:   for all odd  $i$  in the range 1 to  $n$  do
8:     if  $i = n$  then
9:       Find  $J(n) \in [J(n - 1), n]$  by linear search.
10:    else
11:      Find  $J(i) \in [J(i - 1), J(i + 1)]$  by linear search.
12:    end if
13:   end for
14: end if
```

The total number of comparisons needed to find the minima of all odd rows is at most $n - 1$, as illustrated in Fig. 17 below.

Now for REDUCE. The procedure REDUCE operates by maintaining a stack, where each item on the stack is a column (actually, a column index). Initially the stack is empty, and columns are pushed onto the stack one at a time, starting with Column 1. The capacity of the stack is n , the number of rows.

But before any given column is pushed onto the stack, any number (zero or more) of earlier columns are popped off the stack. A column is popped if it is determined that it is *dominated*, which implies that none of its entries can be the minimum of any row. In some cases, if the stack is full, a new column will not be pushed.

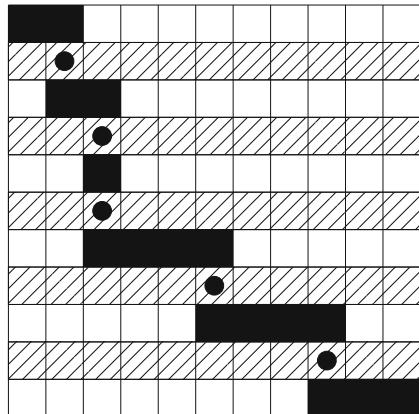
At the end, those columns which remain on the stack form the matrix M' .

Dominance

There is a loop invariant, namely that if the stack size is at least i , and if $S[i] = j$, then all entries of Column j above Row i are known to be useless, i.e., will

Fig. 17 INTERPOLATE.

M' consists of the hatched rows, and the minima of those rows are indicated by *black dots*. By the monotonicity property of J , only search the interval indicated in *black* to find the minimum of any odd numbered row



not be the minima of any row. Formally, $M[i', S[i - 1]] \leq M[i', j]$ for all $1 \leq i' < i$.

Suppose that Column k is the next column to be possibly pushed onto the stack. If the stack is empty, then $S[1] \leftarrow k$, and one is done. Otherwise, it must be checked to see whether the top column in the stack is dominated. Let i be the current size of the stack, and let $S[i] = k$. Then, necessarily, $i \leq j < k$. Column j is dominated if $M[i, k] < M[i, j]$. The reason is that $M[i', j] \geq M[i', S[i - 1]]$ for any $i' < i$ by the loop invariant, and that $M[i', j] < M[i', k]$ for all $i' \geq i$ by monotonicity.

If the stack is popped, then the new top is tested for being dominated. This continues until the test fails. If the stack size is currently less than n , k is pushed onto the stack; otherwise, Column k is useless and is discarded.

Code for REDUCE

```

1: { $M$  is an  $n \times m$  totally monotone matrix.}
2:  $top \leftarrow 0$  {Initialize the stack to empty}
3: for  $j$  from 1 to  $m$  do
4:   while  $top > 0$  and  $M[top, j] < M[top, S[top]]$  do
5:      $top \leftarrow top - 1$  {Pop the stack.}
6:   end while
7:   if  $top < n$  then
8:      $top \leftarrow top + 1$  and then  $S[top] \leftarrow j$  {Push  $j$  onto the stack.}
9:   end if
10: end for
11: Call INTERPOLATE on  $M'$ , consisting of columns remaining on the stack.
The total number of comparisons needed to execute REDUCE (other than the recursive call) is at most  $2m$ . One sees this using an amortization argument. Each column is given two credits initially. When a column is pushed onto the stack, it spends one credit, and when it is popped off the stack it spends one credit. Each of those two events requires at most one comparison. Thus, the total number of comparisons is at most  $2m$ .

```

Fig. 18 REDUCE. M' consists of the hatched columns. All row minima, indicated by black dots, lie within those columns, although possibly not all columns contain minima

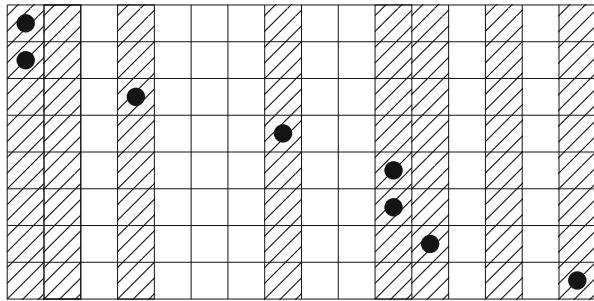


Figure 18 shows an example of an 8×16 matrix M , where the shaded columns survive to form M' .

6.2 The Online Matrix Searching Problem

Define an *almost lower triangular matrix* to be matrix M of n rows, numbered $1 \dots n$, and m columns such that the length of the rows increases as n increases. More formally, let the columns be numbered $0 \dots m - 1$. Then there must exist n row lengths, $1 \leq \ell[1] \leq \ell[2] \leq \dots \leq \ell[n] = m$, such that $M[i, j]$ is defined if and only if $1 \leq i \leq n$ and $0 \leq j < \ell[i]$. If $\ell[i] = i$ for all i , then M is a standard lower triangular matrix.

Given an almost lower triangular matrix, let $J(i)$ be the column index of the minimum entry in the i^{th} row of M . (Ties are broken arbitrarily). The *online matrix search problem* is the problem of finding all row minima of M , with the condition that a given entry $M[i, j]$ is available if and only if $J(k)$ has been computed for every k such that $\ell[k] < j$. For example, if M is a standard lower triangular matrix, $M[i, j]$ is available if and only if $J[k]$ has been computed for all $k \leq j$. Note that $M[i, 0]$ is available initially.

Assume that the computation is done by a process B which is in communication with a supervisor A . The online computation then proceeds as follows.

- A grants permission for B to see Column 0.
- B reports $J[1]$ (which is certainly 0) to A .
- A grants permission for B to see Column 1.
- B reports $J[2]$ to A .
- A grants permission for B to see Column 1.
- B reports $J[3]$ to A .
- etc.

Note that B is unable to compute $J(i)$ until it has seen Columns 0 through $i - 1$, since the minimum of Row i could be in any of those columns. Conversely, one must have computed $J(1)$ through $J(i - 1)$ in order to be allowed to see those columns. Thus, the order of the events in the above computation is strict.

6.3 Algorithm LARSCH

In general, it takes $O(nm)$ time to solve the online matrix search problem. However, if M is totally monotone, the values of J are monotone increasing, and the online matrix search problem can be solved in $O(n + m)$ time by using an online algorithm LARSCH, which is the online version of SMAWK, with some adaptations.

Let M be the input matrix. LARSCH works using a chain of processes, each of which is in communication with its *supervisor*, the process above it in the chain. The supervisor of the top process is presumably an application program. Refer to the process below a process as its *child*.

Each process in the chain works an instance of the online matrix searching problem. The top process works the instance given by the application. Each process reports results interleaved with messages received from its supervisor.

Each process P works the problem on a strictly monotone almost lower triangular matrix M_P , which is a submatrix of M . If Q is the child of P , then P creates a submatrix M_Q of M_P and passes that submatrix to Q . Of course, P never passes the entries of the matrix to Q ; rather, it tells Q which entries of M it is allowed to see. For clarity assume that each process uses its own local row and column indices, but is aware of the global indices of each entry. For example, in example calculation in [Table 5](#), $M_5[3, 2] = M[15, 11]$.

Alternating Types.

In the following detailed description of LARSCH, assume that the initial matrix M is standard lower triangular. (The algorithm can easily be generalized to cover other cases.)

Let P_0, P_1, \dots, P_h , $h = 2(\lfloor \log_2(n+1) \rfloor - 1)$, be the processes, where P_{t+1} is the child of P_t . The processes are of alternating *types*. If t is even, say that P_t has *standard type*, while if t is odd, say that P_t has *stretched type*.

Let M_t be the submatrix of M visible to P_t . $M_0 = M$. If t is odd, then M_t has $n_t = \lfloor 2^{-t/2}(n - 2^{t/2} - 1) \rfloor$ rows and $m_t = 2n_t$ columns, and the length of its i^{th} row is $2i$. If $t \geq 2$ is even, the M_t has $n_t = n_{t-1}$ rows and $m_t = n_t$ columns, and the length of its i^{th} row is i .

The choice of M_{t+1} as a submatrix of M_t is illustrated below. If t is even, then the rows of M_{t+1} are rows $3, 5, 7, \dots, \lfloor \frac{n-1}{2} \rfloor$ of M_t , ie. all rows of odd index other than 1. Row i of M_{t+1} consists of all but the last entry of Row $2i + 1$ of M_t .

If t is odd, then M_{t+1} has the same number of rows as M_t , but only half the columns. Since there are n_t rows in M_t , the number of possible columns of M_t which contain values of J cannot exceed n_t . The REDUCE procedure of LARSCH chooses exactly n_t columns of M_t in a way that makes certain that all columns which contain row minima of M_t are chosen. The figure below illustrates one possible choice ([Fig. 19](#)). The shaded entries are passed to M_{t+1} . Note that not every entry of a selected column is part of M_{t+1} .

Table 5 LARSCH algorithm example: INTERPOLATE and REDUCE

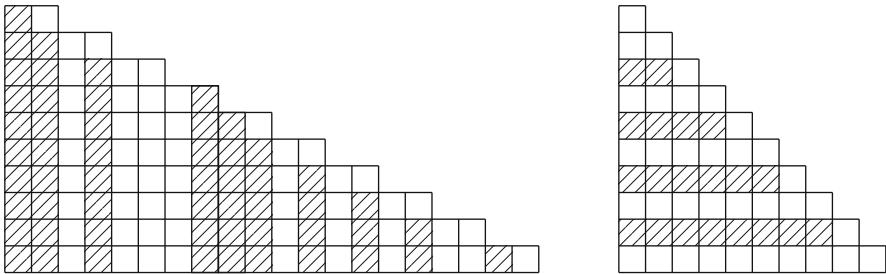


Fig. 19 Shaded entries of M_t indicate the submatrix M_{t+1} passed to the child process. *Left:* odd t . *Right:* even t

6.4 Standard Type Process: P_t for t Even (INTERPOLATE)

Code for a Process of Standard Type: P_t for t Even.

```

1: {Matrix  $M_t$  with  $n_t$  rows and  $n_t$  columns}
2: for  $i$  from 1 to  $n_t$  do
3:   Receive permission to view Column  $i - 1$  from supervisor.
4:   if  $i = 1$  then
5:      $J(1) \leftarrow 0$ 
6:   else if  $i$  is even and  $i < n_t$  then
7:     Grant permission to child to view Columns  $i - 2$  and  $i - 1$ .
8:     Receive  $J\_Partial$  from child.
9:     {Min of all entries but last of Row  $i + 1$  is in Column  $J\_Partial$ .}
10:    Find  $J(i) \in [J(i - 1), J\_Partial]$  by linear search.
11:   else if  $i$  is even and  $i = n_t$  then
12:     Find  $J(n_t) \in [J(i - 1), n_t - 1]$  by linear search.
13:   else [ $i \geq 3$  is odd]
14:     if  $M_t[i, i - 1] < M_t[i, J\_Partial]$  then
15:        $J(i) \leftarrow i - 1$ 
16:     else
17:        $J(i) \leftarrow J\_Partial$ 
18:     end if
19:   end if
20:   Send  $J(i)$  to supervisor. {Min of Row  $i$  is in Column  $J(i)$ }
21: end for

```

6.5 Standard Type Process: P_t for t Odd (REDUCE)

A stack called the *column stack* is maintained. Let *top* be the number of items (which are column indices) stored in the stack, and let $S[i]$ be the i th entry (counting from the bottom) of the stack. That is, $S[i]$ is defined for all $1 \leq i \leq top$.

Each time process P_t is able to view a new column, if the column stack is not empty, it pops all columns which are *dominated* by the new column off the stack, and then pushes the new column. The domination rule is that Column $S[i]$ is dominated by the new column, say Column j , if $M_t[i, j] < M_t[i, S[i]]$. Of course, it is possible that $j \geq 2i$, in which case $M_t[i, j]$ is taken to be infinity.

Code for a Process of Stretched Type: P_t for t Odd.

```

1: {Matrix  $M_t$  with  $n_t$  rows and  $2n_t$  columns}
2:  $top \leftarrow 0$       {Initialize the column stack to empty}
3: for  $i$  from 1 to  $n_t$  do
4:   Receive permission to view Columns  $2i - 2$  and  $2i - 1$  from supervisor.
5:   Pop all columns dominated by Column  $2i - 2$  from column stack.
6:   Push Column  $2i - 2$  onto column stack.
7:   Pop all columns dominated by Column  $2i - 1$  from column stack.
8:   Push Column  $2i - 1$  onto column stack.
9:   Grant permission to child to view Column  $S[i]$ .
10:  {The  $i^{\text{th}}$  column on the stack, which will become Column  $i - 1$  of  $M_{t+1}$ }
11:  Receive  $J(i)$  from child.    {Min of Row  $i$  is in Column  $J(i)$ }
12:  Send  $J(i)$  to supervisor.
13: end for
```

How may a new column possibly dominate existing columns on the column stack? Lines 2, 3, and 4 of the code below are the detail of Line 5 and Line 7 of the code above, while Lines 5 and 6 are the detail of Line 6 and Line 8 of the code above.

Code for Popping and Pushing the Column Stack.

```

1: {Column  $j$  is the new column}
2: while  $top > 0$  and  $2 \cdot top > j$  and  $M_t[top, j] < M_t[top, S[top]]$  do
3:    $top \leftarrow top - 1$       {Pop off the dominated column}
4: end while
5:  $top \leftarrow top + 1$ 
6:  $S[top] \leftarrow j$       {Push the new column}
```

7 The Quadrangle Inequality and Binary Search Trees

Another type of speedup is based in the Knuth-Yao quadrangle inequality. This section discusses this kind of speedup and the relation with SMAWK/LARSCH speedup.

7.1 Background

Recall construction of optimal binary search trees discussed in Sect. 2.5. Gilbert and Moore [59] gave a $O(n^3)$ time algorithm. More than a decade later, in 1971,

it was noticed by Knuth [71] that, using a complicated amortization argument, the $B_{i,j}$ can all be computed using only $O(n^2)$ time. Around another decade later, in the early 1980s, Yao (see [96, 97]) simplified Knuth's proof and, in the process, showed that this *dynamic programming speedup* worked for a large class of problems satisfying a *quadrangle inequality* property. Many other authors then used the Knuth-Yao technique, either implicitly or explicitly, to speed up different dynamic programming problems. For this see for example the work of Wessner [93], the work of Atallah, Kosaraju, Larmore, Miller, and Teng [9] and Bar-No and Ladner [14].

Even though both the SMAWK algorithm and the Knuth-Yao (KY) speedup (best described in [71, 96, 97]) use an implicit quadrangle inequality in their associated matrices, on second glance, they seem quite different from each other. In the SMAWK technique, the quadrangle inequality is on the entries of a given $m \times n$ *input* matrix, which can be any totally monotone matrix. The KY technique, by contrast, uses a quadrangle inequality in the upper-triangular $n \times n$ matrix B . That is, it uses the QI property of its *result* matrix to speed up the evaluation, via dynamic programming, of the entries in the same result matrix. Aggarwal and Park [2] demonstrated a relationship between the KY problem and totally-monotone matrices by building a *3-D monotone matrix* based on the KY problem and then using an algorithm due to Wilber [94] to find *tube* minima in that 3-D matrix. They left as an open question the possibility of using SMAWK directly to solve the KY problem.

Definition 4 A two dimensional upper triangular matrix A , $0 \leq i \leq j \leq n$ satisfies the *quadrangle inequality* (QI) if for all $i \leq i' \leq j \leq j'$,

$$A(i, j) + A(i', j') \leq A(i', j) + A(i, j'). \quad (23)$$

Observation 2 A Monge matrix satisfies a quadrangle inequality, but a totally monotone matrix may not.

Yao's result (of [96]) was formulated as follows: For $0 \leq i \leq j \leq n$ let $w(i, j)$ be a given function and

$$B_{i,j} = \begin{cases} 0, & \text{if } i = j; \\ w(i, j) + \min_{i < t \leq j} (B_{i,t-1} + B_{t,j}), & \text{if } i < j. \end{cases} \quad (24)$$

Definition 5 $w(i, j)$ is *monotone in the lattice of intervals* if $[i, j] \subseteq [i', j']$ implies $w(i, j) \leq w(i', j')$.

As an example, it is not difficult to see that the $w(i, j) = \sum_{l=i+1}^j p_l + \sum_{l=i}^j q_l$ of the BST recurrence (12) satisfies the quadrangle inequality and is monotone in the lattice of intervals.

Definition 6 Let

$$K_B(i, j) = \max\{t : w(i, j) + B_{i,t-1} + B_{t,j} = B_{i,j}\},$$

i.e., the largest index which achieves the minimum in (24).

Yao then proves two Lemmas (see Fig. 21 for an example):

Lemma 6 (Lemma 2.1 in [96]) *If $w(i, j)$ satisfies the quadrangle inequality as defined in Definition 4, and is also monotone on the lattice of intervals, then the $B_{i,j}$ defined in (24) also satisfy the quadrangle inequality.*

Lemma 7 (Lemma 2.2 in [96]) *If the function $B_{i,j}$ defined in (24) satisfies the quadrangle inequality then*

$$K_B(i, j) \leq K_B(i, j+1) \leq K_B(i+1, j+1) \quad \forall i < j.$$

[Lemma 6](#) proves that a QI in the $w(i, j)$ implies a QI in the $B_{i,j}$. Suppose then that one evaluates the values of the $B_{i,j}$ in the order $d = 1, 2, \dots, n$, where, for each fixed d , one evaluates all of $B_{i,i+d}$, $i = 0, 1, \dots, n-d$. Then [Lemma 7](#) says that $B_{i,i+d}$ can be evaluated in time $O(K_B(i+1, i+d) - K_B(i, i+d-1))$. Note that

$$\sum_{i=0}^{n-d} (K_B(i+1, i+d) - K_B(i, i+d-1)) \leq n,$$

and thus all entries for fixed d can be calculated in $O(n)$ time. Summing over all d , it follows that all $B_{i,j}$ can be obtained in $O(n^2)$ time.

As mentioned, [Lemma 7](#) and the resultant $O(n^2)$ running time have long been viewed as unrelated to the SMAWK algorithm. While they seem somewhat similar (a QI leading to an order of magnitude speedup) they appeared not to be directly connected until very recently. In the next section it is shown how to solve the Knuth-Yao problem directly using decompositions into total monotone matrices.

7.2 Decomposition Techniques

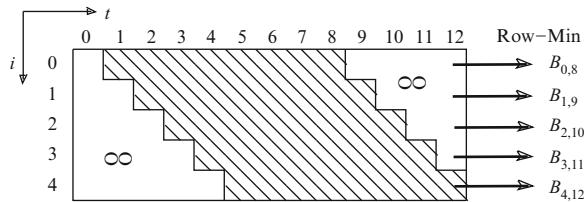
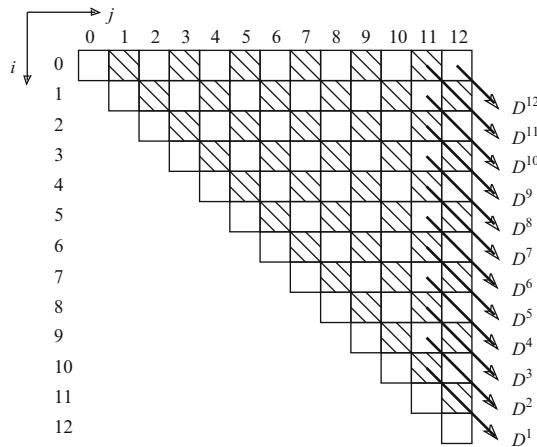
Definition 7 For $1 \leq d \leq n$ define the $(n-d+1) \times (n+1)$ matrix D^d by

$$D_{i,t}^d = \begin{cases} w(i, i+d) + B_{i,t-1} + B_{t,i+d}, & \text{if } 0 \leq i < t \leq i+d \leq n; \\ \infty & \text{otherwise.} \end{cases} \quad (25)$$

[Figure 20](#) illustrates a first decomposition. Note that (24) immediately implies

$$B_{i,i+d} = \min_{0 \leq t \leq n} D_{i,t}^d \quad (26)$$

Fig. 20 The top figure on shows the $B_{i,j}$ matrix for $n = 12$. Each diagonal, $d = j - i$, in the matrix will correspond to a totally monotone matrix D^d . The minimal item of row i in D^d will be the value $B_{i,i+d}$. The other figure shows D^8



so finding the row-minima of D^d yields $B_{i,i+d}$, $i = 0, \dots, n-d$. Put another way, the $B_{i,j}$ entries on diagonal $d = j - i$ are exactly the row-minima of matrix D^d .

Lemma 8 If $w(i, j)$ and the function $B_{i,j}$ defined in (24) satisfies the QI then, for each d ($1 \leq d \leq n$), D^d is a totally monotone matrix.

Proof It suffices to prove that

$$D_{i,t}^d + D_{i+1,t+1}^d \leq D_{i+1,t}^d + D_{i,t+1}^d \quad (27)$$

Note that if $i+1 < t < i+d$, then from Lemma 6,

$$B_{i,t-1} + B_{i+1,t} \leq B_{i+1,t-1} + B_{i,t} \quad (28)$$

and

$$B_{t,i+d} + B_{t+1,i+1+d} \leq B_{t+1,i+d} + B_{t,i+1+d}. \quad (29)$$

Thus,

$$\begin{aligned}
& D_{i,t}^d + D_{i+1,t+1}^d \\
&= [w(i, i+d) + B_{i,t-1} + B_{t,i+d}] + [w(i+1, i+1+d) + B_{i+1,t} + B_{t+1,i+1+d}] \\
&= w(i, i+d) + w(i+1, i+1+d) + [B_{i,t-1} + B_{i+1,t}] + [B_{t,i+d} + B_{t+1,i+1+d}] \\
&\leq w(i, i+d) + w(i+1, i+1+d) + [B_{i+1,t-1} + B_{i,t}] + [B_{t+1,i+d} + B_{t,i+1+d}] \\
&= [w(i+1, i+1+d) + B_{i+1,t-1} + B_{t,i+1+d}] + [w(i, i+d) + B_{i,t} + B_{t+1,i+d}] \\
&= D_{i+1,t}^d + D_{i,t+1}^d
\end{aligned}$$

and (27) is correct (where it is noted that the right hand side is ∞ if $i+1 \not< t$ or $t \not< i+d$). \square

Lemma 9 Assuming that all of the row-minima of D^1, D^2, \dots, D^{d-1} have already been calculated, all of the row-minima of D^d can be calculated using the SMAWK algorithm in $O(n)$ time.

Proof From the previous lemma, D^d is a totally monotone matrix. Also, by definition, its entries can be calculated in $O(1)$ time, using the previously calculated row-minima of $D^{d'}$ where $d' < d$. Thus SMAWK can be applied. \square

Combined with (26) this immediately gives a new $O(n^2)$ algorithm for solving the KY problem; just run SMAWK on the D^d in the order $d = 1, 2, \dots, n$ and report all of the row-minima.

7.3 Online Decomposition

The online problem restricted to the optimal binary search tree would be to construct the OBST for items $\text{Key}_{L+1}, \dots, \text{Key}_R$, and, at each step, add either Key_{R+1} , a new key to the right, or Key_L , a new key to the left. Every time a new element is added, it is desired to update the $B_{i,j}$ (dynamic programming) table and thereby construct the optimal binary search tree of the new full set of elements. (See Fig. 21.)

KY speedup *cannot* be used to do this. The reason that the speedup fails is that the KY speedup is actually an amortization over the evaluation of all entries when done in a particular order. In the online case, adding a new item n to previously existing items $1, 2, \dots, n-1$ requires using (24) to compute the n new entries $B_{i,n}$, in the fixed order $i = n, n-1, \dots, 1, 0$ and it is not difficult to construct an example in which calculating these new entries in this order using (24) requires $O(n^2)$ work.

Neither can the decomposition technique from the previous section solve the online problem. To see why, suppose that items $1, \dots, n$ have previously been given, new item $n+1$ has just been added, and one needs to calculate the values $B_{i,n+1}$ for $i = 0, \dots, n+1$. In this formulation it would correspond to adding a new bottom row to *every* matrix D^d and creating a new matrix D^{n+1} and one would need to

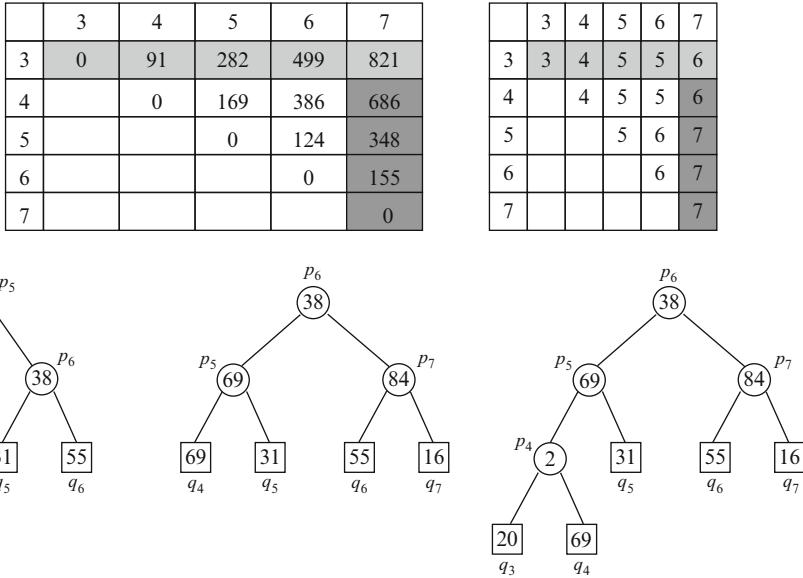


Fig. 21 An example of the online case for optimal binary search trees where $(p_4, p_5, p_6, p_7) = (2, 69, 38, 84)$ and $(q_3, q_4, q_5, q_6, q_7) = (20, 69, 31, 55, 16)$. The left table contains the $B_{i,j}$ values; the right one, the $K_B(i, j)$ values. The unshaded entries in the table are for the problem restricted to only keys 5, 6. The dark gray cells are the entries added to the table when key 7 is added to the right. The light gray cells are the entries added when key 4 is added to the left. The corresponding optimal binary search trees are also given, where circles correspond to successful searches and squares to unsuccessful ones. The values in the nodes are the weights of the nodes (not their keys)

find the row-minima of all of the n new bottom rows. Unfortunately, the SMAWK algorithm only works on the rows of matrices all at once and cannot help to find the row-minima of a single new row.

Note that for the online problem it is certainly possible to *recompute* the entire table; however this comes at the price of $O(n^2)$ time, where $n = R - L$ is the number of keys currently in the table, leading to a total running time of $O(n^3)$ to insert all of the keys. Of interest here is the question of whether one can maintain the speedup while inserting the keys in an online fashion. The goal is an algorithm in which a sequence of n online key insertions will result in a worst case $O(n)$ per step to maintain an optimal tree, yielding an overall run time of $O(n^2)$ (Fig. 22).

To this end now a second decomposition. It is indexed by the *leftmost* element seen so far. See Fig. 23.

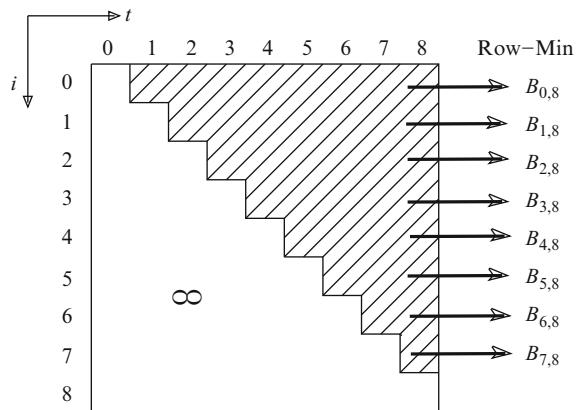
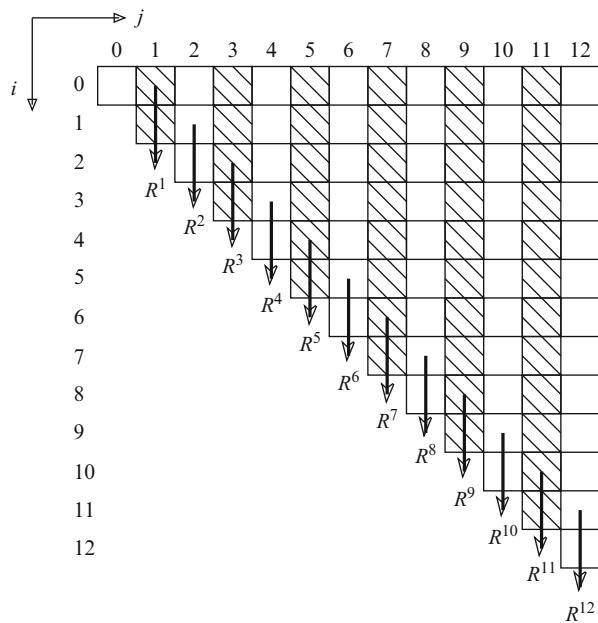
Definition 8 For $0 \leq i < n$ define the $(n - i) \times (n - i)$ matrix L^i by

$$L_{j,t}^i = \begin{cases} w(i, j) + B_{i,t-1} + B_{t,j}, & \text{if } i < t \leq j \leq n; \\ \infty, & \text{otherwise.} \end{cases} \quad (30)$$

Fig. 22 The top figure shows the $B_{i,j}$ matrix for $n = 12$.

Each column in the $B_{i,j}$ matrix will correspond to a totally monotone matrix R^j .

The minimal element of row i in R^j will be the value $B_{i,j}$.
The other figure shows R^8



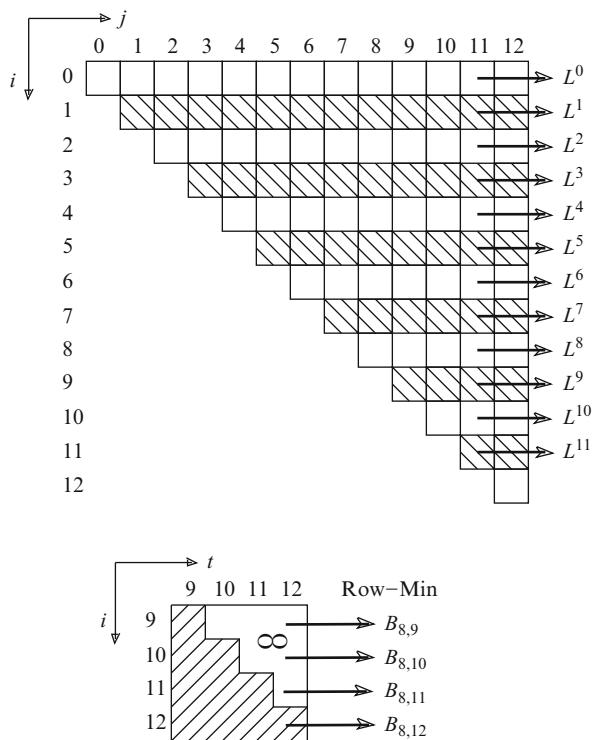
(For convenience, set the row and column indices to run from $(i + 1) \dots n$ and not $0 \dots (n - i - 1)$.) Note that (24) immediately implies

$$B_{i,j} = \min_{i < t \leq n} L_{j,t}^i \quad (31)$$

so finding the row-minima of L^i yields $B_{i,j}$ for $j = i + 1, \dots, n$. Put another way, the $B_{i,j}$ entries in row i are exactly the row minima of matrix L^i .

Lemma 10 If the function defined in (24) satisfies the QI then R^j (resp. L^i) are totally monotone matrices for each fixed j (resp. i).

Fig. 23 The top figure on the left shows the $B_{i,j}$ matrix for $n = 12$. Each row in the $B_{i,j}$ matrix will correspond to a totally monotone matrix L^i . The minimal element of row j in L^i will be the value $B_{i,j}$. The other figure shows L^8 .



Proof The proofs are very similar to that of Lemma 8. To prove R^j is totally monotone, note that if $i + 1 < t < j$, one can again use (28); writing the entries from (28) in boldface gives

$$\begin{aligned} & R_{i,t}^j + R_{i+1,t+1}^j \\ &= [w(i, j) + \mathbf{B}_{i,t-1} + B_{t,j}] + [w(i+1, j) + \mathbf{B}_{i+1,t} + B_{t+1,j}] \\ &\leq [w(i+1, j) + \mathbf{B}_{i+1,t-1} + B_{t,j}] + [w(i, j) + \mathbf{B}_{i,t} + B_{t+1,j}] \\ &= R_{i+1,t}^j + R_{i,t+1}^j \end{aligned}$$

and thus R^j is Monge (where the right hand side is ∞ if $i + 1 \not\prec t$) and thus totally monotone. To prove L^i is totally monotone, if $i < t < j$ then again use (28) (with j replaced by $j + 1$) to get

$$\begin{aligned} & L_{j,t}^i + L_{j+1,t+1}^i \\ &= [w(i, j) + B_{i,t-1} + \mathbf{B}_{t,j}] + [w(i, j+1) + B_{i,t} + \mathbf{B}_{t+1,j+1}] \\ &\leq [w(i, j+1) + B_{i,t-1} + \mathbf{B}_{t,j+1}] + [w(i, j) + B_{i,t} + \mathbf{B}_{t+1,j}] \\ &= L_{j+1,t}^i + L_{j,t+1}^i \end{aligned}$$

and thus L^i is Monge (where the right hand side is ∞ if $t \not\prec j$) and thus totally monotone. \square

Note that this decomposition immediately imply a new proof of [Lemma 7](#) (Lemma 2.2 in [96]) which states that

$$K_B(i, j) \leq K_B(i, j + 1) \leq K_B(i + 1, j + 1). \quad (32)$$

To see this note that $K_B(i, j + 1)$ is the location of the rightmost row-minimum of row i in matrix R^{j+1} , while $K_B(i + 1, j + 1)$ is the location of the rightmost row-minimum of row $i + 1$ in matrix R^{j+1} . Thus, the definition of total monotonicity immediately gives

$$K_B(i, j + 1) \leq K_B(i + 1, j + 1). \quad (33)$$

Similarly, $K_B(i, j)$ is the rightmost row-minimum of row j in L^i while $K_B(i, j + 1)$ is the location of the rightmost row-minimum of row $j + 1$ in L^i . Thus

$$K_B(i, j) \leq K_B(i, j + 1). \quad (34)$$

Combining (33) and (34) yields (32). Since the actual speedup in the KY technique comes from an amortization argument based on (32), it follows that the original KY-speedup itself is also a consequence of total monotonicity.

Up to this point it is not clear how to actually calculate the $B_{i,j}$ using the R^j and L^i . Note first that even though the R^j are totally monotone, their row minima *cannot* be calculated using the SMAWK algorithm. This is because, for $0 \leq i < t \leq j$, the value of entry $R_{i,t}^j = w(i, j) + B_{i,t-1} + B_{t,j}$, which is dependent upon $B_{t,j}$ which is itself the row-minimum of row t in the *same* matrix R^j . Thus, the values of the entries of R^j depend upon other entries in R^j which is something that SMAWK does not allow. The same problem occurs with the L^i .

But despite this dependence, the LARSCH algorithm can still be used to find the row-minima of the R^j . Recall that to execute the LARSCH algorithm one needs only that the matrix X satisfy the following conditions:

1. X is an $n \times m$ totally monotone matrix.
2. For each row index i of X , there is a column index C_i such that for $j > C_i$, $X_{i,j} = \infty$. Furthermore, $C_i \leq C_{i+1}$.
3. If $j \leq C_i$, then $X_{i,j}$ can be evaluated in $O(1)$ time *provided that the row minima of the first $i - 1$ rows are already known*.

If these conditions are satisfied, the LARSCH algorithm then calculates all of the row minima of X in $O(n + m)$ time. This algorithm can now be used to derive

Lemma 11

- Given that all values $B_{i',j}$, $i' < i' \leq j \leq n$ have already been calculated, all of the row-minima of L^i can be calculated in $O(n - i)$ time.
- Given that all values $B_{i,j'}$, $0 \leq i \leq j' < j$ have already been calculated, all of the row-minima of R^j can be calculated in $O(j)$ time.

Proof For the first part, it is easy to see that L^i satisfies the first two conditions required by the LARSCH algorithm with $C_j = j$. For the third condition, note that, for $i < t \leq j$, $L_{j,t}^i = w(i, j) + B_{i,t-1} + B_{t,j}$. The values $w(i, j)$ and $B_{t,j}$ are already known and can be retrieved in $O(1)$ time. $B_{i,t-1}$ is the minimum of row $t - 1$ of L^i but, since we are assuming $t \leq j$, this means that $B_{i,t-1}$ is the minimum of an earlier row in L^i , and the third LARSCH condition is satisfied. Thus, all of the row-minima of the $(n - i) \times (n - i)$ matrix L^i can be calculated in $O(n - i)$ time.

For the second part set X to be the $(j + 1) \times (j + 1)$ matrix defined by $X_{i,t} = R_{j-i,j-t}^j$. Then X satisfies the first two LARSCH conditions with $C_i = i - 1$. For the third condition note that $X_{i,t} = R_{j-i,j-t}^j = w(j - i, j) + B_{j-i,j-t-1} + B_{j-t,j}$. The values $w(j - i, j)$ and $B_{j-i,j-t-1}$ are already known and can be calculated in $O(1)$ time. $B_{j-t,j}$ is the row minima of row t of X ; but, since we are assuming $t \leq C_i = i - 1$ this means that $B_{j-t,j}$ is the row minima of an earlier row in X so the third LARSCH condition is satisfied. Thus, all of the row-minima of X and equivalently R^j can be calculated in $O(j)$ time. \square

Note that [Lemma 11](#) immediately solves the “right-online” and “left-online” problems. Given the new values $w(i, R + 1)$ for $L \leq i \leq R + 1$, simply find the row minima of R^{R+1} in time $O(R - L)$. Given the new values $w(L - 1, j)$ for $L - 1 \leq j \leq R$, simply find the row minima of R^{L-1} . Therefore it was just shown that *any* dynamic programming problem for which the KY speedup can statically improve run time from $O(n^3)$ to $O(n^2)$ time can be solved in an online fashion in $O(n)$ time per step. That is, online processing incurs no penalty compared to static processing. In particular, the optimum binary search tree can be maintained in $O(n)$ time per step as nodes are added to both its left and right.

At this point note that decompositions L^i could also be derived by careful cutting of the 3-D monotone matrices of Aggarwal and Park [2] along particular planes. Aggarwal and Park [2] used an algorithm of Wilber [94] (derived for finding the maxima of certain concave-sequences) to find various tube maxima of their matrices, leading to another $O(n^2)$ algorithm for solving the KY-problem. In fact, even though their algorithm was presented as a static algorithm, careful decomposition of what they do permits using it to solve what is called here the left-online KY-problem. A symmetry argument could then yield a right-online algorithm. This never seems to have been noted in the literature, though.

8 Conclusion

Too small an academic community is aware of the many advanced tools available related to dynamic programming. Routinely, applications are solved by simple-minded dynamic programs, where much faster solutions are possible. In fact, for many massively large problems arising in Molecular Biology, for example, a quadratic solution might be completely useless, equivalent to no solution at all.

It is the hope of the author that this book chapter will open up some of these advanced techniques to a larger community of scientists.

As well the use of dynamic programming in online optimization and by extension the concept of work functions is not as widely embraced as is desirable. Many online algorithms are ad-hoc and work functions make it possible to “de-adhocify” the construction of competitive and efficient algorithms in this setting. The very recent concept of *knowledge states* (see [23]), which has dynamic programming at its core, makes it possible to derive online algorithms even in the randomized case in a systematic way.

Readers are invited to go beyond the obvious when using dynamic programming and to avail themselves of these powerful techniques.

Further Reading

Introductions to dynamic programming with numerous elementary examples can be found in the textbooks by Brassard and Gilles [32], Cormen, Leiserson, Rivest, and Stein [45], Baase and van Gelder [10] and Dasgupta, Papadimitriou, and Vazirani [47].

The classical treatises by Bellmann [24] and [25] are still relevant today though the language is somewhat antiquated and the applications outdated. Other general reading is Dreyfus and Law [50], Stokey, Lucas, and Prescott [88], Bertsekas [28], Denardo [48], Meyn [83], and more recently Sniedovich [87]. One classical algorithm worth mentioning is the algorithm by Viterbi [90] for Hidden Markov Model inference. Many problems in areas such as digital communications can be cast in the Hidden Markov Model. Another classic is Hu and Tucker [65] on alphabetic trees. Bellmore and Nemhauser [27] as well as Lawler, Lenstra, Rinnoy Kan, and Shmoys [79] and Burkard, Deineko, Dal, van der Veen, and Woeginger [39] contain material regarding the use of dynamic programming for the traveling salesman problem. Gusfield [60] is good general resource on dynamic programming for computational biology. Examples with regards to scheduling can be found in Bellman, Esogbue, and Nabeshima [26] as well as in Brucker [33] and in Brucker and Knust [35]. David Eppstein, Zvi Galil, Raffaele Giancarlo, and Giuseppe F. Italiano [52, 53] give a two paper sequence in the Journal of the ACM for sparse dynamic programming.

Regarding online algorithms (discussed in Sect. 3) the book by Borodin and El-Yanif [30] is the standard text. Of note is also the monograph by Karlin [66]. The article Larmore and Chrobak [42], which introduced work functions, extends the material in Sect. 3. For further immersion into the realm of work functions the article by Bein, Chrobak, and Larmore [16] is recommended. Recent work by Bein, Larmore, Noga, and Reischuk [23] on knowledge states has extended the use of work functions to randomized online algorithms: such algorithms are “guided” in their operation by work functions.

Some of the material presented in Sect. 4 can be found in greater detail in Bein, Larmore, Morales, and Sudborough [22]. Further reading regarding this section:

Sorting problems under various operations have been studied extensively, including work on sorting with prefix reversals Gates and Papadimitriou [58] (Gates of Microsoft fame) as well as Heydari and Sudborough [62], transpositions [11] and block moves ([17, 80] as well as Mahajan, Rama, and Vijayakumar [81]).

Some of the material presented in Sect. 5 can found in greater detail in Bein, Noga, and Wiegley [19]. The paper by Brucker and Albers [6] contains an alternate linear time algorithm for list batching. There is a large body of work on dynamic programming and batching; see the work of Baptiste [12], Baptiste and Jouplet [13], Brucker, Gladky, Hoogeveen, Kovalyov, Potts Tautenhahn, and van de Velde [36], Brucker, Kovalyov, Shafransky, and Werner [37], Hoogeveen, and Vestjens [64], as well as the scheduling text book by Peter Brucker [33]. Batching has wide application in manufacturing (see, e.g., [34, 85, 98]), decision management (see, e.g., [74]), and scheduling in information technology (see, e.g., [46]). More recent work on online batching is related to the TCP (Transmission Control Protocol) acknowledgment problem (see [20, 49, 67]).

Regarding Monge properties and total monotonicity the best resource is Park's thesis, [84]. Another excellent survey on Monge properties is Burkard, Klinz and Rudolf [38]. A generalization of the Monge property to an algebraic property is in Bein, Brucker, Larmore, and Park [18]. Interesting applications are in Woeginger [95]. Burkard, Deineko, and Woeginger [40] survey the traveling salesman problem on Monge matrices. Agarwal and Sen [1] give results for selection in monotone matrices for computing k th nearest neighbors. Schieber [86] gives results on k -link paths in graphs with Monge properties.

Section 6 is based on Agarwal, Klawe, Moran, Shor, and Wilber [3] and Larmore and Schieber [78]. Generalizations of the matrix searching techniques are in Klawe [69] and Klawe and Kleitman [70], as well as in Kravets and Park [73], Aggarwal and Park [5] (Parallel Searching). and Wilber [94]. A good survey paper is Galil and Park [57].

An extended version of the material of Sect. 7 can be found in Bein, Larmore, Golin, and Zhang [21]. For Sect. 7 the papers of Knuth [71] followed by Yao [97] are key. Aggarwal and Park used an algorithm of Wilber [94] to find various tube maxima of their matrices, leading to another $O(n^2)$ algorithm for solving the KY-problem. There are two extensions of the Knuth-Yao quadrangle inequality: the first is due to Wachs [91] and the second to Borchers and Gupta (BG) [29]. This is discussed in detail in Bein, Larmore, Golin, and Zhang [21]. Belatedly, Aggarwal, Bar-Noy, Khuller, Kravets, and Schieber [4] describe solutions for matching using the quadrangle inequality.

The main gist of this chapter is dynamic programming speedup: Applications abound. Classic examples include the work by Hirschberg and Larmore [63] on the weight subsequence problem, Larmore and Przytycka [76] on parallel construction of trees with optimal path length, and Larmore and Hirschberg [75] on length limited coding. David Eppstein [51] considers sequence comparisons. Apostolico, Atallah, Larmore, and McFaddin [7] give algorithms for string editing. Highly recommended is the paper by Galil and Giancarlo [56] on speeding up dynamic programming in Molecular Biology. Work by Arslan and Egecioglu [8] is on sequence alignment

Bradford, Golin, Larmore, and Rytter [31] give dynamic programs for optimal prefix-free codes. Recent speedup work is for the online maintenance of k -medians by Fleischer, Golin, and Zhang [54] (see also [61, 89]).

Acknowledgements This chapter is dedicated to Lawrence L. Larmore, a great mentor. A sabbatical (academic year 2006/07) granted by the University of Nevada, Las Vegas, which benefited this book chapter, is acknowledged.

Cross-References

- [Advances in Scheduling Problems](#)
 - [Computing Distances between Evolutionary Trees](#)
 - [Efficient Algorithms for Geometric Shortest Path Query Problems](#)
 - [Geometric Optimization in Wireless Networks](#)
 - [Online and Semi-online Scheduling](#)
 - [Resource Allocation Problems](#)
-

Recommended Reading

1. P.K. Agarwal, S. Sen, Selection in monotone matrices and computing k th nearest neighbors. *J. Algorithms* **20**(3), 581–601 (1996); A preliminary version appeared, in *Proceedings of the 4th Scandinavian Workshop on Algorithm Theory* (1994), pp. 13–24
2. A. Aggarwal, J.K. Park, Notes on searching in multidimensional monotone arrays, in *Proceedings of the 29th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Washington, DC, 1988), pp. 497–512
3. A. Aggarwal, M.M. Klawe, S. Moran, P.W. Shor, R.E. Wilber, Geometric applications of a matrix-searching algorithm. *Algorithmica* **2**(1), 195–208 (1987); A preliminary version appeared, in *Proceedings of the 2nd Annual Symposium on Computational Geometry* (1986), pp. 285–292
4. A. Aggarwal, A. Bar-Noy, S. Khuller, D. Kravets, B. Schieber, Efficient minimum cost matching and transportation using the quadrangle inequality. *J. Algorithms* **19**(1), 116–143 (1995); A preliminary version appeared, in *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science* (1992), pp. 583–592
5. A. Aggarwal, D. Kravets, J.K. Park, S. Sen, Parallel searching in generalized Monge arrays. *Algorithmica* **19**(3), 291–317 (1997); A preliminary version appeared, in *Proceedings of the 2nd Annual ACM Symposium on Parallel Algorithms and Architectures* (1990), pp. 259–268
6. S. Albers, P. Brucker, The complexity of one-machine batching problems. *Discret. Appl. Math.* **47**, 87–10 (1993)
7. A. Apostolico, M. Atallah, L. Larmore, S. McFaddin, Efficient parallel algorithms for string editing and related problems. *SIAM J. Comput.* **19**(5), 968–988 (1990)
8. A.N. Arslan, O. Egecioglu, Dynamic programming based approximation algorithms for sequence alignment with constraints. *INFORMS J. Comput.* **16**(4), 441–458 (2004)
9. M.J. Atallah, S. Rao Kosaraju, L.L. Larmore, G.L. Miller, S-H. Teng, Constructing trees in parallel, in *Proceedings of the 1st Annual ACM Symposium on Parallel Algorithms and Architectures* (ACM, New York, 1989), pp. 421–431
10. S. Baase, A. van Gelder, *Computer Algorithms* (Addison Wesley, Reading, 2000)
11. V. Bafna, P.A. Pevzner, Sorting by transposition. *SIAM J. Discret. Math.* **11**, 224–240 (1998)
12. P. Baptiste, Batching identical jobs. *Math. Methods Oper. Res.* **52**, 355–367 (2000)

13. P. Baptiste, A. Jouplet, On minimizing total tardiness in a serial batching problem. *Oper. Res.* **35**, 107–115 (2001)
14. A. Bar-Noy, R.E. Ladner, Efficient algorithms for optimal stream merging for media-on-demand. *SIAM J. Comput.* **33**(5), 1011–1034 (2004)
15. W. Bein, M. Chrobak, L.L. Larmore, The 3-server problem in the plane, in *Proceedings of 7th European Symposium on Algorithms (ESA)*. Volume 1643 of Lecture Notes in Computer Science (Springer, Berlin/New York, 1999), pp. 301–312
16. W. Bein, M. Chrobak, L.L. Larmore, The 3-server problem in the plane. *Theoret. Comput. Sci.* **287**, 387–391 (2002)
17. W. Bein, L.L. Larmore, S. Latifi, I. Hal Sudborough, Block sorting is hard. *Int. J. Found. Comput. Sci.* **14**(3), 425–437 (2003)
18. W. Bein, P. Brucker, L.L. Larmore, J.K. Park, The algebraic Monge property and path problems. *Discret. Appl. Math.* **145**(3), 455–464 (2005)
19. W. Bein, J. Noga, J. Wiegley, Approximation for batching via priorities. *Sci. Ann. Comput. Sci.* **XVII**, 1–18 (2007)
20. W. Bein, L. Epstein, L.L. Larmore, J. Noga, Optimally competitive list batching. *Theoret. Comput. Sci.* **410**(38–40), 3631–3639 (2009)
21. W. Bein, M. Golin, L. Larmore, Y. Zhang, The Knuth-Yao quadrangle-inequality speedup is a consequence of total-monotonicity. *Trans. Algorithms* **6**(1) (2009)
22. W. Bein, L.L. Larmore, L. Morales, I. Hal Sudborough, A quadratic time 2-approximation algorithm for block sorting. *Theor. Comput. Sci.* **410**, 711–717 (2009)
23. W. Bein, L.L. Larmore, J. Noga, R. Reischuk, Knowledge state algorithms. *Algorithmica* **60**(3), 653–678 (2011)
24. R. Bellman, The theory of dynamic programming. *Bull. Am. Math. Soc.* **60**:503–516 (1954)
25. R. Bellman, *Dynamic Programming*, Dover Paperback edition 2003 edn. (Princeton University Press, Princeton, 1957)
26. R. Bellman, A.O. Esogbue, I. Nabeshima, *Mathematical Aspects of Scheduling and Applications* (Pergamon, Oxford/New York, 1982)
27. M. Bellmore, G.L. Nemhauser, The traveling salesman problem: A survey. *Oper. Res.* **16**(3), 538–558 (1968)
28. D.P. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd edn. (Athena Scientific, Belmont, 2000)
29. A. Borchers, P. Gupta, Extending the quadrangle inequality to speed-up dynamic programming. *Inf. Process. Lett.* **49**(6), 287–290 (1994)
30. A. Borodin, R. El-Yaniv, *Online Computation and Competitive Analysis* (Cambridge University Press, Cambridge/New York , 1998)
31. P.G. Bradford, M.J. Golin, L.L. Larmore, W. Rytter, Optimal prefix-free codes for unequal letter costs: Dynamic programming with the Monge property. *J. Algorithms* **42**(2), 277–303 (2002); A preliminary version appeared, in *Proceedings of the 6th Annual European Symposium on Algorithms* (1998), pp. 43–54
32. G. Brassard, P. Bratley, *Fundamentals of Algorithms* (Prentice Hall, Englewood, 1996)
33. P. Brucker, *Scheduling Algorithms* 5th edn. (Springer, Berlin/New York, 2007)
34. P. Brucker, J. Hurink, Solving a chemical batch scheduling problem by local search. *Ann. Oper. Res.* **96**, 17–38 (2000)
35. P. Brucker, S. Knust, *Complex Scheduling* (Springer, Berlin, 2006)
36. P. Brucker, A. Gladky, H. Hoogeveen, M. Kovalyov, C. Potts, T. Tautenhahn, S. van de Velde, Scheduling a batch processing machine. *J. Sched.* **1**(1), 31–54 (1998)
37. P. Brucker, M. Kovalyov, Y. Shafransky, F. Werner, Batch scheduling with deadline on parallel machines. *Ann. Oper. Res.* **83**, 23–40 (1998)
38. R.E. Burkard, B. Klinz, R. Rudolf, Perspectives of Monge properties in optimization. *Discret. Appl. Math.* **70**(2), 95–161 (1996)
39. R.E. Burkard, V.G. Deineko, R. van Dal, J.A.A. van der Veen, G.J. Woeginger, Well-solvable special cases of the TSP: A survey. *SIAM Rev.* **40**(3), 496–546 (1998)

40. R.E. Burkard, V.G. Deineko, G.J. Woeginger, The travelling salesman problem on permuted Monge matrices. *J. Comb. Optim.* **2**(4), 333–350 (1999)
41. M. Chrobak, L.L. Larmore, An optimal online algorithm for k servers on trees. *SIAM J. Comput.* **20**, 144–148 (1991)
42. M. Chrobak, L.L. Larmore, The server problem and on-line games, in *On-line Algorithms*, ed. by L.A. McGeoch, D.D. Sleator. Volume 7 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science (AMS/ACM, Providence, RI, 1992), pp. 11–64
43. M. Chrobak, L.L. Larmore, Metrical task systems, the server problem, and the work function algorithm, in *Online Algorithms: The State of the Art*, ed. by A. Fiat, G.J. Woeginger (Springer, Berlin/New York, 1998), pp. 74–94
44. M. Chrobak, H. Karloff, T.H. Payne, S. Vishwanathan, New results on server problems. *SIAM J. Discret. Math.* **4**, 172–181 (1991)
45. T. Cormen, C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edn. (McGraw Hill, New York, 2001)
46. A. Dan, D. Sitaran, P. Shahabuddin, Scheduling policies for an on-demand video server with batching, in *Proceedings of the second ACM international conference on Multimedia* (ACM, New York, 1994), pp. 15–23
47. S. Dasgupta, C. Papadimitriou, U. Vazirani, *Algorithms* (McGraw Hill, Boston, 2008)
48. E.V. Denardo, *Dynamic Programming: Models and Applications* (Dover, Mineola, 2003)
49. D.R. Dooley, S.A. Goldman, S.D. Scott, On-line analysis of the TCP acknowledgment delay problem. *J. ACM* **48**(2), 243–273 (2001)
50. S.E. Dreyfus, A.M. Law, *The art and theory of dynamic programming* (Academic, New York, 1977)
51. D. Eppstein, Sequence comparison with mixed convex and concave costs. *J. Algorithms* **11**(1), 85–101 (1990)
52. D. Eppstein, Z. Galil, R. Giancarlo, G.F. Italiano, Sparse dynamic programming I: Linear cost functions. *J. ACM* **39**(3), 519–545 (1992); A preliminary version appeared, in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, (1990), pp. 513–522
53. D. Eppstein, Z. Galil, R. Giancarlo, G.F. Italiano, Sparse dynamic programming II: Convex and concave cost functions. *J. ACM* **39**(3), 546–567 (1992); A preliminary version appeared, in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), pp. 513–522
54. R. Fleischer, M.J. Golin, Y. Zhang, Online maintenance of k -medians and k -covers on a line. *Algorithmica* **45**(4), 549–567 (2006); A preliminary version appeared, in *Proceedings of the 9th Scandinavian Workshop on Algorithm Theory* (2004), pp. 102–113
55. R.F. Floyd, Algorithm 97: Shortest path. *Commun. ACM* **5**(6), 345 (1962)
56. Z. Galil, R. Giancarlo, Speeding up dynamic programming with applications to molecular biology. *Theor. Comput. Sci.* **64**(1), 107–118 (1989)
57. Z. Galil, K. Park, Dynamic programming with convexity, concavity and sparsity. *Theor. Comput. Sci.* **92**(1), 49–76 (1992)
58. W.H. Gates, C.H. Papadimitriou, Bounds for sorting by prefix reversal. *Discret. Math.* **27**, 47–57 (1979)
59. E.N. Gilbert, E.F. Moore, Variable length encodings. *Bell Syst. Tech. J.* **38**, 933–967 (1959)
60. D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology* (Cambridge University Press, Cambridge, 1997)
61. R. Hassin, A. Tamir, Improved complexity bounds for location problems on the real line. *Oper. Res. Lett.* **10**(7), 395–402 (1991)
62. H. Heydari, I. Hal Sudborough, On the diameter of the pancake network. *J. Algorithms* **25**(1), 67–94 (1997)
63. D.S. Hirschberg, L.L. Larmore, The least weight subsequence problem. *SIAM J. Comput.* **16**(4), 628–638 (1987)
64. H. Hoogeveen, A. Vestjens, Optimal on-line algorithms for single-machine scheduling, in *Proceedings of 5th Conference Integer Programming and Combinatorial Optimization (IPCO)* (Springer Verlag, London, 1996), pp. 404–414

65. T.C. Hu, A.C. Tucker, Optimal computer search trees and variable-length alphabetical codes. *SIAM J. Appl. Math.* **21**(4), 514–532 (1971)
66. A. Karlin, On the performance on competitive algorithms in practice, in *Online Algorithms: The State of the Art*, ed. by A. Fiat, G.J. Woeginger (Springer, 1998), pp. 373–382
67. A. Karlin, C. Kenyon, D. Randall, Dynamic TCP acknowledgment and other stories about $e/(e-1)$. *Algorithmica* **36**(3), 209–224 (2003)
68. A. Karlin, M. Manasse, L. Rudolph, D. Sleator, Competitive snoopy caching. *Algorithmica* **3**, 79–119 (1988)
69. M.M. Klawe, Superlinear bounds for matrix searching problems. *J. Algorithms* **13**(1), 55–78 (1992); A preliminary version appeared; in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), pp. 485–493
70. M.M. Klawe, D.J. Kleitman, An almost linear time algorithm for generalized matrix searching. *SIAM J. Discret. Math.* **3**(1), 81–97 (1990)
71. D.E. Knuth, Optimum binary search trees. *Acta Inf.* **1**, 14–25 (1971)
72. E. Koutsoupias, C. Papadimitriou, On the k -server conjecture. *J. ACM* **42**, 971–983 (1995)
73. D. Kravets, J.K. Park, Selection and sorting in totally monotone arrays. *Theory Comput. Syst.* **24**(3), 201–220, (1991); A preliminary version appeared, in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), pp. 494–502
74. R. Kuik, M. Salomon, L.N. van Wassenhove, Batching decisions: structure and models. *Eur. J. Oper. Res.* **75**, 243–263 (1994)
75. L.L. Larmore, D.S. Hirschberg, A fast algorithm for optimal length-limited Huffman codes. *J. ACM* **37**(3), 464–473 (1990); A preliminary version appeared, in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), pp. 310–318
76. L.L. Larmore, T.M. Przytycka, Parallel construction of trees with optimal weighted path length, in *Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures* (ACM Press, New York, 1991), pp. 71–80
77. L.L. Larmore, B. Schieber, On-line dynamic programming with applications to the prediction of RNA secondary structure. *J. Algorithms* **12**, 490–515 (1991)
78. L.L. Larmore, B. Schieber, On-line dynamic programming with applications to the prediction of RNA secondary structure. *J. Algorithms* **12**(3), 490–515 (1991); A preliminary version appeared, in *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms* (1990), pp. 503–512
79. E. Lawler, J. Lenstra, A.H.G Rinnooy Kan, D. Shmoys (eds.), *The Traveling Salesman: A Guided Tour of Combinatorial Optimization* (Wiley, New York, 1985)
80. M. Mahajan, R. Rama, V. Raman, S. Vijayakumar, Approximate block sorting. *Int. J. Found. Comput. Sci.* **12**(2), 337–356 (2006)
81. M. Mahajan, R. Rama, S. Vijayakumar, Block sorting: a characterization and some heuristics. *Nord. J. Comput.* **14**(1), 25 (2007)
82. M. Manasse, L.A. McGeoch, D. Sleator, Competitive algorithms for server problems. *J. Algorithms* **11**, 208–230 (1990)
83. S. Meyn, *Control Techniques for Complex Networks* (Cambridge University Press, Cambridge, MA, 2007)
84. J.K. Park, The Monge array: an abstraction and its applications. PhD thesis, Massachusetts Institute of Technology, 1991
85. C.N. Potts, L.N. van Wassenhove, Integrating scheduling with batching and lot-sizing: A review of algorithms and complexity. *J. Oper. Res. Soc.* **43**, 395–406 (1992)
86. B. Schieber, Computing a minimum weight k -link path in graphs with the concave Monge property. *J. Algorithms* **29**(2), 204–222 (1998); A preliminary version appeared, in *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms* (1995), pp. 405–411
87. M. Sniedovich, *Dynamic Programming: Foundations and Principles*. (Taylor and Francis, Boca Raton, FL, 2010)
88. N. Stokey, R.E. Lucas, E. Prescott, *Recursive Methods in Economic Dynamics* (Harvard University Press, Cambridge, MA, 1989)

89. A. Tamir, An $O(pn^2)$ algorithm for the p -median and related problems on tree graphs. *Oper. Res. Lett.* **19**(2), 59–64 (1996)
90. A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* **13**, 260–269 (1967)
91. M.L. Wachs, On an efficient dynamic programming technique of F. F. Yao. *J. Algorithms* **10**(4), 518–530 (1989)
92. S. Warshall, A theorem on boolean matrices. *J. ACM* **9**(1) 11–12 (1962)
93. R.L. Wessner, Optimal alphabetic search trees with restricted maximal height. *Inf. Process. Lett.* **4**(4), 90–94 (1976)
94. R. Wilber, The concave least-weight subsequence problem revisited. *J. Algorithms* **9**(3), 418–425 (1988)
95. G.J. Woeginger, Monge strikes again: Optimal placement of web proxies in the Internet. *Oper. Res. Lett.* **27**(3), 93–96 (2000)
96. F.F. Yao, Efficient dynamic programming using quadrangle inequalities, in *Proceedings of the 12th Annual ACM Symposium on Theory of Computing* (ACM Press, New York, 1980), pp. 429–435
97. F.F. Yao, Speed-up in dynamic programming. *SIAM J. Matrix Anal. Appl.* **3**(4), 532–540 (1982)
98. G. Zhang, X. Cai, C.Y. Lee, C.K. Wong, Minimizing makespan on a single batch processing machine with nonidentical job sizes. *Naval Res. Logist.* **48**, 226–240 (2001)

Advances in Group Testing

Yongxi Cheng

Contents

1	Introduction	94
2	New Constructions of One- and Two-Stage Pooling Design	98
2.1	Preliminaries	98
2.2	One-Stage Pooling Designs	99
2.3	Two-Stage Pooling Designs	105
2.4	Probabilistic Pooling Designs	111
2.5	Conclusion and Future Studies	112
3	New Complexity Results on Nonunique Probe Selection	113
3.1	Preliminaries	114
3.2	Complexity of Minimal \bar{d} -Separable Matrix	115
3.3	Minimum \bar{d} -Separable Submatrix	118
3.4	Conclusion	120
4	Parameterized Complexity Results on NGT	121
4.1	Preliminaries	122
4.2	Proof of Theorem 8	125
4.3	Discussion	129
5	Upper Bounds on the Minimum Number of Rows of Disjunct Matrices	130
5.1	Upper Bounds on $t(d, n)$	131
5.2	New Upper Bounds on $t(d, n; z)$	134
6	Transformation from Error-Tolerant Separable Matrices to Error-Tolerant Disjunct Matrices	137
6.1	Main Results	138
6.2	Concluding Remarks	141
7	Conclusion	142
	Cross-References	142
	Recommended Reading	142

Y. Cheng

School of Management, Xi'an Jiaotong University, Xi'an, Shaanxi, People's Republic of China
e-mail: chengyx@mail.xjtu.edu.cn

Abstract

In *combinatorial group testing*, there are n items; each has an unknown binary status, *positive* (i.e., *defective*) or *negative* (i.e., *good*), and the number of positives is upper bounded by an integer d . Suppose there is some method to test whether a subset of items contains at least one positive or not. The test result is said to be positive if it indicates that the subset contains at least one positive item; otherwise, the test result is called negative. The problem is to resolve the status of every item using the minimum number of tests.

Group testing (GT) algorithms can be *adaptive* or *nonadaptive*. An adaptive algorithm conducts the tests one by one and allows to design later tests using the outcome information of all previous tests. A nonadaptive group testing (NGT) algorithm specifies all tests before knowing any test results, and the benefit is that all tests can be performed in parallel. For the above group testing problem, nonadaptive algorithms require inherently more tests than adaptive ones.

Though the research of group testing dates back to Dorfman's 1943 paper, a renewed interest in the subject occurred recently mainly due to the applications of group testing to the area of computational molecular biology. In applications of molecular biology, a group testing algorithm is called a *pooling design*, and the composition of each test is called a *pool*. While it is still important to minimize the number of tests, there are two other goals. First, in the biological setting, screening one pool at a time is far more expensive than screening many pools in parallel; this strongly encourages the use of nonadaptive algorithms. Second, DNA screening is error prone, so it is desirable to design *error-tolerant* algorithms, which can detect or correct some errors in the test results.

In this monograph, some recent algorithmic, complexity, and mathematical results on nonadaptive group testing (and on pooling design) are presented.

1 Introduction

In *combinatorial group testing*, there are n items; each has an unknown binary status, *positive* (i.e., *defective*) or *negative* (i.e., *good*), and the number of positives is upper bounded by an integer d . Suppose there is some method to test whether a subset of items contains at least one positive or not. The test result is said to be positive if it indicates that the subset contains at least one positive item; otherwise, the test result is called negative. The problem is to resolve the status of every item using the minimum number of tests.

Group testing (GT) algorithms can be *adaptive* or *nonadaptive*. An adaptive algorithm conducts the tests one by one and allows to design later tests using the outcome information of all previous tests. A nonadaptive group testing (NGT) algorithm must specify all tests before knowing any test results, and the benefit is that all tests can be performed in parallel. For the above group testing problem, nonadaptive algorithms require inherently more tests than adaptive ones. It is known

that any nonadaptive algorithm must use a number of $\Omega(\frac{d^2 \log n}{\log d})$ tests, and the best known nonadaptive algorithm uses $O(d^2 \log n)$ tests. In contrast, the best adaptive algorithm requires $O(d \log n)$ tests (see, e.g., [17]) in the worst case.

Pooling Design. Though the research of combinatorial group testing dates back to Dorfman's 1943 paper [15], probably the most important modern applications of group testing are in the area of computational molecular biology, in which one important subject is clone library screening [3, 17, 26]. In applications to molecular biology, a group testing procedure is called a *pooling design*, and the composition of each test is called a *pool*.

A DNA library consists of thousands of separate DNA *clones*. The basic task of DNA library screening is, for a collection of *probes*, to determine which clone from the library contains which probe. Given a probe, a clone is said to be positive if it contains the probe; otherwise, it is said to be negative. In practice, to identify all positive clones from a library, clones are often pooled together to be tested against each probe, since checking each clone-probe pair is expensive and usually only a few clones in the library contain a given probe. An example is when sequenced-tagged site markers (also called STS probes) are used [46]. In practice, there are experimental tests, for example, the polymerase chain reaction, which can determine in a given pool whether or not there exists at least one clone containing a given probe.

In applications to molecular biology, while it is still important to minimize the number of tests, there are two other goals. First, in the biological setting, screening one pool at a time is far more expensive than screening many pools in parallel; this strongly encourages the use of nonadaptive algorithms. Second, DNA screening is error prone, so it is desirable to design *error-tolerant* algorithms, which can detect or correct some errors in the test outcomes. The reader is referred to the monograph by Du and Hwang [17] for a comprehensive discussion of this topic.

Between fully adaptive and nonadaptive (one stage) algorithms, the so-called trivial two-stage algorithms [36] are of considerable interest for screening problems. Such an algorithm has two stages. In the first stage, the pools are tested in parallel, and a set CP of *candidate positives* from the items is chosen based on the test results; in the second stage, individual tests are performed on all the items in CP to resolve the status of each item. Previous works on two-stage group testing algorithms are, among others, [4, 14, 24, 36, 41]. The following quotation from Knill [36] well emphasizes the importance of such algorithms: "It is generally feasible to construct a number of pools (much fewer than the number of clones) initially by exploiting parallelism, but adaptive construction of pools with many clones during the testing procedure is discouraged. The technicians who implement the pooling strategies generally dislike even the 3-stage strategies that are often used. Thus the most commonly used strategies for pooling libraries of clones rely on a fixed but reasonably small set of non-singleton pools. The pools are either tested all at once or in a small number of stages (usually at most 2) where the previous stage determines which pools to test in the next stage. The potential positives are then inferred and confirmed by testing of individual clones. In most biological applications each

positive clone must be confirmed even if the pool results unambiguously indicate that it is positive. This is to improve the confidence in the results, given that in practice the tests are prone to errors.”

Separating Matrices. A nonadaptive group testing procedure can be represented as a 0-1 matrix $M = (m_{ij})$, in which the columns are associated with the items and the rows are associated with the tests, and $m_{ij} = 1$ indicates that item j is contained in test i . The test outcomes can be represented by a 0-1 vector, the *outcome vector*, where 0 indicates a negative outcome and 1 indicates a positive outcome. It is not hard to verify that if a subset S of columns exactly corresponds to all the positive items, then the outcome vector is equal to vector $U(S)$, the *union* (i.e., the componentwise Boolean sum) of all column vectors in S . Given the matrix representation of an algorithm and the outcome vector, the process of identifying all the positive items is called decoding. For a 0/1 matrix to be a valid nonadaptive group testing algorithm, some separating property is often required. This monograph focuses on two most used and studied separating properties: disjunctness and separability.

In order to identify all positives as long as the number of positives is no more than d , matrix M should satisfy that for any two distinct subsets S_1 and S_2 of columns such that $|S_1| \leq d$ and $|S_2| \leq d$, $U(S_1) \neq U(S_2)$. A matrix satisfying this property is called \bar{d} -separable. In the definition, if the condition “ $|S_1| \leq d$ and $|S_2| \leq d$ ” is replaced by “ $|S_1| = |S_2| = d$,” a matrix satisfying this property is called d -separable. If the matrix representing a nonadaptive pooling design is \bar{d} -separable (or d -separable), then theoretically based on the test outcomes one can unambiguously identify all the up to d (or exactly d) positives. However, the actual process of determining the positives from the outcome vector, that is, the *decoding* process, could be very time-consuming. In practice, one can adopt matrices with stronger property to make the decoding process more efficient.

For two 0-1 vectors u and v with the same number of components, if for any component of u with value 1, the corresponding component of v is also 1, then u is said to be *covered* by v . A 0-1 matrix is said to be d -disjunct if no column is covered by the union of any d other columns. The same structure is also called *cover-free family* in combinatorics [25, 29, 53], and *superimposed code* in information theory [22, 23, 34], and has been extensively studied. Obviously if a matrix is d -disjunct, then it is also \bar{d} -separable, and thus is d -separable. If the matrix M representing a nonadaptive pooling design is d -disjunct and the number of positives is no more than d , then the following efficient decoding procedure exists with running time linear in the size of M : A column c corresponds to a positive item if and only if c is covered by the outcome vector. d -disjunct matrices are important structures in pooling design, and there have been a lot of works on their constructions [2, 10, 20, 23–25, 28, 32–34, 38, 44, 45, 49, 50].

A 0/1 matrix is said to be $(d; z)$ -disjunct [22, 39] if for any set D of d columns and any column $c \notin D$, there exist at least z rows such that each of them has value 1 at column c and value 0 at all the d columns of D . Clearly, d -disjunctness

is just $(d; 1)$ -disjunctness. As mentioned above, if the matrix M representing a nonadaptive group testing algorithm is d -disjunct and the number of positives is no more than d , then there exist efficient decoding procedures with running time linear in the size of M . However, when there are errors in the test outcomes, the above decoding procedure generally does not work, and in this case, the matrix is required to be $(d; z)$ -disjunct, which results in a $\lfloor \frac{z-1}{2} \rfloor$ -error-correcting NGT algorithm. In this case, linear decoding that successfully identifies all positives still exists, provided that there are no more than d positives and at most $\lfloor \frac{z-1}{2} \rfloor$ errors in the test outcomes. d -disjunct and $(d; z)$ -disjunct matrices form the basis of error-free and error-tolerant nonadaptive group testing.

Main Contents. In this monograph, some recent algorithmic, complexity, and mathematical results on nonadaptive group testing (and on pooling design) are presented. The main contents consist of five parts. In the first part, new randomized constructions of one- and two-stage nonadaptive group testing are presented. Comparisons with other known constructions on the number of required tests are also discussed.

In the second part, some complexity results for problems that are basic to nonadaptive group testing are given. The problem to determine whether a given matrix H is \bar{d} -separable and minimal, MIN-SEPARABILITY, is showed to be DP -complete. Here the meaning of being minimal is that the removal of any row from H will make it no longer \bar{d} -separable. The second problem is, given a binary matrix M and a positive integer d , find a minimum \bar{d} -separable submatrix of M with the same number of columns. The complexity of the decision version of this problem, \bar{d} -separable submatrix, is conjectured to be Σ_2^P -complete. As an evidence to support this conjecture, the Σ_2^P -completeness of a problem which is a little more general than \bar{d} -separable submatrix is established.

In the third part, the parameterized complexities of three basic problems in nonadaptive group testing are studied. They are, given an $m \times n$ binary matrix and a positive integer d , to determine whether the matrix is d -separable (\bar{d} -separable, or d -disjunct). Though the three problems are all known to be coNP-complete in the classical complexity theory, the motivation of this study is that in most applications d is very small compared to n ; it is interesting to investigate whether there are efficient algorithms solving the above problems when the value of d is small. In this part, the parameterized versions of the three problems, with d as the parameter, are showed to be co-W[2]-complete. The immediate implications of the results are that, given an $m \times n$ binary matrix and a positive integer d , a deterministic algorithm with running time $f(d) \times (mn)^{O(1)}$ (where f is an arbitrary computable function) to determine whether the matrix is d -separable (\bar{d} -separable, or d -disjunct) should not be expected.

In the fourth part, upper bounds on the minimum number of rows required by any d -disjunct matrix and any $(d; z)$ -disjunct matrix with n columns, $t(d, n)$ and $t(d, n; z)$, respectively, are studied. A very short proof is given for the currently best upper bound on $t(d, n)$; the method is also generalized to obtain a new upper bound

on $t(d, n; z)$. In the final part, a way to transform an error-tolerant separable matrix to an error-tolerant disjunct matrix is given; the optimality of this transformation in some senses is also discussed. If no base is specified, then \log is of base 2 throughout.

2 New Constructions of One- and Two-Stage Pooling Design

In [10] new constructions of one- and two-stage pooling design are given. For one-stage pooling design, the focus is on the construction of disjunct matrices, which are widely studied for various applications including the design of nonadaptive group testing algorithms. There have been a lot of works on the construction of disjunct matrices [2, 10, 20, 23–25, 28, 32–34, 38, 44, 45, 49, 50].

For two-stage pooling designs, De Bonis et al. [14] first present an asymptotically optimal two-stage algorithm that requires a number of tests within a constant factor $7.54(1 + o(1))$ of the information theoretic lower bound $d \log(n/d)$. Eppstein et al. [24] improve the constant factor to $4(1 + o(1))$ by using the concept of (d, k) -*resolvable* matrices (which will be explained later), which is currently the best. There are also probabilistic pooling designs [40, 41, 44] with good performance in practice.

In the sequel of this section, two Las Vegas algorithms for constructing d -disjunct and $(d; z)$ -disjunct matrices are presented. For two-stage pooling designs, an algorithm using a number of $C_d(1 + o(1)) \log n$ tests is presented, where $C_d \leq \frac{3}{\log 3} d$ for $d \geq 1$ and $C_d \rightarrow d \log e$ as $d \rightarrow \infty$. This improves the previously best bound given in [24] by a factor of more than 2. New probabilistic pooling designs are also proposed. Compared to [44], the new probabilistic designs have different type of possible errors and require much fewer tests. All the results presented in this section are from [10].

2.1 Preliminaries

Transversal Design. A pooling design is transversal if the pools can be divided into disjoint families, each of which is a partition of all items. The concept of q -ary $(d, 1)$ -*disjunct* matrix will be first introduced: A q -ary matrix is $(d, 1)$ -disjunct if for any column c and any set D of d other columns, there exists at least one element in c such that the element does not appear in any column of D in the same row.

As described in [17, 20], one can transform a q -ary $(d, 1)$ -disjunct matrix M' into a (binary) d -disjunct matrix M as follows. Replace each row R_i of M' by several rows indexed with entries of R_i ; for each entry x of R_i , the row with index x is obtained from R_i by turning all x 's into 1's and all others into 0's. Thus, the following theorem holds.

Theorem 1 ((Theorem 3.6.1 in [17])) *A $t_0 \times n$ q -ary $(d, 1)$ -disjunct matrix M' yields a $t \times n$ d -disjunct matrix M with $t \leq t_0 q$.*

Clearly, one can perform the above transformation even when the q -ary matrix M' is not $(d, 1)$ -disjunct. Transversal designs are favorable in practice because every column of the resulting matrix M has equal weight, which means that every item is contained in equal number of pools, so that to perform the tests, one needs the same number of copies for each item.

Two Probabilistic Lemmas. The following two lemmas will be useful later. The first is the Markov inequality (see, e.g., Theorem 3.2 in [42]), and the second is commonly known as Chernoff's bounds (Theorems 4.1 and 4.2 in [42]).

Lemma 1 (Markov inequality) *Let Y be a random variable assuming only nonnegative values, then for all $t > 0$,*

$$\Pr[Y \geq t] \leq \frac{E[Y]}{t},$$

where $E[Y]$ is the expectation of Y .

Lemma 2 (Chernoff's bounds) *Let X_1, X_2, \dots, X_n be independent 0/1 random variables, for $1 \leq i \leq n$, $\Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Let $X = \sum_{i=1}^n X_i$ and $\mu = E[X] = \sum_{i=1}^n p_i$. Then, for any $\delta > 0$,*

1. $\Pr[X \geq (1 + \delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$.
2. $\Pr[X \leq (1 - \delta)\mu] \leq e^{-\mu\delta^2/2}$.

2.2 One-Stage Pooling Designs

Two efficient randomized constructions will be given for d -disjunct and $(d; z)$ -disjunct matrices, respectively. The constructions are based on the transversal design.

2.2.1 A New Construction of d -Disjunct Matrices

A Las Vegas algorithm will be presented next, which for given n, d and $0 < p < 1$, successfully constructs a $t \times n$ d -disjunct matrix with probability at least p , with

$$t \leq cd^2 \left(\log \frac{2}{1-p} + \log n \right),$$

where $c \approx 4.28$ is constant.

For given n, d and $0 < p < 1$, define $n_0 = 2n$. Let ϵ be the unique positive root of

$$\ln(1 + \epsilon) = \frac{2\epsilon}{1 + \epsilon}.$$

$\epsilon \approx 3.92$ is chosen to minimize the leading constant of t (see the remarks in later part). Let

Algorithm 1 (constructing d -disjunct matrix $M_{t \times n}$)

1. Construct a random q -ary matrix $M'_{t_0 \times n_0}$ with each cell randomly assigned from $\{1, 2, \dots, q\}$, independently and uniformly.
2. For any $1 \leq i < j \leq n_0$, let $w_{i,j}$ be the random variable denoting the number of rows r such that the two entries $M'(r, i)$ and $M'(r, j)$ are equal. Then,

$$E[w_{i,j}] = \mu \quad (= \frac{t_0}{q}).$$

Create an edge between columns i and j if $w_{i,j} \geq (1 + \epsilon)\mu$.

3. For each edge created in Step 2, remove one of its two columns arbitrarily. Let M'' denote the resulting matrix.
 4. If M'' has less than n ($= \frac{n_0}{2}$) columns, exit and the algorithm fail.
 5. Using the transformation in [Theorem 1](#), turn the first n columns of M'' into a binary matrix $M_{t \times n}$ with $t \leq t_0q$.
-

$$q = (1 + \epsilon)d, \quad t_0 = \frac{1 + \epsilon}{\epsilon} d \ln \frac{2n - 1}{1 - p}, \quad \mu = \frac{t_0}{q}.$$

Please see [Algorithm 1](#) as the algorithm for constructing d -disjunct matrices.

In [Algorithm 1](#), at Step 3, M'' must be q -ary $(d, 1)$ -disjunct since for any column i , the union of any d other columns can only cover less than

$$d \times (1 + \epsilon)\mu = d \times \frac{t_0}{d} = t_0$$

entries of column i . Therefore, if the algorithm successfully returns a matrix, it must be d -disjunct. Moreover,

$$\begin{aligned} t &\leq t_0q \\ &= \frac{(1 + \epsilon)^2}{\epsilon \log e} d^2 \log \frac{2n - 1}{1 - p} \\ &< c d^2 \left(\log \frac{2}{1 - p} + \log n \right), \end{aligned}$$

where $c = \frac{(1 + \epsilon)^2}{\epsilon \log e} \approx 4.28$.

2.2.2 Analysis of Algorithm 1

The analysis of the success probability and running time of [Algorithm 1](#) will be presented next.

Success Probability. First, the expectation of the number of edges created at Step 2 is estimated.

Lemma 3 *Let m be the random variable denoting the number of edges created at Step 2 of [Algorithm 1](#), then $E[m] \leq n(1 - p)$.*

Proof For $1 \leq i < j \leq n_0$, $1 \leq k \leq t_0$, at Step 2 of [Algorithm 1](#), define 0/1 random variable $X_k^{i,j}$ such that

$$X_k^{i,j} = 1 \text{ if and only if } M'(k, i) = M'(k, j).$$

Then,

$$w_{i,j} = X_1^{i,j} + X_2^{i,j} + \cdots + X_{t_0}^{i,j}.$$

Let $X^{i,j}$ be the indicator random variable for the event that there is an edge between column i and column j , that is,

$$X^{i,j} = \begin{cases} 1 & \text{there is an edge between column } i \text{ and column } j, \text{ that is, } w_{i,j} \geq (1+\epsilon)\mu; \\ 0 & \text{otherwise.} \end{cases}$$

Since $w_{i,j}$ is the sum of t_0 -independent 0/1 random variables, the Chernoff bound, (1) in [Lemma 2](#), implies that

$$\Pr[X^{i,j} = 1] = \Pr[w_{i,j} \geq (1+\epsilon)\mu] \leq \left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}} \right)^\mu.$$

Notice that $q = (1+\epsilon)d$, and

$$t_0 = \frac{1+\epsilon}{\epsilon} d \ln \frac{2n-1}{1-p}.$$

Then,

$$\mu = E[w_{i,j}] = \frac{t_0}{q} = \frac{1}{\epsilon} \ln \frac{2n-1}{1-p}.$$

From

$$\ln(1+\epsilon) = \frac{2\epsilon}{1+\epsilon},$$

it follows that $(1+\epsilon)^{1+\epsilon} = e^{2\epsilon}$, and so

$$\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}} = e^{-\epsilon},$$

which implies that

$$\left(\frac{e^\epsilon}{(1+\epsilon)^{1+\epsilon}} \right)^\mu = (e^{-\epsilon})^{\frac{1}{\epsilon} \ln \frac{2n-1}{1-p}} = \frac{1-p}{2n-1}.$$

Thus,

$$E[X^{i,j}] = \Pr[X^{i,j} = 1] \leq \frac{1-p}{2n-1},$$

for $1 \leq i < j \leq n_0$. Since $m = \sum_{1 \leq i < j \leq n_0} X^{i,j}$ and all the $X^{i,j}$'s are identically distributed, and $n_0 = 2n$, it follows that

$$E[m] = \binom{n_0}{2} E[X^{1,2}] \leq \binom{n_0}{2} \frac{1-p}{2n-1} = n(1-p). \quad \square$$

Clearly, m denotes the most number of columns that may be removed at Step 3. Since $E[m] \leq n(1-p)$, by applying the Markov inequality ([Lemma 1](#)), the probability that there are less than n columns left in M'' at Step 4 (i.e., the failure probability of [Algorithm 1](#)) is at most $\Pr[m > n] \leq \frac{E[m]}{n} \leq 1-p$.

Running Time. The time required by [Algorithm 1](#) is dominated by Step 2, which is

$$\binom{n_0}{2} t_0 = O(dn^2 \ln n),$$

by simply counting, for all pairs of columns, the number of rows at which the two columns have equal entry. In fact, an expected $O(n^2 \ln n)$ running time can be obtained by counting along the rows.

For $1 \leq i < j \leq n_0$, let $n(i, j)$ denote the number of equal entries between column i and column j in the same row. Initially, set $n(i, j) = 0$ for $1 \leq i < j \leq n_0$. For each row r , let $S_{r,1}, S_{r,2}, \dots, S_{r,q}$ denote the sets of column indices such that

$$S_{r,k} = \{i : M'(r, i) = k\}.$$

Clearly, the sets $S_{r,k}$, $1 \leq k \leq q$, can be constructed in n_0 time. For each k , increase the values of $n(i, j)$ by 1 for all $i < j$ and $i, j \in S_{r,k}$. The expected number of such pairs (i, j) for each $S_{r,k}$ is $E[\binom{|S_{r,k}|}{2}]$. Since $|S_{r,k}|$ are identically distributed for $1 \leq k \leq q$, the expected running time of Step 2 is

$$t_0 \times \left(n_0 + qE \left[\binom{|S_{r,1}|}{2} \right] \right).$$

Notice that $|S_{r,1}|$ has the binomial distribution with parameters n_0 and $1/q$; thus,

$$n_0 + qE \left[\binom{|S_{r,1}|}{2} \right] = n_0 + q \frac{1}{q^2} (n_0^2 - n_0) = O(n^2/d),$$

and so the expected running time of Step 2, which is also the expected running time of [Algorithm 1](#), is $t_0 \times O(n^2/d) = O(n^2 \ln n)$.

Therefore, the following theorem is established.

Theorem 2 *Given n, d , and $0 < p < 1$, [Algorithm 1](#) successfully constructs a $t \times n$ d -disjunct matrix with probability at least p , with*

$$t < cd^2 \left(\log \frac{2}{1-p} + \log n \right),$$

where $c \approx 4.28$ is constant. The algorithm runs in expected $O(n^2 \ln n)$ time.

Remarks In [Algorithm 1](#), ϵ is chosen to minimize the leading constant of t . It is required that

$$(1 + \epsilon)\mu \leq \frac{t_0}{d},$$

where $\mu = \frac{t_0}{q}$, that is, $q \geq (1 + \epsilon)d$, to guarantee that matrix M'' is $(d, 1)$ -disjunct. To guarantee that with reasonable probability M'' has at least n columns, it is required that

$$n_0 - E[m] \geq n,$$

where $E[m] = \binom{n_0}{2}E[X^{1,2}]$. This implies that

$$\Pr[X^{1,2} = 1] \leq \frac{n_0 - n}{\binom{n_0}{2}}.$$

Since

$$\max_{n_0} \frac{n_0 - n}{\binom{n_0}{2}} = \frac{1}{2n - 1},$$

which can be achieved when $n_0 = 2n - 1$ or $n_0 = 2n$, it should have that

$$\Pr[X^{1,2} = 1] \leq \frac{1}{2n - 1}.$$

This can be guaranteed by

$$\left(\frac{e^\epsilon}{(1 + \epsilon)^{1+\epsilon}} \right)^\mu \leq \frac{1}{2n - 1},$$

that is,

$$\mu \ln \frac{(1 + \epsilon)^{1+\epsilon}}{e^\epsilon} \geq O(1) + \ln n.$$

By plugging in $\mu = \frac{t_0}{q} = \frac{t}{q^2}$ and $q \geq (1 + \epsilon)d$, it follows that

$$t \geq \frac{(1 + \epsilon)^2}{\ln \frac{(1 + \epsilon)^{1+\epsilon}}{e^\epsilon}} d^2 (O(1) + \ln n).$$

Define

$$f(\epsilon) = \frac{(1 + \epsilon)^2}{\ln \frac{(1 + \epsilon)^{1+\epsilon}}{e^\epsilon}} = \frac{(1 + \epsilon)^2}{(1 + \epsilon) \ln(1 + \epsilon) - \epsilon}.$$

To minimize $f(\epsilon)$ for $\epsilon > 0$, from basic calculus, $f'(\epsilon) = 0$ implies that

$$\ln(1 + \epsilon) = \frac{2\epsilon}{1 + \epsilon}.$$

It is easy to verify that this equation has one unique positive root $\epsilon \approx 3.92$. Also, the equation implies that

$$f(\epsilon) = \frac{(1 + \epsilon)^2}{(1 + \epsilon) \ln(1 + \epsilon) - \epsilon} = \frac{(1 + \epsilon)^2}{\epsilon}.$$

2.2.3 Error-Tolerance Case

[Algorithm 1](#) is modified next, so that given $n, d, z > 1$ and $0 < p < 1$, the modified algorithm successfully constructs a $t \times n$ $(d; z)$ -disjunct matrix with probability at least p , with

$$t \leq cd^2 \left(\log \frac{2}{1-p} + \log n \right) + 2(1 + \epsilon)dz + O\left(\frac{z^2}{\ln n}\right),$$

where $\epsilon \approx 3.92$ and $c \approx 4.28$ are constants, and the $O(\cdot)$ notation hides dependencies on p .

First a generalization of $(d, 1)$ -disjunct matrices is given. A q -ary matrix is $(d, 1; z)$ -*disjunct* if for any column c and any set D of d other columns, there exist at least z elements in c such that each of these elements does not appear in any column of D in the same row. Clearly, by applying the same transformation in [Theorem 1](#), one can turn a $t_0 \times n$ q -ary $(d, 1; z)$ -disjunct matrix into a $(d; z)$ -disjunct matrix with n columns and at most t_0q rows.

For given $n, d, z > 1$ and $0 < p < 1$, let n_0, ϵ be as in [Algorithm 1](#). Let

$$q = (1 + \epsilon)d + \frac{\epsilon z}{\ln \frac{2n-1}{1-p}}, \quad t_0 = z + \frac{1 + \epsilon}{\epsilon} d \ln \frac{2n-1}{1-p}, \quad \mu = \frac{t_0}{q}.$$

It can be verified that by this assignment,

$$(1 + \epsilon)\mu = \frac{t_0 - z}{d},$$

and

$$\left(\frac{e^\epsilon}{(1 + \epsilon)^{1+\epsilon}} \right)^\mu = \frac{1-p}{2n-1}.$$

Please see [Algorithm 2](#) as the algorithm for constructing $(d; z)$ -disjunct matrices.

Firstly, at Step 3 of [Algorithm 2](#), M'' must be q -ary $(d, 1; z)$ -disjunct because for any column i , any d other columns can only cover less than $d \times (1 + \epsilon)\mu = t_0 - z$ of its entries. Therefore, if the algorithm successfully returns a matrix, it must be $(d; z)$ -disjunct. Secondly, when $z = o(d \ln n)$, by similar arguments, [Algorithm 2](#) runs in time $O(dn^2 \ln n)$ in the straightforward manner, and can be improved to expected $O(n^2 \ln n)$ time by counting the pairs of equal entries along the rows. Thirdly,

Algorithm 2 (constructing $(d; z)$ -disjunct matrix $M_{t \times n}$)

[Algorithm 2](#) works in the same way as [Algorithm 1](#), except that with $q = (1 + \epsilon)d + \frac{\epsilon z}{\ln \frac{2n-1}{1-p}}$ and $t_0 = z + \frac{1+\epsilon}{\epsilon} d \ln \frac{2n-1}{1-p}$.

$$\begin{aligned} t &\leq t_0 q \\ &= \frac{(1 + \epsilon)^2}{\epsilon \log e} d^2 \log \frac{2n - 1}{1 - p} + 2(1 + \epsilon)dz + \frac{(\epsilon \log e)z^2}{\log \frac{2n-1}{1-p}} \\ &\leq cd^2 \left(\log \frac{2}{1-p} + \log n \right) + 2(1 + \epsilon)dz + O\left(\frac{z^2}{\ln n}\right), \end{aligned}$$

where $c = \frac{(1+\epsilon)^2}{\epsilon \log e} \approx 4.28$.

For the success probability, if one let m^* be the random variable denoting the number of edges created at Step 2, since

$$\left(\frac{e^\epsilon}{(1 + \epsilon)^{1+\epsilon}} \right)^\mu = \frac{1 - p}{2n - 1}$$

still holds, the same result in [Lemma 3](#) also holds here, that is,

$$E[m^*] \leq n(1 - p).$$

Therefore, the probability that there are less than n columns left at Step 4 (i.e., the failure probability of [Algorithm 2](#)) is at most $\Pr[m^* > n] \leq 1 - p$. The following theorem is established.

Theorem 3 *Given $n, d, z > 1$ and $0 < p < 1$, [Algorithm 2](#) successfully constructs a $t \times n$ $(d; z)$ -disjunct matrix with probability at least p , with*

$$t \leq cd^2 \left(\log \frac{2}{1-p} + \log n \right) + 2(1 + \epsilon)dz + O\left(\frac{z^2}{\ln n}\right),$$

where $\epsilon \approx 3.92$ and $c \approx 4.28$ are constants. When $z = o(d \ln n)$, the algorithm runs in expected $O(n^2 \ln n)$ time.

2.3 Two-Stage Pooling Designs

New two-stage pooling designs, which require a number of tests asymptotically no more than a factor of $\frac{3}{\log 3}$ (the factor approaches $\log_2 e \approx 1.44$ as d tends to infinity) of the information-theoretic lower bound $d \log(n/d)$, will be presented next.

This improves the previously best upper bound of $4(1 + o(1))$ times the information theoretic bound in [24] by a factor of more than 2.

For a 0/1 matrix M , let C denote the set of columns of M , recall that M is d -disjunct if for any d -sized subset D of C , each column in $C - D$ is not covered by $U(D)$, where $U(D)$ denotes the union of the columns in D . Such matrices form the basis for nonadaptive (one-stage) pooling designs. However, a d -disjunct matrix with n columns requires no less than $\Omega(\frac{d^2 \log n}{\log d})$ rows, which is a factor of $d / \log d$ of the information-theoretic lower bound.

Instead of determining all the positives immediately, in [24] the authors relax the property of M by introducing the concept of (d, k) -resolvable matrices, which form a good two-stage group testing regimen. A 0/1 matrix M is called (d, k) -resolvable if, for any d -sized subset D of C , there are fewer than k columns in $C - D$ that are covered by $U(D)$. Thus, a matrix is d -disjunct if and only if it is $(d, 1)$ -resolvable.

For a set of n items in which at most d are positives, one can construct a “trivial two-stage” pooling design based on a $t \times n$ (d, k) -resolvable matrix as follows. Define the first round tests according to the rows of the matrix. By identifying the items in a negative pool (a pool with negative test outcome) as negatives, one can restrict the positives to a set D' of size smaller than $d + k$. Then, perform an additional round of tests on each item in D' individually. Thus, the total number of tests of the two stages is less than $t + d + k$.

2.3.1 Near Optimal Two-Stage Pooling Designs

Let M_1 be a q -ary matrix, and let C denote the set of columns of M_1 . Matrix M_1 is said to be $(d, 1; k)$ -resolvable if, for any d -sized subset D of C , there are fewer than k columns in $C - D$ that are *covered* by D . Here by saying a column c is *covered* by D , it means that for each element of c , the element appears at least once in some column of D in the same row. By applying the transformation in [Theorem 1](#), one can turn a $t_0 \times n$ q -ary $(d, 1; k)$ -resolvable matrix into a $t \times n$ (d, k) -resolvable matrix with $t \leq t_0 q$.

Let M' be a random $t_0 \times n$ q -ary (where q will be specified later) matrix with each cell assigned randomly from $\{1, 2, \dots, q\}$, independently and uniformly. For each set D of d columns and a column $c \notin D$, for each element c_i ($i = 1, 2, \dots, t_0$) in c , the probability that c_i appears in some column of D in the same row is $1 - \left(1 - \frac{1}{q}\right)^d$; thus, the probability that every element in c appears in some column of D in the same row, that is, c is covered by D , is $[1 - \left(1 - \frac{1}{q}\right)^d]^{t_0}$. Parameter t_0 is chosen such that

$$\left[1 - \left(1 - \frac{1}{q}\right)^d\right]^{t_0} = \frac{1}{n - d},$$

that is,

$$t_0 = -\frac{\log(n-d)}{\log \left[1 - \left(1 - \frac{1}{q} \right)^d \right]}.$$

Let C denote the set of columns of M' . For any set D of d columns of M' , and for each $c \in C - D$, let X_c be the indicator variable such that $X_c = 1$ if and only if c is covered by D . Then,

$$\Pr[X_c = 1] = \frac{1}{n-d}.$$

Define

$$X_D = \sum_{c \in C-D} X_c.$$

Then, X_D is the random variable denoting the number of columns in $C - D$ that are covered by D . Since X_D is the sum of $(n-d)$ i.i.d. 0/1 random variables and $E[X_D] = 1$, the Chernoff's bound implies that the probability that D covers at least $(1 + \delta)$ columns in $C - D$ is

$$\Pr[X_D \geq (1 + \delta)] \leq \frac{e^\delta}{(1 + \delta)^{1+\delta}}.$$

Therefore, the probability that M' is not $(d, 1; 1 + \delta)$ -resolvable, that is, there exists some set D of d columns that covers at least $(1 + \delta)$ columns in $C - D$, is at most

$$p = \binom{n}{d} \frac{e^\delta}{(1 + \delta)^{1+\delta}}.$$

In order to satisfy $p < 1$, it suffices to assign δ such that

$$\left(\frac{1 + \delta}{e} \right)^{1+\delta} = n^d,$$

since which implies that

$$\frac{(1 + \delta)^{1+\delta}}{e^\delta} > \frac{(1 + \delta)^{1+\delta}}{e^{1+\delta}} = n^d > \binom{n}{d}.$$

Notice that $(\frac{1+\delta}{e})^{1+\delta} = n^d$ implies $(\frac{1+\delta}{e})^{\frac{1+\delta}{e}} = n^{\frac{d}{e}}$; thus,

$$1 + \delta = (1 + o(1)) \frac{d \ln n}{\ln(d \ln n)}.$$

Hence, by probabilistic arguments, the existence of a $t_0 \times n$ q -ary $(d, 1; 1 + \delta)$ -resolvable matrix with t_0 and δ as specified above has been proved.

By applying the transformation in [Theorem 1](#), one can turn the $t_0 \times n$ q -ary $(d, 1; 1 + \delta)$ -resolvable matrix M' into a (binary) $t \times n$ $(d, 1 + \delta)$ -resolvable matrix M with

$$t \leq t_0 q < -\frac{q \log n}{\log \left[1 - (1 - \frac{1}{q})^d \right]}.$$

Define

$$C_d(x) = \frac{x}{-\log \left[1 - (1 - \frac{1}{x})^d \right]}, \quad \text{for } x > 1.$$

Choosing q to be the positive integer that minimizes $C_d(x)$, and let $C_d = C_d(q)$. Then, $t \leq C_d \log n$. For $d \geq 1$,

$$C_d/d \leq C_d(3d)/d = \frac{3}{-\log \left[1 - (1 - \frac{1}{3d})^d \right]} \leq \frac{3}{\log 3}.$$

Also it is not hard to see that when $d = 1$, $C_1 = C_1(3) = \frac{3}{\log 3}$ indeed holds. Furthermore, the following lemma estimates that $q = \Theta(d)$ and $C_d \rightarrow d \log e$ as $d \rightarrow \infty$.

Lemma 4 *For $d \geq 1$, let $q = q(d)$ be the point that minimizes $C_d(x) = \frac{x}{-\log[1-(1-\frac{1}{x})^d]}$ for $x > 1$, and let $C_d = C_d(q)$. Then, $q(d) = \Theta(d)$, and $\lim_{d \rightarrow \infty} C_d/d = \log e$.*

To prove [Lemma 4](#), the following useful fact is proved first.

Fact 1 *Let $f(y) = \ln y \ln(1 - y)$, $0 < y < 1$. Then $f(y)$ achieves maximum at $y = \frac{1}{2}$.*

Proof of Fact 1: By symmetry, it is sufficient to show that $f'(y) > 0$ for $0 < y < \frac{1}{2}$. Since

$$\begin{aligned} f'(y) &= \frac{1}{y} \ln(1 - y) - \frac{1}{1 - y} \ln y \\ &= \frac{1}{y(1 - y)} [(1 - y) \ln(1 - y) - y \ln y], \end{aligned}$$

let

$$g(y) = (1 - y) \ln(1 - y) - y \ln y,$$

Next it will be showed that $g(y) > 0$ for $0 < y < \frac{1}{2}$.

Rewrite

$$g(y) = \ln(1 - y) + y \ln \frac{1}{y(1 - y)}.$$

For $0 < y < \frac{1}{2}$, $\ln \frac{1}{y(1-y)} > \ln 4$ since $y(1-y) < \frac{1}{4}$; thus,

$$g(y) > \ln(1 - y) + y \ln 4.$$

Let

$$h(y) = \ln(1 - y) + y \ln 4.$$

Notice that $h'(y) = \ln 4 - \frac{1}{1-y}$, $h'(y) > 0$ for $0 < y < 1 - \frac{1}{\ln 4}$ and $h'(y) < 0$ for $1 - \frac{1}{\ln 4} < y < \frac{1}{2}$. Thus, $h(y)$ is monotone increasing when $0 < y < 1 - \frac{1}{\ln 4}$ and monotone decreasing when $1 - \frac{1}{\ln 4} < y < \frac{1}{2}$. From $h(0) = h(\frac{1}{2}) = 0$, one can obtain that

$$h(y) > 0 \text{ for } 0 < y < \frac{1}{2}.$$

Therefore, for $0 < y < \frac{1}{2}$, $g(y) > h(y) > 0$, and so $f'(y) = \frac{1}{y(1-y)}g(y) > 0$. \square

Proof of Lemma 4: Notice that if q_1 satisfies $(1 - \frac{1}{q_1})^d = \frac{1}{2}$, then $q_1 = \Theta(d)$ since $\frac{q_1}{d} \rightarrow \log e$ as $d \rightarrow \infty$. Moreover,

$$C_d(q_1) = q_1 = \Theta(d).$$

The lemma is proved by contradiction. First assume that $q(d) = O(d)$ does not hold, that is, for any $c > 0$ and any $d_0 > 0$, there exists $d > d_0$ such that $q(d) > cd$. Then, since $\frac{q}{d} > c$ (for simplicity q is used instead of $q(d)$, if it is clear from the context), as $c \rightarrow \infty$,

$$\begin{aligned} C_d(q) &= \frac{q}{-\log \left[1 - \left(1 - \frac{1}{q} \right)^d \right]} \\ &\sim \frac{q}{-\log \left[1 - \left(1 - \frac{d}{q} \right) \right]} \\ &= d \frac{\frac{q}{d}}{\log \frac{q}{d}} \\ &= \omega(d); \end{aligned}$$

here, $a \sim b$ means that $\lim_{c \rightarrow \infty} \frac{a}{b} = 1$. However, this contradicts since q is the point that minimizes $C_d(q)$ and on the other hand $C_d(q_1) = \Theta(d)$. On the other hand, assume that $q(d) = \Omega(d)$ does not hold, that is, for any $c > 0$ and any $d_0 > 0$, there exists $d > d_0$ such that $q(d) < cd$. Write

$$\begin{aligned} C_d(q) &= \frac{q}{-\log \left[1 - \left(1 - \frac{1}{q} \right)^d \right]} \\ &= \frac{q \ln 2}{-\ln \left\{ 1 - \left[\left(1 - \frac{1}{q} \right)^q \right]^{\frac{d}{q}} \right\}}. \end{aligned}$$

Since $0 < \left(1 - \frac{1}{q}\right)^q < \frac{1}{e}$ for $q > 1$, as $c \rightarrow 0$, $\frac{d}{q} > \frac{1}{c} \rightarrow \infty$, and $\left[\left(1 - \frac{1}{q}\right)^q\right]^{\frac{d}{q}} < e^{-\frac{d}{q}} \rightarrow 0$; thus,

$$C_d(q) \sim \frac{q \ln 2}{\left[\left(1 - \frac{1}{q}\right)^q\right]^{\frac{d}{q}}} > e^{\frac{d}{q}} q \ln 2 = d \frac{e^{\frac{d}{q}} \ln 2}{\frac{d}{q}} = \omega(d);$$

this also contradicts (here $a \sim b$ means that $\lim_{c \rightarrow 0} \frac{a}{b} = 1$). Therefore, $q(d) = \Theta(d)$.

Next C_d as $d \rightarrow \infty$ is estimated. Since $\left(1 - \frac{1}{q}\right)^q < \frac{1}{e}$ for $q > 1$, thus

$$\left(1 - \frac{1}{q}\right)^d < \left(\frac{1}{e}\right)^{\frac{d}{q}} = e^{-\frac{d}{q}},$$

and

$$-\log \left[1 - \left(1 - \frac{1}{q}\right)^d \right] < -\log \left(1 - e^{-\frac{d}{q}}\right),$$

it follows that

$$\begin{aligned} C_d(q) &= \frac{q}{-\log \left[1 - \left(1 - \frac{1}{q}\right)^d \right]} \\ &> \frac{q}{-\log \left(1 - e^{-\frac{d}{q}}\right)} \\ &= \frac{d \ln 2}{\left[-\frac{d}{q} \ln \left(1 - e^{-\frac{d}{q}}\right)\right]}. \end{aligned}$$

Let $y = e^{-\frac{d}{q}}$, then $-\frac{d}{q} = \ln y$, and $C_d(q) > \frac{d \ln 2}{\ln y \ln(1-y)}$. Since $\ln y \ln(1-y)$ achieves maximum at $y = \frac{1}{2}$ (Fact 1), $C_d(q) > d \log e$ for $q > 1$, thus $C_d > d \log e$ for $d \geq 1$. On the other hand, as mentioned at the beginning of the proof, as $d \rightarrow \infty$, $\frac{q_1}{d} \rightarrow \log e$, and $C_d(q_1) = q_1 \rightarrow d \log e$. Therefore, as $d \rightarrow \infty$, $C_d \rightarrow d \log e$. \square

The above arguments showed existence of a $t \times n$ ($d, 1 + \delta$)-resolvable matrix with $t \leq C_d \log n$ and $1 + \delta = (1 + o(1)) \frac{d \ln n}{\ln(d \ln n)}$, which implies the following theorem.

Theorem 4 *Given n and d , there exists a two-stage pooling design for finding up to d positives from n items using no more than $C_d \log n + d + \delta + 1$ tests, where*

$$C_d = \min_{x \in \mathcal{N}} \frac{x}{-\log \left[1 - \left(1 - \frac{1}{x}\right)^d \right]} \leq \frac{3}{\log 3} d,$$

for $d \geq 1$, and $\delta = (1 + o(1)) \frac{d \ln n}{\ln(d \ln n)}$. Moreover,

$$\lim_{d \rightarrow \infty} C_d/d = \log e.$$

2.4 Probabilistic Pooling Designs

A probabilistic pooling design identifying up to d positives from n items with high probability will be presented next. In a probabilistic group testing algorithm, one may identify a positive item as negative; such a wrongly identified item is called a *false negative*; a negative item which is wrongly identified as positive is called a *false positive*. Clearly, the algorithm correctly identifies all positives if and only if there are no false positives or false negatives. Previous works on probabilistic nonadaptive group testing algorithms can be found from, among others, [40, 41, 44].

Algorithm. Given n and d , first construct a $t_0 \times n$ random q -ary matrix M' with each cell randomly assigned from $\{1, 2, \dots, q\}$ independently and uniformly (where t_0 and q will be specified later). Then, use the transformation in [Theorem 1](#) to obtain a $t \times n$ 0/1 matrix M with $t \leq t_0 q$. Associate the n items with the columns of M , and test the pools indicated by the rows of M . The items not in any negative pool are identified as positives.

Analysis. Let D be the set of columns corresponding to the positives, then $|D| \leq d$. First, it is easy to see that no positive item will be identified as negative if there is no error in the test outcomes. For any negative item, let c denote the column associated with it, then the item is wrongly identified if and only if c is covered by $U(D)$ in M , or equivalently, c is covered by D in M' (here the same notations c and D are used for different matrices M and M' , to denote the corresponding columns). The probability that c is covered by D , as analyzed in [Sect. 2.3](#), is

$$\left[1 - \left(1 - \frac{1}{q} \right)^{|D|} \right]^{t_0} \leq \left[1 - \left(1 - \frac{1}{q} \right)^d \right]^{t_0}.$$

Choosing q and t_0 such that

$$\left[1 - \left(1 - \frac{1}{q} \right)^d \right]^{t_0} = \frac{1-p}{n},$$

that is,

$$t_0 = -\frac{\log n + \log \frac{1}{1-p}}{\log[1 - (1 - \frac{1}{q})^d]}.$$

Then, the probability that there exists some negative item wrongly identified is no more than

$$(n - |D|)[1 - (1 - \frac{1}{q})^d]^{t_0} \leq 1 - p,$$

which implies that with probability at least p , the above algorithm successfully identifies all the positives. The number of pools required is no more than

$$t \leq t_0 q = -\frac{q}{\log[1 - (1 - \frac{1}{q})^d]}(\log n + \log \frac{1}{1 - p}).$$

By choosing q to be the positive integer minimizing

$$C_d(x) = \frac{x}{-\log[1 - (1 - \frac{1}{x})^d]} \text{ for } x > 1,$$

obtaining

$$t \leq C_d(\log n + \log \frac{1}{1 - p}).$$

Theorem 5 *The above one-stage algorithm, with probability at least p , correctly identifies up to d positives from n items using no more than $C_d(\log n + \log \frac{1}{1-p})$ tests.*

Remarks

1. The one-stage probabilistic pooling design is also transversal. This design never gets false negatives, while the probabilistic algorithms in [40, 41, 44] never get false positives. The algorithm in [44] identifies up to 9 positives from 18,918,900 items using 5,460 tests, with success probability of 98.5 %. For the method proposed here, $n = 18,918,900$, $d = 9$, and $p = 0.985$, by choosing $q = 14$, it requires $C_d(q)(\log n + \log \frac{1}{1-p}) < 408$ tests, which is much fewer.
2. In contrast to the two-stage design in Sect. 2.3, this probabilistic algorithm is explicitly given and can be easily implemented in practice. In addition, one can extend this algorithm to two stage, by performing an additional round of individual tests on the candidate positives identified by the first round, so that no item will be wrongly identified. It is easy to verify that, for this extended two-stage probabilistic algorithm, by choosing the same value q , and choosing t_0 such that $[1 - (1 - \frac{1}{q})^d]^{t_0} = \frac{1}{n}$, the expected total number of tests required is no more than $C_d \log n + d + 1$, which is better than the deterministic two-stage design in Sect. 2.3.

2.5 Conclusion and Future Studies

New one- and two-stage pooling designs, together with new probabilistic pooling designs, are presented in this section. The approach presented works for both error-free and error-tolerance scenarios. The following remarks end off this section:

1. The constructions of pooling designs in Sects. 2.3 and 2.4 can also be generalized to error-tolerance case, in a similar manner as in the construction of $(d; z)$ -disjunct matrices in Sect. 2.2.3. The details are omitted due to the similarities.

2. Efficient constructions (i.e., in time polynomial in n and d) of the two-stage designs in Sect. 2.3 are not given. Up to now, no efficient construction of two-stage pooling designs using the number of tests within a constant factor of the information theoretic lower bound is known. In [14], the construction requires $\left(\frac{n}{2d}\right)$ time, and in [24], the authors gave existence proof as in this section. Although once such a design is found it can be used as many times as wanted, efficient construction is an important issue.
3. The two-stage pooling design presented in Sect. 2.3 uses the number of tests asymptotically within a factor of C_d/d ($\leq \frac{3}{\log 3}$ for general d , and tends to $\log_2 e \approx 1.44$ as $d \rightarrow \infty$) of the information theoretic bound $d \log(n/d)$. Can two-stage algorithms do as good as fully adaptive algorithms, that is, achieve a factor of asymptotically 1 of the information theoretic bound? Or, how good could it be?
4. Last but the least, efficient (i.e., polynomial time in n and d) deterministic constructions of d -disjunct matrices with $t = O(d^2 \log n)$ are known [2, 50]. Regarding the leading constant within the big-O notation, the results indicate that they are considerably larger than the result given in this section (where the leading constant is approximately 4.28). An efficient randomized construction of d -disjunct matrices with $t = O(d^2 \log n)$ and efficient decoding (in time polynomial in t) is given in [33], and an efficient deterministic construction with the same properties is obtained recently [45]. Improved constructions of disjunct matrices are interesting to investigate.

3 New Complexity Results on Nonunique Probe Selection

Given a collection of n targets and a sample S containing at most d of these targets, and a collection of m probes each of them hybridizes to a subset of the given targets, the goal is to select a subset of probes, such that all targets in S can be identified by observing the hybridization reactions between the selected probes and S . For each probe p , there is hybridization reaction between p and S if S contains at least one target that hybridizes with p , otherwise there is no hybridization reaction. The above probe selection problem has been extensively studied recently [5, 31, 51, 52, 56] due to its important applications, particularly in molecular biology. For example, one application of this identification problem is to identify viruses (targets) from a blood sample. The presence or absence of the viruses is established by observing the hybridization reactions between the blood sample and some probes; here, each probe is a short oligonucleotide of size 8–25 that can hybridize to one or more of the viruses.

A probe is called *unique* if it hybridizes to only one target; otherwise, it is called *nonunique*. Identifying targets using unique probes is straightforward. However, in situations where the targets have a high degree of similarity, for instance when identifying closely related virus subtypes, finding unique probes for every target is difficult. In [54], Schliep, Torney, and Rahmann proposed a group testing method using nonunique probes to identify targets in a given sample. Since each

nonunique probe can hybridize with more than one target, the identification problem becomes more complicated. One important issue is to select a subset from the given nonunique probes so that the hybridization results can be decoded, that is, determine the presence or absence of targets in sample S . Also, the number of selected probes is exactly the number of hybridization experiments required, and it is desirable to select as few probes as possible to reduce the experimental cost. In [35, 54], two heuristics using greedy and linear programming-based techniques, respectively, are proposed for choosing a suitable subset of nonunique probes. In [11], the computational complexities of some basic problems in nonunique probe selection are studied, in the context of the theory of NP -completeness (see Chap. 10 in [17, 19] and [30]). The complexity results in [11] will be presented in this section.

3.1 Preliminaries

The nonunique probe selection problem can be formulated as follows. Given a collection of n targets t_1, t_2, \dots, t_n , and a collection of m nonunique probes p_1, p_2, \dots, p_m , a sample S is known to contain at most d of the n targets. The probe-target hybridizations can be represented by an $m \times n$ 0-1 matrix M . $M_{i,j} = 1$ indicates that probe p_i hybridizes to target t_j , and $M_{i,j} = 0$ indicates otherwise. The subset of probes selected corresponds to a subset of rows in M , which forms a submatrix H of M with the same number of columns. The hybridization results between the selected probes and S also can be represented as a 0-1 vector V . $V_i = 1$ indicates that there is hybridization reaction between p_i and S , that is, p_i hybridizes to at least one target in S , and $V_i = 0$ indicates otherwise. If there is no error in the hybridization experiments, then V is equal to the union of the columns of H that correspond to the targets in S . Here, the union of a subset of columns is simply the Boolean sum of these column vectors. In order to identify all targets in S , the submatrix H should satisfy that all unions of up to d columns in H are different; in other words, H should be \bar{d} -separable. Also, as mentioned above, it is desirable to minimize the number of rows in H .

A matrix H is said to be \bar{d} -separable if all unions of up to d columns in H are different. However, the following equivalent definition is more useful in the proofs here. Let H be a $t \times n$ Boolean matrix. For each $i \in \{1, 2, \dots, t\}$, define

$$H_i = \{j \mid 1 \leq j \leq n, H_{i,j} = 1\}.$$

For any subset S of $\{1, 2, \dots, n\}$ and any $i \in \{1, 2, \dots, t\}$, write

$$H_i(S) = \begin{cases} 1 & \text{if } H_i \cap S \neq \emptyset; \\ 0 & \text{otherwise.} \end{cases}$$

Two sets $S_1, S_2 \subseteq \{1, 2, \dots, n\}$ are said to be *separated* by H if there exists an integer i , $1 \leq i \leq t$, such that $H_i(S_1) \neq H_i(S_2)$. Matrix H is said to be \bar{d} -separable if for any two different subsets S_1, S_2 of $\{1, 2, \dots, n\}$, with $|S_1| \leq d$ and $|S_2| \leq d$, S_1 and S_2 can be separated by H .

3.2 Complexity of Minimal \bar{d} -Separable Matrix

In nonunique probe selection, one natural problem of interests is to determine whether a submatrix H chosen is \bar{d} -separable and minimal. By minimal, it means that the removal of any row from H will make it no longer \bar{d} -separable. The problem can be formulated as follows.

MIN-SEPARABILITY (MINIMAL SEPARABILITY): Given a $t \times n$ Boolean matrix H and an integer $d \leq n$, determine whether it is true that (a) H is \bar{d} -separable, and (b) for any submatrix Q of H of size $(t - 1) \times n$, Q is not \bar{d} -separable.

For a given binary matrix H and a positive integer d , the problem to determine whether H is \bar{d} -separable is known to be $coNP$ -complete ([17], Theorem 10.2.1). Here a DP -completeness proof of problem MIN-SEPARABILITY will be presented.

The class DP is the collection of sets A which are the intersection of a set $X \in NP$ and a set $Y \in coNP$. The notion of DP -completeness has been used to characterize the complexity of the “exact-solution” version of many NP -complete problems. For instance, the exact traveling salesman problem, which asks, for a given edge-weighted complete graph G and a constant K , whether the minimum weight of a traveling salesman tour of the graph G is equal to K , is DP -complete (see [47], Theorem 17.2). In addition, the “critical” version of some NP -complete problems is also known to be DP -complete. For instance, the following problem is the critical version of the 3-satisfiability problem and has been shown to be DP -complete by Papadimitriou and Wolfe [48]:

MIN-3-UNSAT: Given a 3-CNF Boolean formula φ which consists of clauses C_1, C_2, \dots, C_m , determine whether it is true that (a) φ is not satisfiable, and (b) for any j , $1 \leq j \leq m$, the formula φ_j that consists of all clauses C_ℓ , $\ell \in \{1, 2, \dots, m\} - \{j\}$, is satisfiable.

Although most exact-solution version of NP -complete problems have been shown to be DP -complete, many critical versions are not known to be DP -complete. The problem MIN-SEPARABILITY may be viewed as a critical version of the \bar{d} -separability problem. Its DP -completeness will be proved by constructing a reduction from MIN-3-UNSAT.

Theorem 6 MIN-SEPARABILITY is DP -complete.

Proof Recall that $DP = \{X \cap Y \mid X \in NP, Y \in coNP\}$. A problem A is DP -complete if $A \in DP$ and, for all $B \in DP$, $B \leq_m^P A$. For convenience, for any $t \times n$ matrix H , \tilde{H}_j is used to denote the $(t - 1) \times n$ submatrix of H with the j th row removed. First, to see that $\text{MIN-SEPARABILITY} \in DP$, let

$$X = \{(H, d) \mid H \text{ is a } t \times n \text{ 0/1 matrix, } 1 \leq d \leq n, (\forall j, 1 \leq j \leq t) \tilde{H}_j \text{ is not } \bar{d}\text{-separable}\},$$

and

$$Y = \{(H, d) \mid H \text{ is a } t \times n \text{ 0/1 matrix, } 1 \leq d \leq n, H \text{ is } \bar{d}\text{-separable}\}.$$

It is clear that $\text{MIN-SEPARABILITY} = X \cap Y$. It is also not hard to see that $X \in NP$ and $Y \in coNP$. In particular, to see that $X \in NP$, note that $(H, d) \in X$ if and only if there exist $2t$ subsets $S_{j,1}, S_{j,2}$ of $\{1, 2, \dots, n\}$, for $j \in \{1, 2, \dots, t\}$, such that, for each j , $H_k(S_{j,1}) = H_k(S_{j,2})$ for all $k \in \{1, 2, \dots, t\} - \{j\}$.

Next, a reduction from MIN-3-UNSAT to MIN-SEPARABILITY will be described. Let φ be a 3-CNF Boolean formula which consists of m clauses C_1, C_2, \dots, C_m , over n variables x_1, x_2, \dots, x_n . For each $j \in \{1, 2, \dots, m\}$, let φ_j denote the Boolean formula that consists of all clauses C_ℓ for $\ell \in \{1, 2, \dots, m\} - \{j\}$. From φ , a $(3n + m + 1) \times (2n + 2)$ Boolean matrix H will be constructed, and define $d = n + 1$. For convenience, the columns of H are denoted by

$$X = \{x_i, \bar{x}_i \mid 1 \leq i \leq n\} \cup \{y, z\},$$

and denote the rows of H by

$$T = \{x_i, \bar{x}_i, u_i \mid 1 \leq i \leq n\} \cup \{y\} \cup \{C_j \mid 1 \leq j \leq m\}.$$

Next H is defined by specifying each row of it:

1. For each $1 \leq i \leq n$, let $H_{x_i} = \{x_i\}$, $H_{\bar{x}_i} = \{\bar{x}_i\}$, and $H_{u_i} = \{x_i, \bar{x}_i, z\}$.
2. $H_y = \{y\}$.
3. For each $1 \leq j \leq m$, let $H_{C_j} = \{x_i \mid x_i \in C_j\} \cup \{\bar{x}_i \mid \bar{x}_i \in C_j\} \cup \{y, z\}$ (so that $|H_{C_j}| = 5$).

To prove the correctness of the reduction, first verify that if φ is not satisfiable, then H is \bar{d} -separable. To see this, let S_1 and S_2 be two subsets of X , each of size $\leq n + 1$.

Case 1. $S_1 - \{z\} \neq S_2 - \{z\}$. Then, there exists $v \in X - \{z\}$ such that $v \in S_1 \Delta S_2$. Then, $H_v(S_1) \neq H_v(S_2)$.

Case 2. $S_1 - \{z\} = S_2 - \{z\}$. Then, it must be true that $S_1 \Delta S_2 = \{z\}$. Without loss of generality, assume $S_2 = S_1 \cup \{z\}$. Note that $|S_2| \leq n + 1$ implies $|S_1| \leq n$.

Subcase 2.1. There exists an integer i such that $|S_1 \cap \{x_i, \bar{x}_i\}| \neq 1$. First, if $|S_1 \cap \{x_i, \bar{x}_i\}| = 0$ for some i , then $H_{u_i}(S_1) = 0$ and $H_{u_i}(S_2) = 1$ (because $z \in S_2$). Next, if $|S_1 \cap \{x_i, \bar{x}_i\}| = 2$ for some i , then it must have $|S_1 \cap \{x_k, \bar{x}_k\}| = 0$ for some k , because $|S_1| \leq n$. Then, again $H_{u_k}(S_1) = 0 \neq 1 = H_{u_k}(S_2)$.

Subcase 2.2. $|S_1 \cap \{x_i, \bar{x}_i\}| = 1$ for all $i \in \{1, 2, \dots, n\}$. Note that, in this case, $y \notin S_1$. Define a Boolean assignment $\tau : \{x_1, x_2, \dots, x_n\} \rightarrow \{\text{TRUE}, \text{FALSE}\}$ by

$$\tau(x_i) = \text{TRUE} \text{ if and only if } x_i \in S_1.$$

Since φ is not satisfiable, there exists a clause C_j that is not satisfied by τ . It means that $C_j \cap S_1 = \emptyset$, and so $H_{C_j}(S_1) = 0$. However, $H_{C_j}(S_2) = 1$ since $z \in S_2$.

The above completes the proof that H is \bar{d} -separable.

Next, it will be showed that if φ_j is satisfiable for all $j = 1, 2, \dots, m$, then \widetilde{H}_v is not \bar{d} -separable for all $v \in T$. First, for $v \in X - \{z\}$, let $S_1 = \{z\}$ and $S_2 = \{v, z\}$.

Then, for all rows $w \in X - \{z, v\}$, $H_w(S_1) = 0 = H_w(S_2)$. Also, for all other rows $w \in T - X$, $H_w(S_1) = H_w(S_2) = 1$ since $z \in H_w$. So, S_1 and S_2 are not separable by \tilde{H}_v .

Next, consider the case $v = u_i$ for some $i \in \{1, 2, \dots, n\}$. Let

$$S_1 = \{x_k \mid 1 \leq k \leq n, k \neq i\} \cup \{y\},$$

and $S_2 = S_1 \cup \{z\}$. It is clear that $|S_1| = n$ and $|S_2| = n + 1$. Now the following claim is made: S_1 and S_2 are not separable by \tilde{H}_{u_i} .

To prove the claim, note that the rows H_{x_k} , $H_{\bar{x}_k}$, for $1 \leq k \leq n$, and row H_y cannot separate S_1 from S_2 , since $S_1 - \{z\} = S_2 - \{z\}$. Also, rows $H_{u_k}(S_1) = H_{u_k}(S_2) = 1$, for all $k \in \{1, 2, \dots, n\} - \{i\}$, because $|S_1 \cap \{x_k, \bar{x}_k\}| = 1$ if $k \neq i$. In addition, for any $j = 1, 2, \dots, m$, $H_{C_j}(S_1) = 1 = H_{C_j}(S_2)$, since $y \in S_1$. It follows that \tilde{H}_{u_i} cannot separate S_1 from S_2 .

Finally, consider the case $v = C_j$ for some $j \in \{1, 2, \dots, m\}$. Note that φ_j is satisfiable. So, there is a Boolean assignment $\tau : \{x_1, x_2, \dots, x_n\} \rightarrow \{\text{TRUE}, \text{FALSE}\}$ satisfying all clauses C_ℓ , except C_j . Define

$$S_1 = \{x_i \mid \tau(x_i) = \text{TRUE}\} \cup \{\bar{x}_i \mid \tau(x_i) = \text{FALSE}\},$$

and $S_2 = S_1 \cup \{z\}$. Then, similar to the argument for the case $v = u_i$, one can verify that $H_w(S_1) = H_w(S_2)$ for $w \in X - \{z\}$, and for $w \in \{u_i \mid 1 \leq i \leq n\}$. In addition, for any clause C_ℓ , with $\ell \neq j$, C_ℓ is satisfied by τ . It follows that $C_\ell \cap S_1 \neq \emptyset$ and $H_{C_\ell}(S_1) = 1 = H_{C_\ell}(S_2)$. This completes the proof that \tilde{H}_v is not \bar{d} -separable, for all $v \in T$.

Conversely, it will be showed that if $\varphi \notin \text{MIN-3-UNSAT}$, then $(H, n + 1) \notin \text{MIN-SEPARABILITY}$. First, consider the case that φ is a satisfiable formula. Let $\tau : \{x_1, x_2, \dots, x_n\} \rightarrow \{\text{TRUE}, \text{FALSE}\}$ be a Boolean assignment satisfying φ . Define

$$S_1 = \{x_i \mid \tau(x_i) = \text{TRUE}\} \cup \{\bar{x}_i \mid \tau(x_i) = \text{FALSE}\},$$

and $S_2 = S_1 \cup \{z\}$. Then, similar to the earlier proof, one can verify that H cannot separate S_1 from S_2 . In particular, $H_{C_j}(S_1) = 1$ for all $j \in \{1, 2, \dots, m\}$, because τ satisfies C_j and so $C_j \cap S_1 \neq \emptyset$. Thus, $(H, n + 1) \notin \text{MIN-SEPARABILITY}$.

Next, assume that there exists an integer $j \in \{1, 2, \dots, m\}$ such that φ_j is not satisfiable. The following claim is made: \tilde{H}_{C_j} is \bar{d} -separable. The proof of the claim is similar to the first part of the proof (for the statement that if φ is not satisfiable then H is \bar{d} -separable).

Case 1. $S_1 - \{z\} \neq S_2 - \{z\}$. Then, there exists $v \in X - \{z\}$ such that $v \in S_1 \Delta S_2$. So, $H_v(S_1) \neq H_v(S_2)$.

Case 2. $S_1 - \{z\} = S_2 - \{z\}$. Then, it must be true that $S_1 \Delta S_2 = \{z\}$, and one may assume $S_2 = S_1 \cup \{z\}$. It must have $|S_2| \leq n + 1$ and $|S_1| \leq n$.

Subcase 2.1. There exists an integer i such that $|S_1 \cap \{x_i, \bar{x}_i\}| \neq 1$. Similar to the earlier proof, if $|S_1 \cap \{x_i, \bar{x}_i\}| = 0$ for some $i = 1, 2, \dots, n$, then H_{u_i} can be

used to separate S_1 from S_2 . If $|S_1 \cap \{x_i, \bar{x}_i\}| = 2$ for some $i = 1, 2, \dots, n$, then $|S_1 \cap \{x_k, \bar{x}_k\}| = 0$ for some k , and again H_{u_k} separates S_1 from S_2 .

Subcase 2.2. $|S_1 \cap \{x_i, \bar{x}_i\}| = 1$ for all $i \in \{1, 2, \dots, n\}$. Then, since $|S_1| \leq n$, $y \notin S_1$. Define a Boolean assignment $\tau : \{x_1, x_2, \dots, x_n\} \rightarrow \{\text{TRUE}, \text{FALSE}\}$ by $\tau(x_i) = \text{TRUE}$ if and only if $x_i \in S_1$. Since φ_j is not satisfiable, there exists a clause C_ℓ , $\ell \neq j$, such that $\tau(C_\ell) = \text{FALSE}$. It means that $C_\ell \cap S_1 = \emptyset$, and so $H_{C_\ell}(S_1) = 0$. However, $H_{C_\ell}(S_2) = 1$ since $z \in S_2$. So, H_{C_ℓ} separates S_1 from S_2 . This completes the proof that \tilde{H}_{C_j} is \bar{d} -separable, and hence $(H, n + 1) \notin \text{MIN-SEPARABILITY}$. \square

3.3 Minimum \bar{d} -Separable Submatrix

A more important problem in nonunique probe selection is to find a minimum subset of probes that can identify up to d targets in a given sample. In the matrix representation, the problem can be formulated as the following: Given a binary matrix M and a positive integer d , find a minimum \bar{d} -separable submatrix of M with the same number of columns (problem MIN- \bar{d} -SS in [17], Chap. 10).

For $d = 1$, MIN- \bar{d} -SS has been proved to be NP -hard ([17], Theorem 10.3.2), by modifying a reduction used in the proof of the NP -completeness of the problem MINIMUM-TEST-SETS in [30]. For fixed $d > 1$, MIN- \bar{d} -SS is believed to be NP -hard; however, up to now, no formal proof is known. Next the decision version of MIN- \bar{d} -SS is considered.

\bar{d} -SS (\bar{d} -SEPARABLE SUBMATRIX): Given a $t \times n$ Boolean matrix M and two integers $d, k > 0$, determine whether there is a $k \times n$ submatrix H of M that is \bar{d} -separable.

Recall that Σ_2^P is the complexity class of problems that are solvable in nondeterministic polynomial time with the help of an NP -complete set as an oracle. For instance, the following problem SAT₂ is Σ_2^P -complete ([19], Theorem 3.13): Given a Boolean formula φ over two disjoint sets X and Y of variables, determine whether there exists an assignment to variables in X so that the resulting formula (over variables in Y) is a tautology. It is easy to see that \bar{d} -SS is in Σ_2^P . It is conjectured to be Σ_2^P -complete. Here a similar problem that is a little more general than \bar{d} -SS will be considered, and its Σ_2^P -completeness will be proved.

\bar{d} -SSRR (\bar{d} -SEPARABLE SUBMATRIX WITH RESERVED ROWS): Given a $t \times n$ Boolean matrix M and three integers $d > 0$, s , and $k \geq 0$, determine whether there is a \bar{d} -separable $(s + k) \times n$ submatrix H of M that contains the first s rows of M and k rows from the remaining $t - s$ bottom rows of M .

Let φ be a Boolean formula, an *implicant* of φ is a conjunction C of literals that implies φ . The following problem is proved to be Σ_2^P -complete by Umans [55].

SHORTEST IMPLICANT CORE: Given a DNF formula $\varphi = T_1 + T_2 + \dots + T_m$, and an integer p , determine whether φ has an implicant C that consists of p literals from the last term T_m .

By a reduction from SHORTEST IMPLICANT CORE, one can obtain the following result.

Theorem 7 \bar{d} -SSRR is Σ_2^P -complete.

Proof The problem \bar{d} -SSRR can be solved by a nondeterministic machine that guesses a $(s+k) \times n$ submatrix H of M which contains the first s rows of M and then determines whether H is \bar{d} -separable. Note that the problem of determining whether a given matrix H is \bar{d} -separable is in coNP. Thus, \bar{d} -SSRR $\in \Sigma_2^P$.

Next, \bar{d} -SSRR is proved to be Σ_2^P -complete by constructing a polynomial-time reduction from SHORTEST IMPLICANT CORE to it. To define the reduction, let (φ, p) be an instance of the problem SHORTEST IMPLICANT CORE, that is, let

$$\varphi = T_1 + T_2 + \cdots + T_m$$

be a DNF formula over n variables x_1, x_2, \dots, x_n , and let p be an integer > 0 . Note that each term T_j , $1 \leq j \leq m$, of φ is a conjunction of some literals. Also, T_j is used to denote the set of these literals. Assume that the last term T_m of φ has q literals $\ell_1, \ell_2, \dots, \ell_q$. Define a $(3n+m+q) \times (2n+1)$ Boolean matrix M as follows:

1. Let the $2n+1$ columns of M be $X = \{x_1, \bar{x}_1, x_2, \bar{x}_2, \dots, x_n, \bar{x}_n, z\}$ and the $3n+m+q$ rows of M be $T = \{x_i, \bar{x}_i, u_i \mid 1 \leq i \leq n\} \cup \{t_j \mid 1 \leq j \leq m\} \cup \{c_j \mid 1 \leq j \leq q\}$.
2. For $i = 1, 2, \dots, n$, $M_{x_i} = \{x_i\}$, $M_{\bar{x}_i} = \{\bar{x}_i\}$, and $M_{u_i} = \{x_i, \bar{x}_i, z\}$.
3. For $j = 1, 2, \dots, m$, $M_{t_j} = \{x_i \mid \bar{x}_i \in T_j\} \cup \{\bar{x}_i \mid x_i \in T_j\} \cup \{z\}$. (Note that $M_{t_j} \cap T_j = \emptyset$).
4. The bottom q rows of M are $M_{c_j} = \{\ell_j, z\}$, for $j = 1, 2, \dots, q$.

Let $d = n+1$, $s = 3n+m$, and $k = p$, and consider the instance (M, d, s, k) for the problem \bar{d} -SSRR.

First assume that φ has an implicant C of size p that is a subset of T_m . Let H be the submatrix of M that consists of the first $s = 3n+m$ rows plus the $k = p$ rows M_{c_j} for which $\ell_j \in C$. The following claim is made: H is \bar{d} -separable. That is, for any subsets S_1 and S_2 of $\{x_1, \bar{x}_2, \dots, x_n, \bar{x}_n, z\}$ of size $\leq d$, there exists a row in H that separates them.

Case 1. $S_1 - \{z\} \neq S_2 - \{z\}$. Then, there exists $v \in X - \{z\}$ such that $v \in S_1 \Delta S_2$. Then, $M_v(S_1) \neq M_v(S_2)$, and so H separates S_1 from S_2 .

Case 2. $S_1 - \{z\} = S_2 - \{z\}$. Then, it must be true that $S_1 \Delta S_2 = \{z\}$. Without loss of generality, assume $S_2 = S_1 \cup \{z\}$. Note that $|S_2| \leq n+1$ implies $|S_1| \leq n$.

Subcase 2.1. There exists an integer i such that $|S_1 \cap \{x_i, \bar{x}_i\}| \neq 1$. First, if $|S_1 \cap \{x_i, \bar{x}_i\}| = 0$ for some i , then $M_{u_i}(S_1) = 0$ and $M_{u_i}(S_2) = 1$ (because $z \in S_2$). Next, if $|S_1 \cap \{x_i, \bar{x}_i\}| = 2$ for some i , then it must have $|S_1 \cap \{x_k, \bar{x}_k\}| = 0$ for some k , because $|S_1| \leq n$. Then, again $M_{u_k}(S_1) = 0 \neq 1 = M_{u_k}(S_2)$. It follows that H separates S_1 from S_2 .

Subcase 2.2. $|S_1 \cap \{x_i, \bar{x}_i\}| = 1$ for all $i \in \{1, 2, \dots, n\}$. Define a Boolean assignment $\tau : \{x_1, x_2, \dots, x_n\} \rightarrow \{\text{TRUE}, \text{FALSE}\}$ by $\tau(x_i) = \text{TRUE}$ if and only if $x_i \in S_1$. This is further divided into two subcases:

Subcase 2.2.1. τ satisfies the conjunction C . Since C is an implicant of $\varphi = T_1 + T_2 + \dots + T_m$, τ must satisfy some T_j , $1 \leq j \leq m$. Thus, $T_j \subseteq S_1$: For any $x_i \in T_j$, $\tau(x_i) = \text{TRUE}$, and so $x_i \in S_1$, and for any $\bar{x}_i \in T_j$, $\tau(\bar{x}_i) = \text{FALSE}$, and so $\bar{x}_i \in S_1$. It follows that $M_{t_j}(S_1) = 0$ since $M_{t_j} \cap T_j = \emptyset$. On the other hand, $M_{t_j}(S_2) = 1$ since $z \in M_{t_j} \cap S_2$. So, M_{t_j} , and hence H , separates S_1 from S_2 .

Subcase 2.2.2. τ does not satisfy C . Then, for some literal $\ell_j \in C$, $\tau(\ell_j) = 0$. Thus, $\ell_j \notin S_1$, and $M_{c_j}(S_1) = 0$. On the other hand, $M_{c_j}(S_2) = 1$ since $z \in M_{c_j}$. Thus, M_{c_j} , which is a row in H , separates S_1 from S_2 .

Conversely, assume that H is a $(3n + m + k) \times (2n + 1)$ submatrix of M that contains the first $3n + m$ rows of M and is \bar{d} -separable. Let C be the conjunction of literals ℓ_j for which M_{c_j} is a row in H . Then, obviously, $|C| = k$. Now the following claim is made: C is an implicant of φ .

Let $\tau : \{x_1, x_2, \dots, x_n\} \rightarrow \{\text{TRUE}, \text{FALSE}\}$ be a Boolean assignment that satisfies C . It will be showed that τ satisfies φ . Let

$$S_1 = \{x_i \mid \tau(x_i) = \text{TRUE}\} \cup \{\bar{x}_i \mid \tau(x_i) = \text{FALSE}\},$$

and $S_2 = S_1 \cup \{z\}$. Then, S_1 and S_2 can be separated by some row in H . Since $S_2 = S_1 \cup \{z\}$, they are not separable by a row M_{x_i} or $M_{\bar{x}_i}$, for any $i = 1, 2, \dots, n$. In addition, since $|S_1 \cap \{x_i, \bar{x}_i\}| = 1$ for all $i = 1, 2, \dots, n$, they cannot be separated by row M_{u_i} , for any $i = 1, 2, \dots, n$. Furthermore, note that for any literal $\ell_j \in C$, $\tau(\ell_j) = 1$ and so $\ell_j \in S_1$ and $M_{c_j}(S_1) = M_{c_j}(S_2) = 1$. Thus, S_1 and S_2 cannot be separated by any row M_{c_j} of H .

Therefore, S_1 and S_2 must be separable by a row M_{t_j} , for some $j = 1, 2, \dots, m$. That is, $M_{t_j}(S_1) = 0 \neq 1 = M_{t_j}(S_2)$. Since M_{t_j} contains the complements of the literals in T_j , $T_j \subseteq S_1$. It follows that τ satisfies the term T_j , and hence φ . \square

3.4 Conclusion

In this section, the computational complexities of problems related to nonunique probe selection are presented. The problem of verifying the minimality of a \bar{d} -separable matrix is showed to be DP -complete, and hence is intractable, unless $DP = P$. For the problem of finding a minimum \bar{d} -separable submatrix, it is conjectured to be Σ_2^P -complete and, hence, is even more difficult than the minimal \bar{d} -separability problem. To support this conjecture, the problem \bar{d} -SSRR, which is a little more general than the minimum \bar{d} -separable submatrix problem, is shown to be Σ_2^P -complete. The complexity of the original problem MIN- \bar{d} -SS remains open.

4 Parameterized Complexity Results on NGT

Given an $m \times n$ binary matrix and a positive integer d , to decide whether the matrix is d -separable (\bar{d} -separable, or d -disjunct) is a basic problem in nonadaptive group testing (NGT). They are known to be coNP-complete in classical complexity theory [18]. Thus, one should not expect any polynomial time algorithm to solve any of them. However, since in most applications $d \ll n$, an interesting question is whether there are efficient algorithms solving the above decision problems for small values of d .

In [12] by studying the parameterized complexity of the above three problems with d as the parameter, the authors gave a negative answer to the above question. More formally, they studied the parameterized decision problems p -DISJUNCTNESS-TEST, p -SEPARABILITY-TEST, and p -SEPARABILITY*-TEST defined as follows (where \mathcal{N} denotes the set of positive integers).

p -DISJUNCTNESS-TEST

Instance: A binary matrix \mathcal{M} and $d \in \mathcal{N}$.

Parameter: d .

Problem: Decide whether \mathcal{M} is d -disjunct.

p -SEPARABILITY-TEST

Instance: A binary matrix \mathcal{M} and $d \in \mathcal{N}$.

Parameter: d .

Problem: Decide whether \mathcal{M} is d -separable.

p -SEPARABILITY*-TEST

Instance: A binary matrix \mathcal{M} and $d \in \mathcal{N}$.

Parameter: d .

Problem: Decide whether \mathcal{M} is \bar{d} -separable.

The main results obtained in [12] will be presented in this section; they are summarized in the following theorem.

Theorem 8 *p -DISJUNCTNESS-TEST, p -SEPARABILITY*-TEST, and p -SEPARABILITY-TEST are all co-W[2]-complete.*

W[2] is the parameterized complexity class at the second level of the W-hierarchy, and co-W[2] is the class of all parameterized problems whose complements are in W[2]. They will be formally introduced in the sequel.

Theorem 8 indicates that, given an $m \times n$ binary matrix and a positive integer d , a deterministic algorithm with running time $f(d) \times (mn)^{O(1)}$ (where f is an arbitrary computable function) to decide whether the matrix is d -separable (\bar{d} -separable, or d -disjunct) does not exist unless the class W[2] collapses to FPT (the class of all fixed-parameter tractable problems), which is commonly conjectured to be false.

4.1 Preliminaries

Before proving [Theorem 8](#), the notions of fixed-parameter tractability, relational structures, first-order logic, and the W-hierarchy of parameterized complexity classes are formally introduced.

4.1.1 Fixed-Parameter Tractability

The theory of *fixed-parameter tractability* [16, 27] has received considerable attention in recent years, for both theoretical research and practical computation. The notations and conventions in [27] are adopted here. Let Σ denote a fixed finite alphabet. A *parameterization* of Σ^* is a polynomial time computable mapping $\kappa : \Sigma^* \rightarrow \mathcal{N}$. A *parameterized problem* (over Σ) is a pair (Q, κ) consisting of a set $Q \subseteq \Sigma^*$ and a parameterization κ of Σ^* .

An algorithm A with input alphabet Σ is an *fpt-algorithm with respect to κ* , if for every $x \in \Sigma^*$ the running time of A on input x is at most $f(\kappa(x))|x|^{O(1)}$, for some computable function f . A parameterized problem (Q, κ) is *fixed-parameter tractable* if there is an fpt-algorithm with respect to κ that decides Q . The key point of the definition of fpt-algorithm is that the superpolynomial growth of running time is confined to the parameter $\kappa(x)$, which is usually known to be comparatively small. The class of all fixed-parameter tractable problems is denoted by FPT.

Many NP-hard problems such as the VERTEX COVER problem [8] and the ML TYPE-CHECKING problem [37] have been shown to be fixed-parameter tractable. On the other hand, there is strong theoretical evidence that certain well-known parameterized problems, for instance the INDEPENDENT SET problem and the DOMINATING SET problem, are not fixed-parameter tractable [16]. This evidence is provided, similar to the theory of NP-completeness, via a completeness theory based on the following notion of reductions: Let (Q, κ) and (Q', κ') be parameterized problems over alphabets Σ and Σ' , respectively. An *fpt-reduction* from (Q, κ) to (Q', κ') is a mapping $R : \Sigma^* \rightarrow (\Sigma')^*$ such that:

1. For all $x \in \Sigma^*, x \in Q$ if and only if $R(x) \in Q'$.
2. R is computable by an fpt-algorithm (with respective to κ). That is, there is a computable function f such that $R(x)$ is computable in time $f(\kappa(x))|x|^{O(1)}$.
3. There is a computable function $g : \mathcal{N} \rightarrow \mathcal{N}$ such that $\kappa'(R(x)) \leq g(\kappa(x))$, for all $x \in \Sigma^*$.

In the above definition, the last requirement is to ensure that class FPT is closed under fpt-reductions, that is, if a parameterized problem (Q, κ) is reducible to another parameterized problem (Q', κ') and $(Q', \kappa') \in \text{FPT}$, then $(Q, \kappa) \in \text{FPT}$.

4.1.2 Relational Structures

In later discussions, the conventions in descriptive complexity theory are adopted, in which instances of decision problems are viewed as structures of some vocabulary instead of languages over some finite alphabet.

A (*relational*) *vocabulary* τ is a set of relation symbols. Each relation symbol $R \in \tau$ has an *arity* $\text{arity}(R) \geq 1$. A *structure* \mathcal{A} of vocabulary τ consists of a set A called the *universe* and an interpretation $R^{\mathcal{A}} \subseteq A^{\text{arity}(R)}$ of each relation symbol $R \in \tau$. For a tuple $\bar{a} \in A^{\text{arity}(R)}$, write $R^{\mathcal{A}}\bar{a}$ (or $\bar{a} \in R^{\mathcal{A}}$) to denote that \bar{a} belongs to the relation $R^{\mathcal{A}}$. Here only nonempty finite vocabularies and structures with a finite universe are considered.

Recall that a hypergraph is a pair $H = (V, E)$ consisting of a set V of vertices and a set E of hyperedges. Each hyperedge is a subset of V . Graphs are hypergraphs with hyperedges of cardinality two. The following example illustrates how to represent a hypergraph using a relational structure.

Example 1 Let τ_{HG} be the vocabulary consisting of the unary relation symbols *VERT* and *EDGE* and the binary relation symbol *I*. A hypergraph $H = (V, E)$ can be represented by a relational structure \mathcal{H} of vocabulary τ_{HG} as follows:

- The universe of \mathcal{H} is $V \cup E$.
- $\text{VERT}^{\mathcal{H}} := V$ and $\text{EDGE}^{\mathcal{H}} := E$.
- $I^{\mathcal{H}} := \{(v, e) : v \in V, e \in E, \text{ and } v \in e\}$ is the *incidence relation*.

4.1.3 First-Order Logic

First the syntax of first-order logic is briefly recalled. Let τ be a vocabulary. *Atomic* first-order formulas of vocabulary τ are of the form $x = y$ or $Rx_1 \dots x_\ell$, where $R \in \tau$ is ℓ -ary (i.e., has arity ℓ) and x, y, x_1, \dots, x_ℓ are variables. First-order formulas of vocabulary τ are built from atomic formulas using Boolean connectives \wedge (and), \vee (or), \neg (negation), together with the existential and universal quantifiers \exists and \forall . The connectives \rightarrow (implication) and \leftrightarrow (equivalence) are not part of the language defining first-order formulas, but they are used as abbreviations: $\varphi \rightarrow \psi$ stands for $\neg\varphi \vee \psi$, and $\varphi \leftrightarrow \psi$ stands for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.

A variable x is called a free variable of φ if x occurs in φ but is not in the scope of a quantifier binding x . Write $\varphi(x_1, \dots, x_\ell)$ to indicate that all free variables of φ belong to set $\{x_1, \dots, x_\ell\}$. A formula without free variables is called a *sentence*. Let both Σ_0 and Π_0 denote the class of quantifier-free first-order formulas. For $t \geq 0$, let Σ_{t+1} be the class of all formulas $(\exists x_1 \dots \exists x_\ell)\varphi$, where $\varphi \in \Pi_t$, and let Π_{t+1} be the class of all formulas $(\forall x_1 \dots \forall x_\ell)\varphi$, where $\varphi \in \Sigma_t$.

For formulas of second-order logic, in addition to the individual variables, they may also contain *relation variables*; each of the relation variables has a prescribed arity. Here lowercase letters (e.g., x, y, z) are used to denote individual variables, and uppercase letters (e.g., X, Y, Z) are used to denote relation variables. As in [27], for convenience *free* relation variables are allowed to be in first-order formulas, since the crucial difference between first-order and second-order logic is not that second-order formulas can have relation variables, but that second-order formulas can quantify over relations. Therefore, here the syntax of first-order logic is enhanced by

including new atomic formulas of the form $Xx_1 \dots x_\ell$, where X is an ℓ -ary relation variable. The meaning of formula $Xx_1 \dots x_\ell$ is: The tuple of elements interpreting (x_1, \dots, x_ℓ) is contained in the relation interpreting the relation variable X . The classes such as Σ_t and Π_t are also extended to include formulas with free relation variables. It is worth emphasizing again that in first-order logic quantification over relation variables is now allowed.

4.1.4 W-Hierarchy

First a brief introduction to the W-hierarchy of parameterized complexity classes is given. Roughly speaking, the W-hierarchy classifies problems according to the syntactic form of their definitions, and the definitions are formalized using languages of mathematical logic. The W-hierarchy can be defined in several different ways; here the following definition based on the *weighted Fagin-defined problems* is adopted.

Let $\varphi(X)$ be a first-order formula with a free relation variable X of arity s . Define $p\text{-WD}_\varphi$ to be the following parameterized decision problem.

$p\text{-WD}_\varphi$:

Instance: A structure \mathcal{A} and $k \in \mathcal{N}$.

Parameter: k .

Problem: Decide whether there is a relation $S \subseteq A^s$ of cardinality k such that $\mathcal{A} \models \varphi(S)$.

Here, $\mathcal{A} \models \varphi(S)$ stands for that structure \mathcal{A} satisfies sentence $\varphi(S)$ (or, \mathcal{A} is a model of $\varphi(S)$), and S is called a *solution* for φ in structure \mathcal{A} . The readers are referred to, for example, Sect. 4.2 of [27], for more detailed introduction to the semantics of first-order formulas.

For a class Φ of first-order formulas, let $p\text{-WD-}\Phi$ be the class of all parameterized problems $p\text{-WD}_\varphi$ with $\varphi \in \Phi$. For $t \geq 1$, define $W[t] := [p\text{-WD-}\Pi_t]^{\text{fpt}}$, which is the class of all parameterized problems that are fpt-reducible to some problems in $p\text{-WD-}\Pi_t$. The classes $W[t]$, for $t \geq 1$, form the W-hierarchy. Thus, the levels of W-hierarchy essentially correspond to the number of alternations between universal and existential quantifiers in the definitions of their complete problems. Problems hard for $W[1]$ or higher class are assumed not to be fixed-parameter tractable. For instance, the INDEPENDENT SET problem is $W[1]$ -complete, and the DOMINATING SET problem is $W[2]$ -complete.

For a parameterized problem (Q, κ) over the alphabet Σ , let $(Q, \kappa)^c$ denote its complement, that is, the parameterized problem $(\Sigma^* \setminus Q, \kappa)$. Let C be a parameterized complexity class. Then $\text{co-}C$ is defined to be the class of all parameterized problems (Q, κ) such that $(Q, \kappa)^c \in C$. Clearly, $\text{FPT} = \text{co-FPT}$. From the definition of fpt-reductions, it is easy to see that if class C is closed under fpt-reductions, so is $\text{co-}C$. In particular, each class $W[t]$, $t \geq 1$, gives rise to a new parameterized complexity class $\text{co-}W[t]$. Also, it is easy to prove that if (Q, κ) is complete in parameterized complexity class C under fpt-reductions, then $(Q, \kappa)^c$ is complete in class $\text{co-}C$ under fpt-reductions.

4.2 Proof of Theorem 8

Now the proof of [Theorem 8](#) is ready to presented. For a binary matrix M , let R_M be the set consisting of all rows in M , and let C_M be the set consisting of all columns in M .

Relational Structure for a Binary Matrix. Let τ_{BM} be the vocabulary consisting of the unary relation symbols ROW and $COLUMN$ and the binary relation symbol I . Then, the binary matrix M can be represented by a structure \mathcal{M} of vocabulary τ_{BM} , where the universe of \mathcal{M} is $R_M \cup C_M$, and the interpretations for the relation symbols in τ_{BM} are as follows:

- $ROW^{\mathcal{M}} := R_M$.
- $COLUMN^{\mathcal{M}} := C_M$.
- $I^{\mathcal{M}} := \{(r, c) : r \in R_M, c \in C_M, \text{ and } M(r, c) = 1\}$, which is the incidence relation.

The proof of [Theorem 8](#) is partitioned into the following six lemmas:

Lemma 5 p -DISJUNCTNESS-TEST $\in \text{co-W[2]}$.

Proof Consider the following complement problem of p -DISJUNCTNESS-TEST.

p -NONDISJUNCTNESS-TEST

Instance: A binary matrix \mathcal{M} and $d \in \mathcal{N}$.

Parameter: d .

Problem: Decide whether \mathcal{M} is NOT d -disjunct.

A Π_2 formula $nondisj(X)$ with a free relation variable X of arity 2 will be defined, and p -NONDISJUNCTNESS-TEST will be showed to be equal to p -WD $_{nondisj(X)}$ (see [Sect. 4.1.4](#) for the definition of problem p -WD $_{\varphi}$). This implies that p -NONDISJUNCTNESS-TEST is in p -WD- Π_2 , therefore is in W[2].

A binary matrix is not d -disjunct if and only if there is a set D of d columns and another column $c \notin D$ such that $U(D)$ covers c . The idea here is assuming that the solution S to X is of the form $\{(c_i, c) : c_i \in D\}$. Therefore, X should be a binary relation variable, and the solution S to X should have cardinality d .

Define

$$\chi_1 := \forall c_1 \forall c_2 (Xc_1c_2 \rightarrow (COLUMN c_1 \wedge COLUMN c_2 \wedge (c_1 \neq c_2))),$$

and define

$$\chi_2 := \forall c_3 \forall c_4 \forall c_5 \forall c_6 ((Xc_3c_4 \wedge Xc_5c_6) \rightarrow (c_4 = c_6)).$$

Here χ_1 and χ_2 are to guarantee that the solution S to X of cardinality d has the form $\{(c_1, c), \dots, (c_d, c)\}$, where $c \in C_M$, $c_i \in C_M$, and $c_i \neq c$, for $1 \leq i \leq d$. Define

$$\text{nondisj}'(X) := \forall r \exists c_7 \exists c_8 (ROWr \rightarrow (Xc_7c_8 \wedge (Irc_8 \rightarrow Irc_7))).$$

Here $\text{nondisj}'(X)$ is to guarantee that the solution S to X satisfies that the union of columns in $\{c_i : (c_i, c) \in S\}$ covers c (so that \mathcal{M} is not d -disjunct). Finally, define

$$\text{nondisj}(X) := \chi_1 \wedge \chi_2 \wedge \text{nondisj}'(X),$$

which is equivalent to a Π_2 formula.

From the above definition, clearly if there exists a relation $S \subseteq C_M^2$ of cardinality d such that $\mathcal{M} \models \text{nondisj}(S)$, then \mathcal{M} is not d -disjunct. On the other hand, if \mathcal{M} is not d -disjunct, then there exist a subset D of d columns c_1, \dots, c_d and another column $c \notin D$ such that c is covered by $U(D)$. It is not hard to verify that the relation $S = \{(c_1, c), \dots, (c_d, c)\}$ satisfies $\mathcal{M} \models \text{nondisj}(S)$. Therefore, \mathcal{M} is not d -disjunct if and only if there exists a relation S of cardinality d such that $\mathcal{M} \models \text{nondisj}(S)$. That is, p -NONDISJUNCTNESS-TEST is p -WD _{$\text{nondisj}(X)$} . Thus, p -NONDISJUNCTNESS-TEST \in W[2], and so p -DISJUNCTNESS-TEST \in co-W[2]. \square

Lemma 6 p -SEPARABILITY-TEST \in co-W[2].

Proof Consider the following complement problem of p -SEPARABILITY-TEST.

p -NONSEPARABILITY-TEST

Instance: A binary matrix \mathcal{M} and $d \in \mathcal{N}$.

Parameter: d .

Problem: Decide whether \mathcal{M} is NOT d -separable.

A formula $\text{nonsep}(Y)$ with a free relation variable Y of arity 2 will be defined, and p -NONSEPARABILITY-TEST will be shown to be equal to p -WD _{$\text{nonsep}(Y)$} .

A binary matrix is not d -separable if and only if there exist two distinct subsets D_1 and D_2 , each contains d columns such that $U(D_1) = U(D_2)$. Assume that $D_1 = \{c_{11}, c_{12}, \dots, c_{1d}\}$ and $D_2 = \{c_{21}, c_{22}, \dots, c_{2d}\}$. The idea is to assume that the solution S to Y is of the form $\{(c_{11}, c_{21}), (c_{12}, c_{22}), \dots, (c_{1d}, c_{2d})\}$, and so Y should be a binary relation variable, and the solution S to Y should have cardinality d . Define the formula $\text{nonsep}(Y)$ such that it satisfies the following: There exists a relation $S \subseteq C_M^2$ of cardinality d such that $\mathcal{M} \models \text{nonsep}(S)$ if and only if \mathcal{M} is not d -separable.

Define

$$\chi_3 := \forall c_1 \forall c_2 (Yc_1c_2 \rightarrow (\text{COLUMN}c_1 \wedge \text{COLUMN}c_2)),$$

$$\chi_4 := \forall c_3 \forall c_4 \forall c_5 \forall c_6 ((Yc_3c_4 \wedge Yc_5c_6) \rightarrow ((c_3 = c_5) \leftrightarrow (c_4 = c_6))),$$

and

$$\chi_5 := \exists c_7 \exists c_8 \forall c_9 ((Yc_7c_8 \wedge \neg Yc_9c_7)).$$

Here χ_3 is to guarantee that the relation variable $Y \subseteq C_M^2$; χ_4 is to build a bijection between the first component of elements in S and the second component of elements in S , which guarantees that the two subsets $\{c_{1i} : \exists c \text{ s.t. } (c_{1i}, c) \in S\}$ and $\{c_{2j} : \exists c \text{ s.t. } (c, c_{2j}) \in S\}$ (which intend to be D_1 and D_2 , respectively) have the same cardinality; χ_5 is to guarantee that the two subsets $\{c_{1i} : \exists c \text{ s.t. } (c_{1i}, c) \in S\}$ and $\{c_{2j} : \exists c \text{ s.t. } (c, c_{2j}) \in S\}$ are distinct from each other. Define

$$\begin{aligned} \text{nonsep}'(Y) &:= \forall r(\text{ROWr} \rightarrow ((\exists c_{10} \exists c_{11} Y c_{10} c_{11} \wedge Irc_{10}) \\ &\quad \Leftrightarrow (\exists c_{12} \exists c_{13} Y c_{12} c_{13} \wedge Irc_{13}))), \end{aligned}$$

which is to guarantee that the solution S to Y satisfies that the union of columns in $\{c_{1i} : \exists c \text{ s.t. } (c_{1i}, c) \in S\}$ is equal to the union of columns in $\{c_{2j} : \exists c \text{ s.t. } (c, c_{2j}) \in S\}$. From basic logic computation, it is not hard to verify that $\text{nonsep}'(Y)$ is a Π_2 formula with free relation variable Y . Finally, define

$$\text{nonsep}(Y) := \chi_3 \wedge \chi_4 \wedge \chi_5 \wedge \text{nonsep}'(Y).$$

From the above definition of $\text{nonsep}(X)$, if a relation $S \subseteq C_M^2$ of cardinality d satisfies that $\mathcal{M} \models \text{nonsep}(S)$, then the two subsets $\{c_{1i} : \exists c \text{ s.t. } (c_{1i}, c) \in S\}$ and $\{c_{2j} : \exists c \text{ s.t. } (c, c_{2j}) \in S\}$ both contain d columns of \mathcal{M} and are distinct from each other; moreover, their unions are identical. This implies that \mathcal{M} is not d -separable. On the other hand, if \mathcal{M} is not d -separable, then there exist two distinct subsets D_1 and D_2 , each contains d columns such that $U(D_1) = U(D_2)$. Assume that $D_1 = \{c_{11}, \dots, c_{1d}\}$ and $D_2 = \{c_{21}, \dots, c_{2d}\}$. It is not hard to verify that the relation

$$S = \{(c_{11}, c_{21}), (c_{12}, c_{22}), \dots, (c_{1d}, c_{2d})\}$$

satisfies $\mathcal{M} \models \text{nonsep}(S)$.

From above, there exists a relation $S \subseteq C_M^2$ of cardinality d such that $\mathcal{M} \models \text{nonsep}(S)$ if and only if \mathcal{M} is not d -separable; therefore, p -NONSEPARABILITY-TEST is p -WD _{$\text{nonsep}(Y)$} . χ_3 and χ_4 are Π_1 formulas; χ_5 is a Σ_2 formula; $\text{nonsep}'(Y)$ is a Π_2 formula; therefore,

$$\text{nonsep}(Y) = \chi_3 \wedge \chi_4 \wedge \chi_5 \wedge \text{nonsep}'(Y)$$

is equivalent to a Σ_3 formula, which implies that p -NONSEPARABILITY-TEST is in p -WD- Σ_3 . Here the following fact is applied: p -WD- $\Sigma_3 \subseteq p$ -WD- Π_2 (more generally, p -WD- $\Sigma_{t+1} \subseteq p$ -WD- Π_t , for $t \geq 1$). The main idea to prove this conclusion is not complicated; the reader is referred to, e.g., Proposition 5.4 in [27] for the proof). Thus, p -NONSEPARABILITY-TEST $\in W[2]$, and so p -SEPARABILITY-TEST $\in \text{co-W}[2]$. \square

Lemma 7 p -SEPARABILITY*-TEST $\in \text{co-W}[2]$.

Proof Since $\text{W}[2]$ is closed under fpt-reductions, so is $\text{co-W}[2]$. It will be shown next that $p\text{-SEPARABILITY}^*\text{-TEST}$ is fpt-reducible to $p\text{-SEPARABILITY-TEST}$. Since the latter is in $\text{co-W}[2]$ ([Lemma 6](#)), this implies that $p\text{-SEPARABILITY}^*\text{-TEST} \in \text{co-W}[2]$. The reduction can be obtained immediately from the following fact ([Lemma 2.1.6 in \[17\]](#)): A binary matrix M' containing a zero column is d -separable if and only if the matrix M obtained by removing this zero column from M' is \bar{d} -separable.

Let (M, d) be an instance of $p\text{-SEPARABILITY}^*\text{-TEST}$, where M is a binary matrix and the parameter d is a positive integer. Map (M, d) to (M', d) , where M' is obtained by adding a zero column to M . From the above lemma, $(M, d) \in p\text{-SEPARABILITY}^*\text{-TEST}$ if and only if $(M', d) \in p\text{-SEPARABILITY-TEST}$. It is easy to see that this is an fpt-reduction from $p\text{-SEPARABILITY}^*\text{-TEST}$ to $p\text{-SEPARABILITY-TEST}$.

Lemma 8 $p\text{-DISJUNCTNESS-TEST}$ is $\text{co-W}[2]$ -complete.

Proof A *hitting set* in a hypergraph $\mathcal{H} = (V, E)$ is a set T of vertices that intersects each hyperedge, that is, $T \cap e \neq \emptyset$ for all $e \in E$. The classical **HITTING-SET** problem is to find a hitting set of a given cardinality k in a given hypergraph \mathcal{H} , which is known to be NP-complete. The following parameterized hitting set problem is $\text{W}[2]$ -complete (see, e.g., [Theorem 7.14 in \[27\]](#)).

$p\text{-HITTING-SET}$

Instance: A hypergraph \mathcal{H} and $k \in \mathcal{N}$.

Parameter: k .

Problem: Decide whether \mathcal{H} has a hitting set of k vertices.

An fpt-reduction from $p\text{-HITTING-SET}$ to $p\text{-NONDISJUNCTNESS-TEST}$ will be given, based on an idea similar to that in [\[18\]](#). Let (\mathcal{H}, k) with $\mathcal{H} = (V, E)$ be an instance of $p\text{-HITTING-SET}$, where $V = \{1, \dots, n\}$, $E = \{e_1, \dots, e_m\}$, and each e_i , $1 \leq i \leq m$, is a subset of V . Define $d = k$, and define an $(n+m) \times (n+1)$ binary matrix M with rows R_i as follows (here each row is represented as a subset of the set of all columns $\{1, 2, \dots, n+1\}$, in the most natural way):

$$\begin{aligned} R_i &= \{i\}, & i &= 1, \dots, n; \\ R_{n+j} &= e_j \cup \{n+1\}, & j &= 1, \dots, m. \end{aligned}$$

First, assume that \mathcal{H} has a hitting set $T \subseteq V$ of size k . Consider the subset $S_1 = T$ of columns of M . Since T is a hitting set of \mathcal{H} , $U(S_1)$ covers column $n+1$. Notice that $|S_1| = d$ and column $n+1$ is not in S_1 , M is not d -disjunct.

Conversely, assume that M is not d -disjunct. Then, there exist a subset S_1 of d columns in $\{1, 2, \dots, n+1\}$ and another column $c \notin S_1$ such that $U(S_1)$ covers c . From the way the first n rows of matrix M are defined, c can only be column $n+1$. Thus, column $n+1$ is not in S_1 . Set $T = S_1$, then $|T| = k$, and T is a subset of V . Since $U(S_1)$ covers column $n+1$, it is easy to see that T is a hitting set of \mathcal{H} .

It is not hard to verify that the above is an fpt-reduction. Therefore, p -NONDISJUNCTNESS-TEST is W[2]-complete, and so p -DISJUNCTNESS-TEST is co-W[2]-complete. \square

Lemma 9 p -SEPARABILITY*-TEST is co-W[2]-complete.

Proof To prove the lemma, an ftp-reduction from p -HITTING-SET to p -NONSEPARABILITY*-TEST will be given. For an instance (\mathcal{H}, k) of p -HITTING-SET, define matrix M in the same way as in the proof of Lemma 8, and define $d = k + 1$. Next the correctness of this reduction will be shown.

First, assume that \mathcal{H} has a hitting set $T \subseteq V$ of size k . Consider the following two subsets of columns in M : $S_1 = T$ and $S_2 = T \cup \{n + 1\}$. Then, for $1 \leq i \leq n$, it is obvious that $U(S_1)_i = U(S_2)_i$; for $n < i \leq n + m$, since T is a hitting set of \mathcal{H} , $U(S_1)_i = 1 = U(S_2)_i$. Notice that $|S_1|, |S_2| \leq d$ and $S_1 \neq S_2$, M is not \bar{d} -separable.

Conversely, assume that M is not \bar{d} -separable. Then, there exist two subsets S_1 and S_2 of columns in $\{1, 2, \dots, n + 1\}$ such that $|S_1|, |S_2| \leq d$, $S_1 \neq S_2$, and $U(S_1) = U(S_2)$. Since $U(S_1)_i = U(S_2)_i$ for $1 \leq i \leq n$, it follows that

$$S_1 \cap \{1, \dots, n\} = S_2 \cap \{1, \dots, n\}.$$

To have $S_1 \neq S_2$, column $n + 1$ must belong to exactly one of S_1 and S_2 . Without loss of generality, assume that $n + 1 \notin S_1$ and $n + 1 \in S_2$. Set $T = S_1$, then

$$|T| = |S_1| = |S_2| - 1 \leq d - 1 = k,$$

and T is a subset of V . From $U(S_1)_i = U(S_2)_i = 1$ for $n < i \leq n + m$, T is a hitting set of \mathcal{H} .

Therefore, p -NONSEPARABILITY*-TEST is W[2]-complete, and so p -SEPARABILITY*-TEST is co-W[2]-complete.

Lemma 10 p -SEPARABILITY-TEST is co-W[2]-complete.

Proof Since as proved before in Lemma 7 that p -SEPARABILITY*-TEST is fpt-reducible to p -SEPARABILITY-TEST, and in Lemma 9 that p -SEPARABILITY*-TEST is co-W[2]-complete, p -SEPARABILITY-TEST is co-W[2]-hard. Together with Lemma 6 that p -SEPARABILITY-TEST \in co-W[2], it is obtained that p -SEPARABILITY-TEST is co-W[2]-complete.

4.3 Discussion

In this section, the parameterized complexity is established for the following three basic problems in pooling design: Given an $m \times n$ binary matrix and a positive integer d , to decide whether the matrix is d -separable (\bar{d} -separable), or

d -disjunct). It is showed that these problems are co-W[2]-complete; thus, do not admit algorithms with running time $f(d) \times (mn)^{O(1)}$ for any computable function f . The best known algorithms for the above general problems are all in a brute-force manner. It is interesting to investigate that whether these problems admit better algorithms. For instance, are there algorithms with running time $n^{o(d)} O(m)$ to solve these problems when d is small compared to n ?

5 Upper Bounds on the Minimum Number of Rows of Disjunct Matrices

A 0-1 matrix is d -*disjunct* if no column is covered by the union of any d other columns, where the union means the bitwise Boolean sum of these d column vectors. In other words, a 0-1 matrix is called d -disjunct if for any column C and any d other columns, there exists at least one row such that the row has value 1 at column C and value 0 at all d other columns. The same structure is also called *cover-free family* [25, 29, 53] in combinatorics and *superimposed code* [22, 23, 34] in information theory. It is called a d -disjunct matrix in group testing [17, 32, 39]. A 0-1 matrix is $(d; z)$ -*disjunct* [22, 39] if for any column C and any d other columns, there exist at least z rows such that each of them has value 1 at column C and value 0 at all the other d columns. Thus, d -disjunct is $(d; 1)$ -disjunct. Besides other applications, d -disjunct and $(d; z)$ -disjunct matrices form the basis for error-free and error-tolerant nonadaptive group testing algorithms, respectively. These algorithms have applications in many practical areas such as DNA library screening [3, 6, 17, 43] and multi-access communications [57].

Let $t(d, n)$ denote the minimum number of rows required by a d -disjunct matrix with n columns. The bounds on $t(d, n)$ have been extensively studied in the fields of combinatorics, information theory, and group testing, under different terminologies. For lower bounds, it is known that $t(d, n) = \Omega\left(\frac{d^2 \log n}{\log d}\right)$ [21, 29, 53]. In particular, D'yachkov and Rykov [21] proved that $t(d, n) \geq \frac{d^2}{2 \log d} (1 + o(1)) \log n$, which is the best lower bound so far. For upper bounds on $t(d, n)$, it is known that $t(d, n) = O(d^2 \log n)$ [2, 22, 32, 33, 45, 50]. In [22], Dyachkov, Rykov, and Rashad obtained the following asymptotic upper bound on $t(d, n)$ with a rather involved proof, which is currently the best.

Theorem 9 (Dyachkov, Rykov, and Rashad [22]) *For d constant and $n \rightarrow \infty$,*

$$t(d, n) \leq \frac{d}{A_d} [1 + o(1)] \log n,$$

where

$$A_d = \max_{0 \leq p \leq 1} \max_{0 \leq P \leq 1} \left\{ -(1 - P) \log(1 - p^d) + d \left[P \log \frac{p}{P} + (1 - P) \log \frac{1 - p}{1 - P} \right] \right\}.$$

Moreover, $A_d \rightarrow \frac{1}{d \log e}$ as $d \rightarrow \infty$.

For $(d; z)$ -disjunct matrices, let $t(d, n; z)$ denote the minimum number of rows required by a $(d; z)$ -disjunct matrix with n columns. For given d and z , D'yachkov, Rykov, and Rashad [22] studied the value of $\lim_{n \rightarrow \infty} \frac{\log n}{t}$, among others, and they proved that $t(d, n; z) \geq c \left[\frac{d^2 \log n}{\log d} + (z - 1)d \right]$ where c is a constant.

In [13], by using q -ary $(d, 1)$ -disjunct matrices [17, 20] and the probabilistic method (see, e.g., [1]), the authors gave a very short proof for the currently best upper bound on $t(d, n)$. In contrast to the previous result in [22] (**Theorem 9**), which is an asymptotic upper bound, the upper bound on $t(d, n)$ in [13] does not contain the asymptotic term $o(1)$. Also, the method in [13] is generalized to obtain a new upper bound on $t(d, n; z)$. These results will be presented in this section.

5.1 Upper Bounds on $t(d, n)$

Theorem 10 For $n > d \geq 1$,

$$t(d, n) \leq \frac{d + 1}{B_d} \log n,$$

where $B_d = \max_{q > 1} \frac{-\log \left[1 - \left(1 - \frac{1}{q}\right)^d \right]}{q}$. Moreover, $B_d \rightarrow \frac{1}{d \log e}$ as $d \rightarrow \infty$.

Before proving the above theorem, the concept of q -ary $(d, 1)$ -disjunct matrix will be first introduced: A matrix is called q -ary $(d, 1)$ -disjunct if it is q -ary, and for any column C and any set D of d other columns, there exists an element in C such that the element does not appear in any column of D in the same row.

As described in [17, 20], one can transform a q -ary $(d, 1)$ -disjunct matrix M to a (binary) d -disjunct matrix M' as follows. Replace each row R_i of M by several rows indexed with entries of R_i . For each entry x of R_i , the row with index x is obtained from R_i by turning all x 's into 1's and all others into 0's. The following fact is useful in later proof, which is the same as **Theorem 1** only using different notations:

Fact 2 (Theorem 3.6.1 in [17]) A $t \times n$ q -ary $(d, 1)$ -disjunct matrix M yields a $t' \times n$ d -disjunct matrix M' with $t' \leq tq$.

Now it is time to present the proof of **Theorem 10**.

Proof of Theorem 10: Given $n > d \geq 1$, first construct a random $t \times n$ q -ary ($q > 1$) matrix M with each entry assigned randomly and uniformly from $\{1, 2, \dots, q\}$, where q and t will be specified later. For each column C and a set D of d other columns, for each element c_i ($i = 1, 2, \dots, t$) of C , the probability that c_i appears in some column of D in the same row is $1 - (1 - \frac{1}{q})^d$. Thus, the probability that every element of C appears in some column of D in the same row is $[1 - (1 - \frac{1}{q})^d]^t$.

M is not $(d, 1)$ -disjunct if and only if there exist a column C and a set D of d other columns such that the above holds. Therefore, the probability that M is not $(d, 1)$ -disjunct is no more than

$$(n - d) \binom{n}{d} [1 - (1 - \frac{1}{q})^d]^t.$$

What follows next is to try to minimize tq , the number of rows of the d -disjunct matrix M' as in Fact 2, under the condition that q and t satisfy

$$n^{d+1} [1 - (1 - \frac{1}{q})^d]^t \leq 1. \quad (1)$$

Notice that Eq. (1) implies

$$(n - d) \binom{n}{d} [1 - (1 - \frac{1}{q})^d]^t < 1;$$

thus, the probability that M is $(d, 1)$ -disjunct is greater than zero. Therefore, by probabilistic argument, Eq. (1) implies the existence of a $t \times n$ q -ary $(d, 1)$ -disjunct matrix, and so a d -disjunct matrix with n columns and at most tq rows.

To satisfy Eq. (1), let

$$t = \frac{(d + 1) \log n}{-\log[1 - (1 - \frac{1}{q})^d]}.$$

Define

$$B_d(q) = \frac{-\log[1 - (1 - \frac{1}{q})^d]}{q},$$

then

$$tq = \frac{(d + 1) \log n}{B_d(q)}.$$

Let q_0 be the point that maximizes $B_d(q)$, and let $B_d = B_d(q_0)$ (one can estimate that $q_0 = \Theta(d)$ and $B_d = \Theta(\frac{1}{d})$, since the proof here can stand alone without this observation; they are put into Lemma 11) in later part. By assigning $q = q_0$, it follows that

$$t(d, n) \leq (tq)|_{q=q_0} = \frac{(d + 1) \log n}{B_d(q_0)} = \frac{(d + 1) \log n}{B_d}.$$

Next estimate B_d as $d \rightarrow \infty$. Since $(1 - \frac{1}{q})^q < \frac{1}{e}$ for $q > 1$,

$$\left(1 - \frac{1}{q}\right)^d < \left(\frac{1}{e}\right)^{\frac{d}{q}} = e^{-\frac{d}{q}},$$

and

$$-\log[1 - (1 - \frac{1}{q})^d] < -\log(1 - e^{-\frac{d}{q}}).$$

It follows that

$$\begin{aligned} B_d(q) &= \frac{-\log[1 - (1 - \frac{1}{q})^d]}{q} \\ &< \frac{-\log(1 - e^{-\frac{d}{q}})}{q} \\ &= \frac{1}{d \ln 2} \left[-\frac{d}{q} \ln(1 - e^{-\frac{d}{q}}) \right]. \end{aligned}$$

Let $x = e^{-\frac{d}{q}}$, then $-\frac{d}{q} = \ln x$, and

$$B_d(q) < \frac{1}{d \ln 2} \ln x \ln(1 - x).$$

Since $\ln x \ln(1 - x)$ achieves its maximum at $x = \frac{1}{2}$, it follows that $B_d(q) < \frac{\ln 2}{d}$ for $q > 1$. Thus, $B_d < \frac{\ln 2}{d}$ for $d \geq 1$. On the other hand, when q satisfies $(1 - \frac{1}{q})^d = \frac{1}{2}$, as $d \rightarrow \infty$, it is easy to see that $\frac{q}{d} \rightarrow \frac{1}{\ln 2}$, and $B_d(q) = \frac{1}{q} \rightarrow \frac{\ln 2}{d}$. Therefore, as $d \rightarrow \infty$, $B_d \rightarrow \frac{\ln 2}{d} = \frac{1}{d \log e}$. \square

Lemma 11 Given $d \geq 1$, let $q_0 = q_0(d)$ be the point that maximizes $B_d(q) = \frac{-\log[1 - (1 - \frac{1}{q})^d]}{q}$ for $q > 1$. Then, as $d \rightarrow \infty$, $q_0(d) = \Theta(d)$, and $B_d = B_d(q_0) = \Theta(\frac{1}{d})$.

Proof Notice that if q_1 satisfies $(1 - \frac{1}{q_1})^d = \frac{1}{2}$, then $q_1 = \Theta(d)$, since $\frac{q_1}{d} \rightarrow \frac{1}{\ln 2}$ as $d \rightarrow \infty$. Moreover, $B_d(q_1) = \frac{1}{q_1} = \Theta(\frac{1}{d})$. The lemma is proved by contradiction. First assume that $q_0 = O(d)$ does not hold, that is, for any $c > 0$ and any $d_0 > 0$, there exists $d > d_0$ such that $q_0(d) > cd$. Then, since $\frac{q_0}{d} > c$, as $c \rightarrow \infty$,

$$\begin{aligned} B_d(q_0)d &= \frac{-\log[1 - (1 - \frac{1}{q_0})^d]}{q_0}d \\ &\sim \frac{-\log[1 - (1 - \frac{d}{q_0})]}{q_0}d \\ &= \frac{\log \frac{q_0}{d}}{\frac{q_0}{d}} \\ &= o(1), \end{aligned}$$

here $a \sim b$ means that $\lim_{c \rightarrow \infty} \frac{a}{b} = 1$. Thus,

$$B_d(q_0) = \frac{o(1)}{d}.$$

However, this is a contradiction since q_0 is the maximum point of $B_d(q)$ and $B_d(q_1) = \Theta(\frac{1}{d})$ with $(1 - \frac{1}{q_1})^d = \frac{1}{2}$.

On the other hand, assume that $q_0 = \Omega(d)$ does not hold, that is, for any $c > 0$ and any $d_0 > 0$, there exists $d > d_0$ such that $q_0(d) < cd$. Then,

$$\begin{aligned} B_d(q_0)d &= \frac{-\log[1 - (1 - \frac{1}{q_0})^d]}{q_0}d \\ &= \frac{-\ln\{1 - [(1 - \frac{1}{q_0})^{q_0}]^{\frac{d}{q_0}}\}}{q_0 \ln 2}d. \end{aligned}$$

Since $0 < (1 - \frac{1}{q_0})^{q_0} < \frac{1}{e}$ for $q_0 > 1$, as $c \rightarrow 0$, $\frac{d}{q_0} > \frac{1}{c} \rightarrow \infty$, and

$$[(1 - \frac{1}{q_0})^{q_0}]^{\frac{d}{q_0}} < e^{-\frac{d}{q_0}} \rightarrow 0.$$

Thus,

$$\begin{aligned} B_d(q_0)d &\sim \frac{[(1 - \frac{1}{q_0})^{q_0}]^{\frac{d}{q_0}}}{q_0 \ln 2}d \\ &= \frac{1}{\ln 2} \frac{d}{q_0} [(1 - \frac{1}{q_0})^{q_0}]^{\frac{d}{q_0}} \\ &< \frac{1}{\ln 2} \frac{d}{q_0} e^{-\frac{d}{q_0}} \\ &= o(1), \end{aligned}$$

which is also a contradiction (here $a \sim b$ means that $\lim_{c \rightarrow 0} \frac{a}{b} = 1$). Therefore, $q_0(d) = \Theta(d)$. Then, $(1 - \frac{1}{q_0})^d < 1$ is $\Theta(1)$, and thus

$$B_d(q_0) = \frac{\Theta(1)}{q_0} = \Theta\left(\frac{1}{d}\right).$$

□

5.2 New Upper Bounds on $t(d, n; z)$

The above method is now generalized to obtain new upper bounds for $(d; z)$ -disjunct matrices, by establishing the following theorem:

Theorem 11 For d, z constants, and $n \rightarrow \infty$,

$$t(d, n; z) \leq \frac{d+1}{B_d} \log n + \frac{z}{B_d} \log \log n + O(1),$$

where $B_d = \max_{q>1} \frac{-\log[1-(1-\frac{1}{q})^d]}{q}$. Moreover, $B_d \rightarrow \frac{1}{d \log e}$ as $d \rightarrow \infty$.

A q -ary matrix is called $(d, 1; z)$ -disjunct if for any column C and any set D of d other columns, there exists at least z elements in C such that each of these elements does not appear in any column of D in the same row. Clearly, by using the same method mentioned above, one can transform a $t \times n$ q -ary $(d, 1; z)$ -disjunct matrix to a $(d; z)$ -disjunct matrix with n columns and at most tq rows.

Proof of Theorem 14: For given n, d , and z , similarly construct a random $t \times n$ q -ary ($q > 1$) matrix M with each entry assigned randomly and uniformly from $\{1, 2, \dots, q\}$; q and t will be specified later. For each column C and a set D of d other columns, for each element c_i of C , the probability that c_i appears in some column of D in the same row is $1 - (1 - \frac{1}{q})^d$. Thus, the probability that there exist $t - z + 1$ elements of C such that each of them appears in some column of D in the same row is at most

$$\binom{t}{t-z+1} [1 - (1 - \frac{1}{q})^d]^{t-z+1} = \binom{t}{z-1} [1 - (1 - \frac{1}{q})^d]^{t-z+1}.$$

M is not $(d, 1; z)$ -disjunct if and only if there exists a column C and a set D of d other columns such that the above holds. Therefore, the probability that M is not $(d, 1; z)$ -disjunct is no more than

$$(n-d) \binom{n}{d} \binom{t}{z-1} [1 - (1 - \frac{1}{q})^d]^{t-z+1}.$$

The goal is to minimize tq , the number of rows of the corresponding $(d; z)$ -disjunct matrix, under the condition that

$$n^{d+1} t^z [1 - (1 - \frac{1}{q})^d]^{t-z} \leq 1. \quad (2)$$

Notice that Eq. (2) implies

$$(n-d) \binom{n}{d} \binom{t}{z-1} [1 - (1 - \frac{1}{q})^d]^{t-z+1} < 1.$$

Thus, the probability that M is $(d, 1; z)$ -disjunct is greater than zero, which similarly implies the existence of a $t \times n$ q -ary $(d, 1; z)$ -disjunct matrix, and a $(d; z)$ -disjunct matrix with n columns and at most tq rows.

Let q_0 be the point that maximizes

$$B_d(q) = \frac{-\log[1 - (1 - \frac{1}{q})^d]}{q}.$$

Assign $q = q_0$. To satisfy Eq. (2), which is equivalent to

$$(d + 1) \log n + z \log t \leq -(t - z) \log[1 - (1 - \frac{1}{q_0})^d],$$

let

$$t = \frac{(d + 1) \log n}{-\log[1 - (1 - \frac{1}{q_0})^d]} + z + t_1.$$

Then, t_1 should satisfy

$$z \log \left\{ \frac{(d + 1) \log n}{-\log[1 - (1 - \frac{1}{q_0})^d]} + z + t_1 \right\} \leq -t_1 \log[1 - (1 - \frac{1}{q_0})^d] \quad (3)$$

Let

$$t_1 = \frac{z \log \log n}{-\log[1 - (1 - \frac{1}{q_0})^d]} + t_2.$$

From Eq. (3), t_2 should satisfy that

$$\begin{aligned} \frac{z}{-\log[1 - (1 - \frac{1}{q_0})^d]} \log & \left\{ \frac{(d + 1)}{-\log[1 - (1 - \frac{1}{q_0})^d]} \right. \\ & \left. + \frac{1}{\log n} \left(\frac{z \log \log n}{-\log[1 - (1 - \frac{1}{q_0})^d]} + z + t_2 \right) \right\} \leq t_2 \quad (4) \end{aligned}$$

For d and z constants (thus, q_0 is also constant), as $n \rightarrow \infty$, the minimum value of t_2 satisfying Eq. (4) is

$$t_2 = \frac{z}{-\log[1 - (1 - \frac{1}{q_0})^d]} \log \frac{(d + 1)}{-\log[1 - (1 - \frac{1}{q_0})^d]} = O(1).$$

Thus,

$$t = \frac{(d + 1) \log n}{-\log[1 - (1 - \frac{1}{q_0})^d]} + \frac{z \log \log n}{-\log[1 - (1 - \frac{1}{q_0})^d]} + O(1)$$

satisfies Eq. (2) (where the constant term z in t is absorbed in $O(1)$). Therefore, the number of rows of the corresponding $(d; z)$ -disjunct matrix is at most

$$tq_0 = \frac{d+1}{B_d} \log n + \frac{z}{B_d} \log \log n + O(1),$$

where

$$B_d = B_d(q_0) = \max_{q>1} \frac{-\log[1 - (1 - \frac{1}{q})^d]}{q}.$$

Also,

$$B_d \rightarrow \frac{1}{d \log e} \text{ as } d \rightarrow \infty,$$

as proved in [Theorem 10](#). \square

6 Transformation from Error-Tolerant Separable Matrices to Error-Tolerant Disjunct Matrices

Let M be a 0/1 matrix. For any set S of columns of M , $U(S)$ will denote the union of the row indices of 1-entries of all columns in S . When S is the singleton set $\{C\}$, by abusing the notation, $U(S)$ is simply written as C . Matrix M is called d -separable if for any two distinct d -sets S and S' of columns, $U(S) \neq U(S')$. M is called \bar{d} -separable if the restrictions $|S| = d$ and $|S'| = d$ above are changed to $|S| \leq d$ and $|S'| \leq d$, respectively. Finally, M is called d -disjunct if for any d -set S of columns and any column C not in S , C is not contained in $U(S)$. These three properties of 0/1 matrices have been widely studied in the literature of nonadaptive group testing designs (pooling designs), which have applications in DNA screening [[17, 25, 32, 38, 39](#)].

It has long been known that d -disjunctness implies \bar{d} -separability which in turn implies d -separability [[17, Chap. 2](#)]. Recently, Chen and Hwang [[7](#)] found a way to construct a disjunct matrix from a separable matrix to complete the cycle of implications.

Theorem 12 (Chen and Hwang [[7](#)]) *Suppose M is a $2d$ -separable matrix. Then one can construct a d -disjunct matrix by adding at most one row to M .*

The notion of d -separability, \bar{d} -separability, and d -disjunctness has their error-tolerant versions. A 0/1 matrix M is called $(d; z)$ -separable if $|U(S) \Delta U(S')| \geq z$ for any two d -sets of columns of M . It is $(\bar{d}; z)$ -separable if the restriction of d -sets is changed to two sets each with at most d elements. Finally, M is $(d; z)$ -disjunct if for any d -set S of columns and any column C not in S , $|C \setminus U(S)| \geq z$. Note that the variable z represents some redundancy to tolerate errors [[17](#)]. For $z = 1$, the error-tolerant version is reduced to the original version.

In [[17](#)], Du and Hwang attempted to extend [Theorem 12](#) to its error-tolerant version, as stated in the following theorem:

Theorem 13 ([[17](#)], Theorem 2.7.6) *Suppose M is a $(2d; z)$ -separable matrix. Then one can obtain a $(d; z)$ -disjunct matrix by adding at most z rows to M .*

By [Theorem 13](#), Du and Hwang obtained the following corollary:

Corollary 1 ([17], Theorem 2.7.7) *A $(d; 2z)$ -separable matrix can be obtained from a $(2d; z)$ -separable matrix by adding at most z rows.*

Unfortunately, [Theorem 13](#) is incorrect; thus, [Corollary 1](#) is also incorrect, as seen by the following counterexample. Let

$$M_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

It is easily verified that M_1 is $(2; 2)$ -separable. It will be showed next adding two rows to M_1 cannot produce a $(1; 2)$ -disjunct matrix.

Let C_1, C_2, C_3 , and C_4 denote the four columns of M_1 . Suppose setting $C = C_i$ and $S = \{C_j\}$, $i \neq j$. Then two rows are needed such that each containing C_i but not C_j . One such row is already provided by M_1 . So one $(1, 0)$ -pair is needed in a new row. Since this is required for each pair of (i, j) with $i \neq j$, there are $4 \times 3 = 12$ choices of (i, j) pair, and each such pair needs a $(1, 0)$ -pair in a new row, or equivalently, the new rows should provide 12 such $(1, 0)$ -pairs. However, one new row can provide at most four $(1, 0)$ -pairs (achieved by a row with two 1-entries and two 0-entries). So two new rows are not sufficient to provide the 12 $(1, 0)$ -pairs required by the $(1; 2)$ -disjunctness property.

In [9], the authors gave a correct version of [Theorem 13](#) and obtained a more rigorous statement of [Theorem 12](#). Their results will be presented in this section.

6.1 Main Results

Lemma 12 ([17], Lemma 2.1.1) *Suppose M is a d -separable matrix with n columns where $d < n$, then it is k -separable for every positive integer $k \leq d$.*

Note that the condition $d < n$ in [Lemma 12](#) is necessary as seen by the following example: Let

$$M_2 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

M_2 is trivially 3-separable. However, it is not 2-separable, as the union of any pair of its columns is identical. Now [Lemma 12](#) is generalized to an error-tolerant version.

Lemma 13 *If a matrix M with n columns is $(d; z)$ -separable for $d < n$, then it is $(k; z)$ -separable for every positive integer $k \leq d$.*

Proof It suffices to prove M is $(d-1; z)$ -separable. Assume that M is not $(d-1; z)$ -separable. Then there exist two distinct sets S and S' each consisting of $d-1$ columns of M such that $|U(S) \Delta U(S')| < z$.

If $|S \setminus S'| = |S' \setminus S| \geq 2$, then there must exist a pair of columns (C_x, C_y) such that $C_x \in S \setminus S'$ and $C_y \in S' \setminus S$. It is easy to see that

$$|U(S \cup \{C_y\}) \Delta U(S' \cup \{C_x\})| \leq |U(S \cup \{C_y\}) \Delta U(S')| \leq |U(S) \Delta U(S')|.$$

This violates the $(d; z)$ -separability of M , as desired.

Now consider the case of $|S \setminus S'| = |S' \setminus S| = 1$. It is obvious that $|S \cup S'| = d$. Thanks to $d < n$, one can take a column C of M which is in neither S nor S' . It is easily seen that

$$|U(S \cup \{C\}) \Delta U(S' \cup \{C\})| \leq |U(S) \Delta U(S')| < z.$$

This contradicts the $(d; z)$ -separability of M , completing the proof. \square

Now it is ready to give a correct version of [Theorem 13](#).

Theorem 14 Suppose M is a $(2d; z)$ -separable matrix with n columns where $n \geq 2d + 1$. Then one can obtain a $(d; \lceil z/2 \rceil)$ -disjunct matrix by adding at most $\lceil z/2 \rceil$ rows to M .

Proof Suppose M is not $(d; \lceil z/2 \rceil)$ -disjunct. Then there exist a column C and a set S of d other columns such that $|C \setminus U(S)| < \lceil z/2 \rceil$. By adding at most $\lceil z/2 \rceil$ rows to M such that each row has a 1-entry at column C and 0-entries at all columns in S , one can obtain $|C \setminus U(S)| \geq \lceil z/2 \rceil$. Of course, there may exist another pair (C', S') where C' is a column and S' is a set of d columns other than C' , such that $|C' \setminus U(S')| < \lceil z/2 \rceil$ in M . Then break it up by using those $\lceil z/2 \rceil$ rows in the same fashion. Here what one needs to show is that this procedure is not self-conflicting, that is, there does not exist two pairs (C, S) and (C', S') such that $|C \setminus U(S)| < \lceil z/2 \rceil$, yet on the other hand $C \in S'$ while $|C' \setminus U(S')| < \lceil z/2 \rceil$.

Suppose to the contrary that there exist two pairs (C, S) and (C', S') in M as described above with $|S| = |S'| = d$. Define

$$S_0 = \{C'\} \cup S \cup S', \quad S_1 = S_0 \setminus \{C\}, \quad \text{and } S_2 = S_0 \setminus \{C'\}.$$

Let $s = |S_0|$, then $s \leq 2d + 1$ and $|S_1| = |S_2| = s - 1 \leq 2d$.

Note that $S_1 \neq S_2$, but they have the same cardinality which is less than $2d + 1$. Next it will be showed that the symmetric difference of $U(S_1)$ and $U(S_2)$ is less than z , thus violating the assumption of $(2d; z)$ -separability.

Since the only column in S_1 but not in S_2 is C' and $|C' \setminus U(S')| < \lceil z/2 \rceil$, it follows that

$$|U(S_2) \setminus U(S_1)| < \lceil z/2 \rceil. \tag{5}$$

Similarly, one can obtain

$$|U(S_1) \setminus U(S_2)| < \lceil z/2 \rceil. \quad (6)$$

Equation (5) along with Eq. (6) gives $|U(S) \Delta U(S')| < z$, implying that M is not $(s-1; z)$ -separable. This contradicts with Lemma 13, and so the theorem is proved. \square

Corollary 2 Suppose M is a $2d$ -separable matrix with n columns where $n \geq 2d+1$. Then one can obtain a d -disjunct matrix by adding at most one row to M .

Proof It follows from Theorem 14 by setting $z = 1$.

Corollary 2 is a more rigorous version of Theorem 12. The following example shows the necessity of the extra condition $n \geq 2d+1$ in Corollary 2. Let

$$M_3 = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then M_3 is trivially 4-separable, but it can be easily verified that no row can be added to M_3 to make it 2-disjunct. Similarly, any matrix with $2d$ columns is trivially $(2d; z)$ -separable, and one does not expect that adding $\lceil z/2 \rceil$ rows to an arbitrary matrix with $2d$ columns would make it $(d; \lceil z/2 \rceil)$ -disjunct. To see a specific counterexample, note that M_1 is trivially a $(4; 4)$ -separable matrix, but adding two rows does not make it a $(2; 2)$ -disjunct matrix – It is even not $(1; 2)$ -disjunct as indicated at the end of Sect. 1.

Corollary 3 Suppose M is a $(2d; z)$ -separable matrix with n columns where $n \geq 2d+1$. Then, for any positive integer $k \leq \lceil z/2 \rceil$, one can obtain a $(d; k)$ -disjunct matrix by adding at most k rows to M .

Proof The proof of Theorem 14 shows that there does not exist two pairs (C, S) and (C', S') such that $|C \setminus U(S)| < \lceil z/2 \rceil$, yet on the other hand, $C \in S'$ while $|C' \setminus U(S')| < \lceil z/2 \rceil$. In fact, the term $\lceil z/2 \rceil$ can be replaced by any positive integer k which satisfies the symmetric difference of $U(S_1)$ and $U(S_2)$ is less than z . Therefore, for any $k \leq \lceil z/2 \rceil$, one can obtain a $(d; k)$ -disjunct matrix by adding at most k rows to M in the same fashion. \square

The following equivalence relation is given in [17] without giving a proof. Now a proof is given, and a stronger result is obtained by using the equivalence relation.

Lemma 14 ([17], Lemma 2.7.5) A matrix M is $(\bar{d}; z)$ -separable if and only if it is $(d; z)$ -separable and $(d-1; z)$ -disjunct.

Proof Suppose M is $(\bar{d}; z)$ -separable but not $(d - 1; z)$ -disjunct; in other words, there exists a set S of $d - 1$ columns other than a column C such that $|C \setminus U(S)| \leq z$. Then it is easy to see that

$$|U(S \cup \{C\}) \Delta U(S)| = |U(S \cup \{C\}) \setminus U(S)| \leq z,$$

a contradiction to $(\bar{d}; z)$ -separability. Thus, M is $(d - 1; z)$ -disjunct and $(d; z)$ -separable trivially.

Let M be $(d; z)$ -separable and $(d - 1; z)$ -disjunct. It suffices to show that

$$|U(X) \Delta U(Y)| \geq z$$

for any two sets X, Y of at most d columns. If $|X| = |Y| \leq d$, then $|U(X) \Delta U(Y)| \geq z$ by $(d; z)$ -separability and Lemma 13. Assume $|X| < |Y| \leq d$, then there exists a column $C_y \in Y$ but not in X . By $(d - 1; z)$ -disjunctness, it must have $|C_y \setminus U(X)| \geq z$; hence, $|U(X) \Delta U(Y)| \geq z$. It concludes the proof. \square

By Lemmas 14 and 13, Corollary 3 is extended to a stronger version.

Corollary 4 Suppose M is a $(2d; z)$ -separable matrix with n columns where $n \geq 2d + 1$. Then, for any positive integer $k \leq \lceil z/2 \rceil$, one can obtain a $(\bar{d} + 1; k)$ -separable matrix by adding at most k rows to M .

6.2 Concluding Remarks

The following remarks demonstrate the optimality of the results presented in this section.

1. The constraint $k \leq \lceil z/2 \rceil$ in Corollary 3 is necessary to make the number of rows added to be independent of n and d . To see a specific example, consider that M is an $(n \lceil z/2 \rceil) \times n$ matrix such that each column has $\lceil z/2 \rceil$ 1-entries and any 2 columns have no intersection. Then, M is $(2d; z)$ -separable. Since every column has only $\lceil z/2 \rceil$ 1-entries, to make M $(d; k)$ -disjunct by adding rows, the rows added must form a $(d; k - \lceil z/2 \rceil)$ -disjunct submatrix when $k > \lceil z/2 \rceil$. In this case, the minimum number of rows required would depend on n, d , and $k - \lceil z/2 \rceil$.
2. Let N be a 0/1 matrix of constant row sum 1 and constant column sum z , and let M be obtained from N by adding one zero column. It is easy to verify that M is $(2d; z)$ -separable. Since there is a zero column in M , one cannot obtain from M a $(d; k)$ -disjunct matrix by adding less than k rows. This shows that the bound on the number of additional rows given in Corollary 3 is optimal in this sense.

7 Conclusion

Some recent algorithmic, complexity, and mathematical results on nonadaptive group testing (and on pooling design) are presented in this monograph. New construction of disjunct matrices with even reduced number of rows remains interesting to investigate. The complexity of the problem MIN- \bar{d} -SS introduced in Sect. 3.3 remains unsolved. On the bounds of the minimum number $t(d, n)$ of rows of d -disjunct matrices with n columns, closing the gap between $O(d^2 \log n)$ and $\Omega(\frac{d^2 \log n}{\log d})$ remains as a major open problem in extremal combinatorics.

Cross-References

► [Combinatorial Optimization Algorithms](#)

Recommended Reading

1. N. Alon, J.H. Spencer, *The Probabilistic Method* (Wiley, New York, 1992). (Second Edition 2000)
2. N. Alon, D. Moshkovitz, S. Safra, Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms* **2**(2), 153–177 (2006)
3. D.J. Balding, W.J. Bruno, E. Knill, D.C. Torney, A comparative survey of non-adaptive pooling designs, in *Genetic Mapping and DNA Sequencing* (Springer, New York, 1996), pp. 133–154
4. T. Berger, J.W. Mandell, P. Subrahmanyam, Maximally efficient two-stage group testing. *Biometrics* **56**, 833–840 (2000)
5. J. Borneman, M. Chrobak, G. Della Vedova, A. Figueroa, T. Jiang, Probe selection algorithms with applications in the analysis of microbial communities. *Bioinformatics* **17**(Suppl.), S39–S48 (2001)
6. W.J. Bruno, D.J. Balding, E. Knill, D.C. Bruce et al., Efficient pooling designs for library screening. *Genomics*, **26**, 21–30 (1995)
7. H.B. Chen, F.K. Hwang, Exploring the missing link among d -separable, \bar{d} -separable and d -disjunct matrices. *Discret. Appl. Math.* **133**, 662–664 (2007)
8. J. Chen, I.A. Kanj, W. Jia, Vertex cover: further observations and further improvements. *J. Algorithm* **41**(2), 280–301 (2001)
9. H.B. Chen, Y. Cheng, Q. He, C. Zhong, Transforming an error-tolerant separable matrix to an error-tolerant disjunct matrix. *Discret. Appl. Math.* **157**(2), 387–390 (2009)
10. Y. Cheng, D.Z. Du, New constructions of one- and two-stage pooling designs. *J. Comput. Biol.* **15**, 195–205 (2008)
11. Y. Cheng, K.-I Ko, W. Wu, On the complexity of non-unique probe selection. *Theor. Comput. Sci.* **390**(1), 120–125 (2008)
12. Y. Cheng, D.Z. Du, K.-I. Ko, G. Lin, On the parameterized complexity of pooling design. *J. Comput. Biol.* **16**, 1529–1537 (2009)
13. Y. Cheng, D.Z. Du, G. Lin, On the upper bounds of the minimum number of rows of disjunct matrices. *Optim. Lett.* **3**(2), 297–302 (2009)
14. A. De Bonis, L. Gasieniec, U. Vaccaro, Optimal two-stage algorithms for group testing problems. *SIAM J. Comput.* **34**, 1253–1270 (2005)
15. R. Dorfman, The detection of defective members of large populations. *Ann. Math. Stat.* **14**, 436–440 (1943)

16. R.G. Downey, M.R. Fellows, *Parameterized Complexity* (Springer, New York, 1999)
17. D.-Z. Du, F.K. Hwang, *Pooling Designs and Nonadaptive Group Testing: Important Tools for DNA Sequencing* (World Scientific, New Jersey, 2006)
18. D.-Z. Du, K.-I. Ko, Some completeness results on decision trees and group testing. *SIAM J. Algebra. Discret.* **8**(4), 762–777 (1987)
19. D.-Z. Du, K.-I. Ko, *Theory of Computational Complexity* (Wiley, New York, 2000)
20. D.Z. Du, F.K. Hwang, W. Wu, T. Znati, New construction for transversal design. *J. Comput. Biol.* **13**, 990–995 (2006)
21. A.G. D'yachkov, V.V. Rykov, Bounds of the length of disjunct codes. *Probl. Control Inf. Theory* **11**, 7–13 (1982)
22. A.G. D'yachkov, V.V. Rykov, A.M. Rashad, Superimposed distance codes. *Probl. Control Inf. Theory* **18**, 237–250 (1989)
23. A.G. D'yachkov, A.J. Macula, V.V. Rykov, New constructions of superimposed codes. *IEEE Trans. Inf. Theory* **46**, 284–290 (2000)
24. D. Eppstein, M.T. Goodrich, D.S. Hirschberg, Improved combinatorial group testing algorithms for real-world problem sizes. *SIAM J. Comput.* **36**, 1360–1375 (2007)
25. P. Erdős, P. Frankl, Z. Füredi, Families of finite sets in which no set is covered by the union of others. *Isr. J. Math.* **51**, 79–89 (1985)
26. M. Farach, S. Kannan, E. Knill, S. Muthukrishnan, Group testing problems with sequences in experimental molecular biology, in *Proceedings of the Compression and Complexity of Sequences*, ed. by B. Carpentieri et al. (IEEE Press, Los Alamitos, 1997), pp. 357–367
27. J. Flum, M. Grohe, *Parameterized Complexity Theory*. Texts in Theoretical Computer Science, an EATCS Series, vol. XIV (Springer, Berlin, 2006)
28. H.L. Fu, F.K. Hwang, A novel use of t-packings to construct d-disjunct matrices. *Discret. Appl. Math.* **154**, 1759–1762 (2006)
29. Z. Füredi, On r -cover-free families. *J. Comb. Theory Ser. A* **73**, 172–173 (1996)
30. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
31. R. Herwig, A.O. Schmitt, M. Steinfath, J. O' Brien et al., Information theoretical probe selection for hybridisation experiments. *Bioinformatics* **16**, 890–898 (2000)
32. F.K. Hwang, V.T. Sós, Non-adaptive hypergeometric group testing. *Studia Sci. Math. Hung.* **22**, 257–263 (1987)
33. P. Indyk, H.Q. Ngo, A. Rudra, Efficiently decodable non-adaptive group testing, in *Proceedings of 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, Austin, 2010, pp. 1126–1142
34. W.H. Kautz, R.C. Singleton, Nonrandom binary superimposed codes. *IEEE Trans. Inf. Theory* **10**, 363–377 (1964)
35. G.W. Klau, S. Rahmann, A. Schliep, M. Vingron, K. Reinert, Optimal robust non-unique probe selection using integer linear programming. *Bioinformatics* **20**, i186–i193 (2004)
36. E. Knill, Lower bounds for identifying subset members with subset queries, in *Proceedings of 6th ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, USA, 1995, pp. 369–377
37. O. Lichtenstein, A. Pnueli, Checking that finite state concurrent programs satisfy their linear specification, in *Proceedings of 12th ACM Symposium on Principles of Programming Languages (POPL' 85)*, 107, New York, 1985, pp. 97–107
38. A.J. Macula, A simple construction of d-disjunct matrices with certain constant weights. *Discret. Math.* **162**, 311–312 (1996)
39. A.J. Macula, Error-correcting nonadaptive group testing with d^e -disjunct matrices. *Discret. Appl. Math.* **80**, 217–222 (1997)
40. A.J. Macula, Probabilistic nonadaptive group testing in the presence of errors and DNA library screening. *Ann. Comb.* **3**, 61–69 (1999)
41. A.J. Macula, Probabilistic nonadaptive and two-stage group testing with relatively small pools and DNA library screening. *J. Comb. Optim.* **2**, 385–397 (1999)
42. R. Motwani, P. Raghavan, *Randomized Algorithms* (Cambridge University Press, New York, 1995)

43. H.Q. Ngo, D.Z. Du, A survey on combinatorial group testing algorithms with applications to DNA library screening, in *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*, vol. 55 (American Mathematical Society, Providence, 2000), pp. 171–182
44. H.Q. Ngo, D.Z. Du, New constructions of non-adaptive and error-tolerance pooling designs. *Discret. Math.* **243**, 161–170 (2002)
45. H.Q. Ngo, E. Porat, A. Rudra, Efficiently decodable error-correcting list disjunct matrices and applications, in *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*, Zurich, Switzerland, 2011, pp. 557–568
46. M. Olson, L. Hood, C. Cantor, D. Botstein, A common language for physical mapping of the human genome. *Science* **245**, 1434–1435 (1989)
47. C.H. Papadimitriou, *Computational Complexity* (Addison-Wesley, New York, 1994)
48. C.H. Papadimitriou, D. Wolfe, The complexity of facets resolved, *J. Comput. Syst. Sci.* **37**, 2–13 (1988)
49. H. Park, W. Wu, Z. Liu, X. Wu, H.G. Zhao, DNA screening, pooling design and simplicial complex. *J. Comb. Optim.* **7**, 389–394 (2003)
50. E. Porat, A. Rothschild, Explicit non-adaptive combinatorial group testing schemes, in *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, Reykjavik, Iceland, 2008, pp. 748–759
51. S. Rahmann, Rapid large-scale oligonucleotide selection for microarrays, in *Proceedings of the 1st IEEE Computer Society Conference on Bioinformatics (CSB' 02)*, Stanford, CA, USA, 2002, pp. 54–63
52. S. Rahmann, Fast and sensitive probe selection for DNA chips using jumps in matching statistics, in *Proceedings of the 2nd IEEE Computer Society Bioinformatics Conference (CSB' 03)*, Stanford, CA, USA, 2003, pp. 57–64
53. M. Ruszinkó, On the upper bound of the size of the r -cover-free families. *J. Comb. Theory Ser. A* **66**, 302–310 (1994)
54. A. Schliep, D.C. Torney, S. Rahmann, Group testing with DNA chips: generating designs and decoding experiments, in *Proceedings of the 2nd IEEE Computer Society Bioinformatics Conference (CSB' 03)*, Stanford, CA, USA, 2003, pp. 84–93
55. C. Umans, The minimum equivalent DNF problem and shortest implicants, in *Proceedings of 39th IEEE Symposium on Foundation of Computer Science*, Palo Alto, CA, USA, 1998, pp. 556–563
56. X. Wang, B. Seed, Selection of oligonucleotide probes for protein coding sequences. *Bioinformatics* **19**, 796–802 (2003)
57. J.K. Wolf, Born again group testing: multiaccess communications. *IEEE Trans. Inf. Theory* **IT-31**, 185–191 (1985)

Advances in Scheduling Problems

Ming Liu and Feifeng Zheng

Contents

1	Introduction	146
1.1	Emergence of <i>psd</i> Delivery Time	146
1.2	Literature Review	147
1.3	Notation	148
2	Single-Machine Model	150
2.1	Preliminary	150
2.2	Makespan, Maximum Lateness, Maximum Tardiness, and Number of Tardy Jobs	151
2.3	Total Weighted Completion Time	152
2.4	Total Weighted Discounted Completion Time	155
2.5	Total Weighted Tardiness	155
2.6	Total Completion Time with Release Dates	158
2.7	Approximation for Total Completion Time with Release Dates	159
3	Parallel Identical Machine Model	161
3.1	Makespan	161
3.2	Total Completion Time	168
4	Conclusion	168
	Cross-References	168
	Recommended Reading	169

M. Liu (✉)

School of Economics & Management, Tongji University, Shanghai, People's Republic of China
e-mail: mingliu@tongji.edu.cn

F. Zheng

Glorious Sun School of Business and Management, Donghua University, Shanghai,
People's Republic of China
e-mail: zhengff@mail.xjtu.edu.cn

Abstract

This chapter reviews a research focus of scheduling in manufacturing facilities over the last decade. Manufacturing environment has a great influence on the processing times of jobs. The chapter reviews basic concept about past-sequence-dependent delivery times and covers recently results on scheduling with such constraint.

1 Introduction

Scheduling is a form of decision making that plays a crucial role in manufacturing and service industries. Scheduling is a key component of combinatorial optimization. In the current competitive environment, effective scheduling has become a necessity for survival in the marketplace. Companies have to meet shipping dates that have been committed to customers, as failure to do so may result in a significant loss of goodwill. They also have to schedule activities or jobs in such a way as to use the resources available in an efficient manner.

Scheduling deals with the allocation of scarce resources to tasks over time. It is a decision-making process with the goal of optimizing one or more objectives. The resources and tasks in an organization can take many forms. The resources may be machines in a workshop, runways at an airport, crews at a construction site, processing units in a computing environment, and so on. The tasks may be operations in a production process, takeoffs and landings at an airport, stages in a construction project, executions of computer programs, and so on. Each task may have a certain priority level, an earliest possible starting time, and a due date. The objective may be the minimization of the completion time of the last task, and another may be the minimization of the number of tasks completed after their respective due dates [12].

This chapter focuses on a cutting-edge topic in machine scheduling problem, that is, scheduling with past-sequence-dependent (abbr. *psd*) job delivery times. The topic is important because of the theoretical insights it provides and also of its practical implications.

1.1 Emergence of *psd* Delivery Time

In many industries, manufacturing environment has a great influence on the processing times of jobs. A growing body of evidence shows that the manufacturing environment or waiting time may have an adverse effect on the total processing time of a job before its delivery to the customer. In some situations, the waiting time-induced adverse effect does not impede the job's suitability to be processed. Moreover, it can be eliminated after the main processing of the job with an extra

time caused. Such an extra time for eliminating adverse effect between the main processing and the delivery of job is viewed as a *psd* delivery time. Note that the treatment of the adverse effect for a job does not occupy any machine and has no relation with the schedule of the job's main processing. For analysis convenience, it is generally assumed that the *psd* delivery time of a job is proportional to the job's waiting time. One special case is that the *psd* delivery time is a simple linear function of the wait time.

One application with *psd* delivery time is in electronic manufacturing industry. An electronic component may be exposed to certain electromagnetic and/or radioactive fields while waiting in the machine's preprocessing area, and regulatory authorities require the component to be "treated" (e.g., in a chemical solution capable of removing/neutralizing certain effects of electromagnetic/radioactive fields) for an amount of time proportional to the job's exposure time to these fields. The treatment is performed immediately after the component has been processed on the machine to ensure a guaranteed delivery to the customer [6]. Such a postprocessing operation is usually called the job "tail" or the job "delivery time." Unlike the traditional assumption of a job-specific constant delivery time in the scheduling literature [4], it is assumed that the *psd* delivery time of a job is proportional to the job's waiting time.

1.2 Literature Review

It has been widely studied in recent decades for one scheduling scenario such that the waiting time of a job has an adverse effect on the job's main processing. The scenario with waiting time-induced adverse effect is mainly partitioned into three categories.

In the first category, each job has a so-called *deteriorating processing time* such that the processing time of a job increases in its waiting time. The concept of *deterioration* is introduced by Browne and Yechiali [2] and it describes a kind of adverse effect of waiting time.

The second category originated from [5], in which the adverse effect must be removed prior to the main processing of the job by performing a setup operation. The authors introduced the concept of *past-sequence-dependent setup time* to describe the adverse effect.

The third and newest category is from [6], in which the waiting time-induced adverse effect does not impede the suitability to be processed by one machine, and the adverse effect shall be removed prior to delivering the job to the customer. They established a model to incorporate the adverse effect of waiting into a postprocessing operation by introducing the concept of *past-sequence-dependent delivery time*. The treatment of the adverse effect of each job, that is, the job's *psd* delivery time, must follow immediately the main processing of the job. This chapter focuses on this category.

1.3 Notation

In all the following scheduling problems considered, the number of jobs and that of machines are assumed to be finite. Generally, the number of jobs is denoted by n and the number of machines by m (see [12]). The following notations are associated with job J_j :

Processing time (p_{ij}). p_{ij} represents the main processing time of job J_j on machine M_i . It is simply denoted as p_j if the main processing time of J_j is uniform on all machines or if there is only one machine allowed to process job J_j . In the rest of this chapter, sometimes the processing of J_j means its main processing on a machine.

Due date (d_j). d_j represents the committed date at which job J_j is shipped on board or arrived at the customer. There may incur a penalty if the completion time of a job is strictly later than its due date.

Weight (w_j). w_j represents some priority or cost for satisfying job J_j . Its specific meaning depends on manufacturing settings.

Delivery time (q_j). Literally, q_j means the time a completed job needs to be delivered to a customer. In this chapter, delivery time usually represents a postprocessing of a job after its completion on the machine. (Note that this postprocessing operation is not done on the machine).

One scheduling problem is usually described by a triplet $\alpha|\beta|\gamma$ (refer to [1]). The α field describes the machine environment such as single machine or parallel machines, the β field provides details on job-processing characteristics and constraints, and the γ field describes the minimization objectives such as the makespan and the total completion times. Below, possible settings in each of the three fields are given in this chapter.

First, there are two possible machine environments specified in the α field:

Single machine (1). There is a single machine for job processing in the manufacturing system.

Identical machines in parallel (P_m). There are m identical parallel machines in the system. Each job requires only one operation and may be processed on any one of the m machines.

Secondly, there may be multiple entries on job-processing restrictions and constraints specified in the β field, including:

Past-sequence-dependent delivery time (q_{psd}). q_{psd} represents the time required for eliminating waiting time-induced adverse effect after the main processing of job. (Note that this part of processing is not done on the machine).

Release date (r_j). Job J_j becomes available for processing at its release date r_j . If the symbol does not appear in the β field, it implies $r_j = 0$ for any job J_j and all the jobs can be started at the beginning.

Preemption ($prmp$). $prmp$ represents that it is unnecessary to keep a job on a machine, once started, until completion. The scheduler is allowed to interrupt the processing of a job at any time point and arrange another job on the machine instead. In this chapter, the focus is on the resumption case such that if a preempted job is afterward rescheduled on the machine (or another machine in the case of parallel machines), it is started from where it was preempted but not from the start.

Precedence constraint (*prec*). *prec* means that there exist processing precedence constraints among jobs such that one or more jobs are required to be completed before the processing of some other job. In this chapter, three specific forms of precedence constraint are considered: a *chain* in which each job has at most one predecessor and at most one successor, an *intree* in which each job has at most one successor, and an *outtree* in which each job has at most one predecessor.

Before introducing the minimization objectives in the γ field, some fundamental notations are given. In a processing schedule, let C_j be the completion time of job J_j , that is, the completion time of the processing of J_j on a machine plus the job's *psd* delivery time. The *lateness* of job J_j is defined as

$$L_j = C_j - d_j,$$

which is positive or negative when J_j is completed later or earlier than its due date, respectively. The *tardiness* of job J_j is defined as

$$T_j = \max\{C_j - d_j, 0\} = \max\{L_j, 0\}.$$

The difference between the tardiness and the lateness lies in that the former is never negative, that is, the jobs completed earlier than its due date are counted out when counting T_j . The *unit penalty* of job J_j is defined as

$$U_j = \begin{cases} 1, & \text{if } L_j > 0; \\ 0, & \text{otherwise.} \end{cases}$$

The lateness, tardiness, and unit penalty are the three basic due date-related parameters considered in the literature. The possible objective functions specified in γ field are:

Makespan (C_{\max}). The makespan, defined as $\max\{C_1, \dots, C_n\}$, is equivalent to the completion time of the last job to leave the system. A minimum makespan usually implies a high utilization of the machine(s).

Maximum lateness (L_{\max}). The maximum lateness L_{\max} is defined as $\max\{L_1, \dots, L_n\}$. It measures the worst violation of due date among completed jobs.

Maximum tardiness (T_{\max}). The maximum tardiness T_{\max} is defined as $\max\{T_1, \dots, T_n\}$. It also measures the worst violation of due date among completed jobs but omits the jobs with negative lateness.

Total weighted completion time ($\sum w_j C_j$). The sum of the weighted completion times of the n jobs gives an indication of the total holding or inventory costs incurred by the schedule. For the case of $r_j = 0$, the total completion times $\sum C_j$ is equivalent to *total flow time*, and $\sum w_j C_j$ is equivalent to *total weighted flow time*.

Discounted total weighted completion time ($\sum w_j (1 - e^{-r C_j})$). This is a more general cost function than total weighted completion time. The costs are discounted at a rate $r \in (0, 1)$ per unit time. That is, if job J_j is not completed by time t , an additional cost $w_j r e^{-rt} dt$ is incurred over period $[t, t + dt]$. (Note that

$\int_0^{C_j} w_j r e^{-rt} dt = w_j (1 - e^{-rC_j})$ where C_j is a variable). The value of r is usually less than 0.1.

Total weighted tardiness ($\sum w_j T_j$). This objective is also a more general cost function than total weighted completion time.

Number of tardy jobs ($\sum U_j$). This is not only an objective of academic interest but also a measure in common use in practice due to its easy operation.

All the above objective functions belong to so-called *regular* performance measures which are nondecreasing functions of jobs' completion times, that is, C_1, \dots, C_n .

2 Single-Machine Model

This section considers the following single-machine scheduling models related to minimization objectives. For completion time-related objectives, it includes makespan, that is, the maximum completion time, total completion time, and total weighted completion time. For due date-related objectives, it includes maximum lateness and number of tardy jobs. All the objective functions considered in this section are regular, which means the objective function is a nondecreasing function of job's completion time.

For most models considered in this section, jobs are all released at the beginning, and thus there is no advantage in adopting preemptions. For these models, it can be shown that an optimal schedule is a nonpreemptive one, and there are no idle time segments between the processing of any two jobs. However, if jobs are with different release dates in nonpreemptive environment, it may be advantageous to contain unforced idleness.

This section is mainly based on [6, 10, 11].

2.1 Preliminary

Consider a single machine to process a set of n jobs which are available at time zero. Let p_j, d_j denote, respectively, the processing time and due date of job J_j for $j = 1, \dots, n$.

Given a nonpreemptive schedule σ , let $J_{[j]}$, $p_{[j]}$, and $d_{[j]}$ denote the job occupying the j -th position in σ , its processing time, and its due date, respectively. Denote by $S_{[j]}(\sigma)$ or simply $S_{[j]}$ the starting time of job $J_{[j]}$ in σ . In the environment with *psd* delivery time, the processing of $J_{[j]}$ must be followed immediately by its *psd* delivery time $q_{[j]}$. This chapter considers the case of simple linear function for $q_{[j]}$ such that $q_{[j]}$ is proportional to the waiting time or starting time of job $J_{[j]}$. (Readers may extend the discussion to the case of linear or some other function in application necessities). Consequently, $q_{[j]}$ is formulated as

$$q_{[j]} = \gamma S_{[j]}, \quad j = 1, \dots, n, \quad (1)$$

where $\gamma \geq 0$ is a normalizing constant. Observe that in a single-machine environment where the machine is always available for processing and all the n jobs are available at time zero, the starting time $S_{[j]}$ in a schedule without (redundant) idle time satisfies

$$\begin{aligned} S_{[1]} &= 0, \\ S_{[j]} &= \sum_{i=1}^{j-1} p_{[i]}, \quad j = 2, \dots, n. \end{aligned} \tag{2}$$

Let $C_{[j]}$ denote the completion time of job $J_{[j]}$ in a schedule, that is, the completion time of the processing of $J_{[j]}$ on the machine plus the job's *psd* delivery time. Therefore,

$$\begin{aligned} C_{[1]} &= p_{[1]} + q_{[1]} = p_{[1]}, \\ C_{[j]} &= S_{[j]} + p_{[j]} + q_{[j]} \\ &= \sum_{i=1}^{j-1} p_{[i]} + p_{[j]} + q_{[j]} \\ &= (1 + \gamma) \sum_{i=1}^{j-1} p_{[i]} + p_{[j]}, \quad j = 2, \dots, n. \end{aligned} \tag{3}$$

Accordingly,

$$C_{[j]} = (1 + \gamma) \sum_{i=1}^{j-1} p_{[i]} + p_{[j]}, \quad j = 1, \dots, n, \tag{4}$$

where $\sum_{i=1}^0 p_{[i]} = 0$.

2.2 Makespan, Maximum Lateness, Maximum Tardiness, and Number of Tardy Jobs

For convenience, let $P = \sum_{j=1}^n p_j$.

Theorem 1 Problem $1|q_{\text{psd}}|C_{\max}$ can be solved in $O(n)$ time.

Proof It is clear from (Eq. 4) that $C_{[j]} > C_{[j-1]}$, and therefore

$$C_{\max} = C_{[n]} = P + \gamma \sum_{i=1}^{n-1} p_{[i]} = P + \gamma(P - p_{[n]}) = (1 + \gamma)P - \gamma p_{[n]}. \tag{5}$$

Since P is a constant, C_{\max} is minimized when $p_{[n]}$ is maximal. Consequently, any sequence with $p_{[n]} = \max_{j=1,\dots,n}\{p_j\}$ is optimal for the problem. Since the longest job can be identified in $O(n)$ time, the problem can be solved in $O(n)$ time as well. \square

Below it is shown that $1|q_{psd}|L_{\max}$, $1|q_{psd}|T_{\max}$, and $1|q_{psd}|\sum U_j$ problems can be reduced to the corresponding problems without *psd* delivery times by appropriate transformations for p_j and d_j . Specifically, in accordance with expression (Eq. 5), $L_{[j]} = C_{[j]} - d_{[j]} = \sum_{i=1}^j (1 + \gamma) p_{[i]} - (d_{[j]} + \gamma p_{[j]})$ and $T_{[j]} = \max\{L_{[j]}, 0\}$; therefore, the substitutions of $p'_j = (1 + \gamma)p_j$ and $d'_j = d_j + \gamma p_j$, $j = 1, \dots, n$, reduce the above L_j , T_j expressions to the corresponding expressions without *psd* delivery times. Consequently, the $1|q_{psd}|L_{\max}$, $1|q_{psd}|T_{\max}$, and $1|q_{psd}|\sum U_j$ problems reduce to $1||L_{\max}$, $1||T_{\max}$, and $1||\sum_{j=1}^n U_j$ problems, respectively. As a result, $1|q_{psd}|L_{\max}$ and $1|q_{psd}|T_{\max}$ can be solved in $O(n \log n)$ time by implementing the earliest due date (EDD) sequence on due dates d'_j , and $1||\sum U_j$ can be solved in $O(n \log n)$ time by implementing Algorithm 3.3.1 in [12] with processing times p'_j and due dates d'_j ($j = 1, \dots, n$).

2.3 Total Weighted Completion Time

Consider the single-machine scheduling problem to minimize the total weighted completion time with *psd* delivery time. Denote the model as $1|q_{psd}|\sum w_j C_j$. It is shown that shortest weighted processing time (SWPT) rule is optimal for the model.

Theorem 2 For $1|q_{psd}|\sum w_j C_j$, SWPT rule is optimal.

Proof By formula (Eq. 4),

$$\begin{aligned} \sum_{j=1}^n w_{[j]} C_{[j]} &= \sum_{j=1}^n w_{[j]} \left((1 + \gamma) \sum_{i=1}^j p_{[i]} - \gamma p_{[j]} \right) \\ &= (1 + \gamma) \sum_{j=1}^n w_{[j]} \sum_{i=1}^j p_{[i]} - \gamma \sum_{j=1}^n w_{[j]} p_{[j]}. \end{aligned} \quad (6)$$

On the right-hand side of the above second equation, the first item $(1 + \gamma) \sum_{j=1}^n w_{[j]} \sum_{i=1}^j p_{[i]}$ is minimized by SWPT rule, and the second item $\gamma \sum_{j=1}^n w_{[j]} p_{[j]}$ is a constant which has no relation with the processing sequence. This completes the proof. \square

Corollary 1 For $1|q_{psd}|\sum C_j$, SPT rule is optimal.

For an alternative proof of this corollary, readers can refer to [6].

2.3.1 Total Weighted Completion Time with Precedence Constraints

The problem of single-machine scheduling to minimize total weighted completion time with arbitrary precedence constraints is NP-hard, due to the NP-hardness of $1|prec|C_{\max}$ (without *psd* delivery time). So, the focus has now been shifted to several special cases and exploit polynomial time algorithms for the cases.

Consider the simplest form of precedence constraints (i.e., precedence constraints that take the form of parallel chains). This problem can be solved by a relatively simple and efficient (polynomial time) algorithm. The algorithm is based on some fundamental properties of scheduling with precedence constraints.

Consider the case with two job chains. The first chain consists of jobs J_1, \dots, J_k and the second chain consists of jobs J_{k+1}, \dots, J_n . The precedence constraints in the two chains are as follows:

$$J_1 \rightarrow J_2 \rightarrow \dots \rightarrow J_k$$

and

$$J_{k+1} \rightarrow J_{k+2} \rightarrow \dots \rightarrow J_n.$$

In the above chains, J_i precedes J_{i+1} for $i \leq i < k$ and $k+1 \leq i < n$. The following lemma is based on the nonpreemption assumption such that if the first job of one chain is started for processing on the machine, then the whole chain shall be processed till its end without preemption. The lemma answers the question, with the purpose to minimize the total weighted completion time, which one of the two chains should he process first?

Lemma 1 Lemma 3.1.3 [12] If $\frac{\sum_{j=1}^k w_j}{\sum_{j=1}^k p_j} > (<) \frac{\sum_{j=k+1}^n w_j}{\sum_{j=k+1}^n p_j}$, then it is optimal to process the chain of jobs J_1, \dots, J_k before (after) the chain of jobs J_{k+1}, \dots, J_n .

Proof By contradiction. Under sequence $J_1, \dots, J_k, J_{k+1}, \dots, J_n$, the total weighted completion time is

$$\begin{aligned} & w_1 p_1 + \dots + w_k \left((1 + \gamma) \sum_{j=1}^{k-1} p_j + p_k \right) \\ & + w_{k+1} \left((1 + \gamma) \sum_{j=1}^k p_j + p_{k+1} \right) \\ & + \dots + w_n \left((1 + \gamma) \sum_{j=1}^{n-1} p_j + p_n \right), \end{aligned}$$

whereas under sequence $J_{k+1}, \dots, J_n, J_1, \dots, J_k$, it is

$$\begin{aligned} & w_{k+1} p_{k+1} + \dots + w_n \left((1 + \gamma) \sum_{j=k+1}^{n-1} p_j + p_n \right) \\ & + w_1 \left((1 + \gamma) \sum_{j=k+1}^n p_j + p_1 \right) \\ & + \dots + w_k \left((1 + \gamma) (\sum_{j=k+1}^n p_j + \sum_{j=1}^{k-1} p_j) + p_k \right). \end{aligned}$$

The total weighted completion time of the first sequence is strictly less than that of the second sequence if

$$\frac{\sum_{j=1}^k w_j}{\sum_{j=1}^k p_j} > \frac{\sum_{j=k+1}^n w_j}{\sum_{j=k+1}^n p_j}.$$

The lemma follows. \square

Consider the preemptive case other than that in [Lemma 1](#) such that the processing of a chain may be preempted and then it is not necessary to go through each chain at a time. In other words, in the preemptive case, a scheduler may process some jobs of one chain (while adhering to the precedence constraints), switch over to another chain, and later on return to the first chain. A basic concept related to a chain of jobs is present. Given a chain $J_1 \rightarrow J_2 \rightarrow \dots \rightarrow J_k$, let l^* be the smallest index that satisfies

$$\frac{\sum_{j=1}^{l^*} w_j}{\sum_{j=1}^{l^*} p_j} = \max_{l \in \{1, \dots, k\}} \left\{ \frac{\sum_{j=1}^l w_j}{\sum_{j=1}^l p_j} \right\}.$$

The ratio on the left-hand side of the above equation is called the ρ -factor of the chain and it is denoted by $\rho(J_1, \dots, J_k)$. Job J_{l^*} is referred to as the job that determines the ρ -factor of the chain.

For the objective to minimize the total weighted completion time, the following lemma holds in the case of multiple chains.

Lemma 2 *Lemma 3.1.3 [12] If job J_{l^*} determines $\rho(J_1, \dots, J_k)$, then there exists an optimal sequence that processes jobs J_1, \dots, J_{l^*} one after another without any interruption by jobs from other chains.*

Proof See the proof of Lemma 3.1.3 [12]. □

The result of this lemma is intuitive. If one scheduler had decided to start processing a string of jobs, it makes sense to continue processing the string on the machine until job J_{l^*} is completed without processing any other job in between. Given the form of chains for precedence constraints, the above two lemmas are the basis for constructing a simple algorithm that minimizes the total weighted completion time.

Algorithm 3.1.4 [12]:

Whenever the machine is freed, select among the remaining chains the one with the highest ρ -factor. Process this chain without interruption up to and including the job that determines its ρ -factor.

Consider other chains with more general precedence constraints than the parallel chains mentioned above. Some notations on directed tree are introduced, in which each node represents a job and each directed arc represents a precedence constraint between the corresponding two nodes. If there is only one job without successors (or predecessors) in a tree, then it is called an *intree* (or *outtree*). In an intree, the single job with no successors is called the *root* and is located at *level 1*; all jobs immediately preceding the root are at level 2; all those immediately preceding the jobs at level 2 are located at level 3, and so on. All the jobs without predecessors in the intree are referred to in graph theory terminology as *leaves*. In an outtree, the definitions are on the contrary way. The single job without predecessors is called the *root* and is located at level 1; all the jobs immediately following the root are located at level 2, and so on. The jobs without successors in an outtree are called *leaves*.

In both intree and outtree, the jobs with no predecessors are generally referred to as *starting* jobs. There is a single starting job in an outtree while there may be several starting jobs in an intree.

For $1|intree|\sum w_j C_j$ and $1|outtree|\sum w_j C_j$, there are decomposition algorithms (see [13]), which can be directly applied to the corresponding scenarios with *psd* delivery times, that is, $1|q_{psd}, intree|\sum w_j C_j$ and $1|q_{psd}, outtree|\sum w_j C_j$.

2.4 Total Weighted Discounted Completion Time

Consider the objective of total weighted discounted completion time $\sum w_j(1 - e^{-rC_j})$, where r ($0 < r < 1$) is the discount factor. The problem $1|q_{psd}|\sum w_j(1 - e^{-rC_j})$ gives rise to a different priority rule, that is, scheduling jobs in nonincreasing order of $\frac{w_j e^{-rp_j}}{1 - e^{-r(1+\gamma)p_j}}$. This rule is referred to as the Modified weighted discounted shortest processing time (MWDSPT) rule.

Theorem 3 *MWDSPT rule is optimal for $1|q_{psd}|\sum w_j(1 - e^{-rC_j})$.*

Proof By contradiction. Assume otherwise that there exists another optimal schedule S , in which job J_j immediately precedes job J_k , while

$$\frac{w_j e^{-rp_j}}{1 - e^{-r(1+\gamma)p_j}} > \frac{w_k e^{-rp_k}}{1 - e^{-r(1+\gamma)p_k}}.$$

Let t be the time at which job J_j starts its processing in S . An adjacent pairwise interchange between the two jobs results in a new schedule S' . It is observed that for each of the other jobs except J_j and J_k , its starting time and end time in the two sequences are the same. So, the only difference between S and S' related to the objective value is due to jobs J_j and J_k . Together with formula (Eq. 3), the total contribution of the two jobs to the objective under S is

$$\begin{aligned} & w_j(1 - e^{-r(t+p_j+q_j)}) + w_k(1 - e^{-r(t+p_j+p_k+q_k)}) \\ &= w_j(1 - e^{-r(t+p_j+\gamma t)}) + w_k(1 - e^{-r(t+p_j+p_k+\gamma(t+p_j))}). \end{aligned} \tag{7}$$

The total contribution of jobs J_k and J_j to the objective under S' is obtained by interchanging the j s and k s in formula (Eq. 7). By algebraic calculation, it can be shown that the objective value under S' is less than that under S , implying a contradiction. This completes the proof. \square

2.5 Total Weighted Tardiness

It is well known that problem $1||\sum w_j T_j$ is NP-hard in the strong sense, which is a special case of $1|q_{psd}|\sum w_j T_j$. This implies that $1|q_{psd}|\sum w_j T_j$ is also NP-hard

in the strong sense. In this subsection, the main focus is on a polynomially solvable special case. Given a job instance, if any two jobs J_j, J_k with $p_j \leq p_k$ satisfy $d_j \leq d_k$ and $w_j \geq w_k$, then it is said the job instance has *agreeable* due dates and *agreeable* weights.

Theorem 4 *If a job instance has agreeable due dates and agreeable weights, then Earliest Due Date (EDD) rule is optimal for $1|q_{psd}|\sum w_j T_j$.*

Proof Suppose otherwise in an optimal schedule S , there exist two jobs J_j and J_k with $p_j \leq p_k$ such that J_k is scheduled before J_j . By agreeable due dates and weights, $d_j \leq d_k$ and $w_j \geq w_k$ is obtained. Performing a *position interchange* on jobs J_k and J_j in S while keeping the processing order of all the other jobs unchanged, a new schedule S' is obtained. It suffices to prove that the total weighted tardiness under S' is no more than that of S .

Let A and $P(A)$ be the set of jobs between J_j and J_k and its total processing time in S . Let t be the time at which job J_k starts its processing under S . J_k is immediately followed by A and then J_j . Under the new schedule S' , however, J_j will start its processing at time t , and it is immediately followed by A and then J_k . Notice that the total weighted tardiness of all the jobs except J_j and J_k under S' is less than or equal to that under S due to $p_j \leq p_k$. The rest of the proof is to show that the total weighted tardiness of J_j and J_k under S' is no more than that under S .

The total weighted tardiness of J_j and J_k under S is

$$\begin{aligned} w_k T_k + w_j T_j &= w_k \max\{t + p_k + \gamma t - d_k, 0\} \\ &\quad + w_j \max\{t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j, 0\}. \end{aligned}$$

While under S' , the total contribution of J_j and J_k is

$$\begin{aligned} w_j T'_j + w_k T'_k &= w_j \max\{t + p_j + \gamma t - d_j, 0\} \\ &\quad + w_k \max\{t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k, 0\}. \end{aligned}$$

Define $\Delta = (w_j T'_j + w_k T'_k) - (w_k T_k + w_j T_j)$. Depending on Δ , two cases are discussed.

Case 1: $t + p_k + \gamma t - d_k \geq 0$.

Case 1.1: $t + p_j + \gamma t - d_j \geq 0$.

$$\begin{aligned} \Delta &= [w_j(t + p_j + \gamma t - d_j) \\ &\quad + w_k(t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k)] \\ &\quad - [w_k(t + p_k + \gamma t - d_k) \\ &\quad + w_j(t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j)] \end{aligned}$$

$$\begin{aligned}
&= (1 + \gamma)[w_k(p_j + P(A)) - w_j(p_k + P(A))] \\
&\leq 0
\end{aligned}$$

where the above inequality is due to $w_j \geq w_k$ and $p_j \leq p_k$.

Case 1.2: $t + p_j + \gamma t - d_j < 0$.

$$\begin{aligned}
\Delta &= w_k[t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k] \\
&\quad - [w_k(t + p_k + \gamma t - d_k) \\
&\quad + w_j(t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j)] \\
&= w_k(1 + \gamma)(p_j + P(A)) \\
&\quad - w_j(t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j) \\
&\leq w_k[(1 + \gamma)(p_j + P(A)) \\
&\quad - (t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j)] \\
&= w_k[\gamma(p_j - p_k) - (t + p_k + \gamma t - d_j)] \\
&\leq -w_k(t + p_k + \gamma t - d_j) \\
&\leq -w_k(t + p_k + \gamma t - d_k) \\
&\leq 0
\end{aligned}$$

where the first inequality is due to $w_j \geq w_k$ and $t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j \geq t + p_k + \gamma t - d_k \geq 0$ by the condition of Case 1, the second and third inequalities are due to $p_j \leq p_k$ and $d_j \leq d_k$ respectively, and the last inequality is due to case condition $t + p_k + \gamma t - d_k \geq 0$.

Case 2: $t + p_k + \gamma t - d_k < 0$.

Case 2.1: $t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k \geq 0$.

Case 2.1.1: $t + p_j + \gamma t - d_j \geq 0$.

$$\begin{aligned}
\Delta &= [w_j(t + p_j + \gamma t - d_j) \\
&\quad + w_k(t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k)] \\
&\quad - w_j(t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j) \\
&= w_k(t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k) \\
&\quad - w_j(1 + \gamma)(p_k + P(A)) \\
&< w_k(1 + \gamma)(p_j + P(A)) - w_j(1 + \gamma)(p_k + P(A)) \\
&\leq 0
\end{aligned}$$

where the first inequality is due to case condition $t + p_k + \gamma t - d_k < 0$ and the second inequality is due to $w_j \geq w_k$ and $p_j \leq p_k$.

Case 2.1.2: $t + p_j + \gamma t - d_j < 0$.

$$\begin{aligned}\Delta &= w_k(t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k) \\ &\quad - w_j(t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j) \\ &\leq w_j[(\gamma p_j - d_k) - (\gamma p_k - d_j)] \\ &= w_j[\gamma(p_j - p_k) + (d_j - d_k)] \\ &\leq 0,\end{aligned}$$

where the first inequality is due to $w_k \leq w_j$ and the second inequality is due to $p_j \leq p_k$ and $d_j \leq d_k$.

Case 2.2: $t + p_j + P(A) + p_k + \gamma(t + p_j + P(A)) - d_k < 0$.

Case 2.2.1: $t + p_j + \gamma t - d_j \geq 0$.

$$\begin{aligned}\Delta &= w_j(t + p_j + \gamma t - d_j, 0) \\ &\quad - w_j[t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j] \\ &= -w_j(1 + \gamma)(p_k + P(A)) \\ &\leq 0\end{aligned}$$

Case 2.2.2: $t + p_j + \gamma t - d_j < 0$.

$$\begin{aligned}\Delta &= 0 - w_j \max\{t + p_k + P(A) + p_j + \gamma(t + p_k + P(A)) - d_j, 0\} \\ &\leq 0.\end{aligned}$$

This completes the proof. \square

2.6 Total Completion Time with Release Dates

It is already known that $1|r_j| \sum C_j$ is NP-hard (see [9]), and therefore $1|r_j, q_{psd}| \sum C_j$ is NP-hard. The focus is to exploit some polynomially solvable cases of the latter problem. Remember that in Sect. 2.3, Corollary 1 shows that SPT rule produces an optimal schedule for the model without release dates, that is, $1|q_{psd}| \sum C_j$.

This subsection deals with the model with release dates and preemption, that is, $1|q_{psd}, prmp, r_j| \sum C_j$. In the preemptive case, one job on processing may be interrupted at some time and resumed from where it was preempted at a later time. It is proved that a modified shortest remaining processing time (MSRPT) rule is optimal for the model. Notice that in the preemption model, on the case where one job J_j has started processing for more than one time, S_j is defined as the first starting time of the job. The value of psd delivery time depends on J_j 's first starting time, that is, $q_j = \gamma S_j$. The following lemma shows one property in an optimal schedule.

Lemma 3 *In an optimal schedule for $1|q_{psd}, prmp, r_j| \sum C_j$, each job starts processing at its release date.*

Proof By contradiction. Suppose otherwise in an optimal schedule π , job J_j starts processing at time S_j with $S_j > r_j$. Construct a new preemptive schedule π' by inserting processing J_j at time r_j with a time length of zero and keep the processing schedule of all jobs in π unchanged. Since the value of S_j decreases from S_j in π to r_j in π' , together with formula (Eq. 4), the psd delivery time and the completion time of J_j in π' are strictly less than that in π . The completion times of all other jobs are the same in π and π' . Hence, the total completion time in π' is less than that in π , contradicting that π is an optimal schedule. \square

Algorithm MSRPT:

(At any time Principle 1 is superior to Principle 2)

Principle 1: At the release date of each job, process it with a time length of zero.

Principle 2: If the machine is idle or a job arrives at time t , schedule a job by SRPT (shortest remaining processing time) rule. Ties are broken arbitrarily. If there is no available job at the time, wait until a new job arrives. Note that the remaining processing time includes psd delivery time.

Theorem 5 *Algorithm MSRPT is optimal for $1|q_{psd}, prmp, r_j| \sum C_j$.*

Proof First, give a sketch of the proof. By Lemma 3, the psd delivery time of each J_j satisfies $q_j = \gamma S_j = \gamma r_j$, and thus it has no relation with the schedule of the processing of J_j . The problem reduces to $1|prmp, r_j| \sum C_j$. By the argument of position interchange on the processing of any two consecutive jobs, it can be proved that MSRPT produces an optimal schedule.

The theorem follows. \square

2.7 Approximation for Total Completion Time with Release Dates

For the model with release dates, if preemption is allowed, the previous subsection shows that there exists an optimal algorithm. However, if preemption is not allowed, the corresponding problem, denoted by $1|q_{psd}, r_j| \sum C_j$, is NP-hard due to the NP-hardness of $1|r_j| \sum C_j$. Due to the NP-hardness result, the approximation algorithm is developed for problem $1|q_{psd}, r_j| \sum C_j$ in this subsection. The main idea of the proposed approximation algorithm below is to transform a preemptive schedule into a corresponding non-preemptive schedule. The total relaxation of jobs' completion times during the transformation is bounded by a constant factor.

Given an instance of problem $1|q_{psd}, prmp, r_j| \sum C_j$ and a preemptive schedule P which is produced by MSRPT algorithm, let $C_j(P)$ be the completion time of job J_j in P . The following algorithm, named CONVERT, reveals how to convert schedule P into a feasible nonpreemptive schedule N .

Algorithm CONVERT:

Input: Preemptive schedule P

Output: Nonpreemptive schedule N

1. Form a list L of jobs J_1, J_2, \dots, J_n in the increasing order of their completion times $C_j(P)$, $j = 1, 2, \dots, n$.
2. Produce a nonpreemptive and feasible schedule of the n jobs in the same completion order as in L , with the constraint that each job starts processing not earlier than its release date.

Theorem 6 *Given an optimal preemptive schedule P for $1|q_{psd}, prmp, r_j| \sum C_j$. Algorithm CONVERT produces in $O(n)$ time a nonpreemptive schedule N in which $C_j(N) \leq 2(1 + \gamma)C_j(P)$ for each job J_j .*

Proof By the description of Algorithm CONVERT, it works as follows. Consider an arbitrary job J_j in P . Let $S_j^l(P)$ be the last starting time of J_j and k_j the length of time for its last processing. That is, the last processing of J_j is within time interval $[S_j^l(P), S_j^l(P) + k_j]$ in schedule P . Remove all the previous processing time segments of J_j in a total length of $p_j - k_j$ before time $S_j^l(P)$, and insert $p_j - k_j$ extra units of time at time $S_j^l(P) + k_j$ for processing J_j continuously within interval $[S_j^l(P), S_j^l(P) + p_j]$. Note that this time interval might overlap with the processing time intervals of either preceding or succeeding jobs. Hence, in the nonpreemptive schedule N , the starting time of job J_j or $S_j(N)$ may be postponed to another time later than $S_j^l(P)$ with the assurance that there is no unforced idle time between the processing of any two jobs.

It is observed that in the worst case of N , all the jobs completed before J_j are started on or after time $S_j^l(P)$. Together with the fact that the jobs completed before J_j in P are the same as that in N , the following is obtained:

$$\begin{aligned} S_j(N) &\leq S_j^l(P) + \sum_{k:C_k(P) < C_j(P)} p_k \\ &\leq S_j^l(P) + (C_j(P) - \gamma r_j - p_j) \\ &< 2C_j(P) - p_j, \end{aligned}$$

where the second inequality holds since the processing of J_j is completed at time $C_j(P) - \gamma r_j$ and then $\sum_{k:C_k(P) \leq C_j(P)} p_k \leq C_j(P) - \gamma r_j$ and the third inequality is due to $S_j^l(P) < C_j(P)$ and $\gamma r_j \geq 0$. Together with formulae (2) and (4),

$$\begin{aligned} C_j(N) &\leq (1 + \gamma)S_j(N) + p_j \\ &\leq (1 + \gamma)(2C_j(P) - p_j) + p_j \\ &< 2(1 + \gamma)C_j(P). \end{aligned}$$

The theorem follows. \square

Corollary 2 $\sum C_j(N) \leq 2(1 + \gamma) \sum C_j(P)$.

Denote by $\sum C_j(OPT)$ the optimal objective value for $1|q_{psd}, r_j| \sum C_j$. It can be claimed that $\sum C_j(OPT)$ is not less than $\sum C_j(P)$ which is the optimal objective value for $1|q_{psd}, prmp, r_j| \sum C_j$. Hence,

$$\begin{aligned}\sum C_j(N) &\leq 2(1 + \gamma) \sum C_j(P) \\ &\leq 2(1 + \gamma) \sum C_j(OPT).\end{aligned}$$

Theorem 7 Algorithm CONVERT is $2(1 + \gamma)$ -approximation for $1|q_{psd}, r_j| \sum C_j$.

3 Parallel Identical Machine Model

In this section, minimizations of the makespan and the total completion time are considered.

3.1 Makespan

Problem $Pm|q_{psd}|C_{\max}$ is studied in this subsection. It is already known that $Pm||C_{\max}$ is NP-hard in the ordinary sense. The problem is a special case of $Pm|q_{psd}|C_{\max}$ when $\gamma = 0$. Therefore, some heuristics are developed for the problem considered.

A simple idea is to investigate the performance of LIST rule. LIST schedules each job on one machine with smallest current workload.

Theorem 8 LIST rule is $(1 + \frac{(1+\gamma)(m-1)}{m})$ -approximation for $Pm|q_{psd}|C_{\max}$.

Proof Let C_{\max}^{LIST} and C_{\max}^* denote the makespan (including *psd* delivery time) of the schedule produced by LIST rule and the optimal objective value, respectively. Let $J_l (1 \leq l \leq n)$ be the job which determines the makespan C_{\max}^{LIST} . Note that J_l might not be the last scheduled job. The start time of J_l satisfies

$$S_l \leq \frac{\sum_{j=1}^{l-1} p_j}{m} \leq \frac{\sum_{j=1}^n p_j - p_l}{m}.$$

Together with formulae (Eq. 2) and (Eq. 4),

$$\begin{aligned}C_{\max}^{LIST} &= (1 + \gamma)S_l + p_l \\ &\leq \frac{(1 + \gamma) \sum_{j=1}^n p_j}{m} + (1 - \frac{1 + \gamma}{m})p_l.\end{aligned}$$

By LIST rule, $C_{\max}^* \geq \max\{\frac{\sum_{j=1}^n p_j}{m}, p_l\}$. Hence,

$$\frac{C_{\max}^{LIST}}{C_{\max}^*} \leq 1 + \frac{(1+\gamma)(m-1)}{m}.$$

The theorem follows. \square

3.1.1 An FPTAS

Let M_i ($i \in \{1, \dots, m\}$) denote the i th machine in the system. Given an arbitrary schedule, let $J_{[i,j]} (j \in \{1, \dots, n_i\})$ be the j th job scheduled on the machine M_i . Denote by $s_{[i,j]}$ and $C_{[i,j]}$ the starting time and the completion time of job $J_{[i,j]}$, respectively. Let $n_i (i \in \{1, \dots, m\})$ be the number of jobs scheduled on machine M_i . It follows that $n_i \in \{1, \dots, n\}$ for $i \in \{1, \dots, m\}$ and $\sum n_i = n$.

For any problem with some minimization objective, an algorithm \mathcal{A} is called $(1 + \epsilon)$ -approximation if it produces a solution with the objective value at most $1 + \epsilon$ times of the optimal value, and its running time is polynomial in input size. A family of approximation algorithms $\{\mathcal{A}_\epsilon\}$ is called a fully polynomial time approximation scheme (FPTAS) if, for each $\epsilon > 0$, \mathcal{A}_ϵ is a $(1 + \epsilon)$ -approximation algorithm running in polynomial time in both input size and $1/\epsilon$. Without loss of generality, it is assumed that $0 < \epsilon \leq 1$.

For the special case with $m = 1$, that is, $1|q_{psd}|C_{\max}$, it is already known that SPT rule produces an optimal schedule. It immediately leads to the following lemma.

Lemma 4 *In an optimal solution of $Pm|q_{psd}|C_{\max}$, all the jobs on each machine $M_i (i \in \{1, \dots, m\})$ are sequenced by SPT rule.*

According to the above lemma, index the n jobs according to SPT rule. Similar to the establishment of FPTAS in [7,8], introduce variables $x_j (j = 1, \dots, n)$ such that $x_j = i$ if job J_j is assigned to machine $M_i (i \in \{1, \dots, m\})$. Let vector $x = (x_1, x_2, \dots, x_n)$ be one assignment of all the n jobs and $X = \{x\}$ be the vector set containing all the solutions to the problem considered. Note that the only concern is nondelay schedule here since an optimal solution must be such a schedule.

For each vector x , adopt function $f_j^i(x)$ and $g_j^i(x)$, respectively, to denote the processing workload (excluding psd delivery time) and the makespan (including psd delivery time) of machine M_i immediately after scheduling job J_j . Let $h_j(x) = \max_{i=1,\dots,m} \{g_j^i(x)\}$. So $h_n(x)$ is our desired value for any given x .

Define the following initial and recursive functions on X :

Initial function:

$$\begin{aligned} f_0^i(x) &= 0, \quad i = 1, \dots, m; \\ g_0^i(x) &= 0, \quad i = 1, \dots, m; \\ h_0(x) &= 0. \end{aligned}$$

Recursive function for $j = 1, \dots, n$:

$$\begin{aligned}
f_j^i(x) &= f_{j-1}^i(x) + p_j, \quad i = x_j; \\
f_j^i(x) &= f_{j-1}^i(x), \quad \text{otherwise.} \\
g_j^i(x) &= (1 + \gamma) f_{j-1}^i(x) + p_j, \quad i = x_j; \\
g_j^i(x) &= g_{j-1}^i(x), \quad \text{otherwise.} \\
h_j(x) &= \max_{i=1,\dots,m} \{g_j^i(x)\}.
\end{aligned}$$

Therefore, problem $Pm|q_{psd}|C_{\max}$ reduces to minimize

$$h_n(x), \quad x \in X.$$

Since the proposed algorithm \mathcal{A}_ϵ contains procedure $\text{Partition}(A, e, \delta)$ which was proposed in [7, 8], restate the procedure for comprehension completion. In $\text{Partition}(A, e, \delta)$, parameter $A \subseteq X$, e is a nonnegative integer function on X , and $0 < \delta \leq 1$. The procedure partitions A into $k(e) \geq 1$ disjoint subsets which satisfy some specific properties. The value of $k(e)$ is to be determined in the procedure.

Procedure $\text{Partition}(A, e, \delta)$:

- Step 1.** Arrange vectors $x \in A$ in the order $x^{(1)}, x^{(2)}, \dots, x^{(|A|)}$ such that $0 \leq e(x^{(1)}) \leq e(x^{(2)}) \leq \dots \leq e(x^{(|A|)})$.
- Step 2.1.** Assign vectors $x^{(1)}, x^{(2)}, \dots, x^{(i_1)}$ to set A_1^e such that $e(x^{(i_1)}) \leq (1 + \delta)e(x^{(1)})$ and $e(x^{(i_1+1)}) > (1 + \delta)e(x^{(1)})$. If such i_1 does not exist, take $A_{k(e)}^e = A_1^e = A$ and stop.
- Step 2.2.** Assign vectors $x^{(i_1+1)}, x^{(i_1+2)}, \dots, x^{(i_2)}$ to set A_2^e such that $e(x^{(i_2)}) \leq (1 + \delta)e(x^{(i_1+1)})$ and $e(x^{(i_2+1)}) > (1 + \delta)e(x^{(i_1+1)})$. If such i_2 does not exist, take $A_{k(e)}^e = A_2^e = A$ and stop.
- Step 2.3.** Recursively, construct vector set A_i^e for $i \geq 3$ as in Step 2.2 until $x^{(|A|)}$ is included in $A_{k(e)}^e$ for some integer $k(e)$.

Procedure $\text{Partition}(A, e, \delta)$ runs in $O(|A| \log |A|)$ time since Step 1 consumes $O(|A| \log |A|)$ time while the rest of the steps for vector partition takes only $O(|A|)$ time. Further, we use two properties of $\text{Partition}(A, e, \delta)$ proposed in [7, 8] in the development of our FPTAS and the establishment of its time complexity.

Property 1 $|e(x) - e(x')| \leq \delta \min\{e(x), e(x')\}$ for any $\{x, x'\} \in A_j^e$ ($j = 1, \dots, k(e)$).

Property 2 $k \leq \frac{\log e(x^{(|A|})}}{\delta} + 2$ for $0 < \delta \leq 1$ and $1 \leq e(x^{(|A|})}$.

It is now ready to present the FPTAS \mathcal{A}_ϵ for $Pm|q_{psd}|C_{\max}$ below.

Algorithm \mathcal{A}_ϵ :

- Step 1.** (Initialization) Reindex the jobs in the nondecreasing order of p_j . Set $Y_0 = \underbrace{\{(0, 0, \dots, 0)\}}_n$ (which means no job is assigned initially) and $j = 1$.

Step 2.1. (Generate Y_1, \dots, Y_n) For set Y_{j-1} , construct Y'_j by adding $x_j \in \{1, \dots, m\}$ in position j of each vector from Y_{j-1} . Calculate the following recursive function for each $x \in Y'_j$:

$$f_j^i(x) = f_{j-1}^i(x) + p_j, \quad i = x_j;$$

$$f_j^i(x) = f_{j-1}^i(x), \quad \text{otherwise.}$$

$$g_j^i(x) = (1 + \gamma)f_{j-1}^i(x) + p_j, \quad i = x_j;$$

$$g_j^i(x) = g_{j-1}^i(x), \quad \text{otherwise.}$$

$$h_j(x) = \max_{i=1, \dots, m} \{g_j^i(x)\}.$$

Step 2.2. If $j = n$, take $Y_n = Y'_n$ and go to Step 3.

Step 2.3. Otherwise, if $j < n$, set $\delta = \frac{\epsilon}{2n}$ and perform the following operations:

Step 2.3.1. Call $\text{Partition}(Y'_j, f_j^i, \delta)$ ($i = 1, \dots, m$) to divide set Y'_j into $k(f^i)$ disjoint subsets $Y_1^{f^i}, Y_2^{f^i}, \dots, Y_{k(f^i)}^{f^i}$.

Step 2.3.2. Call $\text{Partition}(Y'_j, h_j, \delta)$ to divide set Y'_j into $k(h)$ disjoint subsets $Y_1^h, Y_2^h, \dots, Y_{k(h)}^h$.

Step 2.3.3. Divide set Y'_j into disjoint subsets $Y_{a_1 \dots a_m b} = Y_{a_1}^{f^1} \cap \dots \cap Y_{a_m}^{f^m} \cap Y_b^h$ for $a_1 = 1, 2, \dots, k(f^1); \dots; a_m = 1, 2, \dots, k(f^m); b = 1, 2, \dots, k(h)$.

Step 2.3.4. For each nonempty subset $Y_{a_1 \dots a_m b}$, choose a vector $x^{(a_1 \dots a_m b)}$ such that $h_j(x^{(a_1 \dots a_m b)}) = \min\{h_j(x) | x \in Y_{a_1 \dots a_m b}\}$.

Step 2.3.5. Set $Y_j := \{x^{(a_1 \dots a_m b)} | a_1 = 1, 2, \dots, k(f^1); \dots; a_m = 1, 2, \dots, k(f^m); b = 1, 2, \dots, k(h)\}$, and $Y_{a_1}^{f^1} \cap \dots \cap Y_{a_m}^{f^m} \cap Y_b^h \neq \emptyset\}; j = j + 1$, and go to Step 2.1.

Step 3. (Solution) Select vector $x^0 \in Y_n$ such that $h_n(x^0) = \min\{h_n(x) | x \in Y_n\}$.

The purpose of Step 2.3 is to reduce the solution space in the current iteration which will be used to generate the solution space in the next iteration. In each iteration, such reduction guarantees that the function values of omitted solutions are within $(1 + \delta)$ times of the values of remained solutions.

Theorem 9 For $Pm|q_{psd}|C_{\max}$, algorithm \mathcal{A}_ϵ finds a solution x^0 such that $h_n(x^0) \leq (1 + \epsilon)h_n(x^*)$.

Proof Let vector $x^* = (x_1^*, \dots, x_n^*)$ be an optimal solution for $Pm|q_{psd}|C_{\max}$. Suppose there is a partial optimal solution $(x_1^*, \dots, x_j^*, 0, \dots, 0) \in Y_{a_1 \dots a_m b} \subseteq Y'_j$ for some j and a_1, \dots, a_m, b . By Step 2 of algorithm \mathcal{A}_ϵ , such j always exists, for instance, $j = 1$. By Step 2.3.4 of algorithm \mathcal{A}_ϵ , no matter whether or not $(x_1^*, \dots, x_j^*, 0, \dots, 0)$ is chosen as $x^{(a_1 \dots a_m b)}$, it is obtained by [Property 1](#) that

$$|f_j^i(x_1^*, \dots, x_j^*, 0, \dots, 0) - f_j^i(x^{(a_1 \dots a_m b)})| \leq \delta f_j^i(x_1^*, \dots, x_j^*, 0, \dots, 0),$$

$$i = 1, \dots, m,$$

and

$$|h_j(x_1^*, \dots, x_j^*, 0, \dots, 0) - h_j(x^{(a_1 \dots a_m b)})| \leq \delta h_j(x_1^*, \dots, x_j^*, 0, \dots, 0),$$

which means

$$h_j(x^{(a_1 \cdots a_m b)}) \leq (1 + \delta)h_j(x_1^*, \dots, x_j^*, 0, \dots, 0). \quad (8)$$

Now establish the relation between solutions obtained by \mathcal{A}_ϵ and a partial optimal solution in the next iteration.

Consider two vectors which assign $j + 1$ jobs, namely, $(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0)$ and $\hat{x}^{(a_1 \cdots a_m b)} = (x_1^{(a_1 \cdots a_m b)}, \dots, x_j^{(a_1 \cdots a_m b)}, x_{j+1}^*, 0, \dots, 0)$. (\hat{x} is chosen as a bridge connecting a partial optimal solution and the solutions produced by \mathcal{A}_ϵ in the next iteration). Without loss of generality, assume $x_{j+1}^* = u$. It follows that

$$\begin{aligned} & |g_{j+1}^u(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0) - g_{j+1}^u(\hat{x}^{(a_1 \cdots a_m b)})| \\ &= |(1 + \gamma)f_j^u(x_1^*, \dots, x_j^*, 0, \dots, 0) + p_j - (1 + \gamma)f_j^u(x^{(a_1 \cdots a_m b)}) - p_j| \\ &= (1 + \gamma)|f_j^u(x_1^*, \dots, x_j^*, 0, \dots, 0) - f_j^u(x^{(a_1 \cdots a_m b)})| \\ &\leq (1 + \gamma)\delta f_j^u(x_1^*, \dots, x_j^*, 0, \dots, 0) \\ &\leq \delta g_{j+1}^u(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0). \end{aligned}$$

Therefore,

$$g_{j+1}^u(\hat{x}^{(a_1 \cdots a_m b)}) \leq (1 + \delta)g_{j+1}^u(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0).$$

According to the above analysis, the following is obtained:

$$\begin{aligned} & g_{j+1}^i(\hat{x}^{(a_1 \cdots a_m b)}) \\ &\leq (1 + \delta)g_{j+1}^i(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0), \quad i = 1, \dots, m. \end{aligned} \quad (9)$$

Combining inequality (Eq. 9) and the definition of $h_j(x)$,

$$\begin{aligned} & |h_{j+1}(\hat{x}^{(a_1 \cdots a_m b)}) - h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0)| \\ &= |\max_{i=1, \dots, m}\{g_{j+1}^i(\hat{x}^{(a_1 \cdots a_m b)})\} \\ &\quad - \max_{i=1, \dots, m}\{g_{j+1}^i(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0)\}| \\ &\leq \delta g_{j+1}^i(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0) \\ &\leq \delta \max_{i=1, \dots, m}\{g_{j+1}^i(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0)\} \\ &= \delta h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0). \end{aligned}$$

Consequently,

$$h_{j+1}(\hat{x}^{(a_1 \cdots a_m b)}) \leq (1 + \delta)h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0). \quad (10)$$

Set $\delta_j = \delta$ in this, namely, j th, iteration. Then the following inequalities follows directly by inequalities (Eq. 8)–(Eq. 10):

$$h_j(x^{(a_1 \cdots a_m b)}) \leq (1 + \delta_j)h_j(x_1^*, \dots, x_j^*, 0, \dots, 0). \quad (11)$$

$$\begin{aligned} & g_{j+1}^i(\hat{x}^{(a_1 \cdots a_m b)}) \\ & \leq (1 + \delta_j)g_{j+1}^i(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0), \quad i = 1, \dots, m. \end{aligned} \quad (12)$$

$$h_{j+1}(\hat{x}^{(a_1 \cdots a_m b)}) \leq (1 + \delta_j)h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0). \quad (13)$$

Without loss of generality, assume that $\hat{x}^{(a_1 \cdots a_m b)} \in Y_{c_1 \cdots c_m d} \subseteq Y'_{j+1}$ and algorithm \mathcal{A}_ϵ chooses $x^{(c_1 \cdots c_m d)}$ instead of $\hat{x}^{(a_1 \cdots a_m b)}$ in its $(j+1)$ th iteration. By Property 1 and inequalities (Eq. 12) and (Eq. 13), the following are obtained:

$$\begin{aligned} & |g_{j+1}^i(\hat{x}^{(a_1 \cdots a_m b)}) - g_{j+1}^i(x^{(c_1 \cdots c_m d)})| \\ & \leq \delta g_{j+1}^i(\hat{x}^{(a_1 \cdots a_m b)}) \\ & \leq \delta(1 + \delta_j)g_{j+1}^i(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0), \quad i = 1, \dots, m \end{aligned}$$

and

$$\begin{aligned} & |h_{j+1}(\hat{x}^{(a_1 \cdots a_m b)}) - h_{j+1}(x^{(c_1 \cdots c_m d)})| \\ & \leq \delta h_{j+1}(\hat{x}^{(a_1 \cdots a_m b)}) \\ & \leq \delta(1 + \delta_j)h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0). \end{aligned} \quad (14)$$

Together with inequalities (Eq. 13) and (Eq. 14),

$$\begin{aligned} & |h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0) - h_{j+1}(x^{(c_1 \cdots c_m d)})| \\ & \leq |h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0) - h_{j+1}(\hat{x}^{(a_1 \cdots a_m b)})| \\ & \quad + |h_{j+1}(\hat{x}^{(a_1 \cdots a_m b)}) - h_{j+1}(x^{(c_1 \cdots c_m d)})| \\ & \leq (\delta_j + \delta(1 + \delta_j))h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0) \\ & = (\delta + (1 + \delta)\delta_j)h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0). \end{aligned}$$

Set $\delta_{j+1} = \delta + (1 + \delta)\delta_j$ in this $(j+1)$ th iteration. It follows that

$$h_{j+1}(x^{(c_1 \cdots c_m d)}) \leq (1 + \delta_{j+1})h_{j+1}(x_1^*, \dots, x_j^*, x_{j+1}^*, 0, \dots, 0).$$

Repeat the above analysis for $j+2, \dots, n$. There exists an $x' \in Y_n$ such that

$$h_n(x') \leq (1 + \delta_n)h_n(x^*), \quad (15)$$

where δ_n is by the following induction:

$$\begin{aligned} \delta_n &= \delta + (1 + \delta)\delta_{n-1} \\ &= \delta + (1 + \delta)(\delta + (1 + \delta)\delta_{n-2}) \\ &= \dots \\ &= \delta \sum_{i=0}^{n-j-1} (1 + \delta)^i + (1 + \delta)^{n-j}\delta_j. \end{aligned}$$

Remember that it has been set $\delta_j = \delta$. By the above formula, δ_n achieves its maximal value when $j = 1$. That is,

$$\begin{aligned} \delta_n &\leq \delta \sum_{i=0}^{n-2} (1 + \delta)^i + (1 + \delta)^{n-1}\delta \\ &= \delta \sum_{i=0}^{n-1} (1 + \delta)^i \\ &= (1 + \delta)^n - 1 \\ &= \sum_{i=1}^n \frac{n(n-1)\dots(n-i+1)}{i!} \delta^i. \end{aligned}$$

By Step 2.3, substituting $\delta = \frac{\epsilon}{2n}$ in the above formula on the right-hand side of the third equation, the following is obtained:

$$\begin{aligned} \delta_n &\leq \sum_{i=1}^n \frac{n(n-1)\dots(n-i+1)}{i!} \left(\frac{\epsilon}{2n}\right)^i \\ &\leq \sum_{i=1}^n \frac{1}{i!} \left(\frac{\epsilon}{2}\right)^i \\ &\leq \epsilon \sum_{i=1}^n \left(\frac{1}{2}\right)^i \\ &\leq \epsilon. \end{aligned}$$

Together with inequality (Eq. 15),

$$h_n(x') \leq (1 + \epsilon)h_n(x^*).$$

By Step 3 of the algorithm, vector x^0 is chosen such that

$$h_n(x^0) \leq h_n(x') \leq (1 + \epsilon)h_n(x^*).$$

This completes the proof. \square

Theorem 10 Algorithm \mathcal{A}_ϵ requires $O(\frac{m(m+1)n^2}{\epsilon} \log L \log n)$ time.

Proof It suffices to consider the most time-consuming operation of iteration j in Step 2, that is, *Partition*. The time complexity of procedure $\text{Partition}(Y'_j, f_j^i, \delta)$ as well as $\text{Partition}(Y'_j, h_j, \delta)$ is $O(Y'_j \log Y'_j)$. In Step 2, by the construction of Y'_{j+1} , $|Y'_{j+1}| \leq m|Y_j| \leq m \cdot k(f^1) \cdots k(f^m) \cdot k(h)$ is obtained. Define $L = \sum_{i=1}^n p_i$. By Property 2, for each $i \in \{1, \dots, m\}$,

$$k(f^i) \leq \frac{2n}{\epsilon} \log L + 2.$$

Similarly,

$$k(h) \leq \frac{2n}{\epsilon} \log L + 2.$$

Therefore, $|Y'_j| \leq \frac{2mn}{\epsilon} \log L + 2 = O(\frac{mn}{\epsilon} \log L)$ and then $|Y'_j| \log |Y'_j| = O(\frac{mn}{\epsilon} \log L \log n)$. There are at most n iterations, in each of which procedure *Partition* is called $m + 1$ times. So the time complexity of algorithm \mathcal{A}_ϵ is $O(\frac{m(m+1)n^2}{\epsilon} \log L \log n)$. The theorem follows. \square

3.2 Total Completion Time

For problem $Pm|q_{psd}|\sum C_j$, earliest completion time (ECT) rule is equal to SPT rule. Below, a fundamental property for an ECT schedule is presented.

Property 3 Among all feasible schedules for $Pm|q_{psd}|\sum C_j$, an ECT schedule has the most number of completed jobs at any time point during processing.

The above property directly deduces the following theorem.

Theorem 11 *ECT (or SPT) rule is optimal for $Pm|q_{psd}|\sum C_j$.*

4 Conclusion

This chapter has provided an overview of scheduling with past-sequence-dependent delivery times. It has described the major results on single and identical parallel machine settings.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grant 71101106, 71172189, and 71090404/71090400.

Cross-References

- ▶ Complexity Issues on PTAS
- ▶ Greedy Approximation Algorithms

Recommended Reading

1. A. Borodin, R. El-yaniv, *Online Computation and Competitive Analysis* (Cambridge University Press, Cambridge, 1998)
2. S. Browne, U. Yechiali, Scheduling deteriorating jobs on a single processor. *Oper. Res.* **38**, 495–498 (1990)
3. P. Brucker, *Scheduling Algorithms*, 5th edn. (Springer, Berlin, 2006)
4. R.M. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic machine scheduling: a survey. *Ann. Discret. Math.* **5**, 287–326 (1979)
5. C. Koulamas, G.J. Kyparisis, Single-machine scheduling problems with past-sequence-dependent setup times. *Eur. J. Oper. Res.* **187**, 1045–1049 (2008)
6. C. Koulamas, G.J. Kyparisis, Single-machine scheduling problems with past-sequence-dependent delivery times. *Int. J. Prod. Econ.* **126**, 264–266 (2010)
7. M.Y. Kovalyov, W. Kubiak, A fully polynomial approximation scheme for minimizing makespan of deteriorating jobs. *J. Heuristics* **3**, 287–297 (1998)
8. M.Y. Kovalyov, W. Kubiak, A fully polynomial approximation scheme for the weighted earliness-tardiness problem. *Oper. Res.* **47**, 757–761 (1999)
9. J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, Complexity of machine scheduling problems. *Ann. Discret. Math.* **1**, 343–362 (1977)
10. M. Liu, F. Zheng, C. Chu, Y. Xu, New results on single-machine scheduling; past-sequence-dependent delivery time. *Theor. Comput. Sci.* **438**, 55–61 (2012)
11. M. Liu, F. Zheng, C. Chu, Y. Xu, Single-machine scheduling with past-sequence-dependent delivery times and release times. *Inf. Process. Lett.* **112**, 835–838 (2012)
12. M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 2nd edn. (Prentice Hall, New York, 2005)
13. F.B. Sidney, Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs. *Oper. Res.* **23**, 283–298 (1975)

Algebrization and Randomization Methods

Liang Ding and Bin Fu

Contents

1	Introduction	172
2	Preliminaries	173
2.1	Basic Notations and Definitions	173
2.2	Rudimentary Knowledge of Algebra	174
2.3	Schwartz-Zippel Lemma	176
3	k-Path Problem	177
3.1	A Brief History of the k -Path Problem	178
3.2	$O^*(2^k)$ Time Randomized Algorithm	178
4	Hamiltonian Path Problem	185
4.1	A Brief History of TSP and Hamiltonian Path Problem	185
4.2	Overall Idea of Björklund's Approach	186
4.3	Notations and Definitions	187
4.4	Cycle Cover Cancelation in Characteristic Two	188
4.5	Determinants and Inclusion-Exclusion	190
4.6	Hamiltonicity in Undirected Graphs	195
5	Minimum Common String Partition Problem	201
5.1	A Brief History of MCSP	202
5.2	An $O(2^n n^{O(1)})$ Time Algorithm for General Cases	203
6	Algebraic FPT Algorithms for Channel Scheduling Problem	204
6.1	A Brief History of LTDR	205
6.2	Algebraic FPT Algorithms for LTDR	206
7	The Complexity of Testing Monomials	208
7.1	Notations and Definitions	208
7.2	3SAT and Related NP-Complete Problems	210
7.3	$\prod \sum \prod$ Polynomials	211
7.4	Complexity for Coefficient of Monomial	212
8	Hardness of Approximation of Integration and Differentiation	213
8.1	Inapproximation of Integration	213
8.2	Inapproximation of Differentiation	215

B. Fu (✉) • L. Ding

Department of Computer Science, The University of Texas-Pan American, Edinburg, TX, USA
e-mail: lding@uga.edu; adamdingliang@gmail.com; bifu@cs.utpa.edu; binfu56a@gmail.com

9 Conclusion.....	216
Recommended Reading.....	217

Abstract

Over the past decades, people have dedicated great efforts to solve some NP-complete problems. Hamiltonian path problem is one of classic problems which baffle people many years. In recent years, some algebrization and randomization techniques have been developed and applied in the design of algorithms, which provides a credible alternative to traditional methods. Then the testing monomials theories were developed based on above techniques, which have been found applications in solving some critical problems in graph theory, networking, bioinformatics, etc. We give a self-contained description of the examples of successful applications by combining algebrization techniques and randomization techniques.

1 Introduction

This chapter presents examples to show some important algebrization and randomization techniques that have been effectively applied in the design of algorithms for some classic or burgeoning problems. These techniques were initialized to solve the classic path and packing problem. Then they are found applications in wide range areas such as graph theory, networking, bioinformatics, etc. Through exploiting the techniques of algebrization and randomization and combining them, some classic intractable problems such as k -path problem and Hamiltonian path problem, which baffle people for a long time, have got breakthroughs. Our aim is to make readers recognize that the new developed algebrization and randomization methods are very powerful tools for designing algorithms.

The chapter is organized as follows. [Section 2](#) provides the basic notations and definitions adopted in the rest of this chapter. Also the proof of some well-known results are showed to make the chapter self-contained. The senior readers may skip this section. In the following sections, several examples which are based on the algebrization and randomization techniques are presented. The organization for each section is similar: the beginning paragraphs give a brief glance of the history of the problem, and then the details of the algorithms and the proof the correctness are showed next.

[Section 3](#) presents an $O^*(2^k)$ time algorithm for the k -path problem. The k -path problem, which is also known as the longest path problem, has been studied for decades. With the development of bioinformatics, the k -path problem has found many applications. Combining algebrization and randomization, a big step improvement is achieved to solve the k -path problem.

In [Sect. 4](#), the similar technique is used to solve the famous Hamiltonian path problem. The Hamiltonian path problem is an important graph problem that has

been studied extensively. It is presented as an example for many standard textbooks about complexity and algorithm. For a general graph with n vertices, the fastest algorithm to solve the Hamiltonian path problem runs in $O^*(2^n)$ time. And this runtime went on about half a century since 1950s. This section shows an $O^*(1.657^n)$ time algorithm to solve the Hamiltonian path problem for the general direct graphs.

The theory talked about in Sect. 7, which is named testing monomial theory, comes from the methods used in previous two sections. Because of the important applications, it is necessary to investigate the testing monomial theory and understand how the theory relates to critical problems in complexity. This section works on the question which ask whether a polynomial represented by certain economically compact structure has a multilinear monomial in its sum-product expansion. It discusses the hardness of testing monomial for some special polynomials and gives algorithms to solve it.

Sections 5 and 6 are devoted to discussions on the applications of the testing monomial theory. Two problems, which both are newly burgeoning problems, are presented separately in two sections. Section 5 shows an $O(2^n n^{O(1)})$ time algorithm for minimum common string partition problem (MCSP). Since the applications in genome sequencing of computational biology, the MCSP problem has drawn a lot of attentions. In Sect. 6, algebraic fixed parameter tractable algorithms are presented for Channel Scheduling Problem. The Channel Scheduling Problem focuses on developing efficient scheduling algorithms form the client point of view with the objective of minimizing the access time and number of channel switches for the wireless data broadcast. By solving these two problems, the readers can see the practical applicability of the testing monomial theory. Also it reflects the power of the algebrization and randomization.

As most chapters of this book talk about the approximation algorithms for the combinatorial optimization problems, the reader can see the discussion about the parameterized complexity of combinatorial optimization problems in Chapter ▶ [Connections Between Continuous and Discrete Extremum Problems, Generalized Systems, and Variational Inequalities](#). This chapter contains some recent technologies that combine algebrization and randomization to develop parameterized algorithms and exact algorithms for some NP-hard problems. These methods also bring the new development of complexity theory. We give a self-contained description for all the results in the chapter.

2 Preliminaries

This section reviews some basic notations and theorems which will be adopted throughout the chapter.

2.1 Basic Notations and Definitions

In mathematics, polynomial is one of the most basic algebra concepts. It can be described as an expression constituted by a sum of powers in one or more

variables multiplied by coefficients. It sounds simple, but it hides many profound and complicated computation problems. The concept of polynomial runs through all the chapter. Let $\mathcal{P} \in \{Z, Z_2, \dots, Z_N\}$, $N > 2$. For variables x_1, \dots, x_n , let $\mathcal{P}[x_1, \dots, x_n]$ denote the communicative ring of all the n -variate polynomials with coefficients from \mathcal{P} . For $1 \leq i_1 \leq \dots \leq i_k \leq n$, $\pi = x_{i_1}^{j_1} \dots x_{i_k}^{j_k}$ is called a *monomial*. The *degree* of π , denoted by $\deg(\pi)$, is $\sum_{s=1}^k j_s$. π is *multilinear*, if $j_1 = \dots = j_k = 1$, i.e., π is linear in all its variables x_{i_1}, \dots, x_{i_k} . For any given integer $c \geq 1$, π is called a *c-monomial*, if $1 \leq j_1, \dots, j_k < c$.

An *arithmetic circuit*, or *circuit* for short, is a direct acyclic graph with $+$ gates of unbounded fan-ins, \times gates of two fan-ins, and all terminals corresponding to variables. The *size*, denoted by $s(n)$, of a circuit with n variables is the number of gates in it. A circuit is called a *formula*, if the fan-out of every gate is at most one, i.e., the underlying direct acyclic graph is a tree. A polynomial $p(x_1, \dots, x_n)$ can be represented by a circuit easily.

By definition, any polynomial $p(x_1, \dots, x_n)$ can be expressed as a sum of a list of monomials, called the *sum-product expansion*. The *degree* of the polynomial is the largest degree of its monomials in the expansion.

2.2 Rudimentary Knowledge of Algebra

Senior readers who are familiar with algebra may skip this section. Most of notations which are used in the following sections coincide with standard notations. However, to make the chapter self-contained, it is necessary to introduce some of them which are infrequently seen in combinatorial optimization. Rank and characteristic are very important concepts which appear throughout this chapter. The definitions are given as follows.

Definition 1 Let G be a group and a be an element of G , e is the identity of G , and the *rank* of the element a is defined as the smallest positive integer n which makes $a^n = e$. If n does not exist, a has infinite rank. The rank of a is usually represented as $|a|$.

Definition 2 The characteristic of a ring R , often denoted $Char(R)$, is defined to be the smallest number of times to get the additive identity element 0 by summarizing the multiplicative identity element 1 of R ; the ring R is said to have characteristic zero if this repeated sum never reaches the additive identity.

Theorem 1 ([25]) *If R is a ring with identity and without zero divisors, then all of the elements except the additive identity element 0 have the same characteristic which is either 0 or a prime p .*

In abstract algebra, a field F is a commutative ring whose nonzero elements form a group under multiplication. Thus, each nonzero element a of F has inverse element a^{-1} . It is easy to see that there is no zero divisor in F . Therefore, the above

theorem holds for a field F . Similar to a ring, a field also has its characteristic. Because of [Theorem 1](#), the characteristic of a finite field F must be a prime number.

The main methods based on algebrization and randomization are established over some special algebraic structures. Two special algebraic space $GF(p^n)$ and \mathbb{Z}_2^k are frequently used here. $GF(p^n)$ represents the Galois fields (or finite field) with p^n elements. The Galois fields are classified by size. There is exactly one finite field up to isomorphism of size p^k for each prime p and positive integer k . Specifically, the Galois fields or the finite field has the following properties [44]: first, the Galois fields can be classified by the number of elements, so-called the order, which has the form p^n . Here, p is the characteristic of the field which is a prime number, and n is a positive integer. Second, for every prime number p and positive integer n , there exists a finite field with p^n elements. Third, any two finite fields with the same number of elements are isomorphic. That is, under some renaming of the elements of one of these, both its addition and multiplication tables become identical to the corresponding tables of the other one. For more proofs of the properties, readers can refer to Jacobson's book [44].

For example, the elements of $GF(2^2)$, which has characteristic two, can be represented as the four polynomials $0, 1, x, 1 + x$ over $GF(2)$, where computations are done modulo $x^2 + x + 1$, such as $x^3 = x \cdot x^2 = x \cdot (1 + x) = x + x^2 = 1$ and $x^2 = x \cdot x = (x^2 + 1) \cdot x = x^3 + x = 1 + x$ in $GF(2^2)$. Given the operation with componentwise addition modulo 2 over binary k -vectors, \mathbb{Z}_2^k is the group of all these binary k -vectors. If W_0 is used to denote the all-zeros vector of \mathbb{Z}_2^k , then for each $v \in \mathbb{Z}_2^k$, $v^2 = W_0$ is achieved.

In linear algebra, determinant and permanent are two fundamental concepts over square matrix which has many applications. Given an n -by- n matrix $A = (a_{i,j})$, the *determinant* of A is defined as

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{i=1}^n a_{i,\sigma(i)}$$

Here, the sum is computed over all permutations σ of the $\{1, 2, \dots, n\}$. For each permutation σ , $\operatorname{sgn}(\sigma)$ denotes the signature of σ . It is positive 1 for even σ and negative 1 for odd σ . Evenness or oddness of σ can be defined as follows: the permutation is even(odd) if the new sequence can be obtained by an even(odd) number of switches of numbers. The permanent is a function of the square matrix similar to the determinant. The *permanent* of an n -by- n matrix A is

$$\operatorname{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^n a_{i,\sigma(i)}$$

Over a ring of characteristic two, the well-known fact for a square matrix A is that the determinant is identical to the permanent.

$$\det(A) = \operatorname{perm}(A) \tag{1}$$

2.3 Schwartz-Zippel Lemma

The Schwartz-Zippel lemma is a well-known tool that is commonly used in probabilistic polynomial identity testing. It was developed independently by Schwartz and Zippel in around 1980 [67, 77]. And it is the theoretical basis of the techniques which will be presented in the following sections.

Lemma 1 ([65]) *Let $Q(x_1, \dots, x_n) \in \mathbb{F}[x_1, \dots, x_n]$ be a multivariate polynomial of total degree d . Fix any finite set $\mathbb{S} \subseteq \mathbb{F}$, and let r_1, \dots, r_n be chosen independently and uniformly at random from \mathbb{S} . Then*

$$\Pr[Q(r_1, \dots, r_n) = 0 | Q(x_1, \dots, x_n) \neq 0] \leq \frac{d}{|\mathbb{S}|} \quad (2)$$

Proof The idea derives from single variable case: a single variable polynomial of degree d has no more than d roots. It seems intuitively that a similar statement would hold for multivariate polynomials. The proof is by induction on the number of variables n . As was mentioned before, for base case $n = 1$, the probability that $Q(r_1) = 0$ is at most $\frac{d}{|\mathbb{S}|}$. Assume the lemma holds for a multivariate polynomial with up to $n - 1$ variables. Consider the polynomial $Q(x_1, \dots, x_n)$ by factoring out the variable x_1 :

$$Q(x_1, \dots, x_n) = \sum_{i=0}^k x_1^i Q_i(x_2, \dots, x_n)$$

where $k \leq d$ is the largest exponent of x_1 in Q . Since P is not identical to 0, the coefficient of $x_1^k Q_i(x_2, \dots, x_n)$ is not equal to zero by our choice. It is easy to see that the total degree of $Q_k(x_2, \dots, x_n)$ is at most $d - k$, by induction hypothesis:

$$\Pr[Q_k(r_2, \dots, r_n) = 0] \leq \frac{d - k}{|\mathbb{S}|}$$

Then consider the case that $Q_k(r_2, \dots, r_n) \neq 0$, $Q(x_1, \dots, x_n)$ becomes a univariate polynomial:

$$q(x_1) = Q(x_1, r_2, \dots, r_n) = \sum_{i=0}^k x_1^i Q_i(r_2, \dots, r_n)$$

The polynomial $q(x_1)$ has degree k , so the base case implies that

$$\Pr[Q(r_1, \dots, r_n) = 0 | Q_k(r_2, \dots, r_n) \neq 0] \leq \frac{k}{|\mathbb{S}|}$$

Using events A and B to denote $Q(r_1, \dots, r_n) = 0$ and $Q_k(r_2, \dots, r_n) = 0$, respectively, by probability theory,

$$\begin{aligned}
\Pr[A] &= \Pr[A \cap B] + \Pr[A \cap B^c] \\
&= \Pr[B]\Pr[A|B] + \Pr[B^c]\Pr[A|B^c] \\
&\leq \Pr[B] + \Pr[A|B^c] \\
&\leq \frac{d-k}{|\mathbb{S}|} + \frac{k}{|\mathbb{S}|} \\
&= \frac{d}{|\mathbb{S}|}
\end{aligned}$$

So the induction completes. \square

3 k-Path Problem

In graph theory, given a graph $G = (V, E)$, a *simple path* of G is a path that does not have any repeated vertices. The *longest path problem* is the problem of finding a simple path of maximum length in the given graph G . The decision version of the longest path problem is called the *k-path problem*. The graph G and an integer k are given as input; the aim of the k -path problem is to determine if graph G contains a simple path of length at least k .

Unlike the shortest path problem, even with negative edge weights, as long as the graph G does not have negative-weight cycles, the shortest paths between two vertices can be found in polynomial time. Finding the longest simple path between two vertices is NP-complete even if all edge weights are equal to 1. The NP-completeness of the decision problem can be easily proved by a reduction from the Hamiltonian path problem.

Theorem 2 *The k -path problem is NP-complete.*

Proof For a given graph $G = (V, E)$ and an integer k , a certificate P can be easily verified in polynomial time, so the k -path problem is in class NP.

It is known that Hamiltonian path is NP-complete [23]. The next is to prove Hamiltonian path \leq_p k -path, which shows that the k -path problem is NP-hard. Given a instance of Hamiltonian path $G_1 = (V_1, E_1)$, it can be transformed to a k -path instance by simple asking whether there exists a simple path of length $|V_1|$ in G_1 .

Clearly, if G_1 has a Hamiltonian path, since it traverses all possible vertices, this Hamiltonian path is the simple path of G_1 with length $|V_1|$. Conversely, if there is simple path of length $|V_1|$ in the graph G_1 , it is a path that visits all the vertices of G_1 without repetition, and this is a Hamiltonian path by definition. \square

The longest path problem can be transformed into the shortest path problem by utilizing the dual nature of optimizations. Given a longest path problem with input graph G , G can be converted into the shortest simple path problem on graph H which is same as the original graph G but with edge weights negated. It is

known that the shortest path problem can be solved in polynomial time for graph without negative-weight cycle. This approach cannot be used to solve H because any positive-weight cycle in G produces a negative-weight cycle in H . Thus, the shortest path problem on a graph with negative-weight cycle is also NP-complete. However, the longest path problem is polynomial-time solvable on acyclic graphs. Since the acyclic graph has no cycles, there is no negative-weight cycle on H . Thus, any shortest path algorithm can be used to solve the shortest problem on H .

3.1 A Brief History of the k -Path Problem

For a graph $G = (V, E)$ with the number of nodes $|V| = n$ and an integer k , the naive algorithm of the k -path problem enumerates all sequences of length $k + 1$ with time bound $O(n^k)$. When $k = n$, it has been known very early that the problem can be solved in $O^*(2^k)$ time using dynamic programming method [8, 38, 51]. In 1985, B. Monien first improved the time bound to $O^*(k!)$. Thus, the problem can be solved in polynomial time when $k \leq \log n / \log \log n$. After about 10 years, Papadimitriou and Yannakakis [66] conjectured that the $(\log n)$ -path problem can be solved in polynomial time. Almost in the meantime, Alon, Yuster, and Zwick [2] verified the conjecture. They gave a randomized algorithm running in $O^*((2e)^k) \leq O^*(5.44^k)$ time and a deterministic $O^*(c^k)$ time algorithm, where c is a large constant.

With the development of bioinformatics, the k -path problem has found applications both in biological subnetwork matchings [53] and in detecting signaling pathways in protein interaction networks [68]. Probably driven by the demand of applications, faster algorithms appeared after 2006. J. Kneis et al. [56] provided a $O^*(4^k)$ randomized algorithms and $O^*(c^k)$ deterministic algorithm with $c = 16$. The deterministic complexity was further reduced by Chen et al. [19], who presented more efficient color-coding schemes and gave an $O^*(12.8^k)$ time deterministic algorithm. The space complexity of the algorithm is $O(k \log k n + m)$ which is a remarkable improvement over the exponential $\ln k$ space complexity of the color-coding approach.

Very recently, a big step was taken by Koutis [61] and Williams [74]. Koutis gave a randomized algorithm for k -path that runs in $O^*(2^{3k/2}) \leq O^*(2.83^k)$ time. By improving Koutis's approach, Williams presented a $O^*(2^k)$ time randomized algorithm in 2008. They used very typical approaches which combine algebraization methods with randomization methods to obtain more efficient algorithms.

3.2 $O^*(2^k)$ Time Randomized Algorithm

For this section, the main techniques are from the works of Williams [74] and Koutis [61]. Our objective is to make their works more readable for the inexperienced. Also an introduction of this line of research will make readers have clear mentality. The general idea is to reduce the k -path problem to the

multilinear monomial testing for the sum-product expansion of a polynomial which is constructed by the input graph. Improving Koutis's result [61], Williams [74] developed an $O(2^k n^{O(1)})$ time randomized algorithm such that it outputs yes with high probability if there is a multilinear monomial and always outputs no if there is no multilinear monomial.

According to above idea, the first problem is how to transform the given graph G for the k -path problem into a polynomial. Assume graph G has vertex set v_1, v_2, \dots, v_n . Let A be the adjacency matrix of G , and let x_1, x_2, \dots, x_n be variables. Define a matrix $B[i, j] = A[i, j]x_i$. Let $\vec{1}^T$ be the row n -dimension vector of all 1's and \vec{x} be the column n -dimension vector defined by $\vec{x}[i] = x_i$. So the graph G can be transformed into the k -walk polynomial $P_k(x_1, \dots, x_k) = \vec{1}^T \cdot B^{k-1} \cdot \vec{x}$. The k -walk polynomial has a good property that it reflects all the walk of length k in its sum-product expansion. Then the following claim can be proved.

Define the polynomial

$$P_k(x_1, \dots, x_k) = \sum_{i_1, \dots, i_k \text{ is a walk in } G} x_{i_1} \cdots x_{i_k}$$

It is easy to see that the graph G has a k -path v_{i_1}, \dots, v_{i_k} iff $P_k(x_1, \dots, x_k)$ has a multilinear monomial of $x_{i_1} \cdots x_{i_k}$ of degree k in its sum-product expansion (see an example below). So the k -path problem for G can be reduced to the problem that detects whether $P_k(x_1, \dots, x_k)$ has multilinear monomial in its sum-product expansion. It is trivial to see whether $P_k(x_1, \dots, x_k)$ has a multilinear monomial when its sum-product expansion is already known. Unfortunately, the sum-product expression is essentially a troublesome thing which is infeasible to realize. That is because a polynomial may often have exponentially many monomials in its expansion.

Lemma 2 *There is a polynomial-time algorithm such that given an undirected graph G and a parameter k , it outputs a polynomial size circuit S for a multivariate polynomial $C(\cdot)$ such that G has a k -path if and only if the sum-product expansion of $C(\cdot)$ contains a multilinear monomial.*

Proof For each vertex $v_i \in V$, define a polynomial $p_{k,i}$ as follows:

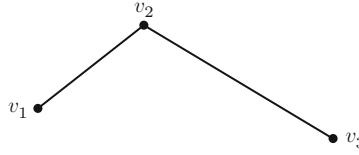
$$\begin{aligned} p_{1,i} &= x_i, \\ p_{k+1,i} &= x_i \left(\sum_{(v_i, v_j) \in E} p_{k,j} \right), \quad k > 1. \end{aligned}$$

A polynomial for G is defined as

$$p(G, k) = \sum_{i=1}^n p_{k,i}.$$

Obviously, $p(G, k)$ can be represented by an arithmetic circuit. It is easy to see that the graph G has a k -path $v_{i_1} \cdots v_{i_k}$ iff $p(G, k)$ has a monomial of $x_{i_1} \cdots x_{i_k}$ in its sum-product expansion. \square

Example 1 Consider the following simple graph G_1 with three nodes:



The adjacency matrix A_1 of G_1 and the matrix $B_1[i, j] = A_1[i, j]x_i$ are given below:

$$A_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad B_1 = \begin{pmatrix} 0 & x_1 & 0 \\ x_2 & 0 & x_2 \\ 0 & x_3 & 0 \end{pmatrix}$$

Then the 3-walk polynomial $P_3(x_1, x_2, x_3) = \vec{1}^T \cdot B_1^2 \cdot \vec{x} = 2x_1x_2x_3 + x_1^2x_2 + x_1x_2^2 + x_2^2x_3 + x_2x_3^2$. Clearly, $x_1x_2x_3$ is a multilinear monomial in $P_3(x_1, x_2, x_3)$, so there is a corresponding 3-path $v_1 \rightarrow v_2 \rightarrow v_3$ in G_1 . It is easy to see that P_k can be represented by a circuit of size $O(k(m+n))$ where m is the number of edges in G . Let $F = GF(2^{3+\log k})$ and W_0 be the all-zeros vector of \mathbb{Z}_2^k . Williams's randomized algorithm is introduced as follows:

Algorithm 1 Williams's Algorithm

Input: a graph G with vertex set v_1, v_2, \dots, v_n and a parameter k

Output: If G does not have a k -path, the algorithm always outputs *false*; if G has a k -path, the algorithm outputs *true* with probability no less than $1/5$.

Steps:

- 1: Calculate P_k as described above.
- 2: Implement P_k with a circuit C .
- 3: Pick n uniform random vectors z_1, z_2, \dots, z_n from \mathbb{Z}_2^k .
- 4: For each multiplication gate g_i in C , pick a uniform random $w_i \in F \setminus 0$.
Insert a new gate that multiplies the output of g_i with w_i and feeds the output to those gates that read the output of g_i .
- 5: Let C' be the new arithmetic circuit and P' be the new polynomial represented by C' .
Output true iff $P'(W_0 + z_1, \dots, W_0 + z_n) \neq 0$.

End of Algorithm

The basic idea of Williams's algorithm is to substitute variables of P_k with random group algebra elements of \mathbb{Z}_2^k such that in the sum-product expansion of P_k , all the non-multilinear monomials in P_k become zero and some multilinear monomials survive. And in order to avoid making the coefficients of all multilinear monomials zero, the original scalar-free multiplication circuit is augmented by adding random scalar multiplications over a field large enough that the remaining multilinear monomial evaluates to nonzero with decent probability. Let $s(n)$ be the size of the arithmetic circuit used to represent P_k . Then [Theorem 3](#) gives theoretical support for Williams's algorithm.

In above algorithm, readers need to pay attention to two points. First, all the additions and multiplications are over a ring $F(\mathbb{Z}_2^k)$. Elements of $F(\mathbb{Z}_2^k)$ have the form $\sum_{g \in \mathbb{Z}_2^k} a_g g$, where each $a_g \in F$. The addition of $F(\mathbb{Z}_2^k)$ is defined in a point-wise manner:

$$\left(\sum_{g \in \mathbb{Z}_2^k} a_g g \right) + \left(\sum_{g \in \mathbb{Z}_2^k} b_g g \right) = \sum_{g \in \mathbb{Z}_2^k} (a_g + b_g) g$$

The multiplication of two vectors $(a_1, \dots, a_k)^T$ and $(b_1, \dots, b_k)^T$ in $F(\mathbb{Z}_2^k)$ is equal to $(c_1, \dots, c_k)^T$ with $c_i = a_i + b_i \pmod{2}$ for $i = 1, 2, \dots, k$. The multiplication has the form of a convolution:

$$\left(\sum_{g \in \mathbb{Z}_2^k} a_g g \right) \cdot \left(\sum_{g \in \mathbb{Z}_2^k} b_g g \right) = \sum_{g, v \in \mathbb{Z}_2^k} (a_g a_v)(gv) = \sum_{g \in \mathbb{Z}_2^k} \left(\sum_{h \in \mathbb{Z}_2^k} a_h b_{hg} \right) g$$

Example 2 Let $u_1, u_2 \in GF(2^2)[\mathbb{Z}_2^3]$, where

$$u_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + x \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad u_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Then

$$\begin{aligned} u_1 + u_2 &= \left[\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + x \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right] + \left[\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right] \\ &= (1+x) \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \end{aligned}$$

$$\begin{aligned}
u_1 \cdot u_2 &= \left[\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + x \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \right] \cdot \left[\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right] \\
&= (1+x) \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + x \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + (1+x) \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}
\end{aligned}$$

Second point is that in the evaluation of the k -path polynomial P_k , the algorithm corresponds to picking random $y_{i,j,c}$ in F for $c = 1, \dots, k-1$, $i, j = 1, \dots, n$, letting $B_c[i, j] = y_{i,j,c} B[i, j]$, then evaluating $P'_k(x_1, \dots, x_n) = \vec{1}^T \cdot B_{k-1} \cdots B_1 \cdot \vec{x}$ on the appropriate vectors.

Before proving the correctness of Williams's algorithm, a lemma which was proved by Koutis [61] and then further developed by Williams [74] over any field of characteristic two needs to be shown first.

Lemma 3 *If $z_1, z_2, \dots, z_i \in \mathbb{Z}_2^k$ are linearly dependent over $GF(2)$, then $\prod_{j=1}^i (W_0 + z_j) = 0$ in $F[\mathbb{Z}_2^k]$. Otherwise, if $z_1, z_2, \dots, z_i \in \mathbb{Z}_2^k$ are linearly independent over $GF(2)$, then $\prod_{j=1}^i (W_0 + z_j) = \sum_{z \in \mathbb{Z}_2^k} z$.*

Proof If z_1, z_2, \dots, z_i are linearly dependent, there is a nonempty subset T of the vectors that sum to the all-zeros vector. In $F[\mathbb{Z}_2^k]$, this is equivalent to

$$\prod_{j \in T} z_j = W_0.$$

Let $S \subseteq T$ be arbitrary. By multiplying both sides by $\prod_{j \in (S \Delta T)} z_j$,

$$\prod_{j \in S} z_j = \prod_{j \in (S \Delta T)} z_j.$$

Thus, $\prod_{j \in T} (W_0 + z_j) = \sum_{S \subseteq T} (\prod_{j \in S} z_j) = 0 \bmod 2$, since each product appears twice in the sum. Since F is characteristic two, $\prod_{j=1}^i (W_0 + z_j) = 0$ over $F[\mathbb{Z}_2^k]$.

Therefore, linearly dependent vectors lead to a cancellation of monomials. On the other hand, when z_1, z_2, \dots, z_i are linearly independent, $\prod_{j=1}^i (W_0 + z_j)$ is just the sum over all vectors in the span of z_1, z_2, \dots, z_i , since each vector in the span is of the form $\prod_{j \in S} z_j$ for some $S \subseteq [i]$, and there is a unique way to generate each vector in the span. \square

Lemma 4 *Let v_1, \dots, v_n be n random vectors in \mathbb{Z}_2 . Then with probability at least $\frac{1}{4}$, they are linearly independent.*

Proof. For the first m vectors v_1, \dots, v_k , there are 2^k vectors that can be expressed as $\sum_{i=1}^k a_i v_k$, where $a_i \in Z_2$ for $i = 1, \dots, m$. Thus, for a random n -dimensional vector v_{m+1} from Z_2 , with probability $\frac{2^k}{2^n}$, v_{m+1} cannot be expressed as $\sum_{i=1}^k a_i v_k$ for some $a_i \in Z_2$ for $i = 1, \dots, m$. The probability that v_1, \dots, v_n linearly independent is

$$\left(1 - \frac{1}{2^n}\right) \left(1 - \frac{2}{2^n}\right) \cdots \left(1 - \frac{2^{n-1}}{2^n}\right) \geq \frac{1}{2} \prod_{k=2}^n \left(1 - \frac{1}{k^2}\right) \quad (3)$$

$$= \frac{1}{4}. \quad (4)$$

□

Example 3 Let $z_1, z_2, z_3, z_4 \in \mathbb{Z}_2^3$ be four vectors:

$$z_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad z_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \quad z_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad z_4 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$$

Since $z_1 \cdot z_2 \cdot z_4 = W_0$, z_1, z_2, z_3, z_4 are linearly dependent. To coincide with above proof, let $T = \{z_1, z_2, z_3\}$. Then $\prod_{j \in T} (W_0 + z_j) = (W_0 + z_1) \cdot (W_0 + z_2) \cdot (W_0 + z_4) = W_0 + z_1 + z_2 + z_4 + z_1 \cdot z_2 + z_1 \cdot z_4 + z_2 \cdot z_4 + z_1 \cdot z_2 \cdot z_4 = 0$. The above equation equals to zero because $z_1 \cdot z_2 \cdot z_4 = W_0$, $z_1 = z_2 \cdot z_4$, $z_2 = z_1 \cdot z_4$, $z_4 = z_1 \cdot z_2$ in the sum-product and F is characteristic two.

On the other hand, it is easy to see that z_1, z_2, z_3 are linearly independent. So it can be easily verified that $\prod_{j=1}^3 (W_0 + z_j)$ is the sum over all vectors in the span of z_1, z_2, z_3 .

Theorem 3 ([74]) Given a graph G with vertex set v_1, v_2, \dots, v_n and a parameter $k \leq n$, Williams's algorithm is a randomized $O^*(2^k n^{O(1)})$ time algorithm such that if there is a k -path in G , it outputs true with probability no less than $1/5$ and always outputs false if there is no k -path in G .

Proof By the observation of Koutis [61] that for any $z_i \in \mathbb{Z}_2^k$,

$$(W_0 + z_i)^2 = W_0^2 + 2v_i + v_i^2 = W_0 + 0 + W_0 = 0 \pmod{2}.$$

Thus, since F has characteristic two, all squares in P equal to 0 in $P'(W_0 + z_1, \dots, W_0 + z_n)$. It illustrates that if $P(x_1, \dots, x_n)$ does not have a multilinear monomial, then for any choice of v_i , $P'(W_0 + z_1, \dots, W_0 + z_n) = 0$ over $F[\mathbb{Z}_2^k]$.

Without loss of generality, assume that every multilinear monomial of P has degree at least k , and at least one multilinear monomial has degree exactly k . If not, let $k' < k$ be the minimum degree of a multilinear monomial in P . By trying

all $j = 1, \dots, k$ and multiplying the final output of the circuit for P by j new variables x_{n+1}, \dots, x_{n+j} , a polynomial P^j is obtained to be fed to the randomized algorithm. Note that when $j = k - k'$, our assumption holds.

By above assumption, every multilinear monomial in the sum-product expansion of P has the form $c \cdot x_{i_1} \cdots x_{i_{k''}}$, where $k'' \geq k$ and $c \in \mathbb{Z}$. For each such monomial, there is a corresponding collection of multilinear monomial P' , each of the form $w_1 \cdots w_{k'-1} \prod_{j=1}^{k'} (W_0 + z_{i_j})$, where the sequence $w_1, \dots, w_{k'-1}$ is distinct for every monomial in the collection (as the sequences of multiplication gates $g_1, \dots, g_{k'-1}$ are distinct). Since there are no scalar multiplication in the arithmetic circuit, these monomials do not have leading coefficients. Then it can be proved that if the sum-product expansion of $P(x_1, \dots, x_n)$ has a multilinear monomial, for random choices of w_i 's and z_i 's, $P'(W_0 + z_1, \dots, W_0 + z_n) \neq 0$ with probability at least $1/5$.

Since $k'' > k$ vectors are linearly dependent, by Lemma 3, $P'(W_0 + z_1, \dots, W_0 + z_n)$ equals to either 0 or $c \sum_{z \in \mathbb{Z}_2^k}$ for some $c \in F$. The vectors z_{l_1}, \dots, z_{l_k} chosen for the variables in a multilinear monomial P are linearly independent with probability at least $1/4$, because the probability that a random $k \times k$ matrix over $GF(2)$ has full rank is at least $1/4$ according to Lemma 4.

If S is used to represent the set of those multilinear monomials in P which correspond to k linearly independent vectors in $P'(W_0 + z_1, \dots, W_0 + z_n)$, then for some $c_i \in F$, the coefficient $c = \sum_i c_i \in F$ mentioned in the last paragraph corresponds to the i th multilinear monomial in S . Here, each coefficient c_i comes from a sum of products of $k - 1$ elements $w_{i,1}, \dots, w_{i,k-1}$ corresponding to some multiplication gates $g_{i,1}, \dots, g_{i,k-1}$. Note that in the k -path case, each c_i is a sum of products of the form $w_{i,1}w_{i,2} \cdots w_{i,k-1}$. Imagining the w_i 's as variables, the sum $Q = \sum_i c_i$ is a *degree-k polynomial* over F . Assuming $S \neq \emptyset$, since each monomial in Q has coefficient 1, Q is not identical to zero. Then by Schwartz-Zippel Lemma 1, randomly assigning values to the variables of Q results in an evaluation of 0 $\in F$ with probability at most $k/|F| = 1/2^3$. Since $S \neq \emptyset$ with probability at least $1/4$, and the coefficient $c \neq 0$ with probability at least $1 - 1/2^3 = 7/8$, the overall probability of success is at least $1/4 \cdot 7/8 > 1/5$.

In the remaining paragraph, the runtime of the algorithm is discussed. However, since all the additions and multiplications takes place over $F[\mathbb{Z}_2^k]$, it is necessary to account for the cost of each arithmetic.

Consider the sum-product expansion of $P'(W_0 + z_1, \dots, W_0 + z_n)$. It will be converted into $\sum C_i(\cdot)v_i$, where each v_i is a k -dimensional vector over Z_2 , and $C_i(\cdot)$ is a circuit for a multivariate polynomial. According to the construction of in Lemma 2, each multiplication gate is for $(m_i v_j)(\sum C_i(\cdot)v_i)$, where m_i is a monomial. It takes $O(2^k)$ steps to convert it into the format $\sum C'_i(\cdot)v_i$.

Assume that the final evaluation has the format $\sum C_i^*(\cdot)v_i$. The size of each $C_i^*(\cdot)$ can be maintained in the size $O(n^{O(1)})$ as the original size of circuit $P(\cdot)$ is $O(n^{O(1)})$. The existing algorithm needs to be used for the polynomial identity, which takes $O(n^{O(1)})$ time for each $C_i^*(\cdot)$, to test if each $C_i(\cdot)^*$ is zero by Lemma 1. Therefore, the total time is $O(2^k n^{O(1)})$. \square

4 Hamiltonian Path Problem

In graph theory, a *Hamiltonian path* is a path in an undirected graph that visits each vertex exactly once. A *Hamiltonian cycle* is a cycle in an undirected cycle which visits each vertex exactly once and also returns to the starting vertex. For example, every cycle graph and complete graph with more than two vertices is Hamiltonian. Hamiltonian paths and cycles are named after William Rowan Hamilton who invented the Icosian game, which involves finding a Hamiltonian cycle in the edge graph of the dodecahedron.

The Hamiltonian path problem and the Hamiltonian cycle problem are two classic problems of determining whether there exists a Hamiltonian path and a Hamiltonian cycle, respectively, for a given directed or undirected graph. For a given graph G , the Hamiltonian path problem is equivalent to the Hamiltonian cycle problem by adding a new vertex and connecting it to all the vertices of G . Both problems are in Karp's original list [51] of NP-complete problems. For the specific proof of NP-completeness of the problems, please refer to standard algorithm books (e.g., [23]). Actually, the NP-hardness of Hamiltonian path problem has been used in the proof of [Theorem 2](#). Through setting the distance between two cities to a finite constant if they are adjacent and infinity otherwise, the Hamiltonian path problem is a special case of the *traveling salesman problem* (TSP). The TSP was first formulated as a mathematical problem in 1930s. It is one of the most intensively studied problems in computational mathematics. Given a list of cities and their pairwise distances, the TSP asks for a shortest possible tour that visits each city exactly once. Even though the problem is computational difficult, a large amount of heuristics and exact methods are given to solve different instances of it [3]. Furthermore, the TSP has many applications in planning, logistics, and the manufacture of microchips. It is also equivalent to some bioinformatics problems such as DNA sequencing after slightly modified.

4.1 A Brief History of TSP and Hamiltonian Path Problem

The TSP was defined formally in the 1800s and was first studied by K. Menger and other mathematicians during the 1930s. The naive solution of the TSP would be try all permutations and see which one is cheapest using brute force search. The running time for this approach is in $O(n!)$, the factorial of the number of cities. So this approach becomes impractical even for only 20 cities. In almost 200 years from 1930s, plenty of algorithms including heuristics, approximations, and exact algorithms have been developed. Since the topic mainly focus on the Hamiltonian path detection problem in this section, we only introduce the exact algorithms for brevity. Later in the 1950s and 1960s, the TSP was reduced to an integer linear program by G. Dantzig et al., and the cutting plane method was developed to solve it. In the same period, Bellman [7, 8] and Karp [38] provided the dynamic programming algorithm for the TSP which runs in $O(n^2 2^n)$ time. Given a complete

graph with n vertices and edge weights $l : E \rightarrow R^+$, the algorithm is based on defining $w_{s,t}(X)$ for $s, t \in X \subseteq V$ as the weight of the lightest path from s to t in the induced graph $G[X]$ visiting all vertices in X exactly once. Here, $w_{s,t}(X)$ obeys the following simple recursive equation:

$$w_{s,t}(X) = \begin{cases} \min_{u \in X \setminus \{s,t\}} w_{s,u}(X \setminus \{t\}) + l(ut) & |X| > 2 \\ l(st) & |X| \leq 2 \end{cases}$$

The dynamic programming solution requires exponential space. Using inclusion-exclusion, the problem can be solved in time within a polynomial factor of 2^n and polynomial space [5, 52, 57]. Then great progress was made by Grotschel et al., which uses cutting planes and branch and bound to solve instances with up to 2,392 cities. After that, Applegate, Bixby, Chvatal, and Cook developed the program Concorde that has been used in many recent record solutions. Except for the improvements for the general graph, there are some commendable results for restricted graph classes. Broersma et al. [13] described how to solve the Hamiltonian path problem for claw-free graphs in $O^*(1.682^n)$ time. Iwama and Nakashima [43] showed that the TSP in cubic graphs admits an $O^*(1.251^n)$ time algorithm by improving slightly on Eppstein [28]. In graphs of maximum degree 4, Gebauer [35] provided an algorithm to count the Hamiltonian cycles in $O^*(1.715^n)$ time. Björklund et al. [12] presented that the traveling salesman problem in bounded degree graphs can be solved in time $O((2 - \epsilon)^n)$, where $\epsilon > 0$ depends only on the degree bound but not on the number of cities, n .

In 2005, by transforming a TSP instance into an instance of microchip layout problem, Cook and others computed an optimal tour for an instance which contains 33,810 cities. Though for the instances with tens of thousands of cities, the TSP can be solved. There is still no algorithm which breaks the time bound $O(2^k)$ for the general graph. So Woeginger [75] asked in open problem that whether there exists an exact algorithm which runs in time $O^*(c^n)$ for the TSP and the Hamiltonian path problem for some $c < 2$. After 7 years, this problem has been solved very recently by Björklund [9], who improved the bounds to $O(1.414^n)$ and $O(1.657^n)$ for the bipartite graph and the general graph, respectively.

4.2 Overall Idea of Björklund's Approach

About 50 years ago, the $O^*(2^n)$ time bound for detecting Hamiltonian path problem has been established. No exact algorithm which was developed to beat the bound until half a century later. Björklund [9] presented a Monte Carlo algorithm for n -vertex undirected graph running in $O^*(1.657^n)$ time, which is a breakthrough for this fundamental and classic problem. This crucial contribution was inspired by the algebraic sieving method for k -path problem [61, 74] which is introduced in Sect. 3.

The basic idea of Björklund's approach is similar to the inclusion-exclusion algorithm which can be summarized as first counting all closed n -walks through

s and then canceling out crossing n -walks. However, comparing with inclusion-exclusion algorithm, Björklund's approach has some limitations. It only works for undirected graph, it is randomized, and also it cannot count the number of solutions. The same as Williams's algorithm [74] for the k -path, Björklund's approach evaluates multivariate polynomials over fields of characteristic two. The difference between them is that instead of sieving the k -path among all the walks, Björklund's approach sieves the Hamiltonian cycles among all the cycle covers. A cycle cover in a directed n -vertex graph is a set of n arcs such that every vertex is the origin of one arc and the end of another. The arcs together describe disjoint cycles covering all vertices of the graph. In particular, the cycle covers contain the Hamiltonian cycle.

The main contribution of Björklund's work is that it not only further develops Koutis-William's ideas but also introduces determinants to count weighted cycle covers to extend the ideas of using determinants to count perfect matchings [10]. However, unlike Koutis-William who construct small arithmetic circuits, Björklund depends on the efficient algorithms for computing a matrix determinant numerically. In the following paragraphs of this section, a concrete introduction about Björklund's approach is given, which further shows the tremendous ability of algebrization and randomization for the algorithm design.

4.3 Notations and Definitions

This section first introduces some particular notations and definitions for Björklund's algorithm. Efforts were made to explain the details of the methods more clearly; however, for letting readers simply refer to the original papers, most of the notations are consistent with Björklund's. A cycle cover can be formally defined as follows.

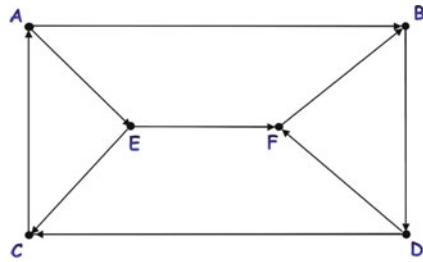
Definition 3 For a direct graph $D = (V, E)$, a *cycle cover* is a subset $C \subseteq E$ such that for every vertex $v \in V$, there is exactly one arc $a_{v_1} \in C$ starting in v and exactly one arc $a_{v_2} \in C$ ending in v . Let $cc(D)$ be the family of all cycle covers of D and let $hc(D)$ be the set of Hamiltonian cycle covers.

It is easy to see that $hc(D) \subseteq cc(D)$. Because among all the cycle covers of G , the Hamiltonian cycle covers are some particular cycle covers which consist of only one big cycle passing through all vertices of D . Let $cc(D) \setminus hc(D)$ be set of *non-Hamiltonian cycle covers*, i.e., $cc(D)/hc(D)$ contains all the cycle covers which have more than one small cycles. A Hamiltonian cycle cover of a direct graph can be simply written as a vertex order $(v_0, v_1, \dots, v_{n-1})$ (cf. Fig. 1).

Given a function $f : X \rightarrow Y$, f is *surjective* (or onto) if each element of Y can be obtained by applying f to some element of X . Let $f^{-1} : Y \rightarrow 2^X$ be the preimage of f , then $f^{-1}(b) = \{a \in X : f(a) = b\}$.

Then an important definition for *labeled cycle cover sum* will be given next. The name stems from the fact that every arc in cycle cover is labeled by a nonempty

Fig. 1 The direct graph with a non-Hamiltonian cycle cover $[(A, E, C), (B, D, F)]$ and a Hamiltonian cycle (A, E, F, B, D, C)



subset of a set of labels. It is important because the Hamiltonian path detection problem can be reduced to labeled cycle cover sum calculating problem.

Definition 4 Suppose $D = (V, E)$ is a directed graph, L is a label set, and f is a function on a ring R which takes an edge a in G and the a subset of labels in L as inputs. The *labeled cycle cover sum* for D is defined as

$$\Lambda(D, L, f) = \sum_{C \in cc(D)} \sum_{g: L \twoheadrightarrow C} \prod_{a \in C} f(a, g^{-1}(a)) \quad (5)$$

In above definition, all the cycles covers are summarized in the outer sum, and the inner sum is over all surjective function g from the label set L to the cycle cover set C with the two head arrow \twoheadrightarrow representing a surjective function.

4.4 Cycle Cover Cancelation in Characteristic Two

In Sect. 4.2, it has been mentioned that the basic idea is first counting too much and then canceling out every false contribution. Following this idea, since the cycle covers of the graph contains all the Hamiltonian paths, the objective is to cancel out all the non-Hamiltonian paths and hence keep the Hamiltonian cycle covers. Thus, this section first describes how the non-Hamiltonian cycle covers are canceled out in the labeled cycle cover sum over the ring R with characteristic two and then discusses how to keep the Hamiltonian cycle covers.

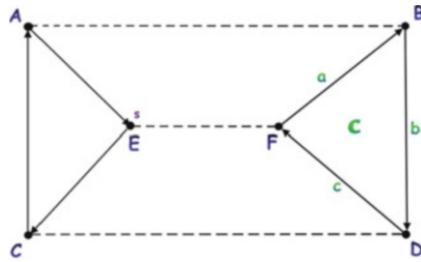
Before presenting the lemma, two definitions are necessary.

Definition 5

- A direct graph $G = (V, E)$ is *bidirected* if for every arc $uv \in E$, it has an arc $vu \in E$ in the opposite direction.
- For an arbitrarily chosen special vertex s , $f: A \times 2^L \setminus \{\emptyset\} \rightarrow R$ is an *s-oriented mirror function* if $f(uv, Z) = f(vu, Z)$ for all Z and all $u \neq s$ and $v \neq s$.

The next lemma proves that how the non-Hamiltonian cycle covers vanish, which also imply that the nonexistence of false positives in Björklund's randomized algorithm for detecting the Hamiltonian path.

Fig. 2 Considering Fig. 1 as a bidirected graph, this figure shows a cycle cover $[(AEC), (BDF)]$ which contains a labeled cycle $\mathbf{C} = (BDF)$ with the arcs along the cycle \mathbf{C} is to pair with the cycle cover in Fig. 3



Lemma 5 Given a bidirected graph $D = (V, E)$, a finite set L , and special vertex $s \in V$, let f be an s -oriented mirror function over a ring R of characteristic two. Then

$$\Lambda(D, L, f) = \sum_{H \in hc(D)} \sum_{g: L \rightarrow H} \prod_{a \in H} f(a, g^{-1}(a)) \quad (6)$$

Proof To cancel out all the labeled non-Hamiltonian cycle covers in the labeled cycle cover sum, a mapping M is created to partition all the non-Hamiltonian cycle covers into dual pairs such that both cycle covers in each pair contribute the same term to the sum. Since the ring R is of characteristic two, all these terms cancel.

Let $C \in cc(D)$ and \mathbf{C} be the first cycle of C not passing through s . Since all the non-Hamiltonian cycle cover consists of at least two cycles and all cycles are vertex disjoint, there must exist one such \mathbf{C} . There are two cases. In the general case, consider any fixed order of the cycles. Let $C' = C$ except for the cycle \mathbf{C} which is reversed in C' , i.e., every arc $uv \in \mathbf{C}$ is replaced by the arc in the opposite direction vu in C' . Since G is bidirected, this arc must exist. In the special case when \mathbf{C} contains only two arcs, C' is identical to C . The function g'^{-1} is identical to g^{-1} on $C \setminus \mathbf{C}$ and is defined by $g'^{-1}(uv) = g^{-1}(vu)$ for all arcs $uv \in \mathbf{C}$. Then the mapping is constructed by $M(C, g) = (C', g')$. And clearly, $(C, g) \neq M(C, g)$ and $(C, g) = M(M(C, g))$. Therefore, the mapping M uniquely pairs up the labeled non-Hamiltonian cycle covers. Figures 2 and 3 give an example.

From the known conditions of the lemma, f is an s -oriented mirror function. Thus, $f(uv, Z) = f(vu, Z)$ for all arcs uv not incident to s and all $Z \in 2^L \setminus \emptyset$. Also since F is a ring of characteristic two and (C, g) and $M(C, g)$ have same labels, the equality $\prod_{a \in C} f(a, g^{-1}(a)) = \prod_{a \in C'} f(a, g'^{-1}(a))$ satisfies. Therefore, the addition of the above tow products from (C, g) and $M(C, g)$ equals to 0. \square

The above lemma shows how the non-Hamiltonian cycle covers cancel out in labeled cycle cover sum if the function f meets some requirements. However, in order to detect the Hamiltonian cycle covers, at least one Hamiltonian cycle cover reserves is still needed. The technique used here is very similar to the

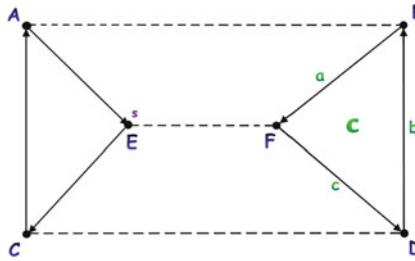


Fig. 3 This Figure is achieved by applying mapping M to Fig. 2. It has a cycle cover $[(AEC), (BFD)]$ which contains a labeled cycle $C = (BFD)$. The cycle (BDF) in Fig. 2 and the cycle (BFD) in this figure keep the same labeling and hence pair up to cancel out the label cycle cover sum

one used in the k -path problem. The labeled cycle cover sum are transformed into a multivariate polynomial such that it would have monomial with nonzero coefficient if Hamiltonian cycle exists. Moreover, it would not have monomials with nonzero coefficient resulting from non-Hamiltonian cycle covers, as a consequence of Lemma 5. Then Freivalds's fingerprint technique is employed to detect if the multivariate polynomial is identical to zero or not. If it is zero, the original direct graph does not contain Hamiltonian cycles, whereas if it is not zero, the original direct graph contains at least one Hamiltonian cycle.

A careful reader could discover that for the same Hamiltonian cycle in a bidirected graph, it has two opposite directions. If the two directions of the Hamiltonian cycle contribute the same terms to the sum, they would vanish in the labeled cycle cover sum. That is why the s -oriented mirror function is defined early in this section. In particular for the special vertex s and any $su, us \in E$, the above transformation ensures that $f(su, X)$ and $f(us, X)$ will not share variables. The details of the transformation will be discussed in Sect. 4.6.

4.5 Determinants and Inclusion-Exclusion

In this section, the readers would know how to compute labeled cycle cover sum efficiently. This technique can be found in Björklund's recent work [10]. In Sect. 2.2, it has been mentioned that for a matrix A over a ring of characteristic two, the result $\det(A) = \text{perm}(A)$ holds. Moreover, given a weighted direct graph $D = (V, E)$, let the weights of the edges be defined by a function $w : A \rightarrow R$ and define a $|V| \times |V|$ matrix with rows and columns representing the vertices V

$$\mathbf{A}_{i,j} = \begin{cases} w(ij) & : ij \in E \\ 0 & : \text{otherwise,} \end{cases}$$

then

$$\text{perm}(A) = \sum_{C \in cc(D)} \prod_{a \in C} w(a). \quad (7)$$

The above Eq. (7) holds because of two aspects. First, a cycle cover of the direct graph D is a collection of vertex-disjoint directed cycles that covers all nodes of D . Hence each vertex i in D has a unique successor $\sigma(i)$ in the cycle cover and σ is a permutation on $\{1, 2, \dots, n\}$ where n is the number of vertices in D . Then every cycle cover of D corresponds to a permutation σ on $\{1, 2, \dots, n\}$. Thus, according to the definition of permanent in Sect. 2.2, $\text{perm}(A) \geq \sum_{C \in cc(D)} \prod_{a \in C} w(a)$. On the other hand, any permutation σ on $\{1, 2, \dots, n\}$ corresponds to a possible cycle cover in which there is an edge from vertex i to vertex $\sigma(i)$ for each i in G . If there exists an edge created by the permutation σ which is not a right edge of D , the cycle cover created by the permutation σ is not a right cycle cover of D . Thus, the weight function w will make the contribution from this permutation zero. Thus, $\text{perm}(A) \leq \sum_{C \in cc(D)} \prod_{a \in C} w(a)$.

The remaining paragraphs describe how to compute labeled cycle cover sum through a sum of an exponential number of determinants. To this end the matrices $\mathbf{M}_f(Z)_{i,j}$ are defined as for every $Z \subseteq L$

$$\mathbf{M}_f(Z)_{i,j} = \begin{cases} f(ij, Z) & : ij \in E, Z \neq \emptyset \\ 0 & : \text{otherwise.} \end{cases} \quad (8)$$

Then the following polynomial $p(f, r)$ is introduced to compute an associated labeled cycle cover sum in characteristic two:

$$p(f, r) = \sum_{Y \subseteq L} \det \left(\sum_{Z \subseteq Y} r^{|Z|} \mathbf{M}_f(Z) \right) \quad (9)$$

It is easy to see that $p(f, r)$ consists of a sum of an exponential number of determinants in which r is indeterminate whose aim is to control the total rank of the subsets used as labels in the labeled cycle covers. The following lemma shows that the labeled cycle cover sum is identical to the coefficient of the monomial $r^{|L|}$ in $p(f, r)$. For a polynomial $q(r)$ in an indeterminate r , $[r^l]q(r)$ is the coefficient of monomial r^l in $q(r)$.

Lemma 6 *For a directed graph D , a set L of labels, and any $f : A \times 2^L \setminus \emptyset \rightarrow GF(2^k)$,*

$$[r^{|L|}]p(f, r) = \Lambda(D, L, f). \quad (10)$$

Proof Since the determinant is equivalent to the permanent in rings of characteristic two (Eq. 1) and the relationship between permanent and cycle covers,

$$\begin{aligned}
p(f, r) &= \sum_{Y \subseteq L} \text{perm} \left(\sum_{Z \subseteq Y} r^{|Z|} \mathbf{M}_f(Z) \right) \\
&= \sum_{Y \subseteq L} \sum_{C \in cc(D)} \prod_{a \in C} \left(\sum_{Z \subseteq Y} r^{|Z|} f(a, Z) \right).
\end{aligned}$$

After calculating the inner product,

$$p(f, r) = \sum_{Y \subseteq L} \sum_{C \in cc(D)} \sum_{q: C \rightarrow 2^Y \setminus \{\emptyset\}} \prod_{a \in C} r^{|q(a)|} f(a, q(a)).$$

Changing the order of summation,

$$p(f, r) = \sum_{C \in cc(D)} \sum_{q: C \rightarrow 2^L \setminus \{\emptyset\}} \sum_{\substack{\bigcup_{a \in C} q(a) \subseteq Y \\ Y \subseteq L}} \left(\prod_{a \in C} r^{|q(a)|} f(a, q(a)) \right). \quad (11)$$

Claim 1 In Eq.(11), all the terms from the functions $q : C \rightarrow 2^L \setminus \{\emptyset\}$ satisfied $\bigcup_{a \in C} q(a) \subset L$ cancel in rings of characteristic two.

From $\bigcup_{a \in C} q(a) \subset L$, the union of $q(a)$ over all the elements of C does not cover all of L , i.e., $L \setminus \bigcup_{a \in C} q(a) \neq \emptyset$. Then for each subset $T \subseteq L \setminus \bigcup_{a \in C} q(a)$, there is a term derived from T (let $Y = L \setminus T$) in the innermost summation. It is easy to see that for all the subsets of $L \setminus \bigcup_{a \in C} q(a)$, the terms derived from them are equal. And in total there are $2^{|L \setminus \bigcup_{a \in C} q(a)|}$ equal terms. Since the ring characteristic is two, these terms cancel.

Because of Claim 1, in Eq.(11),

$$p(f, r) = \sum_{C \in cc(D)} \sum_{\substack{q: C \rightarrow 2^L \setminus \{\emptyset\} \\ \bigcup_{a \in C} q(a) = L}} r^{\sum_{a \in C} |q(a)|} \left(\prod_{a \in C} f(a, q(a)) \right). \quad (12)$$

Since $\bigcup_{a \in C} q(a) = L$ and $\sum_{a \in C} |q(a)| = |L|$ implies $\forall a \neq b : q(a) \cap q(b) = \emptyset$, the coefficient of $r^{|L|}$ is

$$[r^{|L|}]p(f, r) = \sum_{C \in cc(D)} \sum_{\substack{q: C \rightarrow 2^L \setminus \{\emptyset\} \\ \bigcup_{a \in C} q(a) = L \\ \forall a \neq b : q(a) \cap q(b) = \emptyset}} \left(\prod_{a \in C} f(a, q(a)) \right).$$

Therefore, after inverting the function q , $[r^{|L|}]p(f, r)$ identifies the labeled cycle cover sum in Definition 4. \square

Based on the above lemma, the labeled cycle cover sum can be computed by calculating the coefficient of a polynomial. This is a typical algebrization which transforms a graph theoretical problem into a algebraic problem. The algebrization enables a relatively efficient algorithm for computing the labeled cycle cover sum. Before presenting the details of the algorithm, some definitions and a Lemma are given to compute the labeled cycle cover sum. These results are developed by Björklund et al. in [11]. For more information, please refer to their paper [11] and Yates' fast zeta transform [76].

Definition 6 Let N be a set of n elements with $n \geq 1$ and let us assume that $N = \{1, 2, \dots, n\}$. The set of all subsets of N is denoted by 2^N . Assume that R is an arbitrary ring. Let f be a function that associates with every subset $S \subseteq N$ an element $f(S)$ of the ring R . The Möbius transform of f is the function \hat{f} that associates with every $X \subseteq N$ the ring element

$$\hat{f}(X) = \sum_{S \subseteq X} f(S). \quad (13)$$

Given the Möbius transform \hat{f} , the original function f may be recovered via the Möbius inversion formula

$$f(S) = \sum_{X \subseteq S} (-1)^{|S \setminus X|} \hat{f}(X). \quad (14)$$

The *fast Möbius transform* [54, 76] is the following algorithm for computing the Möbius transform in $O(n2^n)$ ring operations.

Lemma 7 *There is an $O(n2^n)$ time algorithm to compute $\hat{f}(X)$ for all $X \subseteq N$.*

Proof To compute \hat{f} given f , let initially

$$\hat{f}_0(X) = f(X)$$

for all $X \subseteq N$ and then iterate for all $j = 1, 2, \dots, n$ and $X \subseteq N$ as follows:

$$\hat{f}_j(X) = \begin{cases} \hat{f}_{j-1}(X) & : \text{if } j \notin X, \\ \hat{f}_{j-1}(X \setminus \{j\}) + \hat{f}_{j-1}(X) & : \text{if } j \in X. \end{cases} \quad (15)$$

It is straightforward to verify by induction on j that this recurrence gives $\hat{f}_n(X) = \hat{f}(X)$ for all $X \subseteq N$ in $O(n^2 2^n)$ ring operations.

Let $N_j = \{1, 2, \dots, j\}$ and $N_0 = \emptyset$. Using induction on j , we can show that $\hat{f}_j(X) = \sum_{X_j \subseteq N_j \cap X} f(X_j \cup Y_j)$, where $Y_j = X - N_j$.

It is trivial for $j = 0$ since $\hat{f}_0(X) = f(X)$. Assume that $\hat{f}_j(X) = \sum_{X_j \subseteq N_j \cap X} f(X_j \cup Y_j)$.

It is easy to see that $\hat{f}_{j+1}(X) = \sum_{X_{j+1} \subseteq N_{j+1} \cap X} f(X_{j+1} \cup Y_{j+1})$ from the definition of $\hat{f}_{j+1}(X)$ and the hypothesis of induction. \square

Then the following lemma gives the details about computing the labeled cycle cover sum for a direct graph.

Lemma 8 *The labeled cycle cover sum $\Lambda(D, L, f)$ for a function f with codomain $GF(2^k)$ on a directed graph G on n vertices, and with $2^k > |L|n$, can be computed in $O(|L| + n)^{O(1)} 2^{|L|}$ arithmetic operations over $GF(2^k)$, where ω is the square matrix multiplication exponent.*

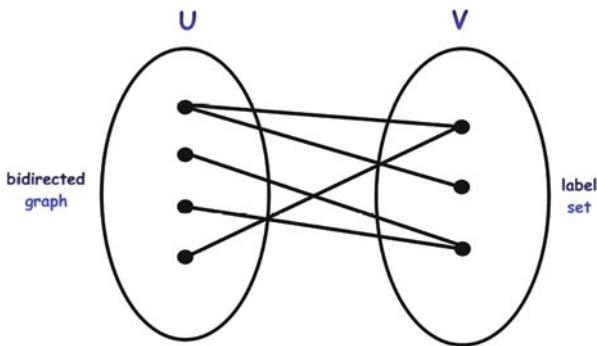
Proof The labeled cycle cover sum $\Lambda(D, L, f)$ can be computed by means of **Lemma 6**. It would be recognized that the runtime of the algorithm is exponential in the number of labels but polynomial in the size of the input graph. According to **Lemma 6**, for polynomial $p(f, r)$, the aim is to achieve the coefficient of $r^{|L|}$. It is easy to see from [Eq. \(12\)](#) that $p(f, r)$ is a polynomial of r with maximum degree $|L|n$. Thus, the polynomial for $|L|n$ choices of r is evaluated, and interpolation is used to solve for the sought coefficient. The condition $2^k > |L|n$ is required to make sure the interpolation has enough different points. Here, a generator g of the multiplicative group in $GF(2^k)$ is used to generate the interpolation point $r = g^0, g^1, \dots, g^{|L|n}$. To do interpolation, for instance, the $O(|L|^2 n^2)$ time Lagrange interpolation is one of the choices.

From the above analysis, it can be achieved that the second part of the runtime bound. However, the much more time-consuming procedures are matrixes tabulating and determinant calculating in formula ([Eq. 9](#)). Let $T(Y) = \sum_{Z \subseteq Y} r^{|Z|} M_f(Z)$: tabulating $T(Y)$ for all $Y \subseteq L$ can be done in $O(|L|2^{|L|})$ time using [Lemma 7](#). Moreover, for the determinant calculating of $p(f, r) = \sum_{Y \subseteq L} \det(T(Y))$, it can be done in $O(n^{O(1)} 2^{|L|})$ as the determinant of an $n \times n$ matrix can be computed in $O(n^{O(1)})$ time using any standard method such as Gauss elimination to convert the matrix of the determinant into a right triangular one. Since there are $|L|n$ values of r , the total runtime for the determinant calculating is bounded by $O((|L| + n)^{O(1)} 2^{|L|})$.

Therefore, by summing up all the time required by the matrix tabulating, the determinant calculating, and the Lagrange interpolation, the labeled cycle cover sum can be computed in $O((|L| + n)^{O(1)} 2^{|L|})$ time. \square

The complexity of the above lemma can be made as low as $O((|L|^2 n + |L|n^{1+\omega}) 2^{|L|} + |L|^2 n^2)$ as mentioned in [9] by using the algorithms of Bunch and Hopcroft [14] for computing determinant and the Coppersmith-Winograd square matrix multiplication exponent [22], where ω is the square matrix multiplication exponent.

Fig. 4 A bipartite graph contains two halves. To reduce the Hamiltonian path to the labeled cycle cover sum, the *left half* U is used as a bidirected graph and the *right half* V is a label set



4.6 Hamiltonicity in Undirected Graphs

In this section, an $O^*(1.657^n)$ time algorithm is introduced to detect Hamiltonian path for direct graphs. For the direct graphs $G = (V, E)$, the overall idea is to reduce Hamiltonian path to labeled cycle cover sum. In Björklund et al.'s paper, they first apply the reduction to bipartite graphs. It gives an $O^*(1.414^n)$ time algorithm for bipartite graphs.

Definition 7 A *bipartite graph* is a graph whose vertices can be divided into two disjoint sets V_1 and V_2 such that every edge connects a vertex in V_1 to one in V_2 .

For the bipartite graph, a smaller bidirected graph D is constructed on one of the halves, and the other half is used as labels in a labeled cycle cover sum on D . (cf. Fig. 4).

For the general direct graph, the core idea is same as the bipartite case. However, there are some slight differences. Compared with the bipartite case, the general graphs do not have the bipartite features. Hence, it seems impossible to partition the vertices into two equal parts of which the Hamiltonian cycle alternate vertices from one part to the other part. Then it is difficult to know which subset of the vertices could serve as labels. To solve this problem, a uniformly randomly chosen partition $V = V_1 \cup V_2$ with $|V_1| = |V_2|$ can be exploited. That is because the number of transitions along a fixed Hamiltonian cycle from a vertex in one part to a vertex in the other part is $n/2$ in expectation. Note that in the bipartite case it is n . So for random partition of the general graph, there still is a good chance that the transitions will happen along the fixed Hamiltonian cycle. The vertices in V_1 in the bipartite case were handled at a polynomial-time cost, whereas the vertices in V_2 case at a price of a factor 2 each in the runtime. In the same vein, the vertices in V_1 followed by a vertex in V_2 along a fixed Hamiltonian cycle in the general case will be computationally cheap. The alternation would not happen each time along the Hamiltonian cycle. Therefore, arcs need to be distinguished according to the partition.

The arcs connecting adjacent vertex pairs v_i, v_{i+1} along a Hamiltonian cycle H are called *unlabeled* by V_2 if both v_i and v_{i+1} belong to V_1 . The remaining arcs are referred to as *labeled* by V_2 . Then the arcs of H can be partitioned into two parts of which $\mathcal{L}(H)$ and $\mathcal{U}(H)$ represent the set of the labeled arcs and the set of the unlabeled arcs, respectively. Define $hc_{V_2}^m(G)$ as the subset of $hc(G)$ of Hamiltonian cycles H which have precisely m arcs unlabeled by V_2 along H .

A vertex s is fixed: two variables x_{uv} and x_{vu} are introduced for every edge $uv \in E$ such that $u \in V_2$ or $v \in V_2$ (or both). $x_{uv} = x_{vu}$ works except when $u = s$ or $v = s$.

For a complete bidirected graph $D = (V_1, F)$, F is the set of arcs connecting every two points u and v in V_1 such that there exists a path from u to v only through points in V_2 in G or edge $uv \in G[V_1]$.

V_2 contains some of the labels. In addition to V_2 , a set L_m contains m extra labels is added to handle arcs unlabeled by V_2 . For each edge uv in $G[V_1]$ and every element $d \in L_m$, new variables $x_{uv,d}$, $x_{vu,d}$ are introduced and set to $x_{uv,d} = x_{vu,d}$ except when $u = s$ or $v = s$. See Fig. 5 for an example.

For two vertices $u, v \in V$ and a nonempty subset $X \subseteq V$, $\mathcal{P}_{u,v}(X)$ is defined as the family of all simple paths in G from u to v passing through exactly the vertices in X (in addition to u and v). For $uv \in F$ and $\emptyset \subset X \subseteq V_2$,

$$f(uv, X) = \sum_{P \in \mathcal{P}_{u,v}(X)} \prod_{wz \in P} x_{wz}.$$

For every arc $uv \in F$ such that uv is an edge in $G[V_1]$, and every $d \in L_m$,

$$f(uv, d) = x_{uv,d}. \quad (16)$$

In all other points, f is set to zero. For simplicity, the general graphs with even number of vertices is considered.

Algorithm 2 Björklund's Algorithm for Hamiltonian Path

Input: A graph $G = (V_1, V_2, E)$ with vertex set v_1, v_2, \dots, v_n

Output: If G does not have a Hamiltonian path, the algorithm always outputs “No”; if G has a Hamiltonian path, the algorithm outputs “Yes” with probability no less than $1 - (1/2)^n$.

Steps:

- 1: Uniformly yield a partition $V_1 \cup V_2 = V$ at random, with $|V_1| = |V_2| = n/2$. Select k large enough such that $2^k > cn$ for some $c > 1$.
- 2: Let $m = 0$.
- 3: Transform the input graph G into a symbolic labeled cycle cover sum $\Lambda(D, V_2 \cup L_m, f)$ with D and f defined as above and the label set $L = V_2 \cup L_m$.
- 4: Pick a random point p from the field $GF(2^k)$ and evaluate $\Lambda(D, V_2 \cup L_m, f)$ in p over $GF(2^k)$ using the method from Lemmas 8 and 9.
- 5: Let $m = m + 1$. Repeat steps 3–5 until $m = m_{\max} = 0.205n$.
- 6: Repeat steps 2–5 for $r = n^{c20.024n}$ runs with constant c to be selected later.
- 7: If any result shows that $\Lambda(D, V_2, f)$ is a nonzero polynomial, output “Yes”; otherwise, output “No.”

End of Algorithm

Hamiltonian undirected graph $G = (V, E)$

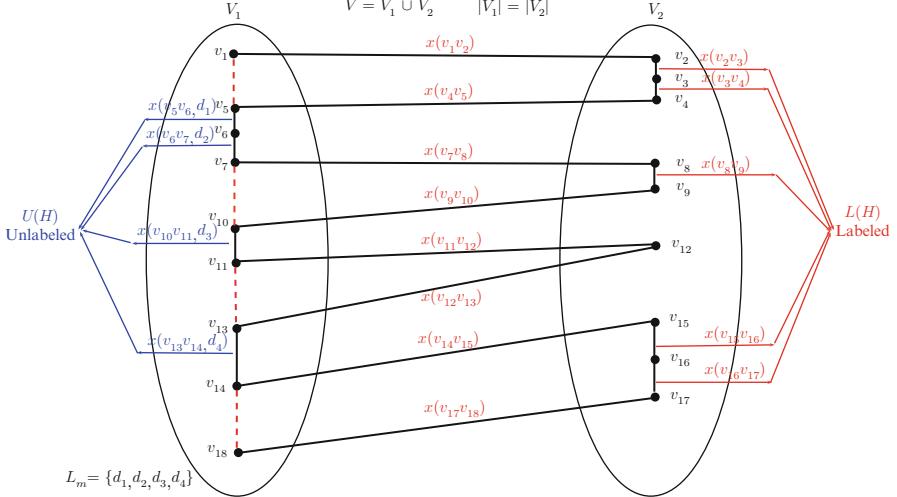


Fig. 5 The undirected graph $G = (V, E)$ with a Hamiltonian path is uniformly randomly partitioned into two equal parts V_1 and V_2 . Each part has nine vertices. The red variables show the labeled arcs, whereas the blue variables represent the unlabeled arcs. Here, for clearness of the figure, one unlabeled arc which is from v_{18} to v_1 is omit. Thus, the extra label set L_m has five elements handling arcs unlabeled by V_2 . This figure is very similar to the bipartite graph except that some unlabeled

The above algorithm has two loops. Since the partition yield randomly in step 1, the inner loop on m is to make sure all possible numbers of unlabeled arcs by V_2 can be evaluated. The algorithm runs for $r = n^{O(1)} 2^{0.024n}$ times to guarantee that the probability of the false negatives is exponentially small in n . Then the following lemma shows the relationship between labeled cycle cover sum $\Lambda(D, V_2 \cup L_m, f)$ and the Hamiltonian cycles of G . And Lemma 9 ensures the correctness of the algorithm for the general graphs.

Lemma 9 With $G, D, V_2, \mathcal{U}, \mathcal{V}, \mathcal{L}, m, L_m$, and f defined above,

1. $\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc_{V_2}^m(G)} (\sum_{\sigma: \mathcal{U}(H) \rightarrow L_m} \prod_{uv \in \mathcal{U}(H)} x_{uv, \sigma(uv)}) (\prod_{uv \in \mathcal{L}(H)} x_{uv})$ with σ one to one.
2. $\Lambda(D, V_2 \cup L_m, f)$ is the zero polynomial iff $hc_{V_2}^m(G) = \emptyset$.

Proof (i) Since D is bidirected and f is an s -oriented mirror function, by Lemma 5

$$\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc(D)} \sum_{g: V_2 \cup L_m \rightarrow H} \prod_{a \in H} f(a, g^{-1}(a)).$$

By omitting the vertices which belong to V_2 , it is easy to see that there is a mapping from a Hamiltonian cycle $H \in hc(G)$ to a Hamiltonian cycle

in D . Observing that the above equation is over all the Hamiltonian cycles in D , detecting the Hamiltonicity of G can be done in the backward way, to expand the Hamiltonian cycles in D into Hamiltonian cycles of G . Since previous works were first done on the Hamiltonian cycles in D , it is necessary to extend the definition of labeled and unlabeled arc which were defined before for Hamiltonian cycles in G only. For a Hamiltonian cycle $H \in hc(D)$ labeled by the function $g : V_2 \cup L_m \rightarrow H$, an arc $uv \in H$ is labeled by V_2 if $g^{-1}(uv) \subseteq V_2$ and unlabeled by V_2 if $g^{-1}(uv) \in L_m$. H_L and H_U are used to represent the labeled and unlabeled set of arcs, respectively.

According to the definition of the function f , the arcs of a Hamiltonian cycle in D in the labeled cycle cover sum either are labeled by an element of L_m or a nonempty subset of V_2 , since these are the only subsets of the labels for which f is nonzero. Moreover, the definition of f at Eq.(16) tells us that every arc unlabeled by V_2 along a Hamiltonian cycle consumes exactly one of the m labels in L_m , and all labels are used. Thus, only the Hamiltonian cycles in D with exactly m arcs unlabeled by V_2 leave a nonzero contribution. Note that there is another restriction that a cycle H leaves a nonzero result only if the m arcs unlabeled by V_2 along the cycle are also edges in G . Considering all the above restrictions on the inner summation in above formula,

$$\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc(D)} \sum_{\substack{H_U \cup H_L = H \\ H_U \cap H_L = \emptyset \\ |H_U| = m \\ \forall a \in H_U : a \in E}} \Lambda_{H_U}(L_m) \Lambda_{H_L}(V_2)$$

with

$$\Lambda_{H_U}(L_m) = \left(\sum_{\sigma : H_U \rightarrow L_m} \prod_{a \in H_U} f(a, \sigma(a)) \right)$$

and

$$\Lambda_{H_L}(V_2) = \left(\sum_{g : V_2 \rightarrow H_L} \prod_{a \in H_L} f(a, g^{-1}(a)) \right).$$

Through analysis above, it is easy to see that the summation in $\Lambda_{H_U}(L_m)$ is over all functions σ which are one to one. Replacing f by its definition, the inner expressions is expanded to

$$\Lambda_{H_U}(L_m) = \left(\sum_{\sigma : H_U \rightarrow L_m} \prod_{uv \in H_U} x_{uv, \sigma(uv)} \right)$$

and

$$\Lambda_{H_L}(V_2) = \left(\sum_{g: V_2 \rightarrow H_L} \prod_{uv \in H_L} \sum_{P \in \mathcal{P}_{u,v}(g^{-1}(uv))} \prod_{wz \in P} x_{wz} \right).$$

Since every vertex in V_2 is mapped to by precisely one arc in F on a Hamiltonian cycle H in D , for arcs $a_1, a_2 \in H_L$ and $a_1 \neq a_2$, $g^{-1}(a_1) \cap g^{-1}(a_2) = \emptyset$. Thus, for every arc uv in the labeled set H_L , $\Lambda_{H_L}(V_2)$ exhausts all the possible simple paths from u to v passing through the vertices in X with $\emptyset \subset X \subseteq V_2$. Then through picking one simple path for each labeled arc in H_L , the combination of all the simple paths grouped with all the unlabeled arcs actually forms a Hamiltonian cycle in G . Therefore, rewriting the expression as a sum of Hamiltonian cycles in G , the result establishes as claimed:

$$\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc_{V_2}^m(G)} \left(\sum_{\sigma: \mathcal{U}(H) \rightarrow L_m} \prod_{uv \in \mathcal{U}(H)} x_{uv, \sigma(uv)} \right) \left(\prod_{uv \in \mathcal{L}(H)} x_{uv} \right)$$

- (ii) If the graph G has no Hamiltonian cycles, the sum is clearly zero from above equation. Otherwise, if G has Hamiltonian cycles, then from the inner summation of the above equation, it is easy to see that each Hamiltonian cycle contributes a set of $m!$ different monomials per orientation of the cycle (one for each permutation of σ). Each edge of the cycle contributes a variable for the monomials. Since two different Hamiltonian cycles each has an edge and the other has not, monomials resulting from different Hamiltonian cycles are different. By the definition of s -oriented mirror function, the variables associated to the oppositely directed arcs incident to s are different. Thus, the monomials formed by oppositely oriented Hamiltonian cycles are unique in the sum.

□

Lemma 10 *There is an $O(n)$ time algorithm to randomly partition a set of (even) n elements into two equal parts with $n/2$ elements each.*

Proof Let V be a set of $n = 2m$ elements. Let V_1 be empty in the beginning. Select a random element from V , put it into V_1 , and remove the selected element from V . Repeat the above step until V_1 contains m elements. It is easy to see equal subset of size m of V has the same probability to be generated. Let U be a subset of m elements in V . With probability $\frac{m}{2m}$, the first selected is selected from U . Assume that the first i elements are all selected from U for some $i < m$. With probability $\frac{m-i}{2m-i}$, the $i+1$ -th element is selected from U . Thus, the probability that U is generated by the algorithm is

$$\frac{m}{2m} \cdot \frac{m-1}{2m-1} \cdots \frac{1}{m+1} = \frac{1}{\binom{2m}{m}}.$$

□

We need to follow probability amplification lemma for the randomized algorithm for Hamiltonian cycle problem. [Lemma 11](#) is a standard method in the randomized algorithm design.

Lemma 11 *Let A be an randomized algorithm for the Hamiltonian cycle problem such that for every $G = (V, E)$ without Hamiltonian cycle, it always returns “No” (zero false positive), and for every $G = (V, E)$ with a Hamiltonian cycle, it returns “Yes” with probability at least $\frac{1}{\alpha(n)}$ with $\alpha(n) > 1$. Then there is another algorithm A' by repeating A $n^{O(1)}\alpha(n)$ times such that for every $G = (V, E)$ without Hamiltonian cycle, it always returns “No,” and for every $G = (V, E)$ with a Hamiltonian cycle, it returns “Yes” with probability at least $1 - \frac{1}{2^n}$.*

Proof Let $r = n^c\alpha(n)$ with constant c to be determined later. The algorithm A is repeated r times. If one of the r times returns “Yes,” then the algorithm A' returns “Yes.” Otherwise, the algorithm A' returns “No.” It is easy to see that A' has a zero false positive. If the input graph $G = (V, E)$ has a Hamiltonian path, then with probability at most $(1 - \frac{1}{\alpha(n)})^r$, all of the r times execution of A returns “No.” Therefore, with probability at least $1 - (1 - \frac{1}{\alpha(n)})^r \geq 1 - \frac{1}{2^n}$, at least one of the r times of A executions returns “Yes” if constant c is selected to be large enough. \square

Theorem 4 ([10]) *There is a Monte Carlo algorithm detecting whether an undirected graph on n vertices is Hamiltonian or not running in $O^*(1.657^n)$ time, with no false positives and false negatives with probability exponentially small in n .*

Proof From [Lemma 9](#), the readers can realize the practicability of Björklund’s algorithm for the general graphs. The only thing left is to analyze the runtime and the probability of false negatives of our run results.

For the runtime, from the definition of $f(uv, X)$ for $\emptyset \subset X \subseteq V_2$, to evaluate the labeled cycle cover sum $\Lambda(D, V_2 \cup L_m, f)$, the function f needs to be tabulated for all subsets of V_2 . This can be achieved by running a variant of the Bellman-Held-Karp recursion. Formally, let $\hat{f} : (V \times V) \times 2^{V_2} \rightarrow GF(2^k)$ be defined for $u \neq v$ and $\emptyset \subset X \subseteq V_2$ by

$$\hat{f}(uv, X) = \sum_{P \in \mathcal{P}_{u,v}(X)} \prod_{w \in P} x_{wz}$$

then $f(uv, X) = \hat{f}(uv, X)$ for $uv \in F$ and $\emptyset \subset X \subseteq V_2$. For $|X| > 1$ the recursion

$$\hat{f}(uv, X) = \sum_{w \in X, uv \in E} x_{uw} \hat{f}(wv, X \setminus w)$$

can be used to tabulate $f(uv, X)$ for all $\emptyset \subset X \subseteq V_2$. Since V_2 has $n/2$ vertices, the tabulation for f can be done in $O^*(2^{n/2})$ time. After tabulation, $\Lambda(D, V_2 \cup L_m, f)$ can be evaluated by [Lemma 8](#). Then the runtime of the evaluation is close relevant

to the size of the label set $V_2 \cup L_m$. Since the size of V_2 is fixed to $n/2$, the only need is the analysis of the maximum size of L_m .

Let $G = (V, E)$ be a Hamiltonian undirected graph and $V_1 \cup V_2 = V$ with $|V_1| = |V_2|$. We will show

$$\Pr(|hc_{V_2}^m(G)| > 0) \geq \frac{\binom{n/2-1}{m-1}^2}{\binom{n}{n/2}} \in \Theta\left(\frac{m(\frac{1}{2} - \frac{m}{n})^{2m-n-1}}{(\frac{m}{n})^{2m} 4^n \sqrt{2\pi n^{1.5}}}\right). \quad (17)$$

To obtain the inequality in (Eq. 17), it is necessary to count the number of possibility that one fixed Hamiltonian cycle $H = (v_0, v_1, c \dots, v_{n-1})$ has exactly m arcs unlabeled by V_2 and moreover has $v_0 \in V_1$ and $v_{n-1} \in V_2$. For such a H , there are exactly $n/2 - m$ indices i such that $x_i \in V_1$ and $x_{i+1} \in V_2$, and just as many indices i where $x_i \in V_2$ and $x_{i+1} \in V_1$. Take Fig. 5 as an example, since $m = 5$, there are $n/2 - m = 4$ vertices $(v_1, v_7, v_{11}, v_{14})$ which satisfy $x_i \in V_1$ and $x_{i+1} \in V_2$. And similarly there are 4 vertices $(v_4, v_9, v_{12}, v_{17})$ which satisfy $x_i \in V_2$ and $x_{i+1} \in V_1$. The ordered list of these transition indices $i_1, i_2, \dots, i_{n-2m}$ uniquely describes the partition. In other words, any such list with $0 \leq i_1, \forall j : i_j < i_{j+1}$, and $i_{n-2m} = n-1$ corresponds to a unique partition. Set $i_0 = -1$ and define the positive integers $d_j = i_j - i_{j-1}$ for all $0 < j \leq n-2m$. Then $d_1, d_2, \dots, d_{n-2m-1}$ describes a partition of the vertices in V_1 in $n/2 - m$ groups. Analogously, $d_2, d_4, \dots, d_{n-2m}$ describes a partition of the vertices in V_2 in $n/2 - m$ groups. The number of ways to write a positive integer as a sum of k positive integers is $\binom{p-1}{k-1}$. Thus, by multiplying the number of ways to partition the vertices in V_1 with the number of ways for V_2 , and dividing with the total number of balanced partition $\binom{n}{n/2}$, the inequality in Eq. (17) is achieved.

The second bound is derived by replacing the binomial coefficients with their factorial definition and using Stirling's approximation for $n! \in \Theta((n/e)^n \sqrt{2\pi n})$.

Since $\Pr(\sum_{m=0}^{m_{\max}} |hc_{V_2}^m(G)| = 0) \leq 1 - \Pr(|hc_{V_2}^{m_{\max}}| > 0)$, to bring the probability of false negatives down to $\exp(-\Omega(n))$, the number of runs $r = n^{O(1)} \Pr^{-1}(|hc_{V_2}^{m_{\max}}| > 0)$ is needed by Lemma 11. Solving for the local minimum of the total runtime, using the inequality in (Eq. 17) to bound the probability, $m_{\max} = 0.205n$ and $r = n^{O(1)} 2^{0.024n}$ runs. Therefore, the total runtime is bounded by $O^*(1.657^n)$. \square

5 Minimum Common String Partition Problem

In this section, an application of the multilinear testing problem is presented to illustrate the practicability of algebrization. By means of multilinear testing for polynomials, a new exponential algorithm was recently developed by Fu et al. [33] to solve the minimum common string partition problem (MCSP). The problem MCSP has attracted a lot of attention, partly because of its applications in computational biology, text processing, and data compression. More specifically, MCSP has a close connection with the genome rearrangement problems such as edit

distance and sorting by reversals. In fact, MCSP was initially studied exactly as a problem on “edit distance with moves” [70].

Definition 8 Given a pair of strings S and S' , the common string partition problem is to find partitions for $S = S_1 \dots S_k$ and $S' = S'_1 \dots S'_k$ such that S'_1, \dots, S'_k is a rearrangement of S_1, \dots, S_k and k is the least.

Naturally, in the minimum common string partition problem, two strings X and Y of length n are given over an alphabet Σ . Let each symbol appear the same number of times in X and Y . This condition is certainly sufficient and necessary for us to compute a common string partition for X and Y . For example, two strings $X = beadcdb$ and $Y = abdebc$ have a common partition ($< b, e, ab, c, db >$, $< ab, db, e, b, c >$). There are several variants of MCSP. Restricted version is called where each letter occurs at most d times in each input string as d -MCSP. When the input strings are alphabet of size c , the corresponding problem is called $MCSP^c$.

5.1 A Brief History of MCSP

The problem d -MCSP has been well studied. 2-MCSP (and therefore MCSP) is NP-hard and APX-hard [37]. Several approximation algorithms have been proposed for MCSP problem [37, 59]. Chen et al. [18] studied the problem of computing signed reversal distance with duplicates (SRDD). They introduced the signed minimum common partition problem as a tool for dealing with SRDD and observed that for any two related signed strings X and Y , the size of a minimum common partition and the minimum number of reversal operations needed to transform X and Y are within a factor of 2. Kolman and Walen [60] devised an $O(d^2)$ -approximation algorithm running in $O(n)$ time for SRDD.

Chrobak et al. [21] analyzed the greedy algorithm for MCSP; they showed that for 2-MCSP the approximation ratio is exactly 3, for 4-MCSP the approximation ratio is $\Omega(\log n)$, and for the general MCSP, the approximation ratio is between $\Omega(n^{0.43})$ and $O(n^{0.67})$. Kaplan and Shafrir [50] improved the lower bound to $\Omega(n^{0.46})$ when the input strings are over an alphabet of size $O(\log n)$. Kolman [58] described a simple modification of the greedy algorithm; the approximation ratio of the modified algorithm is $O(p^2)$ for p -MCSP. Most recently, Goldstein and Lewenstein presented an greedy algorithm for MCSP [36].

In the framework of parameterized complexity, Damaschke first solved MCSP by an FPT algorithm with respect to parameters k (size of the optimum solution), r (the repetition number), and t (the distance ratio, depending on the shortest block in the optimum solution) [24]. Jiang et al. showed that both d -MCSP and $MCSP^c$ admit FPT algorithms when d and c are (constant) parameters [47]. However, it remains open whether MCSP admits any FPT algorithm parameterized only on k .

5.2 An $O(2^n n^{O(1)})$ Time Algorithm for General Cases

This section shows that algebrization, especially multilinear monomial testing, is a very useful tool by presenting an exact solution for the MCSP problem. The overall idea is to convert it into a detection of a multilinear monomial in the sum-product expansion of a multivariate polynomial.

Definition 9 Let $S = a_1 a_2 \cdots a_n$ be a string of length n ;

- For an integer $i \in [1, n]$, define $S[i] = a_i$.
- For an interval $[i, j] \subseteq [1, n]$, define $S[i, j]$ to be the substring $a_i a_{i+1} \cdots a_j$ of S .

The aim is to create a polynomial in which a multilinear monomial is used to encode a solution for the partition problem. Then the following theorem explains how to construct polynomial according to two given strings S and S' and gives the proof of the correctness of encoding.

Theorem 5 ([33]) *There is an $O(2^n n^{O(1)})$ time algorithm for the minimum common string partition problem.*

Proof Let S and S' be two input strings of length n . Let each position i of S' have a unique variable x_i to represent it. Define $P[a, b] = x_a x_{a+1} \cdots x_b$. For each interval $[s, t]$, define $G_{s,t} = \sum_{S'[s',t']=S[s,t]} P[s', t']$.

For each $i \leq n$, define a polynomial $F_{i,1} = G_{1,i}$. It represents all the substrings of S' that match $S[1, i]$. This is formally stated by Claim 2 below.

Claim 2 $S'[u, v]$ is a substring of S' with $S'[u, v] = S[1, i]$ if and only if there is a multilinear monomial $P[u, v]$ in $F_{i,1}$.

Let $F_{0,t} = 1$ for each $t \geq 1$. For each $i \leq n$, define

$$F_{i,t+1} = \sum_{1 \leq j \leq i} G_{j,i} \cdot F_{j-1,t}. \quad (18)$$

This polynomial has the property for encoding some partial common partition. It is stated in Claim 3.

Claim 3 Let i be an arbitrary integer parameter in the range $[1, n]$. Then there is a partition $S[i_1, j_1], \dots, S[i_t, j_t]$ for $S[1, i]$ such that $S[i_r, j_r] = S'[i'_r, j'_r]$ for some disjoint intervals $[i'_1, j'_1], \dots, [i'_t, j'_t]$ that are subset of $[1, n]$ if and only if there is a multilinear monomial Q in the sum of product expansion of $F_{i,t}$ with $Q = \prod_{r=1}^t P[i'_r, j'_r]$.

Proof Prove by induction. For the case $t = 1$, it follows from Claim 2. Assume that the claim is true for t .

Considering the case $t + 1$, the following two cases for the induction are needed to be analyzed:

- Assume that there is a partition $S[i_1, j_1], \dots, S[i_{t+1}, j_{t+1}]$ for $S[1, i]$ such that $1 = i_1 \leq j_1 = i_2 - 1 < j_2 = \dots = j_t = i_{t+1} - 1 < j_{t+1} = i$ and $S[i_r, j_r] = S'[i'_r, j'_r]$ for $r = 1, \dots, t + 1$ with some disjoint intervals $[i'_1, j'_1], \dots, [i'_{t+1}, j'_{t+1}]$. By our inductive hypothesis, there is a multilinear monomial $Q' = \prod_{r=1}^t P[i'_r, j'_r]$ that is in the sum of product expansion of $F_{j_r, t}$. By the definition of $G_{i_{t+1}, j_{t+1}}$, $G_{i_{t+1}, j_{t+1}}$ contains the multilinear monomial $P[i'_{t+1}, j'_{t+1}]$ in its sum of product expansion. By the definition of $F_{i, t+1}$ in Eq.(18), multilinear monomial $Q = Q' \cdot P[i'_{t+1}, j'_{t+1}]$ in the sum of expansion of $F_{i, t+1}$.
- Assume that there is multilinear monomial Q in $F_{i, t+1}$. By the definition of $F_{i, t+1}$ in Eq.(Eq. 18), there is an integer j and an interval $[i'_{t+1}, j'_{t+1}]$ such that $S[j, i] = S'[i'_{t+1}, j'_{t+1}]$, and there is a multilinear monomial Q' with $Q = P[i'_{t+1}, j'_{t+1}] \cdot Q'$.

By the inductive hypothesis, there is a partition $S[i_1, j_1], \dots, S[i_t, j_t]$ for $S[1, j - 1]$ such that for each $1 \leq r \leq t$, $S[i_r, j_r] = S'[i'_r, j'_r]$ for some disjoint intervals $[i'_1, j'_1], \dots, [i'_t, j'_t]$, and $Q' = \prod_{r=1}^t P[i'_r, j'_r]$. Since Q is a multilinear monomial, there is no intersection between $[i'_{t+1}, j'_{t+1}]$ and $\cup_{r=1}^t [i'_r, j'_r]$. \square

By Claim 3, it is easy to see that there is a partition of k segments if and only if there is a multilinear monomial $x_1 x_2 \cdots x_n$ in the sum of product expansion of $F_{n, k}$.

Then there is an $O(2^n n^{O(1)})$ time algorithm to find the multilinear monomial by evaluating $F_{n, k}$ from bottom-up. During the evaluation, only the multilinear monomials are kept in $F_{i, t}$. Since the total number of variables is n , the total number of multilinear monomials is at most 2^n . Note that $G_{j, i}$ has at most $O(n)$ monomials. Therefore, each multiplication $G_{j, i} F_{j-1, t}$ takes $O(2^n n^{O(1)})$ time. The arithmetic expression for $F_{n, k}$ involves $n^{O(1)}$ $+$ and \cdot operations. Therefore, the total time for generating all the multilinear monomials in the sum of product expansion of $F_{n, k}$ is $O(2^n n^{O(1)})$.

Testing the existence of multilinear monomial $x_1 x_2 \cdots x_n$ in the sum of product expansion over $F_{n, k}$ gives the existence of a common partition of at most k blocks. Tracing how the multilinear monomial $x_1 x_2 \cdots x_n$ is formed with a bottom-up approach to find all multilinear monomial in the sum of product expansion of $F_{n, k}$ shows how a common partition of at most k blocks is computed. This can be seen in the proof of Claim 3. \square

6 Algebraic FPT Algorithms for Channel Scheduling Problem

In this section, the algorithms for scheduling data retrieval in multichannel wireless data broadcast environments will be introduced. These algorithms are derived from the multilinear testing theory. Moreover, they also use algebraic method and randomized method to minimize the total time for channel scheduling. Generally, the Channel Scheduling Problem, which is also named the least time data retrieving (LTDR)

problem, is belong to wireless data broadcast. In many applications, such as stock quotes, flight schedules, and traffic news [1], the LTDR problem appears when people may want to download multiple data items at a time.

There are two major performance concerns with respect to wireless data broadcast, one is tuning time and the other is access time. The tuning time, defined as the amount of time a client spends on listening to the channels, is a performance criterium to evaluate the energy consumption, while the access time, defined as the time duration from the moment a client submits a query to the moment that all the required data are downloaded, reflects the response delay of queries. The LTDR problem is the problem of finding a schedule to download a set of required data, which are cyclically broadcasted via multiple broadcast channels, such that all the required data can be downloaded in a short time. The formal definition of the LTDR problem is as follows.

Definition 10 Given a query data set S and a broadcast program of k channels, find a valid schedule to download all the data items in S in some order so that the total access time is minimized.

In the following paragraphs, the related works about LTDR is first introduced. Then the algebraic FPT algorithms for LTDR will be presented in the next section. These works can be found in the recently accepted paper by Gao, Lu, Wu, and Fu [34].

6.1 A Brief History of LTDR

As it is talked about before, *access time* is one of the major factors with respect to the performance of the wireless data broadcast. Many works have been done to minimize the *access time* [42, 46, 71], etc. The LTDR problem, which is from the client point of view, is one of the optimization problems in the field of wireless data broadcast.

Under the assumption that the indices of all required data are already obtained, the LTDR problem focuses on developing efficient scheduling algorithms from the client point of view with the objective of minimizing the access time. Compared to the massive amount of works for scheduling the data retrieval process at the client side, there has been little work done on scheduling the data retrieval process at the client side. When there is only one broadcast channel, or the client requests single data item at a time, the data retrieval scheduling problem is straightforward. But in many cases, the clients may want to download multiple data per request [39–42, 48, 62, 71]. Juran et al. and Hurson et al. presented several data retrieval heuristics in [42, 48] to reduce the access time and the number of channel switchings. In [71], Shi et al. investigated how to schedule the downloading process at the client side by using multiple parallel processes. It is worthy to mention that the studies in [42, 48, 71] assume the data are partitioned over multiple channels without replications. This assumption simplifies the algorithm design, but there are many data allocation scheduling may result in the replicative data appearing on channels [1, 30, 41].

Here, the LTDR problem assumes that the data are allowed replicatively. Compared with the model studied in [42, 48, 71], this is more generalized assumption which fits all wireless data broadcast programs. Under this assumption, Du et al. give the complexity analysis and some approximation algorithms of the LTDR problem in their recently work [27].

6.2 Algebraic FPT Algorithms for LTDR

In this section, a fixed parameter tractable algorithm with computational time $O(2^t(nkh)^{O(1)})$ for the LTDR problem is introduced. Here, k is the number of channels, t is the least number of programs to be downloaded in a set of programs S , n is the maximal time slots, and h is the maximal number of channel switches.

A problem with input size n and parameter k has a fixed parameter tractable (FPT) algorithm if there is an algorithm that runs in an $O(f(k)n^{O(1)})$ time. Looking for FPT algorithms for NP-hard problems has been widely studied in the last two decades. Although it is unknown if the LTDR problem is NP-hard, it looks a challenging problem and hard to find the optimal solution.

Specifically, the LTDR problem can be described as the clients want to download a set S of m programs from k channels by a receiver. Each channel C_i is partitioned into many discrete time slots, and a program p_i takes integer $s_{i,j}$ of time slots in channel j . If the receiver switches from channel i at time t to channel j , it can start downloading the program in channel j at time $t+2$. As described before, the target is to minimize the total time and the number of switches among those channels.

Definition 11 A subset program $S' = \{x_{i_1}, \dots, x_{i_a}\} \subseteq S$ is (i, t, h) -downloadable if the receiver can download all programs in S' in time interval $[1, t]$, the total number of channel switches is at most h , and a program x_{i_j} is downloaded from channel i in the final time phase.

The following lemma transform the channel switches allowed LTDR problem to the monomial testing problem.

Lemma 12 *There is polynomial-time algorithm such that it constructs a circuit for a polynomial $F_{i,t,h,a}$ such that for each (i, t, h) -downloadable subset $S' = \{p_{i_1}, \dots, p_{i_a}\} \subseteq S$, the sum of product expansion of $F_{i,t,h,a}$ contains a multilinear monomial $(x_{i_1}, \dots, x_{i_a})Y$, where each variable x_i represents program p_i for $i = 1, \dots, m$, and Y is a multilinear monomial with only y variables.*

Proof For each program $p_i \in S$, let variable x_i represent it. A recursive way is used to define the polynomial $F_{i,t,h,a}$:

1. $F_{i,t,0,0} = 0$.
2. $F_{i,t,0,1} = \sum_{\substack{p_j \in S, \\ p_j \text{ is entirely in} \\ \text{the time interval} \\ [t_1, t_2] \text{ of channel } i}} x_j$.

$$3. F_{i,t,h+1,b+1} = y_{i,t,h+1,b+1,1} (\sum_{t_1 < t} F_{i,t_1,h+1,b} \cdot P_{i,t_1+1,t,1}) + y_{i,t,h+1,b+1,1} (\sum_{t_1 < t, j \neq i} F_{j,t_1-1,h,b} \cdot P_{i,t_1+1,t,1}).$$

In the recursion for $F_{i,t,h+1,b+1}$, it is based on the two cases. The first case, which corresponds to the term $(\sum_{t_1 < t} F_{i,t,h+1,b} \cdot P_{i,t_1+1,t,1})$, is that the second last program is still downloaded from channel i and so is the last program. The second case, which corresponds to the term $(\sum_{t_1 < t, j \neq i} F_{j,t_1-1,h,b} \cdot P_{i,t_1+1,t,1})$, is that the second last program is downloaded from another channel $j \neq i$ and the last program is downloaded from channel i . The switch needs one unit additional time than the case without channel switch. \square

If the clients just need to download a subset of t programs in set S , we have the following theorem to get an $O(2^t(nkh)^{O(1)})$ time randomized algorithm with two layers of randomization.

Theorem 6 ([34]) *There is an $O(2^t(nkh)^{O(1)})$ time randomized algorithm to determine if there is a scheduling to download t programs from S in at most n time slots and at most h channel switches.*

Proof By Lemma 12, let polynomial $H_{n,h,t} = \sum_{i=1}^k F_{i,n,h,t}$. It is easy to see that there is a scheduling for downloading at least t programs in time n and k channel switches if and only if the sum-product expansion of $H_{n,h,t}$ has a multilinear monomial $(x_{i_1}, \dots, x_{i_t})$, where each variable x_i represents program p_i for $i = 1, \dots, m$.

Replace each x_i with a vector $w_i = u_0^T + v_i^T$, where $u_0 = (0, \dots, 0)$ is the zero vector of dimension t and v_i is a random vector of dimension t .

Define the operation $(a_1, \dots, a_m)^T \cdot (b_1, \dots, b_m)^T = ((a_1 + b_1)(mod 2), \dots, (a_m + b_m)(mod 2))^T$. The replacement $x_i = w_i$, ($i = 1, \dots, m$) makes all monomials, which has non-multilinear monomial in its x part, be zero.

Assume that the sum of product expansion of $H_{n,h,t}$ has a multilinear monomial $(x_{i_1} \cdots x_{i_t})$. For a series of random vectors of dimension t : v_{j_1}, \dots, v_{j_t} , with probability at most $2^{t-1}/2^t = 1/2^{t-i+1}$, v_{j_i} is a linear combination of $v_{j_1}, \dots, v_{j_{i-1}}$. Therefore, with probability at most $\sum_{i=1}^t \frac{1}{2^{t-i+1}} \leq \frac{3}{4}$, v_{j_i} is a linear combination of $v_{j_1}, \dots, v_{j_{i-1}}$ for some $i \leq t$. When v_{i_1}, \dots, v_{i_t} are linearly independent, the product w_{i_1}, \dots, w_{i_t} is nonzero. Every monomial in the sum of product expansion has different product Y of its y variables since it is determined by a unique path forming the polynomial. Therefore, for those random vectors v_i , every multilinear monomial has a chance at least $1 - \frac{3}{4} = \frac{1}{4}$ be nonzero. There is solution if and only if with probability at least $\frac{1}{4}$, $H_{n,h,t}|_{x_i=w_i(i=1,\dots,m)}$ is not a zero polynomial in the field G_2 .

It is well known that there is randomized polynomial-time algorithm to check if a polynomial is identical to zero.

After the replacements, it generates 2^t terms since there are 2^t vectors of dimension t . The coefficient of each vector is kept as a polynomial size circuit. Therefore, the time is $O(2^t(nkh)^{O(1)})$. \square

7 The Complexity of Testing Monomials

Through the discussions of previous sections, the readers may have understood the motivation and necessity of the study about the monomial testing problem for multivariate polynomials. In this section, the question which asks whether a polynomial represented by certain economically compact structure has a multilinear monomial in its sum-product expansion will be discussed. The main results of this part come from the recent works by Chen and Fu [15]. The complexity aspects of this problem and its variants are investigated with twofolds of objectives. One is to understand how this problem relates to critical problems in complexity and if so to what extent. The other is to exploit possibilities of applying algebraic properties of polynomials to the study of those problems. The monomial testing problem is related to, and somehow complements with, the low-degree testing and the identity testing of polynomials. In addition to the k -path problem and the Hamiltonian path problem, applications has been found in genome sequencing [33] and wireless data broadcast [34]. Also it has been used to interpret the complexity of multivariate integration and derivative [26, 32]. In the following subsection, a brief history about the complexity of polynomials is first introduced. Then a series of results about $\prod \sum \prod$ and $\prod \sum$ polynomials are presented. These results are a part of the foundation of testing monomials in multivariate polynomials [15, 16, 20].

There has been a long history in complexity theory with heavy involvement of studies and applications of polynomials. More notably, low-degree polynomial testing or representing and polynomial identity testing have played invaluable roles in many major breakthroughs in complexity theory. For example, low-degree polynomial testing is involved in the proof of the PCP theorem, the cornerstone of the theory of computational hardness of approximation, and the culmination of a long line of research on IP and PCP [4, 29]. Polynomial identity testing has been extensively studied due to its role in various aspects of theoretical computer science (e.g., [17, 49]) and its application in various fundamental results such as Shamir's IP=PSPACE [69] and the AKS primality testing [63]. Low-degree polynomial representing [64] has been sought so as to prove important results in circuit complexity, complexity class separation, and subexponential time learning of boolean functions (e.g., [6, 31, 55]). These are a just few examples. A survey of the related literature is certainly beyond the scope of this chapter.

7.1 Notations and Definitions

Let $\mathcal{P} \in \{Z, Z_{\geq 0}\}$, where Z is the set of all integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$ and $Z_{\geq 0}$ is the set of nonnegative integers $\{0, 1, 2, \dots\}$. For variables x_1, x_2, \dots, x_n , let $\mathcal{P}[x_1, \dots, x_n]$ denote the n -variate polynomials with coefficients from \mathcal{P} . For $1 \leq i_1 \leq \dots \leq i_k \leq n$, $\pi = x_{i_1}^{j_1} \cdots x_{i_k}^{j_k}$ is called a *monomial*. The degree of π , denoted by $\deg(\pi)$, is $\sum_{s=1}^k j_s$. π is multilinear, if $j_1 = \dots = j_k = 1$, i.e., π is linear in all

its variables x_{i_1}, \dots, x_{i_k} . For any given integer $c \neq 1$, π is called a c -monomial, if $1 \leq j_1, \dots, j_k < c$.

An arithmetic circuit, or circuit for short, is a direct acyclic graph with $+$ gates of unbounded fan-ins, \times gates of two fan-ins, and all terminals corresponding to variables. The size, denoted by $s(n)$, of a circuit with n variables is the number of gates in it. A circuit is called a formula, if the fan-out of every gate is at most one, i.e., the underlying direct acyclic graph is a tree.

By definition, any polynomial $p(x_1, \dots, x_n)$ can be expressed as a sum of a list of monomials, called the *sum-product expansion*. The degree of the polynomial is the largest degree of its monomials in the expansion. With this expression, it is trivial to see whether $p(x_1, \dots, x_n)$ has a multilinear monomial or a monomial with any given pattern. Unfortunately, this expression is essentially problematic with any given pattern. Unfortunately, this expression is essentially problematic and infeasible to realize, because a polynomial may often have exponentially many monomials in its expansion.

In general, a polynomial $p(x_1, \dots, x_n)$ can be represented by a circuit or some even simpler structure as defined in the following. This type of representation is simple and compact and may have a substantially smaller size, say, polynomially in n , in comparison with the number of all monomials in the sum-product expansion. The challenge is how to test whether $p(x_1, \dots, x_n)$ has a multilinear monomial or some needed monomial, efficiently without unfolding it into its sum-product expansion.

Definition 12 Let $p(x_1, \dots, x_n) \in \mathcal{P}[x_1, \dots, x_n]$ be any given polynomial. Let $m, s, t \geq 1$ be integers:

- $p(x_1, \dots, x_n)$ is said to be a $\prod_m \sum_s \prod_t$ polynomial, if $p(x_1, \dots, x_n) = \prod_{i=1}^m F_i$, $F_i = \sum_{j=1}^{r_i} X_{ij}$ and $1 \leq r_i \leq s$, and $\deg(X_{ij}) \leq t$. Each F_i is called a clause. Note that X_{ij} is not a monomial in the sum-product expansion of $p(x_1, \dots, x_n)$ unless $m = 1$. To differentiate this subtlety, X_{ij} is called a term.
- In particular, $p(x_1, \dots, x_n)$ is a $\prod_m \sum_s$ polynomial, if it is a $\prod_m \sum_s \prod_1$ polynomial. Here, each clause is a linear addition of single variables. In other word, each term has degree 1.
- When no confusing arises from the correct, $\prod \sum \prod$ and $\prod \sum$ is used to stand for $\prod_m \sum_s \prod_t$ and $\prod_m \sum_s$, respectively. Similarly, $\prod \sum_s \prod$ and $\prod \sum_s$ is used to stand for $\prod_m \sum_s \prod_t$ and $\prod_m \sum_s$, respectively, emphasizing that every clause in a polynomial has at most s terms or is a linear addition of at most s single variables.
- For any given integer $k > 1$, $p(x_1, \dots, x_n)$ is called a $k\text{-}\prod \sum \prod$ polynomial, if each of its terms has at most k distinct variables.
- $p(x_1, \dots, x_n)$ is called a $\prod \sum \prod \times \prod \sum$ polynomial, if $p(x_1, \dots, x_n) = p_1 p_2$ such that p_1 is a $\prod \sum \prod$ polynomial and p_2 is a $\prod \sum$ polynomial. Similarly, $p(x_1, \dots, x_n)$ is called a $k\text{-}\prod \sum \prod \times \prod \sum$ polynomial, if $p(x_1, \dots, x_n) = p_1 p_2$ such that p_1 is a $k\text{-}\prod \sum \prod$ polynomial and p_2 is a $\prod \sum$ polynomial.

On the surface, a $\prod_m \sum_s \prod_t$ polynomial “resembles” a 3SAT (2SAT) formula, especially when $t = 1$. Likewise, a $\prod_m \sum_3 \prod_t (\prod_m \sum_2 \prod_t)$ polynomial “resembles” a 3SAT (2SAT) formula, especially when $t = 1$. However, negated variables are not involved in a polynomials. Furthermore, as pointed out in the previous section, is not easy, if not impossible, to have some easy algebra to deal with the properties of $x^2 = x$ and $x \cdot \bar{x} = 0$ in a field, especially when the field is larger than Z_2 . Also, as pointed out before, the arithmetization technique is Shamir is not applicable to this case.

7.2 3SAT and Related NP-Complete Problems

In this section, we give the definition of 3SAT problem and its restricted version, which is called (3, 3)-SAT.

Definition 13

- A 3SAT instance is a conjunctive form $C_1 \cdot C_2 \cdots C_m$ such that each C_i is a disjunction of at most three literals.
- 3SAT is the language of those 3SAT instances that have satisfiable assignments.
- A (3, 3)-SAT instance is an instance G for 3SAT such that for each variable x , the total number of times of x and \bar{x} in G is at most 3, and the total number of times of \bar{x} in G is at most 1.
- (3, 3)-SAT is the language of those (3, 3)-SAT instances that have satisfiable assignments.

It is well known 3SAT is NP-complete. For example, $(x_1 + x_2 + x_3)(x_1 + \bar{x}_2)$ ($\bar{x}_1 + x_2$) is both 3SAT and (3, 3)-SAT instance, and also belongs to both 3SAT and (3, 3)-SAT. On the other hand, $(x_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2)(\bar{x}_1 + x_2)$ is not a (3, 3)-SAT instance since \bar{x}_1 appears twice in the formula. The following lemma is similar to a result derived by Tovey [72].

Lemma 13 *There is a polynomial-time reduction from 3SAT to (3, 3)-SAT.*

Proof Let F be an instance for 3SAT. Let's focus on one variable x_i that appears m times in F . Introduce a series of new variables $y_{i,1}, \dots, y_{i,m}$ for x_i . Convert F to F' by changing the j -th occurrence of x_i in F to $y_{i,j}$ for $j = 1, \dots, m$. Define

$$\begin{aligned} G_{x_i} &= (x_i \rightarrow y_{i,1}) \cdot (y_{i,1} \rightarrow y_{i,2}) \cdot (y_{i,2} \rightarrow y_{i,3}) \cdot (y_{i,3} \rightarrow y_{i,4}) \cdots (y_{i,m-1} \rightarrow y_{i,m}) \cdot \\ &\quad (y_{i,m} \rightarrow x_i) \\ &= (\bar{x}_i + y_{i,1}) \cdot (\bar{y}_{i,1} + y_{i,2}) \cdot (\bar{y}_{i,2} + y_{i,3}) \cdot (\bar{y}_{i,3} + y_{i,4}) \cdots (\bar{y}_{i,m-1} + y_{i,m}) \cdot \\ &\quad (\bar{y}_{i,m} + x_i). \end{aligned}$$

Each logical formula $(x \rightarrow y)$ is equivalent to $(\bar{x} + y)$. If G_{x_i} is true, then $x_i, y_{i,1}, \dots, y_{i,m}$ are equivalent.

Convert F' into F'' such that $F'' = F'G_{x_1} \cdots G_{x_k}$, where x_1, \dots, x_k are all variables in F .

For each variable x in F'' with more than one \bar{x} , create a new variable y_x and replace each positive x of F by \bar{y}_x and each negative \bar{x} by y_x . Thus, F'' becomes F''' . It is easy to see that $F \in 3\text{SAT}$ iff F'' is satisfiable iff $F''' \in (3,3)\text{-SAT}$. \square

7.3 $\prod \sum \prod$ Polynomials

Given any $\prod_m \sum_s \prod_t$ polynomial $p(x_1, \dots, x_n) = p_1 \cdots p_m$, one can nondeterministically choose a term π_i from the clause p_i and then check whether $\pi_1 \cdots \pi_m$ is a multilinear monomial. So the problem of testing multilinear monomials in a $\prod \sum \prod$ polynomial is in NP. In the following, the reader will realize that this problem is also NP-hard.

Theorem 7 ([15]) *It is NP-hard to test whether a $2\text{-}\prod_m \sum_3 \prod_2$ polynomial has a multilinear monomial in its sum-product expansion.*

Proof By Lemma 13, $(3,3)\text{-SAT}$ is NP-complete. The $(3,3)\text{-SAT}$ problem can be reduced to the given problem. Let $f = f_1 \wedge \cdots \wedge f_m$ be a $(3,3)\text{-3SAT}$ instance. Every variable x_i in f appears at most three times, and if x_i appears three times, then x_i itself occurs twice and \bar{x}_i once.

Let x_i be any given variable in f ; new variables are introduced to replace it. If x_i appears only once, then the appearance of x_i (or \bar{x}_i) is replaced by a new variable y_{i1} . When x_i appears twice: if x_i occurs twice, then replace the first occurrence by a new variable y_{i1} and the second by y_{i2} . If both x_i and \bar{x}_i occur, then replace both occurrences by y_{i1} . When x_i occurs three times with x_i appearing twice and \bar{x}_i once, then replace the first x_i by y_{i1} and the second by y_{i2} , and replace \bar{x}_i by $y_{i1}y_{i2}$. This procedure of replacing all variables in f , negated or not, with new variables can be carried out easily in quadratic time.

Let $p = p_1 \cdots p_m$ be polynomial resulting from the above replacement process. Here, p_i corresponds to f_i with boolean literals being replaced. Clearly, p is a $2\text{-}\prod_m \sum_3 \prod_2$ polynomial.

Considering the sum-product expansion of $f = f_1 \cdots f_m$, it is easy to see that f is satisfiable iff its sum-product expansion has a product

$$\psi = \tilde{x}_{i_1} \cdots \tilde{x}_{i_m},$$

where the literal \tilde{x}_{i_j} is from the clause f_j and is either x_{i_j} or \bar{x}_{i_j} , $1 \leq j \leq m$. Furthermore, the negation of \tilde{x}_{i_j} must not occur in ψ .

Let $t(\tilde{x}_{i_j})$ denote the replacement of \tilde{x}_{i_j} by new variables $y_{i,j,1}$ and/or $y_{i,j,2}$ as described above to transform f to p . Then, $t(\tilde{x}_{i_j})$ is a term in the clause p_j . Hence,

$$t(\psi) = t(\tilde{x}_{i_1}) \cdots t(\tilde{x}_{i_m})$$

is a monomial in the sum-product expansion of p . Moreover, $t(\psi)$ is multilinear, because a variable and its negation cannot appear in ψ at the same time.

On the other hand, assume that

$$\pi = \pi_1 \cdots \pi_m$$

is a multilinear monomial in p with the term π_{i_j} in the clause p_j . Let $t^{-1}(\cdot)$ denote the reversal replacement of $t(\cdot)$. Then, by the procedure of the replacement above, $t^{-1}(\pi_{i_j})$ is a variable or the negation of a variable in f_j . Thus,

$$t^{-1}(\pi) = t^{-1}(\pi_1) \cdots t^{-1}(\pi_m)$$

is a product in the sum-product expansion of f . Since π is multilinear, a variable and its negation cannot appear in $t^{-1}(\pi)$ at the same time. This implies that f is satisfiable by an assignment of setting all the literals in $t^{-1}(\pi)$ true. \square

An example is given below to illustrate the variable replacement procedure given in the above proof. Given a 3SAT formula

$$f = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_4 \vee x_5),$$

the polynomial for f after variable replacement is

$$p(f) = (y_{11} + y_{21}y_{22} + y_{31})(y_{11}y_{12} + y_{21} + y_{41})(y_{12} + y_{22} + y_{31})(y_{42} + y_{51}).$$

The truth assignment satisfying f as determined by the product $x_3 \cdot \bar{x}_1 \cdot x_2 \cdot x_4$ is one-to-one correspondent to the multilinear monomial $y_{31} \cdot y_{11}y_{12} \cdot y_{22} \cdot y_{42}$ in $p(f)$.

Two corollaries follow immediately from this theorem.

Corollary 1 ([15]) *It is NP-hard to test whether a $\prod \sum \prod$ polynomial has multilinear monomials in its sum-product expansion.*

7.4 Complexity for Coefficient of Monomial

In this section, it is shown that computing the coefficient of a multilinear monomial in a formula is #P-hard.

Theorem 8 ([16]) *It is #P-hard that given a formula of homogenous polynomial, compute the coefficient of any multilinear monomial.*

Proof It is well known that computing the number of perfect matchings in a bipartite graph is $\#P$ -hard. The counting of perfect matching problem can be reduced to this problem. Assume that $G = (V_1 \cup V_2, E)$ is a bipartite graph. A formula is constructed below.

Assume that $V_1 = \{v_1, \dots, v_n\}$ and $V_2 = \{u_1, \dots, u_m\}$. For each vertex $v_i \in V_1$, let x_i represent it. For each vertex $u_i \in V_2$, let y_i represent it.

For each vertex $v_i \in V_1$, define the polynomial $F_i = \sum_{(v_i, u_j) \in E} x_i y_j$. The formula F_G is $F_1 \cdot F_2 \cdots \cdot F_n$. The number of perfect matchings in G is equal to the coefficient of the multilinear monomial $x_1 x_2 \cdots x_n y_1 y_2 \cdots y_n$ in the sum of product expansion of F_G . \square

8 Hardness of Approximation of Integration and Differentiation

Integration and differentiation are basic operations in classical mathematics. Valiant showed that computing the high-dimensional integration and differentiation are both $\#P$ -hard via reducing permanent problem to them [73]. Valiant's results do not imply any inapproximability of the two problems since permanent has a polynomial-time approximation scheme [45]. Using the complexity theory of monomial testing, Fu [32] showed some hardness results for high-dimensional integration and differentiation of multivariate polynomials. This is an application of the complexity theory of testing monomial.

Let $Z_{\geq 0} = \{0, 1, 2, \dots\}$ be the set of all natural numbers. Let $Z_+ = \{1, 2, \dots\}$ be the set of all positive natural numbers.

Assume that function $r(n)$ is from $Z_{\geq 0}$ to Z^+ . For a functor $F(\cdot)$ that converts a multivariate polynomial into a real number, an algorithm $A(\cdot)$ gives an $r(n)$ -factor approximation to $F(f)$ if it satisfies the following conditions: if $F(f) \geq 0$, then $\frac{F(f)}{r(n)} \leq A(f) \leq r(n)F(f)$, and if $F(f) < 0$, then $r(n)F(f) \leq A(f) \leq \frac{F(f)}{r(n)}$, where n is the length of f .

8.1 Inapproximation of Integration

In this section, we show that it is NP-hard to approximate high-dimensional integration of multivariate polynomials.

Lemma 14 is our main technical lemma. It is used to convert a $(3, 3)$ -SAT instance into a $\prod \sum \prod_2$ polynomial.

Lemma 14 Let $g_1(x) = 30x^2 - 36x + 9$, $g_2(x) = -6x + 4$ and $f(x) = 2x$. They satisfy the following conditions:

1. $\int_0^1 g_1(x) dx$, $\int_0^1 g_2(x) dx$, $\int_0^1 f(x) dx$, and $\int_0^1 g_1(x)g_2(x) dx$ are all positive integers,
2. $\int_0^1 g_1(x)f(x) dx$, $\int_0^1 g_2(x)f(x) dx$, and $\int_0^1 g_1(x)g_2(x)f(x) dx$ are all equal to 0.

Proof For a polynomial $h(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$, we can compute its integration as $\int_0^1 h(x) dx = \frac{a_n}{n+1} + \frac{a_{n-1}}{n} + \cdots + a_0$. It is straightforward to verify the lemma with the concrete expressions for the three functions $g_1(x)$, $g_2(x)$ and $f(x)$. \square

Lemma 15 *There is a polynomial-time algorithm h such that given a $(3, 3)$ -SAT instance $s(x_1, \dots, x_d)$, it produces a $\prod \sum \prod_2$ polynomial $h(s(x_1, \dots, x_d)) = q(y_1, \dots, y_d)$ to satisfy the following two conditions:*

1. *If $s(x_1, \dots, x_d)$ is satisfiable, then $\int_{[0,1]^d} q(y_1, \dots, y_d) dy_1 \cdots dy_d$ is a positive integer; and*
2. *If $s(x_1, \dots, x_d)$ is not satisfiable, then $\int_{[0,1]^d} q(y_1, \dots, y_d) dy_1 \cdots dy_d$ is zero.*

Proof Let polynomials $g_1(y)$, $g_2(y)$, and $f(y)$ be defined according to those in Lemma 14.

For a $(3, 3)$ -SAT problem $s(x_1, \dots, x_d)$, let $q(y_1, \dots, y_d)$ be defined as follows:

- For the first positive literal x_i in $s(x_1, \dots, x_d)$, replace it with $g_1(y_i)$.
- For the second positive literal x_i in $s(x_1, \dots, x_d)$, replace it with $g_2(y_i)$.
- For the negative literal \bar{x}_i in $s(x_1, \dots, x_d)$, replace it with $f(y_i)$.

The formula $s(x_1, \dots, x_d)$ has a sum of product form. It is satisfiable if and only if one term does not contain both positive and negative literals for the same variable. If a term contains both x_i and \bar{x}_i , the corresponding term in the sum of product for $q(\cdot)$ contains both $g_j(y_i)$ and $f(y_i)$ for some $j \in \{1, 2\}$. This makes it zero after integration by Lemma 14. Therefore, $s(x_1, \dots, x_d)$ is satisfiable if and only if $\int_{[0,1]^d} q(y_1, \dots, y_d) dy_1 \cdots dy_d$ is not zero. Furthermore, it is satisfiable and the integration is a positive integer by Lemma 14. The computing time of h is clearly polynomial since we convert s to $h(s)$ by replacing each literal by a single variable function of degree at most 2. \square

Theorem 9 ([32]) *Let $a(n)$ be an arbitrary function from $Z_{\geq 0}$ to Z_+ . Then there is no polynomial-time $a(n)$ -factor approximation for the integration of a $\prod \sum \prod_2$ polynomial $p(x_1, \dots, x_d)$ in the region $[0, 1]^d$ unless $P = NP$.*

Proof Assume that $A(\cdot)$ is a polynomial-time $a(n)$ -factor approximation for the integration $\int_{[0,1]^d} p(y_1, \dots, y_d) dy_1 \cdots dy_d$ with $\prod \sum \prod_2$ polynomial $p(y_1, \dots, y_d)$. For a $(3, 3)$ -SAT instance $s(x_1, \dots, x_d)$, let $p(y_1, \dots, y_d) = h(s(x_1, \dots, x_d))$ according to Lemma 15. By Lemma 15, a $(3, 3)$ -SAT instance $s(x_1, \dots, x_d)$ is satisfiable if and only if the integration $J = \int_{[0,1]^d} p(y_1, \dots, y_d) dy_1 \cdots dy_d$ is not zero. Assume that $s(x_1, \dots, x_d)$ is not satisfiable, then we have $A(J) \in [J/a(n), J \cdot a(n)] = [0, 0]$ that implies $A(J) = 0$. Assume that $s(x_1, \dots, x_d)$ is satisfiable, then we have $A(J) \in [J/a(n), J \cdot a(n)] \subseteq (0, +\infty)$ that implies $A(J) > 0$. Thus, $s(x_1, \dots, x_d)$ is satisfiable if and only if $A(J) > 0$.

Therefore, there is a polynomial-time algorithm for solving $(3, 3)$ -SAT that is NP-complete by Lemma 13. So, $P = NP$. \square

8.2 Inapproximation of Differentiation

In this section, we study the hardness of high-dimensional differentiation. We derive the inapproximation results under both $\text{NP} \neq \text{P}$.

Definition 14 A monomial is an expression $x_1^{a_1} \cdots x_d^{a_d}$, and its degree is $a_1 + \cdots + a_d$. A monomial $x_1^{a_1} \cdots x_d^{a_d}$, in which x_1, \dots, x_d are different variables, is a *multilinear* if $a_1 = a_2 = \cdots = a_d = 1$.

For example, $(x_1x_3 + x_2^2)(x_2x_4 + x_3^2)$ is a $\prod \sum \prod_2$ polynomial. It has a multilinear monomial $x_1x_2x_3x_4$ in its sum of products expansion.

We give [Lemma 16](#) to convert an instance f for $(3, 3)$ -SAT into a $\prod \sum \prod_2$ polynomial. Its proof is similar to that of [Theorem 7](#).

Lemma 16 *Let $a(1^n)$ be a polynomial-time computable function from $Z_{\geq 0}$ to Z_+ . Then there is a polynomial-time algorithm A such that given a $(3, 3)$ -SAT instance $F(y_1, \dots, y_d)$, the algorithm returns a $\prod \sum \prod_2$ polynomial $G(x_1, \dots, x_d)$ such that:*

1. *If F is not satisfiable, then G does not have a multilinear monomial with a nonzero coefficient in its sum of product expansion.*
2. *If F is satisfiable, then G has the multilinear monomial $x_1 \cdots x_d$ with a positive integer coefficient at least $3a(1^n)^2$ in its sum of product expansion.*

Proof Let $(3, 3)$ -SAT instance F be $C_1C_2 \cdots C_k$. Each clause C_i has format $y_{i_1}^* + y_{i_2}^* + y_{i_3}^*$, where literal y_j^* is either y_j or its negation \bar{y}_j . Since F is a $(3, 3)$ -SAT instance, for each variable y_i in F , y_i and \bar{y}_i totally appear at most three times in F , and \bar{y}_i appears at most once in F .

For each variable y_i in F , create four new variables $z_{i,1}, z_{i,2}, u_{i,1}$, and $u_{i,2}$. Convert formula F into polynomial G_1 such that for each y_i in F , the first positive occurrence y_i is changed into $z_{i,1}u_{i,1}$, the second positive occurrence y_i is changed into $z_{i,2}u_{i,2}$, and the negative occurrence \bar{y}_i is changed to $z_{i,1}z_{i,2}$. After the conversion for all the variables, formula F is transformed into a polynomial G_1 . We have that F is satisfiable if and only if G_1 has a multilinear monomial with positive coefficient in its sum of products expansion. This is because a multilinear monomial in the sum of product expansion of G_1 corresponds a consistent conjunctive term, which does not contain both y_i and its negation \bar{y}_i for some variable y_i , in the sum of product expansion of F .

Let H_1 be the set of all variables in G_1 . Assume that d_1 is the degree of G_1 (it is easy to see that all monomials in the sum-product expansion of G_1 have the same degree d_1). Let m be the number of variables in H_1 . Let d be the number of boolean variables in F . Assume that no clause C_i in F contains a single literal (otherwise, we can force the literal to be true to simplify F). The number of clauses in F is at most $\frac{3d}{2}$ since each variable appears in F at most three times and each clause C_i of F contains at least two literals. The degree G_1 is at most $3d$ since each literal

of F is replaced by a product of two variables. The number of variables in G_1 is $m = 4d$ (we create four new variables for each variable in F), which is larger than the degree of G_1 .

Create new variables v_1, \dots, v_{m-d_1} . For $j = 1, \dots, m-d_1$, let $q_j = \sum_{x \in H_1} xv_j$. Finally, we get the polynomial $G = 3a(1^n)^2 \cdot G_1 \cdot q_1 \cdots q_{m-d_1}$, where $n = m + (m - d_1)$. Note that $3a(1^n)^2$ in the polynomial G is considered an integer constant which does not contain any variable. The degree of G is $n = d_1 + 2(m - d_1) = m + (m - d_1)$. Thus, the degree of G is the same as the total number of variables in g . We can show that f is satisfiable if and only if there is a multilinear monomial, which is the product of all variables in G , with positive coefficient of size at least $3a(1^n)^2$. \square

Theorem 10 ([32]) Assume that $r(n)$ is a function from $Z_{\geq 0}$ to Z_+ . If there is a polynomial-time algorithm A such that given a $\prod \sum \prod_2$ polynomial $g(x_1, \dots, x_d)$, it gives an $r(n)$ -factor approximation to $\frac{\partial g^{(n)}(x_1, \dots, x_d)}{\partial x_1 \cdots \partial x_d}$ at the origin point $(x_1, \dots, x_d) = (0, \dots, 0)$, then $P = NP$.

Proof Assume that $A(\cdot)$ is a polynomial-time $r(n)$ -approximation for computing $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \cdots \partial x_n}$ at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$.

Assume that f is an arbitrary formula in a $(3, 3)$ -SAT problem. By Lemma 16, we can get a polynomial $g(x_1, \dots, x_n)$. The derivative $\frac{\partial g^{(n)}(0, \dots, 0)}{\partial x_1 \cdots \partial x_n}$ is equal to the coefficient of $x_1 \cdots x_n$ in the sum of product expansion of g .

If f is satisfiable, we have $A(g) > 0$, and if f is not satisfiable, we have $A(g) = 0$ since $A(\cdot)$ is a $r(n)$ -approximation and $r(n) \geq 1$. We can know if the coefficient of $x_1 \cdots x_n$ in the sum of product expansion of g is positive in polynomial time. Thus, $(3, 3)$ -SAT is solvable in polynomial time. Since $(3, 3)$ -SAT is NP-complete, we have $P = NP$. \square

9 Conclusion

The combination of randomization and polynomial brings efficient algorithm for the classical problems k -path and Hamiltonian path. The polynomial method has applications in the areas of wireless networking and bioinformatics. It can be seen that testing the existence of multilinear monomials in the sum-product expansion of a polynomial is NP-complete and computing the coefficient of a multilinear monomial in the sum-product expansion is #P-hard. These results shows the polynomial methods links to the central areas of computational complexity theory. This chapter explains some methods that are related to algorithm and computational complexity. It is expected to see the further development of the theory in the future.

Acknowledgements The authors would like to thank Dr. Dingzhu Du for his encouragements when writing this book chapter. We would also like to thank Zhixiang Chen, Yang Liu, and Robert Schweller for a wonderful experience of collaboration working in this area. Finally, we would like

to thank the reviewers for their helpful comments for an earlier version of this chapter. Bin Fu is supported in part by the National Science Foundation Early Career Award CCF-0845376.

Recommended Reading

1. S. Acharya, R. Alonso, M. Franklin, S. Zdonik, Broadcast disks: data management for asymmetric communication environments, in *The 1995 ACM SIGMOD/PODS Conference*, San Jose, CA, 1995, pp. 199–210
2. N. Alon, R. Yuster, U. Zwick, Color-coding. *J. ACM* **42**, 844–856 (1995)
3. D.L. Applegate, R.E. Bixby, V. Chvatal, W.J. Cook, *The Traveling Salesman Problem: A Computational Study* (Princeton University Press, Princeton, 2007)
4. S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and the hardness of approximation problems. *J. ACM* **45**(3), 501555 (1998)
5. E.T. Bax, Inclusion and exclusion algorithm for the Hamiltonian Path Problem. *Inf. Process. Lett.* **47**(4), 203–207 (1993)
6. R. Beigel, The polynomial method in circuit complexity, in *Proceedings of the Eighth Conference on Structure in Complexity Theory*, San Diego, CA, 1993, pp. 82–95
7. R. Bellman, Combinatorial processes and dynamic programming, in *Proceedings of Symposia in Applied Mathematics 10*, American Mathematical Society, Columbia University, 24–26 Apr 1960, pp. 217–249. Providence (R.I.)
8. R. Bellman, Dynamic programming treatment of the travelling salesman problem. *J. ACM* **9**(1), 61–63 (1962)
9. A. Björklund, Determinant sums for undirected hamiltonicity, in *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, FOCS*, Las Vegas, NV, 2010
10. A. Björklund, Exact covers via determinants, in *Symposium on Theoretical Aspects of Computer Science 2010*, Nancy, France, 2010, pp. 95–106
11. A. Björklund, P. Kaski, T. Husfeldt, Fourier meets Möbius: fast subset convolution, in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, San Diego, CA, 2007
12. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, The travelling salesman problem in bounded degree graphs, in *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, ICALP*, Reykjavik, Iceland, 2008
13. H. Broersma, F.V. Fomin, P. Hof, D. Paulusma, Fast exact algorithms for Hamiltonicity in claw-free graphs, in *Graph-Theoretic Concepts in Computer Science*, Montpellier, France, vol. 5911/2010, 2010, pp. 44–53
14. J.R. Bunch, J.E. Hopcroft, Triangular factorization and inversion by fast matrix multiplication. *Math. Comput.* **28**, 231–236 (1974)
15. Z. Chen, B. Fu, The complexity of testing monomials in multivariate polynomials, in *Proceedings of the 5th International Conference on Combinatorial Optimization and Applications*, Zhangjiajie, China, 4–6 Aug 2011. Lecture notes in computer science, vol. 6831, pp. 1–15
16. Z. Chen, B. Fu, Approximating multilinear monomial coefficients and maximum multilinear monomials in multivariate polynomials. *J. Comb. Optim.* **25**(2), 234–254 (2013)
17. Z. Chen, M. Kao, Reducing randomness via irrational numbers. *SIAM J. Comput.* **29**(4), 1247–1256 (2000)
18. X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi, T. Jiang, Computing the assignment of orthologous genes via genome rearrangement, in *Proceedings of the 3rd Asia-Pacific Bioinformatics Conference (APBC'05)*, Singapore, 2005, pp. 363–378
19. J. Chen, S. Lu, S. Sze, F. Zhang, Improved algorithms for path, matching, and packing problems, in *Proceedings of ACM-SIAM Symposium on Discrete Algorithms (SODA)*, New Orleans, LA, 2007, pp. 298–307

20. Z. Chen, B. Fu, Y. Liu, R. Schwellner, Algorithms for testing monomials in multivariate polynomials, in *Proceedings of the 5th International Conference on Combinatorial Optimization and Applications*, Zhangjiajie, China. Lecture Notes in Computer Science, vol. 6831, pp. 16–30 (2011)
21. M. Chrobak, P. Kolman, J. Sgall, The greedy algorithm for the minimum common string partition problem, in *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'04)*, Cambridge, MA. Lecture Notes in Computer Science, vol. 3122, 2004, pp. 84–95
22. D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, *J. Symb. Comput.* **9**, 251–280 (1990)
23. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd edn. (MIT, Cambridge, 2009)
24. P. Damaschke, Minimum common string partition parameterized, in *Proceedings of the 8th Workshop on Algorithms in Bioinformatics (WABI'08)*, Karlsruhe, Germany. Lecture Notes in Computer Science, vol. 5251, 2008, pp. 87–98
25. W.E. Deskins, *Abstract Algebra*, Rep Una edn. (Dover Publications, New York, 1996)
26. Liang Ding, *Intractability of Integration and Derivative for Multivariate Polynomial and Trigonometric Function*. Master thesis, Advisor Bin Fu, University of Texas-Pan American, Aug 2012
27. B. Fu, C. Huang, Z. Lu, W. Wu, K. Yang, W. Lee, Minimizing the access time of multi-item requests in wireless data broadcast environments. (Submitted for Conference Publication)
28. D. Eppstein, The traveling salesman problem for cubic graphs. *J. Graph Algorithms Appl.* **11**(1), 61–81 (2007)
29. U. Feige, S. Goldwasser, L. Lovasz, S. Safra, M. Szegedy, Interactive proofs and the hardness of approximating cliques. *J. ACM* **43**(2), 268–292 (1996)
30. K. Foltz, L. Xu, J. Bruck, Scheduling for efficient data broadcast over two channels, in *Proceedings of 2004 IEEE International Symposium on Information Theory*, Chicago, IL, 2004, pp. 113–116
31. B. Fu, Separating PH from PP by relativization. *Acta Math. Sin.* **8**(3), 329–336 (1992)
32. B. Fu, Multivariate polynomial integration and derivative are polynomial time inapproximable unless $P = NP$, in *Proceedings of the Joint International Conference on Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*, Beijing, China, 2012, pp. 182–191
33. B. Fu, H. Jiang, B. Yang, B. Zhu, Exponential and polynomial time algorithms for the minimum common string partition problem. In *Proceedings of the 5th International Conference on Combinatorial Optimization and Applications*, COCOA 2011, Zhangjiajie, China. Lecture notes in computer science, vol. 6831, 2011, pp. 299–310
34. X. Gao, Z. Lu, W. Wu, B. Fu, Algebraic algorithm for scheduling data retrieval in multi-channel wireless data broadcast environments, in *Proceedings of Combinatorial Optimization and Applications - 5th International Conference, COCOA 2011*, Zhangjiajie, China, 4–6 Aug 2011
35. H. Gebauer, On the number of Hamilton cycles in bounded degree graphs, in *Proceedings of the Fourth Workshop on Analytic Algorithms and Combinatorics, ANALCO*, 2008 (SIAM, Philadelphia, 2008)
36. I. Goldstein, M. Lewenstein, Quick greedy computation for minimum common string partitions, in *Proceedings of the 22nd Annual Symposium on Combinatorial Pattern Matching (CPM'11)*, Palermo, Italy, 2011. To appear
37. A. Goldstein, P. Kolman, J. Zheng, Minimum common string partitioning problem: hardness and approximations, in *Proceedings of the 15th International Symposium on Algorithms and Computation (ISAAC'04)*, Hong Kong, China. Lecture Notes in Computer Science, vol. 3341, 2004, pp. 473–484. Also in: *The Electronic Journal of Combinatorics* 12 (2005), paper R50
38. M. Held, R.M. Karp, A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.* **10**(1), 196–210 (1962)

39. J.L. Huang, M.S. Chen, Broadcast program generation for unordered queries with data replication, in *Proceedings of the 8th ACM Symposium on Applied Computing*, Melbourne, FL, 2003, pp. 866–870
40. J.L. Huang, M.S. Chen, Dependent data broadcasting for unordered queries in a multiple channel mobile environment. *IEEE Trans. Knowl. Data Eng.* **16**, 1143–1156 (2004)
41. J.L. Huang, M.S. Chen, W.C. Peng, Broadcasting dependent data for ordered queries with data replication in a multi-channel mobile environment, in *Proceedings of the 19th IEEE International Conference on Data Engineering*, Bangalore, India, 2003, pp. 692–694
42. A.R. Hurson, A.M. Munoz-Avila, N. Orchowski, B. Shirazi, Y. Jiao, Power aware data retrieval protocols for indexed broadcast parallel channels. *Previous Mobile Comput.* **2**(1), 85–107 (2006)
43. K. Iwama, T. Nakashima, An improved exact algorithm for cubic graph TSP. *Computing and Combinatorics*. Lecture Notes in Computer Science, vol. 4598 (Springer, New York/Berlin/Heidelberg 2007), pp. 108–117
44. N. Jacobson, *Basic Algebra I*, 2nd edn. (Dover Publications, Mineola, 2009)
45. M. Jerrum, A. Sinclair, E. Vigoda, A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM* **51**(4), 671–697 (2004)
46. T. Jiang, W. Xiang, H.H. Chen, Q. Ni, Multicast broadcast services support in OFDMA-based WiMAX systems. *IEEE Commun. Mag.* **45**(8), 78–86 (2007)
47. H. Jiang, B. Zhu, D. Zhum H. Zhu, Minimum common string partition revisited. *J. Comb. Optim.* **23**, 519–527 (2010). doi:10.1007/s10878-010-9370-2
48. J. Juran, A.R. Hurson, N. VijayKrishnan, S. Kim, Data organization and retrieval on parallel air channels: performance and energy issues. *Wirel. Netw.* **10**(2), 183–195 (2004)
49. V. Kabanets, R. Impagliazzo, Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.* **13**(1-2), 1–46 (2004)
50. H. Kaplan, N. Shafir, The greedy algorithm for edit distance with moves. *Inf. Process. Lett.* **97**(1), 23–27 (2006)
51. R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations* (Plenum, New York, 1972), pp. 85–103
52. R.M. Karp, Dynamic programming meets the principle of inclusion and exclusion. *Oper. Res. Lett.* **1**(2), 49–51 (1982)
53. B.P. Kelley, R. Sharan, R.M. Karp, T. Sittler, D.E. Root, B.R. Stockwell, T. Ideker, Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proc. Natl. Acad. Sci. USA* **100**, 11394–11399 (2003)
54. R. Kennes, Computational aspects of the Möbius transform of a graph. *IEEE Trans. Syst. Man Cybern.* **22**, 201–223 (1991)
55. A. Klivans, R.A. Servedio, Learning DNF in time $2^{O(n^{1/3})}$. *J. Comput. Syst. Sci. - STOC 2001*. **68**(2), 303–318 (2004)
56. J. Kneis, D. Molle, S. Richter, P. Rossmanith, Divide-and-color, in *Proceedings of the International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, Bergen, Norway. Lecture Notes in Computer Science, vol. 4271 (Springer, 2006), pp. 58–67
57. S. Kohn, A. Gottlieb, M. Kohn, A generating function approach to the traveling salesman problem, in *ACM Annual Conference*, Seattle, WA (ACM, 1977), pp. 294–300
58. P. Kolman, Approximating reversal distance for strings with bounded number of duplicates, in *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS'05)*, Gdansk, Poland. Lecture Notes in Computer Science, vol. 3618, 2005, pp. 580–590
59. P. Kolman, T. Walen, Reversal distance for strings with duplicates: linear time approximation using hitting set, in *Proceedings 4th Workshop on Approximation and Online Algorithms (WAOA'06)*, Zurich, Switzerland. Lecture Notes in Computer Science, vol. 4368, 2006, pp. 279–289
60. P. Kolman, T. Walen, Approximating reversal distance for strings with bounded number of duplicates. *Discret. Appl. Math.* **155**(3), 327–336 (2007)

61. I. Koutis, Faster algebraic algorithms for path and packing problems, in *Automata, Languages and Programming*. Lecture Notes in Computer Science, vol. 5125/2008 (Springer, Berlin/Heidelberg, 2008), pp. 575–586
62. G. Lee, M.S. Yeh, S.C. Lo, A. Chen, A strategy for efficient access of multiple data items in mobile environment, in *Proceedings of the 3rd International Conference on Mobile Data Management*, Singapore, 2002, pp. 71–78
63. A. Manindra, K. Neeraj, S. Nitin, PRIMES is in P. *Ann. Math.* **160**(2), 781–793 (2004)
64. M. Minsky, S. Papert, *Perceptrons* (expanded edition 1988) (MIT Press, Cambridge, MA, 1968)
65. R. Motwani, P. Raghavan, *Randomized Algorithms* (Cambridge University Press, Cambridge/New York, 1995)
66. C. Papadimitriou, M. Yannakakis, On limited nondeterminism and the complexity of the V-C dimension. *J. Comput. Syst. Sci.* **53**, 161–170 (1996)
67. J.T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities. *J. ACM* **27**(4), 701–717 (1980)
68. J. Scott, T. Ideker, R.M. Karp, R. Sharan, Efficient algorithms for detecting signaling pathways in protein interaction networks. *J. Comput. Biol.* **13**, 133–144 (2006)
69. A. Shamir, IP = PSPACE. *J. ACM* **39**(4), 869–877 (1992)
70. D. Shapira, J. Storer, Edit distance with move operations, in *Proceedings of the 13th Annual Symposium on Combinatorial Pattern Matching (CPM'02)(SODA'02)*, Fukuoka, Japan, 2002, pp. 667–676
71. Y. Shi, X. Gao, J. Zhong, W. Wu, Efficient parallel data retrieval protocols with MIMO antennae for data broadcast in 4G wireless communications, in *Proceedings of the 21st International Conference on Database and Expert Systems Applications, DEXA (2)*, Bilbao, Spain, 2010, pp. 80–95
72. C.A. Tovey, A simplified satisfiability problem. *Discret. Appl. Math.* **8**, 85–89 (1984)
73. L. Valiant, Completeness classes in algebra, in *Proceedings of the eleventh Annual ACM Symposium on Theory of Computing*, Atlanta, USA, 1979, pp. 249–261
74. R. Williams, Finding paths of length k in $O * (2^k)$ time. *Inf. Process. Lett.* **109**(6), 315–318 (2009)
75. G.J. Woeginger, Exact algorithms for NP-hard problems: a survey, in *Combinatorial Optimization Eureka, You Shrink!* Lecture Notes in Computer Science, vol. 2570 (Springer, Berlin, 2003), pp. 185–207
76. F. Yates, The design and analysis of factorial experiments, Technical Communication No.35, Commonwealth Bureau of Soil Science, Harpenden, UK, 1937
77. R.E. Zippel, Probabilistic algorithms for sparse polynomials, *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, Marseille, France, vol. 72, 1979, pp. 216–226

Algorithmic Aspects of Domination in Graphs

Gerard Jennhwa Chang

Contents

1	Introduction	222
2	Definitions and Notation	224
2.1	Graph Terminology	224
2.2	Variations of Domination	227
2.3	Special Classes of Graphs	230
3	Trees	232
3.1	Basic Properties of Trees	232
3.2	Labeling Algorithm for Trees	232
3.3	Dynamic Programming for Trees	237
3.4	Primal–Dual Approach for Trees	240
3.5	Power Domination in Trees	242
3.6	Tree Related Classes	244
4	Interval Graphs	245
4.1	Interval Orderings of Interval Graphs	245
4.2	Domatic Numbers of Interval Graphs	246
4.3	Weighted Independent Domination in Interval Graphs	248
4.4	Weighted Domination in Interval Graphs	249
4.5	Weighted Total Domination in Interval Graphs	250
4.6	Weighted Connected Domination in Interval Graphs	250
5	Chordal Graphs and NP-Completeness Results	251
5.1	Perfect Elimination Orderings of Chordal Graphs	251
5.2	NP-Completeness for Domination	252
5.3	Independent Domination in Chordal Graphs	256
6	Strongly Chordal Graphs	261
6.1	Strong Elimination Orderings of Strongly Chordal Graphs	261
6.2	Weighted Domination in Strongly Chordal Graphs	262
6.3	Domatic Partition in Strongly Chordal Graphs	264
7	Permutation Graphs	266
8	Cocomparability Graphs	268

G.J. Chang

Department of Mathematics, National Taiwan University, Taipei, Taiwan

e-mail: gjchang@math.ntu.edu.tw

9 Distance-Hereditary Graphs.....	271
9.1 Hangings of Distance-Hereditary Graphs.....	271
9.2 Weighted Connected Domination.....	272
Cross-References.....	274
Recommended Reading.....	275

Abstract

Domination in graph theory has many applications in the real world such as location problems. A dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D . The domination problem is to determine the domination number $\gamma(G)$ of a graph G that is the minimum size of a dominating set of G . Although many theoretic theorems for domination and its variations have been established for a long time, the first algorithmic result on this topic was given by Cockayne, Goodman, and Hedetniemi in 1975. They gave a linear-time algorithm for the domination problem in trees by using a labeling method. On the other hand, at about the same time, Garey and John constructed the first (unpublished) proof that the domination problem is NP-complete. Since then, many algorithmic results are studied for variations of the domination problem in different classes of graphs. This chapter is to survey the development on this line during the past 36 years. Polynomial-time algorithms using labeling method, dynamic programming method, and primal-dual method are surveyed on trees, interval graphs, strongly chordal graphs, permutation graphs, cocomparability graphs, and distance-hereditary graphs. NP-completeness results on domination are also discussed.

1 Introduction

Graph theory was founded by Euler [88] in 1736 as a generalization of the solution to the famous problem of the Königsberg bridges. From 1736 to 1936, the same concept as graph, but under different names, was used in various scientific fields as models of real-world problems; see the historic book by Biggs et al. [22]. This chapter intends to survey the domination problem in graph theory from an algorithmic point of view.

Domination in graph theory is a natural model for many location problems in operations research. As an example, consider the following fire station problem. Suppose a county has decided to build some fire stations, which must serve all of the towns in the county. The fire stations are to be located in some towns so that every town either has a fire station or is a neighbor of a town which has a fire station. To save money, the county wants to build the minimum number of fire stations satisfying the above requirements.

Domination has many other applications in the real world. The recent book by Haynes et al. [112] illustrates many interesting examples, including dominating

Fig. 1 King domination on a 3×5 Chessboard (a) Chessboard (b) Chinese chessboard

a				
	K			K

b		K		
K				K
		K		

queens, sets of representatives, school bus routing, computer communication networks, (r, d) -configurations, radio stations, social network theory, land surveying, and kernels of games.

Among them, the classical problems of covering chessboards by the minimum number of chess pieces are important in stimulating the study of domination, which commenced in the early 1970s. These problems certainly date back to De Jaenisch [83] and have been mentioned in the literature frequently since that time.

A simple example is to determine the minimum number of kings dominating the entire chessboard. The answer to an $m \times n$ chessboard is $\lceil \frac{m}{3} \rceil \lceil \frac{n}{3} \rceil$. In the Chinese chess game, a king only dominates the four neighbor cells which have common sides with the cell the king lies in. In this case, the Chinese king domination problem for an $m \times n$ chessboard is harder. Figure 1 shows optimal solutions to both cases for a 3×5 board.

The above problems can be abstracted into the concept of domination in terms of graphs as follows. A *dominating set* of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D . The *domination number* $\gamma(G)$ of a graph G is the minimum size of a dominating set of G .

For the fire station problem, consider the graph G having all towns of the county as its vertices and a town is adjacent to its neighbor towns. The fire station problem is just the domination problem, as $\gamma(G)$ is the minimum number of fire stations needed.

For the king domination problem on an $m \times n$ chessboard, consider the king's graph G whose vertices correspond to the mn squares in the chessboard, and two vertices are adjacent if and only if their corresponding squares have a common point. For the Chinese king domination problem, the vertex set is the same, but two vertices are adjacent if and only if their corresponding squares have a common side. Figure 2 shows the corresponding graphs for the king and the Chinese king domination problems on a 3×5 chessboard. The king domination problem is just the domination problem, as $\gamma(G)$ is the minimum number of kings needed. Black vertices in the graph form a minimum dominating set.

Although many theoretic theorems for the domination problem have been established for a long time, the first algorithmic result on this topic was given by Cockayne et al. [56] in 1975. They gave a linear-time algorithm for the domination problem in trees by using a labeling method. On the other hand, at about the same time, Garey and Johnson (see [99]) constructed the first (unpublished) proof that the domination problem is NP-complete for general graphs. Since then, many

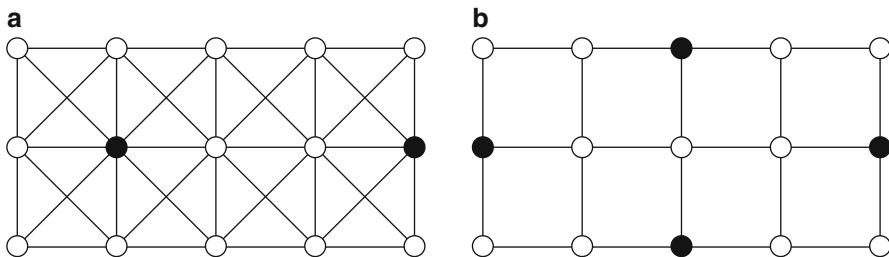


Fig. 2 King's graphs for the chess and the Chinese chess **(a)** For chess; **(b)** For Chinese chess

algorithmic results are studied for variations of the domination problem in different classes of graphs. The purpose of this chapter is to survey these results.

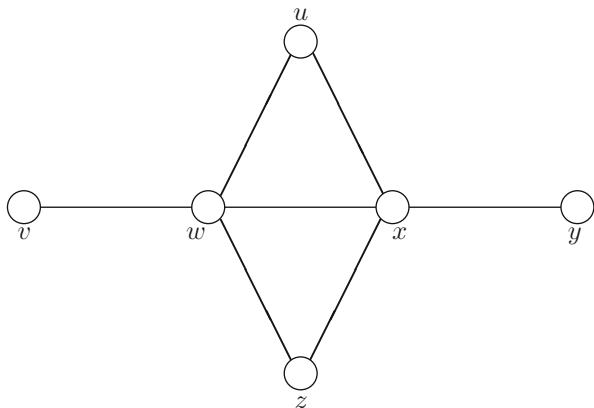
This chapter is organized as follows. [Section 2](#) gives basic definitions and notation. In particular, the classes of graphs surveyed in this chapter are introduced. Among them, trees and interval graphs are two basic classes in the study of domination. While a tree can be viewed as many paths starting from a center with different branches, an interval graph is a “thick path” in the sense that a vertex of a path is replaced by a group of vertices tied together. [Section 3](#) investigates different approaches for domination in trees, including labeling method, dynamic programming method, and primal–dual approach. These techniques are used not only for trees but also for many other classes of graphs in the study of domination as well as many other optimization problems. [Section 4](#) is for domination in interval graphs. It is in general not clear to which classes of graphs the results in trees and interval graphs can be extended. For some classes of graphs, the domination problem becomes NP-complete, while for some classes it is polynomially solvable. [Section 5](#) surveys NP-completeness results on domination, for which chordal graphs play an important role. The remaining sections are for classes of graphs in which the domination problem is solvable, including strongly chordal graphs, permutation graphs, cocomparability graphs, and distance-hereditary graphs.

2 Definitions and Notation

2.1 Graph Terminology

A *graph* is an ordered pair $G = (V, E)$ consisting of a finite nonempty set V of *vertices* and a set E of 2-subsets of V , whose elements are called *edges*. Sometimes $V(G)$ is used for the vertex set and $E(G)$ for the edge set of a graph G . A graph is *trivial* if it contains only one vertex. For any edge $e = \{u, v\}$, it is said that vertices u and v are *adjacent* and that vertex u (respectively, v) and edge e are *incident*. Two distinct edges are *adjacent* if they contain a common vertex. It is convenient to henceforth denote an edge by uv rather than $\{u, v\}$. Notice that uv and vu represent the same edge in a graph.

Fig. 3 A graph G with six vertices and seven edges



Two graphs $G = (V, E)$ and $H = (U, F)$ are *isomorphic* if there exists a bijection f from V to U such that $uv \in E$ if and only if $f(u)f(v) \in F$. Two isomorphic graphs are essentially the same as one can be obtained from the other by renaming vertices.

It is often useful to express a graph G diagrammatically. To do this, each vertex is represented by a point (or a small circle) in the plane and each edge by a curve joining the points (or small circles) corresponding to the two vertices incident to the edge. It is convenient to refer to such diagram of a graph as the graph itself. In Fig. 3, a graph G with vertex set $V = \{u, v, w, x, y, z\}$ and edge set $E = \{uw, ux, vw, wx, xy, wz, xz\}$ is shown.

Suppose A and B are two sets of vertices. The *neighborhood* $N_A(B)$ of B in A is the set of vertices in A that are adjacent to some vertex in B , that is,

$$N_A(B) = \{u \in A : uv \in E \text{ for some } v \in B\}.$$

The *closed neighborhood* $N_A[B]$ of B in A is $N_A(B) \cup B$. For simplicity, $N_A(v)$ stands for $N_A(\{v\})$, $N_A[v]$ for $N_A[\{v\}]$, $N(B)$ for $N_V(B)$, $N[B]$ for $N_V[B]$, $N(v)$ for $N_V(\{v\})$, and $N[v]$ for $N_V[\{v\}]$. The notion $u \sim v$ stands for $u \in N[v]$.

The *degree* $\deg(v)$ of a vertex v is the size of $N(v)$ or, equivalently, the number of edges incident to v . An *isolated vertex* is a vertex of degree zero. A *leaf* (or *end vertex*) is a vertex of degree one. The minimum degree of a graph G is denoted by $\delta(G)$ and the maximum degree by $\Delta(G)$. A graph G is r -regular if $\delta(G) = \Delta(G) = r$. A 3-regular graph is also called a *cubic graph*.

A graph $G' = (V', E')$ is a *subgraph* of another graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. In the case of $V' = V$, G' is called a *spanning subgraph* of G . For any nonempty subset S of V , the (*vertex*) *induced subgraph* $G[S]$ is the graph with vertex set S and edge set

$$E[S] = \{uv \in E : u \in S \text{ and } v \in S\}.$$

A graph is *H-free* if it does not contain H as an induced subgraph. The *deletion* of S from $G = (V, E)$, denoted by $G - S$, is the graph $G[V \setminus S]$. $G - v$ is a short notation for $G - \{v\}$ when v is a vertex in G . The *deletion* of a subset F of E from $G = (V, E)$ is the graph $G - F = (V, E \setminus F)$. $G - e$ is a short notation for $G - \{e\}$ if e is an edge of G . The *complement* of a graph $G = (V, E)$ is the graph $\overline{G} = (V, \overline{E})$, where

$$\overline{E} = \{uv \notin E : u, v \in V \text{ and } u \neq v\}.$$

Suppose $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are two graphs with $V_1 \cap V_2 = \emptyset$. The *union* of G_1 and G_2 is the graph $G_1 \cup G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. The *join* of G_1 and G_2 is the graph $G_1 + G_2 = (V_1 \cup V_2, E_+)$, where

$$E_+ = E_1 \cup E_2 \cup \{uv : u \in V_1 \text{ and } v \in V_2\}.$$

The *Cartesian product* of G_1 and G_2 is the graph $G_1 \square G_2 = (V_1 \times V_2, E_\square)$, where

$$V_1 \times V_2 = \{(v_1, v_2) : v_1 \in V_1 \text{ and } v_2 \in V_2\},$$

$$E_\square = \{(u_1, u_2)(v_1, v_2) : (u_1 = v_1, u_2 v_2 \in E_2) \text{ or } (u_1 v_1 \in E_1, u_2 = v_2)\}.$$

In a graph $G = (V, E)$, a *clique* is a set of pairwise adjacent vertices in V . An *i-clique* is a clique of size i . A 2-clique is just an edge. A 3-clique is called a *triangle*. A *stable* (or *independent*) set is a set of pairwise nonadjacent vertices in V .

For two vertices x and y of a graph, an *x-y walk* is a sequence $x = x_0, x_1, \dots, x_n = y$ such that $x_{i-1}x_i \in E$ for $1 \leq i \leq n$, where n is called the *length* of the walk. In a walk x_0, x_1, \dots, x_n , a *chord* is an edge $x_i x_j$ with $|i - j| \geq 2$. A *trail* (*path*) is a walk in which all edges (vertices) are distinct. A *cycle* is an $x-x$ walk in which all vertices are distinct except the first vertex is equal to the last. A graph is *acyclic* if it does not contain any cycle.

A graph is *connected* if for any two vertices x and y , there exists an $x-y$ walk. A graph is *disconnected* if it is not connected. A (*connected*) *component* of a graph is a maximal subgraph which is connected. A *cut-vertex* is a vertex whose deletion from the graph results in a disconnected graph. A *block* of a graph is a maximal connected graph which has no cut-vertices.

The *distance* $d(x, y)$ from a vertex x to another vertex y is the minimum length of an $x-y$ path; and $d(x, y) = \infty$ when there is no $x-y$ path.

Digraphs or *directed graphs* can be defined similar to graphs except that an edge (u, v) is now an ordered pair rather than a 2-subset. All terms in graphs can be defined for digraphs with suitable modifications by taking into account the directions of edges.

An *orientation* of a graph $G = (V, E)$ is a digraph (V, E') such that for each edge $\{u, v\}$ of E , exactly one of (u, v) and (v, u) is in E' and the edges in E' all come from this way.

2.2 Variations of Domination

Due to different requirements in the applications, people have studied many variations of the domination problem. For instance, in the queen's domination problem, one may ask the additional property that two queens do not dominate each other or any two queens must dominate each other. The following are most commonly studied variants of domination.

Recall that a dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one vertex in D . This is equivalent to that $N[x] \cap D \neq \emptyset$ for all $x \in V$ or $\cup_{y \in D} N[y] = V$.

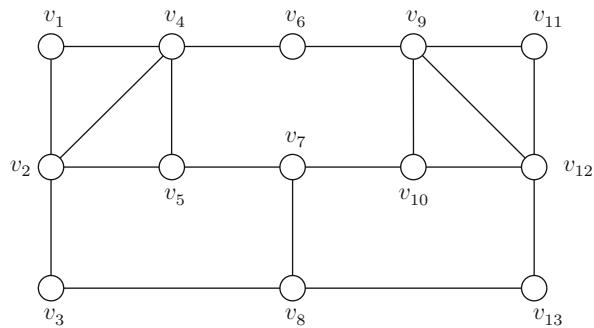
A dominating set D of a graph $G = (V, E)$ is *independent*, *connected*, *total*, or *perfect* (*efficient*) if $G[D]$ has no edge, $G[D]$ is connected, $G[D]$ has no isolated vertex, or $|N[v] \cap D| = 1$ for any $v \in V \setminus D$. An *independent* (respectively, *connected* or *total*) *perfect dominating set* is a perfect dominating set which is also independent (respectively, connected or total). A *dominating clique* (respectively, *cycle*) is a dominating set which is also a clique (respectively, cycle). For a fixed positive integer k , a k -*dominating set* of G is a subset D of V such that for every vertex v in V , there exists some vertex u in D with $d(u, v) \leq k$. An *edge dominating set* of $G = (V, E)$ is a subset F of E such that every edge in $E \setminus F$ is adjacent to some edge in F . For the above variations of domination, the corresponding *independent*, *connected*, *total*, *perfect*, *independent perfect*, *connected perfect*, *total perfect*, *clique*, *cycle*, and k - and *edge domination numbers* are denoted by $\gamma_i(G)$, $\gamma_c(G)$, $\gamma_t(G)$, $\gamma_{\text{per}}(G)$, $\gamma_{\text{iper}}(G)$, $\gamma_{\text{oper}}(G)$, $\gamma_{\text{tp}}(G)$, $\gamma_{\text{cl}}(G)$, $\gamma_{\text{cy}}(G)$, $\gamma_k(G)$, and $\gamma_e(G)$, respectively.

A dominating set D of a graph $G = (V, E)$ corresponds to a *dominating function* which is a function $f : V \rightarrow \{0, 1\}$ such that $\sum_{u \in N[v]} f(u) \geq 1$ for any $v \in V$. The *weight* of f is $w(f) = \sum_{v \in V} f(v)$. Then $\gamma(G)$ is equal to the minimum weight of a dominating function of G . Several variations of domination are defined in terms of functions as follows. A *signed dominating function* is a function $f : V \rightarrow \{+1, -1\}$ such that $\sum_{u \in N[v]} f(u) \geq 1$. The *signed domination number* $\gamma_s(G)$ of G is the minimum weight of a signed dominating function. A *Roman dominating function* is a function $f : V \rightarrow \{0, 1, 2\}$ such that any vertex v with $f(v) = 0$ is adjacent to some vertex u with $f(u) = 2$. The *Roman domination number* $\gamma_{\text{Rom}}(G)$ of G is the minimum weight of a Roman dominating function.

A set-valued variation of domination is as follows. For a fixed positive integer k , a k -*rainbow dominating function* is a function $f : V \rightarrow 2^{\{1, 2, \dots, k\}}$ such that $f(v) = \emptyset$ implies $\cup_{u \in N(v)} f(u) = \{1, 2, \dots, k\}$. The *weight* of f is $w(f) = \sum_{v \in V} |f(v)|$. The k -*rainbow domination number* $\gamma_{\text{krain}}(G)$ of G is the minimum weight of a k -rainbow dominating function. Notice that 1-rainbow domination is the same as the ordinary domination.

A quite different variation motivated from the power system monitoring is as follows. For a positive integer k , suppose D is a vertex subset of a graph $G = (V, E)$. The following two observation rules are applied iteratively:

Fig. 4 A graph G of 13 vertices and 19 edges



- **Observation rule 1 (OR1).** A vertex in D observes itself and all of its neighbors.
- **Observation rule 2 (OR2).** If an observed vertex is adjacent to at most k unobserved vertices, then these vertices become observed as well.

The set D is a k -power dominating set of G if all vertices of the graph are observed after repeatedly applying the above two observation rules. Alternatively, let $\mathcal{S}_0(D) = N[D]$ and

$$\mathcal{S}_{i+1}(D) = \cup\{N[v] : v \in \mathcal{S}_i(D), |N(v) \setminus \mathcal{S}_i(D)| \leq k\}$$

for $i \geq 0$. Notice that $\mathcal{S}_0(D) \subseteq \mathcal{S}_1(D) \subseteq \mathcal{S}_2(D) \subseteq \dots$ and there is a t such that $\mathcal{S}_t(D) = \mathcal{S}_i(D)$ for $i \geq t$. Using this notation, D is a k -power dominating set if and only if $\mathcal{S}_t(D) = V$. The k -power domination number $\gamma_{k\text{pow}}(G)$ of G is the minimum size of a k -power dominating set. For the case of $k = 1$, 1-power domination is called power domination.

Figure 4 shows a graph G of 13 vertices and 19 edges, whose values of $\gamma_\pi(G)$ for variations of domination and the corresponding optimal sets/functions are given below.

$\gamma(G)$	$= 3,$	$D^* = \{v_2, v_8, v_9\};$
$\gamma_i(G)$	$= 3,$	$D^* = \{v_2, v_8, v_9\};$
$\gamma_c(G)$	$= 6,$	$D^* = \{v_2, v_4, v_5, v_7, v_{10}, v_{12}\};$
$\gamma_t(G)$	$= 5,$	$D^* = \{v_2, v_4, v_7, v_{10}, v_{12}\};$
$\gamma_{\text{per}}(G)$	$= 4,$	$D^* = \{v_2, v_3, v_8, v_9\};$
$\gamma_{\text{iper}}(G)$	$= \infty,$	infeasible;
$\gamma_{\text{cp}}(G)$	$= 10,$	$D^* = \{v_1, v_2, v_4, v_5, v_6, v_7, v_9, v_{10}, v_{11}, v_{12}\};$
$\gamma_{\text{tp}}(G)$	$= 6,$	$D^* = \{v_1, v_2, v_4, v_5, v_{12}, v_{13}\};$
$\gamma_{\text{cl}}(G)$	$= \infty,$	infeasible;
$\gamma_{\text{cy}}(G)$	$= 8,$	$D^* = \{v_2, v_4, v_6, v_9, v_{12}, v_{10}, v_7, v_5\};$
$\gamma_2(G)$	$= 2,$	$D^* = \{v_6, v_7\};$
$\gamma_k(G)$	$= 1,$	$D^* = \{v_7\} \text{ for } k \geq 3;$
$\gamma_e(G)$	$= 3,$	$D^* = \{v_2v_4, v_9v_{12}, v_7v_8\};$
$\gamma_s(G)$	$= 5,$	$f^*(v_i) = -1 \text{ for } i = 1, 6, 8, 11 \text{ and } f^*(v_i) = +1 \text{ for other } i;$

$$\begin{aligned}
\gamma_{\text{Rom}}(G) &= 6, & f^*(v_i) &= 2 \text{ for } i = 2, 8, 9 \text{ and } f^*(v_i) = 0 \text{ for other } i; \\
\gamma_{1\text{rain}}(G) &= 3, & f^*(v_i) &= \{1\} \text{ for } i = 2, 8, 9 \text{ and } f^*(v_i) = \emptyset \text{ for other } i; \\
\gamma_{2\text{rain}}(G) &= 6, & f^*(v_i) &= \{1, 2\} \text{ for } i = 2, 8, 9 \text{ and } f^*(v_i) = \emptyset \text{ for other } i; \\
\gamma_{3\text{rain}}(G) &= 8, & f^*(v_i) &= \{1\} \text{ for } i = 3, 6, 8, 13, f^*(v_1) = f^*(v_{10}) = \{2\}, \\
&& f^*(v_5) &= f^*(v_{11}) = \{3\} \text{ and } f^*(v_i) = \emptyset \text{ for other } i; \\
\gamma_{4\text{rain}}(G) &= 10, & f^*(v_i) &= \{1\} \text{ for } i = 3, 6, 8, 13, f^*(v_1) = \{2\}, f^*(v_{10}) = \{2, 4\}, \\
&& f^*(v_5) &= \{3, 4\}, f^*(v_{11}) = \{3\} \text{ and } f^*(v_i) = \emptyset \text{ for other } i; \\
\gamma_{5\text{rain}}(G) &= 12, & f^*(v_i) &= \{1\} \text{ for } i = 3, 6, 8, 13, f^*(v_1) = \{2\}, f^*(v_{10}) = \\
&& &\{2, 4, 5\}, \\
&& f^*(v_5) &= \{3, 4, 5\}, f^*(v_{11}) = \{3\} \text{ and } f^*(v_i) = \emptyset \text{ for other } i; \\
\gamma_{k\text{rain}}(G) &= 13, & f^*(v_i) &= \{1\} \text{ for all } i \text{ for } k \geq 6; \\
\gamma_{k\text{pow}}(G) &= 1, & D^* &= \{v_2\} \text{ for } k \geq 1.
\end{aligned}$$

The (*vertex*-)*weighted* versions of all of the above vertex-subset variations of domination can also be considered. Now, every vertex v has a weight $w(v)$ of real number. The problem is to find a dominating set D in a suitable variation such that

$$w(D) = \sum_{v \in D} w(v)$$

is as small as possible. Denote this minimum value by $\gamma_\pi(G, w)$, where π stands for a variation of the domination problem. When $w(v) = 1$ for all vertices v , the weighted cases become the cardinality cases.

For some variations of domination, the vertex weights may assume to be nonnegative as the following lemma shows.

Lemma 1 Suppose $G = (V, E)$ is a graph in which every vertex is associated with a weight $w(v)$ of real number. If $w'(v) = \max\{w(v), 0\}$ for all vertices $v \in V$, then for any $\pi \in \{\emptyset, c, t, k, k\text{pow}\}$ we have

$$\gamma_\pi(G, w) = \gamma_\pi(G, w') + \sum_{w(v) < 0} w(v).$$

Proof Denote by A the set of all vertices v with $w(v) < 0$. Suppose D is a π -dominating set of G with $\sum_{v \in D} w(v) = \gamma_\pi(G, w)$. Then

$$\begin{aligned}
\gamma_\pi(G, w') &\leq \sum_{v \in D} w'(v) \\
&= \sum_{v \in D} w(v) - \sum_{v \in D \cap A} w(v) \\
&\leq \gamma_\pi(G, w) - \sum_{w(v) < 0} w(v).
\end{aligned}$$

On the other hand, for any π -dominating set D of G with $\sum_{v \in D} w'(v) = \gamma_\pi(G, w')$, $D \cup A$ is also a π -dominating set of G and so

$$\begin{aligned}
\gamma_\pi(G, w) &\leq \sum_{v \in D \cup A} w(v) \\
&= \sum_{v \in D} w'(v) + \sum_{v \in A} w(v) \\
&= \gamma_\pi(G, w') + \sum_{w(v) < 0} w(v).
\end{aligned}$$

The lemma then follows. \square

More generally, one may consider *vertex-edge-weighted* cases of the domination problems as follows. Now, besides the weights of vertices, each edge e has a weight $\bar{w}(e)$. The object is then to find a dominating set D in a suitable variation such that

$$\bar{w}(D) = \sum_{v \in D} w(v) + \sum_{u \in V \setminus D} \bar{w}(uu')$$

is as small as possible, where u' is a vertex in D that is adjacent to u . Note that there are many choices of u' except for the perfect domination and its three variations. Denote this minimum value by $\gamma_\pi(G, w, \bar{w})$, where π stands for a variation of domination. When $\bar{w}(e) = 0$ for all edges e , the vertex-edge-weighted cases become the vertex-weighted cases.

Another parameter related to domination is as follows. The *domatic number* $d(G)$ of a graph G is the maximum number r such that G has r pairwise disjoint dominating sets D_1, D_2, \dots, D_r . One can also define *independent*, *connected*, *total*, *perfect*, *independent perfect*, *connected perfect*, *total perfect*, *clique*, *cycle*, *k -edge*, *k -power domatic numbers* $d_l(G), d_c(G), d_t(G), d_{\text{per}}(G), d_{\text{iper}}(G), d_{\text{cper}}(G), d_{\text{tper}}(G), d_{\text{cl}}(G), d_{\text{cy}}(G), d_k(G), d_e(G)$, and $d_{k^{\text{pow}}}(G)$, respectively, according to above variations of domination in similar ways.

2.3 Special Classes of Graphs

In this subsection, special classes of graphs are introduced. They are not only important in the study of domination but also fundamental in graph theory.

A *complete graph* is a graph whose vertex set is a clique. The complete graph with n vertices is denoted by K_n . The complement \overline{K}_n of the complete graph K_n is then a graph with no edge.

The *n -path*, denoted by P_n , is a graph with n vertices that contains a chordless path of length n . The *n -cycle*, denoted by C_n , is a graph with n vertices that contains a chordless cycle of length n .

An *r -partite graph* is a graph whose vertex set can be partitioned into r stable sets, which are called its *partite sets*. A 2-partite graph is usually called a *bipartite* graph. A *complete r -partite graph* is an r -partite graph in which vertices in different partite sets are adjacent. A complete r -partite graph with partite sets having n_1, n_2, \dots, n_r vertices, respectively, is denoted by K_{n_1, n_2, \dots, n_r} .

A *tree* is a connected graph without any cycle. A *directed tree* is an orientation of a tree. A *rooted tree* is a directed tree in which there is a special vertex r , called the *root*, such that for every vertex v there is a directed $r-v$ path. Trees are probably the simplest structures in graph theory. Problems looking hard in general graphs are often investigated in trees first as a warm up. Domination in trees is introduced in Sect. 3. Many ideas for domination in trees are then generalized to other classes of graphs.

Suppose \mathcal{F} is a family of sets. The *intersection graph* of \mathcal{F} is the graph obtained by representing each set in \mathcal{F} as a vertex and joining two distinct vertices with an edge if their corresponding sets intersect. It is well known that any graph is the intersection graph of some family \mathcal{F} . The problem of characterizing the intersection graphs of families of sets having some specific topological or other pattern is often interesting and frequently has applications in the real world. A typical example is the class of interval graphs. An *interval graph* is the intersection graph of intervals in the real line. They play important roles in many applications. Domination in interval graphs is investigated in Sect. 4.

A graph is *chordal* (or *triangulated*) if every cycle of length greater than three has a chord. The class of chordal graphs is one of the classical classes in the *perfect graph theory*; see the book by Golumbic [101]. It turns out to be also very important in the domination theory. It is well known that a graph is chordal if and only if it is the intersection graph of some subtrees of a certain tree. If these subtrees are paths, this chordal graph is called an *undirected path graph*. If these subtrees are directed paths of a rooted tree, the graph is called a *directed path graph*. If these subtrees are paths in some n -path, the graph is just an interval graph.

Most variations of the domination problem are NP-complete even for chordal graphs; see Sect. 5. As an important subclass of chordal graphs, the class of strongly chordal graphs is a star of the domination theory. Strongly chordal graphs include directed path graphs, which in turn include trees and interval graphs. Domination in strongly chordal graphs is studied in Sect. 6.

A permutation diagram consists of n points on each of two parallel lines and n straight line segments matching the points. The intersection graph of the line segments is called a *permutation graph*. Domination in permutation graphs is introduced in Sect. 7.

A *comparability graph* is a graph $G = (V, E)$ that has a *transitive* orientation $G' = (V, E')$, that is, $uv \in E'$ and $vw \in E'$ imply $uw \in E'$. A *cocomparability graph* is the complement of a comparability graph. Cocomparability graphs are generalizations of permutation graphs and intervals graphs. Domination in cocomparability graphs is investigated in Sect. 8.

A graph is *distance hereditary* if the distance between any two vertices is the same in any connected induced subgraph containing them. Domination in distance-hereditary graphs is studied in Sect. 9.

For more detailed discussions of these classes of graphs, see the remaining sections of this chapter.

3 Trees

3.1 Basic Properties of Trees

Recall that a tree is an acyclic connected graph. The following characterizations are well known; see the textbook by West [193].

Theorem 1 *The following statements are equivalent for any graph $G = (V, E)$:*

- (1) G is a tree.
- (2) G is connected and $|V| = |E| + 1$.
- (3) G is acyclic and $|V| = |E| + 1$.
- (4) For any two vertices u and v , there is a unique u - v path.
- (5) The vertices of G have an ordering $[v_1, v_2, \dots, v_n]$ such that v_i is a leaf of $G_i = G[\{v_i, v_{i+1}, \dots, v_n\}]$ for $1 \leq i \leq n - 1$, or equivalently

for each $1 \leq i \leq n - 1$, v_i is adjacent to exactly one v_j with $j > i$. (TO)

The ordering in [Theorem 1](#) (5) is called a *tree ordering* of the tree, where the only neighbor v_j of v_i with $j > i$ is called the *parent* of v_i and v_i is a *child* of v_j . The tree ordering plays an important role in many algorithms dealing with trees. Many algorithms on trees process from leaves by passing information to their parents iteratively or, equivalently, doing a loop according to a tree ordering. From an algorithmic point of view, the testing of a tree and finding a tree ordering can be done in linear time. [Figure 5](#) shows a tree of 11 vertices and a tree ordering.

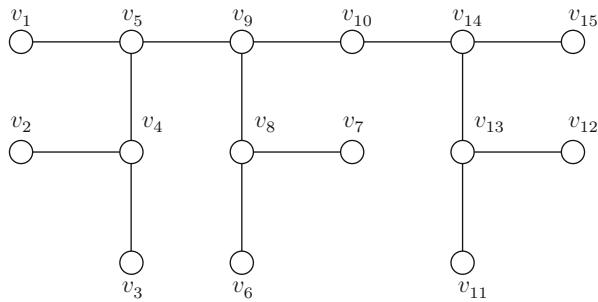
For some algorithms in trees, it is necessary to process simultaneously a group of leaves which is adjacent to a vertex with only one non-leaf neighbor. This corresponds to a tree ordering, which is called a *strong tree ordering*, with the property that all children of a vertex are consecutive in the ordering. The tree order $[v_2, v_3, v_1, v_4, v_6, v_7, v_5, v_8, v_9, v_{11}, v_{12}, v_{10}, v_{13}, v_{14}, v_{15}]$ is strong for the tree in [Fig. 5](#).

3.2 Labeling Algorithm for Trees

Cockayne et al. [56] gave the first linear-time algorithm for the domination problem in trees by a labeling method, which is a naive but useful approach.

The algorithm starts processing a leaf v of a tree T , which is adjacent to a unique vertex u . To dominate v , a minimum dominating set D of T must contain u or v . However, since $N[v] \subseteq N[u]$, it is better to have u in D rather than v in D . So one can keep an information “required” in u and delete v from T . At some iteration, if a “required” leaf v adjacent to u which is not labeled by “required” is processed, it is necessary to put v into D and delete it from T . But now, there is a vertex in D that dominates u , so u is labeled by “free.” For convenience, all vertices are labeled by “bound” initially.

Fig. 5 An example of the tree ordering



More precisely, suppose the vertex set of a graph $G = (V, E)$ is partitioned into three sets F , B , and R , where F consists of *free* vertices, B consists of *bound* vertices, and R consists of *required* vertices. A *mixed dominating set* of G (with respect to F , B , R) is a subset $D \subseteq V$ such that

$$R \subseteq D \text{ and every vertex in } B \setminus D \text{ is adjacent to some vertex in } D.$$

Free vertices need not be dominated by D but may be included in D in order to dominate bound vertices. The *mixed domination number* $\gamma^m(G)$ is the minimum size of a mixed dominating set in G , such a set is called an *md-set* of G . Note that mixed domination is the ordinary domination when $B = V$ and $F = R = \emptyset$.

The construction and correctness of the algorithm is based on the following theorem.

Theorem 2 Suppose T is a tree having free, bound, and required vertices F , B , and R , respectively. Let v be a leaf of T , which is adjacent to u . Then the following statements hold:

- (1) If $v \in F$, then $\gamma^m(T) = \gamma^m(T - v)$.
- (2) If $v \in B$ and T' is the tree which results from T by deleting v and relabeling u as “required,” then $\gamma^m(T) = \gamma^m(T')$.
- (3) If $v \in R$ and $u \in R$, then $\gamma^m(T) = \gamma^m(T - v) + 1$.
- (4) If $v \in R$, $u \notin R$ and T' is the tree which results from T by deleting v and relabeling u as “free”, then $\gamma^m(T) = \gamma^m(T') + 1$.

Proof (1) Since v is free, any md-set D' of $T - v$ is also a mixed dominating set of T . Thus, $\gamma^m(T) \leq |D'| = \gamma^m(T - v)$. On the other hand, suppose D is an md-set of T . If $v \notin D$, then D is also a mixed dominating set of $T - v$. If $v \in D$, then $(D \setminus \{v\}) \cup \{u\}$ is a mixed dominating set of $T - v$, whose size is at most $|D|$. Thus, in either case, $\gamma^m(T - v) \leq |D| = \gamma^m(T)$.

(2) Since u is required in T' , any md-set D' of T' always contains u and hence is also a mixed dominating set of T . Thus, $\gamma^m(T) \leq |D'| = \gamma^m(T')$. On the other hand, suppose D is an md-set of T . Since v is bound in T , either u or v is in D .

- In any case, $D' = (D \setminus \{v\}) \cup \{u\}$ is a mixed dominating set of T' , in which u is considered as a required vertex. So, $\gamma^m(T') \leq |D'| \leq |D| = \gamma^m(T)$.
- (3) If D' is an md-set of T' , then $D' \cup \{v\}$ is a mixed dominating set of T . Thus, $\gamma^m(T) \leq |D' \cup \{v\}| = \gamma^m(T') + 1$. On the other hand, any md-set D of T contains both u and v . Then $D \setminus \{v\}$ is a mixed dominating set of T' . So, $\gamma^m(T') \leq |D \setminus \{v\}| = \gamma^m(T) - 1$.
 - (4) If D' is an md-set of T' , then $D' \cup \{v\}$ is a mixed dominating set of T . Thus, $\gamma(T) \leq |D' \cup \{v\}| = \gamma^m(T') + 1$. On the other hand, any md-set D of T contains v . Since u is free in T' , $D \setminus \{v\}$ is a mixed dominating set in T' . So, $\gamma^m(T') \leq |D \setminus \{v\}| = \gamma^m(T) - 1$. \square

The above theorem then gives the following algorithm for the mixed domination problem in trees.

Algorithm DomTreeL. Find a minimum mixed dominating set of a tree.

Input. A tree T whose vertices are labeled by free, bound, or required. A tree ordering $[v_1, v_2, \dots, v_n]$ of T .

Output. A minimum mixed dominating set D of T .

Method.

```

 $D \leftarrow \phi;$ 
for  $i = 1$  to  $n - 1$  do
    let  $v_j$  be the parent of  $v_i$ ;
    if ( $v_i$  is bound) then
        relabel  $v_j$  as required;
    else if ( $v_i$  is required) then
         $D \leftarrow D \cup \{v_i\}$ ;
        if  $v_j$  is bound then relabel  $v_j$  as free;
        end if;
    end if;
end for;
if  $v_n$  is not free then  $D \leftarrow D \cup \{v_n\}$ .

```

Slater [183] generalized the above idea to solve the k -domination problem in trees. In fact he solved a slightly more general problem called R -domination. Now, each vertex v is associated with an ordered pair $R_v = (a_v, b_v)$, where a_v is a nonnegative integer and b_v a positive integer. The dominating set D is chosen so that each vertex v is within distance a_v from some vertex in D . The integer b_v indicates that there is a vertex in the current D that is at distance b_v from v . More precisely, an R -dominating set of $G = (V, E)$ is a vertex subset D such that for any vertex v in G , either there is some $u \in D$ with $d(u, v) \leq a_v$ or else there is some $u \in V$ with $b_u + d(u, v) \leq a_v$. The R -domination number $\gamma^R(G)$ of G is the minimum size of an R -dominating set. Notice that R -domination with each $R_v = (k, k + 1)$ is the same as the k -domination.

The construction and correctness of Slater's algorithm are based on the following theorem whose proof is omitted here.

Theorem 3 Suppose T is a tree in which each vertex v has a label $R_v = (a_v, b_v)$, where a_v is a nonnegative integer and b_v a positive integer. Let v be a leaf of T , which is adjacent to u . Then the following statements hold:

- (1) If $a_v \geq b_v$ and T' is the tree which results from T by deleting v and resetting b_u by $\min\{b_u, b_v + 1\}$, then $\gamma^R(T) = \gamma^R(T')$.
- (2) If $a_v = 0$ and T' is the tree which results from T by deleting v and resetting b_u by 1, then $\gamma^R(T) = \gamma^R(T') + 1$.
- (3) If $0 < a_v < b_v$ and T' is the tree which results from T by deleting v and resetting a_u by $\min\{a_u, a_v - 1\}$ and b_u by $\min\{b_u, b_v + 1\}$, then $\gamma^R(T) = \gamma^R(T')$.

This then gives the following algorithm for R -domination in trees.

Algorithm RDomTreeL. Find a minimum R -dominating set of a tree.

Input. A tree T with a tree ordering $[v_1, v_2, \dots, v_n]$, in which each vertex v has a label $R_v = (a_v, b_v)$, where $a_v \geq 0$ and $b_v > 0$ are integers.

Output. A minimum R -dominating set D of T .

Method.

```

 $D \leftarrow \phi;$ 
for  $i = 1$  to  $n - 1$  do
    let  $v_j$  be the parent of  $v_i$ ;
    if  $(a_{v_i} \geq b_{v_i})$  then
         $b_{v_j} \leftarrow \min\{b_{v_j}, b_{v_i} + 1\};$ 
    else if  $(a_{v_i} = 0)$  then
         $D \leftarrow D \cup \{v_i\};$ 
         $b_{v_j} \leftarrow 1;$ 
        end if;
    else if  $(0 < a_{v_i} < b_{v_i})$  then
         $a_{v_j} \leftarrow \min\{a_{v_j}, a_{v_i} - 1\};$ 
         $b_{v_j} \leftarrow \min\{b_{v_j}, b_{v_i} + 1\};$ 
        end if;
    end for;
if  $(a_{v_n} < b_{v_n})$  then  $D \leftarrow D \cup \{v_n\}.$ 

```

The labeling algorithm is also used for many variations of domination. For instance, Mitchell and Hedetniemi [162] and Yannakakis and Gavril [196] gave labeling algorithms for the edge domination problem in trees. Laskar et al. [149] gave a labeling algorithm for the total domination problem in trees.

Next, an example of labeling algorithm using strong tree orderings is demonstrated. Chang et al. [47] investigated the k -rainbow domination problem on trees. For technical reasons, they in fact dealt with a more general problem. A k -rainbow assignment is a mapping L that assigns each vertex v a label $L(v) = (a_v, b_v)$ with $a_v, b_v \in \{0, 1, \dots, k\}$. A k - L -rainbow dominating function is a function $f : V(G) \rightarrow 2^{\{1, 2, \dots, k\}}$ such that for every vertex v in G the following conditions hold:

- (L1) $|f(v)| \geq a_v$.
(L2) $|\cup_{u \in N(v)} f(u)| \geq b_v$ whenever $f(v) = \emptyset$.

The k -L-rainbow domination number $\gamma_{k\text{Rain}}(G)$ of G is the minimum weight of a k -L-rainbow dominating function. A k -L-rainbow dominating function f of G is *optimal* if $w(f) = \gamma_{k\text{Rain}}(G)$. Notice that k -rainbow domination is the same as k -L-rainbow domination if $L(v) = (0, k)$ for each $v \in V(G)$.

Theorem 4 Suppose v is a leaf adjacent to u in a graph G with a k -rainbow assignment L . Let $G' = G - v$ and L' be the restriction of L on $V(G')$, except that when $a_v > 0$ we let $b'_u = \max\{0, b_u - a_v\}$. Then the following hold:

- (1) If $a_v > 0$, then $\gamma_{k\text{Rain}}(G) = \gamma_{k\text{Rain}}(G') + a_v$.
- (2) If $a_v = 0$ and $a_u \geq b_v$, then $\gamma_{k\text{Rain}}(G) = \gamma_{k\text{Rain}}(G')$.

Theorem 5 Suppose $N(u) = \{z, v_1, v_2, \dots, v_s\}$ such that v_1, v_2, \dots, v_s are leaves in a graph G with a k -rainbow assignment L . Assume $a_{v_i} = 0$ for $1 \leq i \leq s$ and $b_{v_1} \geq b_{v_2} \geq \dots \geq b_{v_s} > a_u$. Let $b^* = \min\{b_{v_i} + i - 1 : 1 \leq i \leq s + 1\} = b_{v_{i^*}} + i^* - 1$, where $b_{v_{s+1}} = a_u$ and i^* is chosen as small as possible. If $G' = G - \{v_1, v_2, \dots, v_s\}$ and L' is the restriction of L on $V(G')$ with modifications that $a'_u = b_{v_{i^*}}$ and $b'_u = \max\{0, b_u - i^* + 1\}$, then $\gamma_{k\text{Rain}}(G) = \gamma_{k\text{Rain}}(G') + i^* - 1$.

Remark that for the case when the component of G containing u is a star, the vertex z does not exist. There is in fact no vertex v_{s+1} . The assignment of $b_{v_{s+1}} = a_u$ is for the purpose of convenience. In the case of $i^* = s + 1$, it just means that a'_u is the same as a_u .

The theorems above then give the following linear-time algorithm for the k -L-rainbow domination problem in trees.

Algorithm RainbowDomTreeL. Find the k -L-domination number of a tree.

Input. A tree $T = (V, E)$ in which each vertex v is labeled by $L(v) = (a_v, b_v)$.

Output. The minimum k -L-rainbow dominating number r of T .

Method.

```

 $r \leftarrow 0$ ;
get a breadth first search ordering  $x_1, x_2, \dots, x_n$  for the tree  $T$  rooted at  $x_1$ ;
for  $j = 1$  to  $n$  do  $s_j \leftarrow 0$ ; {number of children  $x_i$  with  $a_{x_i} = 0$  and  $b_{x_i} > a_{x_j}\}$ 
for  $j = n$  to  $2$  step by  $-1$  do
     $s \leftarrow s_j$ ;
     $v \leftarrow x_j$ ;
    if  $s > 0$  then {apply Theorem 5}
        let  $u = v$  and  $z, v_1, v_2, \dots, v_s, b^*, i^*$  be as described in Theorem 5;
         $r \leftarrow r + i^* - 1$ ;
         $a_u \leftarrow b_{v_{i^*}}$ ;
         $b_u \leftarrow \max\{0, b_u - i^* + 1\}$ ;
        end if;
    else {apply Theorem 4}
        let  $u = x_{j'}$  be the parent of  $v$ ;
        if  $a_v > 0$  then {  $b_u \leftarrow \max\{0, b_u - a_v\}$ ;  $r \leftarrow r + a_v$  };
    end if;
end if;

```

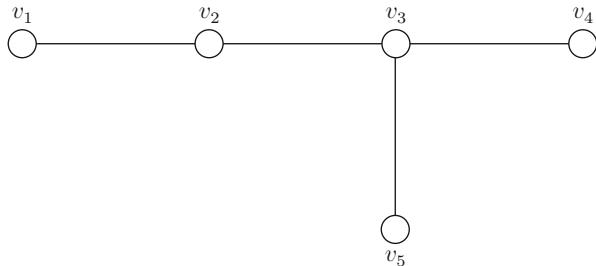


Fig. 6 A tree T with a unique minimum independent dominating set

```

else if  $a_u < b_v$  then  $s_{j'} \leftarrow s_{j'} + 1$ ;
end else;
end do;
if  $a_{x_1} > 0$  then  $r \leftarrow r + a_{x_1}$ ; else if  $b_{x_1} > 0$  then  $r \leftarrow r + 1$ .

```

As these algorithms suggest, the labeling algorithm may only work for problems whose solutions have “local property.” For an example of problem without local property, the independent domination problem in the tree T of Fig. 6 is considered. The only minimum independent dominating set of T is $\{v_1, v_3\}$. If the tree ordering $[v_1, v_2, v_4, v_3, v_5]$ is given, the algorithm must be clever enough to put the leaf v_1 into the solution at the first iteration. If another tree ordering $[v_5, v_4, v_3, v_2, v_1]$ is given, the algorithm must be clever enough not to put the leaf v_5 at the first iteration. So, the algorithm must be one that not only looks at a leaf and its only neighbor but also has some idea about the whole structure of the tree. This is the meaning that the solution does not have a “local property.”

3.3 Dynamic Programming for Trees

Dynamic programming is a powerful method for solving many discrete optimization problems; see the books by Bellman and Dreyfus [14], Dreyfus and Law [85], and Nemhauser [167]. The main idea of the dynamic programming approach for domination is to turn the “bottom-up” labeling method into “top-down.” Now a specific vertex u is chosen from G . A minimum dominating set D of G either contains or does not contain u . So it is useful to consider the following two domination problems which are the ordinary domination problem with boundary conditions:

$$\gamma^1(G, u) = \min\{|D| : D \text{ is a dominating set of } G \text{ and } u \in D\}.$$

$$\gamma^0(G, u) = \min\{|D| : D \text{ is a dominating set of } G \text{ and } u \notin D\}.$$

Lemma 2 $\gamma(G) = \min\{\gamma^1(G, u), \gamma^0(G, u)\}$ for any graph G with a specific vertex u .

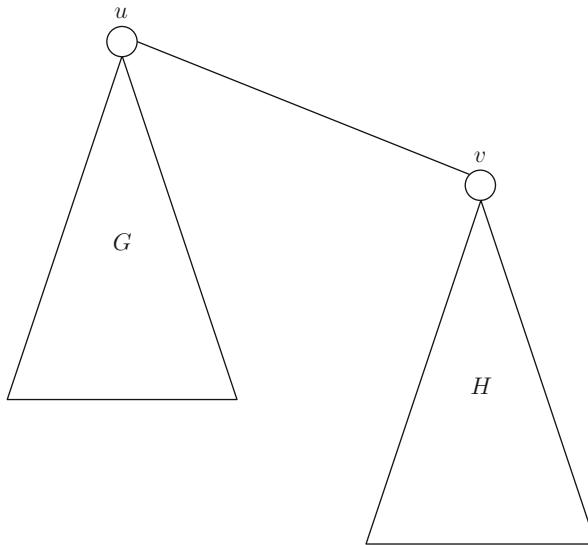


Fig. 7 The composition of two trees G and H

Suppose H is another graph with a specific vertex v . Let I be the graph with the specific vertex u , which is obtained from the disjoint union of G and H by joining a new edge uv ; see Fig. 7.

The aim is to use $\gamma^1(G, u)$, $\gamma^0(G, u)$, $\gamma^1(H, v)$, and $\gamma^0(H, v)$ to find $\gamma^1(I, u)$ and $\gamma^0(I, u)$. Suppose D is a dominating set of I with $u \in D$. Then $D = D' \cup D''$, where D' is a dominating set of G with $v \in D'$ and D'' is a subset of $V(H)$ which dominates $V(H) - \{v\}$. There are two cases. In the case of $v \in D''$, D'' is a dominating set of H . On the other hand, if $v \notin D''$, then D'' is a dominating set of $H - v$. In order to cover the latter case, the following new problem is introduced:

$$\gamma^{00}(G, u) = \min\{|D| : D \text{ is a dominating set of } G - u\}.$$

Note that $\gamma^{00}(G, u) \leq \gamma^0(G, u)$, since a dominating set D of G with $u \notin D$ is also a dominating set of $G - u$.

Theorem 6 Suppose G and H are graphs with specific vertices u and v , respectively. Let I be the graph with the specific vertex u , which is obtained from the disjoint union of G and H by joining a new edge uv . Then the following statements hold:

- (1) $\gamma^1(I, u) = \gamma^1(G, u) + \min\{\gamma^1(H, v), \gamma^{00}(H, v)\}.$
- (2) $\gamma^0(I, u) = \min\{\gamma^0(G, u) + \gamma^0(H, v), \gamma^{00}(G, u) + \gamma^1(H, v)\}.$
- (3) $\gamma^{00}(I, u) = \gamma^{00}(G, u) + \gamma^0(H) = \gamma^{00}(G, u) + \min\{\gamma^1(H, v), \gamma^0(H, v)\}.$

- Proof* (1) This follows from the fact that D is a dominating set of I with $u \in D$ if and only if $D = D' \cup D''$, where D' is a dominating set of G with $u \in D'$ and D'' is a dominating set of H with $v \in D''$ or a dominating set of $H - v$.
- (2) This follows from the fact that D is a dominating set of I with $u \notin D$ if and only if $D = D' \cup D''$, where either D' is a dominating set of G with $u \notin D'$ and D'' is a dominating set of H with $v \notin D''$ or D' is a dominating set of $G - u$ and D'' is a dominating set of H with $v \in D''$.
- (3) This follows from the fact that D is a dominating set of $I - u$ if and only if $D = D' \cup D''$, where D' is a dominating set of $G - u$ and D'' is a dominating set of H . \square

The lemma and the theorem above then give the following dynamic programming algorithm for the domination problem in trees.

Algorithm DomTreeD. Determine the domination number of a tree.

Input. A tree T with a tree ordering $[v_1, v_2, \dots, v_n]$.

Output. The domination number $\gamma(T)$ of T .

Method.

```

for  $i = 1$  to  $n$  do
     $\gamma^1(v_i) \leftarrow 1$ ;
     $\gamma^0(v_i) \leftarrow \infty$ ;
     $\gamma^{00}(v) \leftarrow 0$ ;
end do;
for  $i = 1$  to  $n - 1$  do
    let  $v_j$  be the parent of  $v_i$ ;
     $\gamma^1(v_j) \leftarrow \gamma^1(v_j) + \min\{\gamma^1(v_i), \gamma^{00}(v_i)\}$ ;
     $\gamma^0(v_j) \leftarrow \min\{\gamma^0(v_j) + \gamma^0(v_i), \gamma^{00}(v_j) + \gamma^1(v_i)\}$ ;
     $\gamma^{00}(v_j) \leftarrow \gamma^{00}(v_j) + \min\{\gamma^1(v_i), \gamma^0(v_i)\}$ ;
end do;
 $\gamma(T) \leftarrow \min\{\gamma^1(v_n), \gamma^0(v_n)\}.$ 

```

The advantage of the dynamic programming method is that it also works for problems whose solutions have no local property. As an example, Beyer et al. [21] solved the independent domination problem by this method. Moreover, the method can be used to solve the vertex-edge-weighted cases. The following derivation for the vertex-edge-weighted domination in trees is slightly different from that given by Natarajan and White [166]:

Define $\gamma^1(G, u, w, \bar{w})$, $\gamma^0(G, u, w, \bar{w})$, and $\gamma^{00}(G, u, w, \bar{w})$ in the same way as $\gamma^1(G, u)$, $\gamma^0(G, u)$, and $\gamma^{00}(G, u)$, except that $|D|$ is replaced by $\bar{w}(D)$.

Lemma 3 $\gamma(G, w, \bar{w}) = \min\{\gamma^1(G, u, w, \bar{w}), \gamma^0(G, u, w, \bar{w})\}$ for any graph G with a specific vertex u .

Theorem 7 Suppose G and H are graphs with specific vertices u and v , respectively. Let I be the graph with the specific vertex u , which is obtained from the

disjoint union of G and H by joining a new edge uv . The following statements hold:

- (1) $\gamma^1(I, u, w, \bar{w}) = \gamma^1(G, u, w, \bar{w}) + \min\{\gamma(H, v, w, \bar{w}), \bar{w}(uv) + \gamma^{00}(H, v, w, \bar{w})\}.$
- (2) $\gamma^0(I, u, w, \bar{w}) = \min\{\gamma^0(G, u, w, \bar{w}) + \gamma(H, w, \bar{w}), \gamma^{00}(G, u, w, \bar{w}) + \bar{w}(uv) + \gamma^1(H, v, w, \bar{w})\}.$
- (3) $\gamma^{00}(I, u, w, \bar{w}) = \gamma^{00}(G, u, w, \bar{w}) + \gamma(H, w, \bar{w}).$

The lemma and the theorem above then give the following algorithm for the vertex-edge-weighted domination problem in trees.

Algorithm VEWDomTreeD. Determine the vertex-edge-weighted domination number of a tree.

Input. A tree T with a tree ordering $[v_1, v_2, \dots, v_n]$, and each vertex v has a weight $w(v)$ and each edge e has a weight $\bar{w}(e)$.

Output. The vertex-edge-weighted domination number $\gamma(T, w, \bar{w})$ of T .

Method.

for $i = 1$ **to** n **do**

$\gamma(v_i) \leftarrow \gamma^1(v_i) \leftarrow w(v_i);$
 $\gamma^0(v_i) \leftarrow \infty;$
 $\gamma^{00}(v_i) \leftarrow 0;$

end do;

for $i = 1$ **to** $n - 1$ **do**

let v_j be the parent of v_i ;
 $\gamma^1(v_j) \leftarrow \gamma^1(v_j) + \min\{\gamma(v_i), \bar{w}(uv) + \gamma^{00}(v_i)\};$
 $\gamma^0(v_j) \leftarrow \min\{\gamma^0(v_j) + \gamma(v_i), \gamma^{00}(v_j) + \bar{w}(uv) + \gamma^1(v_i)\};$
 $\gamma^{00}(v_j) \leftarrow \gamma^{00}(v_j) + \gamma(v_i);$
 $\gamma(v_j) \leftarrow \min\{\gamma^1(v_j), \gamma^0(v_j)\};$

end do;

$\gamma(T, w, \bar{w}) \leftarrow \gamma(v_n).$

The dynamic programming method is also used in several papers for solving variations of the domination problems in trees; see [12, 96, 114, 122, 184, 201].

3.4 Primal–Dual Approach for Trees

The most beautiful method used in domination may be the primal–dual approach. In this method, besides the original domination problem, the following dual problem is also considered. In a graph $G = (V, E)$, a 2-stable set is a subset $S \subseteq V$ in which every two distinct vertices u and v have distance $d(u, v) > 2$. The 2-stability number $\alpha_2(G)$ of G is the maximum size of a 2-stable set in G . It is easy to see the following inequality.

Weak Duality Inequality: $\alpha_2(G) \leq \gamma(G)$ for any graph G .

Note that the above inequality can be strict, as shown by the n -cycle C_n that $\alpha_2(C_n) = \lfloor \frac{n}{3} \rfloor$ but $\gamma(C_n) = \lceil \frac{n}{3} \rceil$.

For a tree T , an algorithm which outputs a dominating set D^* and a 2-stable set S^* with $|D^*| \leq |S^*|$ is designed. Then

$$|S^*| \leq \alpha_2(T) \leq \gamma(T) \leq |D^*| \leq |S^*|,$$

which implies that all inequalities are equalities. Consequently, D^* is a minimum dominating set, S^* is a maximum 2-stable, and the *strong duality equality* $\alpha_2(T) = \gamma(T)$ holds.

The algorithm starts from a leaf v adjacent to u . It also uses the idea as in the labeling algorithm that u is more powerful than v since $N[v] \subseteq N[u]$. Instead of choosing v , u is put into D^* . Besides, v is also put into S^* . More precisely, the algorithm is as follows.

Algorithm DomTreePD. Find a minimum dominating set and a maximum 2-stable set of a tree.

Input. A tree T with a tree ordering $[v_1, v_2, \dots, v_n]$.

Output. A minimum dominating set D^* and a maximum 2-stable set S^* of T .

Method.

```

 $D^* \leftarrow \phi;$ 
 $S^* \leftarrow \phi;$ 
for  $i = 1$  to  $n$  do
    let  $v_j$  be the parent of  $v_i$ ;
    (assume  $v_j = v_n$  for  $v_i = v_n$ )
    if  $(N[v_i] \cap D^* = \phi)$  then
         $D^* \leftarrow D^* \cup \{v_j\};$ 
         $S^* \leftarrow S^* \cup \{v_i\};$ 
    end if:
end do.

```

To verify the algorithm, it is sufficient to prove that D^* is a dominating set, S^* is a 2-stable set, and $|D^*| \leq |S^*|$.

D^* is clearly a dominating set as the if-then statement does.

Suppose S^* is not a 2-stable set, that is, there exist v_i and $v_{i'}$ in S^* such that $i < i'$ but $d_T(v_i, v_{i'}) \leq 2$. Let $T_i = T[\{v_i, v_{i+1}, \dots, v_n\}]$. Then T_i contains v_i, v_j , and $v_{i'}$. Since $d(v_i, v_{i'}) \leq 2$, the unique v_i - $v_{i'}$ path in T (and also in T') is either $v_i, v_{i'}$ or $v_i, v_j, v_{i'}$. In any case, $v_j \in N[v_{i'}]$. Thus, at the end of iteration i , D^* contains v_j . When the algorithm processes $v_{i'}$, $N[v_{i'}] \cap D^* \neq \phi$ which causes that S^* does not contain $v_{i'}$, a contradiction.

$|D^*| \leq |S^*|$ follows from that when v_j is added into D^* , which may or may not already be in D^* , a new vertex v_i is always added into S^* .

Theorem 8 *Algorithm DomTreePD gives a minimum dominating set D^* and a maximum 2-stable set S^* of a tree T with $|D^*| = |S^*|$ in linear time.*

Theorem 9 (Strong Duality) $\alpha_2(T) = \gamma(T)$ for any tree T .

The primal–dual approach was in fact used by Farber [90] and Kolen [143] for the weighted domination problem in strongly chordal graphs. It was also used by Cheston et al. [55] for upper fraction domination in trees.

3.5 Power Domination in Trees

Power domination is a most different variation of domination. The observation rules make it so different from the ordinary domination as well as other variations. Haynes et al. [111] gave a linear-time algorithm for the power domination problem in trees. This subsection demonstrates a neat linear-time algorithm offered by Guo et al. [103].

Algorithm PowerDomTree. Find a minimum power dominating set of a tree.

Input. A tree T rooted at vertex r .

Output. A minimum power dominating set D of T .

Method.

sort the non-leaf vertices of T in a list L according to a post-order traversal of T ;

$D \leftarrow \emptyset$;

while $L \neq \{r\}$ **do**

$v \leftarrow$ the first vertex in L ;

$L \leftarrow L \setminus \{v\}$;

if v has at least two unobserved children **then**

$D \leftarrow D \cup \{v\}$;

exhaustively apply the two observation rules to T ;

end if;

end while;

if r is unobserved **then** $D \leftarrow D \cup \{r\}$.

Theorem 10 *Algorithm PowerDomTree gives a minimum power dominating set of a tree in linear time.*

Proof First is to prove that the output D of the algorithm is a power dominating set. The proof is based on an induction on the depth of the vertices u in T , denoted by $\text{depth}(u)$. Note that the proof works top-down, whereas the algorithm works bottom-up. For $\text{depth}(u) = 0$, it is clear that u , which is r , is observed due to the second “if”-condition of the algorithm. Suppose that all vertices u with $\text{depth}(u) < k$ with $k > 0$ are observed. Consider a vertex u with $\text{depth}(u) = k$, which is a child of the vertex v . If $v \in D$, then u is observed; otherwise, due to the induction hypothesis, v is observed. Moreover, if $\text{depth}(v) \geq 1$, then the parent of v is observed as well. Because of the first “if”-condition of the algorithm, v is not in D only if v has at most one unobserved child during the “while”-loop of the algorithm processing v . If vertex u is this only unobserved child, then it gets observed by applying OR2 to v , because v itself and all its neighbors (including the parent of v and its children)

with the only exception of u are observed by the vertices in D . In summary, it is concluded that all vertices in T are observed by D . \square

Next is to prove the optimality of D by showing a more general statement:

Claim Given a tree $T = (V, E)$ rooted at r , Algorithm PowerDomTree outputs a power dominating set D such that $|D \cap V_u| \leq |D' \cap V_u|$ for any minimum power dominating set D' of T and any tree vertex u , where V_u denotes the vertex set of the subtree of T rooted at u .

Proof of the Claim. Let $T_u = (V_u, E_u)$ denote the subtree of T rooted at vertex u . Let ℓ denote the depth of T , that is, $\ell := \max_{v \in V} \text{depth}(v)$. For each vertex $u \in V$, define $d_u := |D \cap V_u|$ and $d'_u := |D' \cap V_u|$. The claim is to be proved by an induction on the depth of tree vertices, starting with the maximum depth ℓ and proceeding to 0.

Since vertices u with $\text{depth}(u) = \ell$ are leaves and the algorithm adds no leaf to D , it is the case that $d_u = 0$ and so $d_u \leq d'_u$ for all u with $\text{depth}(u) = \ell$.

Suppose that $d_u \leq d'_u$ holds for all u with $\text{depth}(u) > k$ with $k < \ell$. Consider a vertex u with $\text{depth}(u) = k$. Let C_u denote the set of children of u . Then, for all $v \in C_u$, by the induction hypothesis, $d_v \leq d'_v$. In order to show $d_u \leq d'_u$, one only has to consider the case that

$$(a) u \in D, \quad (b) u \notin D', \quad \text{and (c)} \sum_{v \in C_u} d_v = \sum_{v \in C_u} d'_v. \quad (1)$$

In all other cases, $d_u \leq d'_u$ always holds. In the following, it will be shown that this case does not apply. Assume that $u \neq r$. The argument works also for $u = r$.

From (c) and that $d_v \leq d'_v$ for all $v \in C_u$ (induction hypothesis), $d_v = d'_v$ for all $v \in C_u$. Moreover, (a) is true only if u has two unobserved children v_1 and v_2 during the “while”-loop of the algorithm processing u (the first “if”-condition). In other words, this means that vertices v_1 and v_2 are not observed by the vertices in $(D \cap V_u) \setminus \{u\}$. In the following, it is shown that u has to be included in D' in order for D' to be a valid power dominating set. To this end, the following statement is needed:

$$\text{For all } x \in D' \cap V_{v_i} \text{ with } 1 \leq i \leq 2 \text{ there is some } x' \in W_{(v_i, x)} \cap D, \quad (2)$$

where $W_{(v_i, x)}$ denotes the path between v_i and x (including v_i and x).

Without loss of generality, consider only $i = 1$. Assume that statement (2) is not true for a vertex $x \in D' \cap V_{v_1}$. Since $x \notin D$, one can infer that $d_x < d'_x$. Since no vertex from $W_{(v_1, x)}$ is in D and, by induction hypothesis, $d_y \leq d'_y$ for all $y \in V_{v_1}$, it follows that $d_{x'} < d'_{x'}$ for all vertices $x' \in W_{(v_1, x)}$, in particular, $d_{v_1} < d'_{v_1}$. This contradicts the fact that $d_{v_i} = d'_{v_i}$ for all children v_i of u . Thus, statement (2) is true.

By statement (Eq. 2), for each vertex $x \in D' \cap V_{v_i}$ ($i \in \{1, 2\}$), there exists a vertex $x' \in D$ on the path $W_{(v_i, x)}$. Thus, if vertex v_i is observed by a vertex $x \in D' \cap V_{v_i}$, then there is a vertex $x' \in D \cap W_{(v_i, x)}$ that observes v_i . However, (a) in (Eq. 1) is true only if v_1 and v_2 are unobserved by the vertices in $D \cap V_{v_1}$ and $D \cap V_{v_2}$. Altogether, this implies that v_1 and v_2 cannot be observed by the vertices in $D' \cap V_{v_1}$ and $D' \cap V_{v_2}$. Since D' is a power dominating set of T , vertex u is taken into D' ; otherwise, u has two unobserved neighbors v_1 and v_2 . OR2 can never be applied to u whatever vertices from $V \setminus V_u$ are in D' . It concludes that the case that $u \in D, u \notin D'$, and $\sum_{v \in C_u} d_v = \sum_{v \in C_u} d'_v$ does not apply. This completes the proof of the claim. \square

Concerning the running time, the following explains how to implement the exhaustive applications of OR1 and OR2. Observe that if OR2 is applicable to a vertex u during the bottom-up process, then all vertices in T_u can be observed at the current stage of the bottom-up process. In particular, after adding vertex u to D , all vertices in T_u can be observed. Thus, exhaustive applications of OR1 and OR2 to the vertices of T_u can be implemented as pruning T_u from T which can be done in constant time. Next, consider the vertices in $V \setminus V_u$. After adding u to D , the only possible application of OR1 is that u observes its parent v . Moreover, the applicability of OR2 to the vertices x lying on the path from u to r is checked, in the order of their appearance, the first v , and the last r . As long as OR2 is applicable to a vertex x on this path, x is deleted from the list L that is defined in the first line of the algorithm and prune T_x from T .

The linear running time of the algorithm is then easy to see: The vertices to which OR2 is applied are removed from the list L immediately after the application of OR2 and are never processed by the instruction in the first line inside the “while”-loop of the algorithm. With proper data structures such as integer counters storing the number of observed neighbors of a vertex, the applications of the observation rules to a single vertex can be done in constant time. Post-order traversal of a rooted tree is clearly doable in linear time. \square

3.6 Tree Related Classes

Besides the methods demonstrated in the previous subsections, the “transformation method” sometimes is also used in the study of domination. Roughly speaking, the method transforms the domination problem in certain graphs to another well-known problem, which is solvable. As this method depends on the variation of domination and the type of graphs, it will be mentioned only when it is used in the problem surveyed in this chapter.

There are some classes of graphs which have treelike structures, including block graphs, (partial) k -trees, and cacti.

A *block graph* is a graph whose blocks are complete graphs. For results on variations of domination in block graphs, see [39, 43, 126, 130, 202].

A *cactus* is a graph whose blocks are cycles. Hedetniemi [118] gave a linear-time algorithm for the domination problem in cacti.

For a positive integer k , k -*trees* are defined recursively as follows: (a) A complete graph of $k + 1$ vertices is a k -tree; (b) the graph obtained from a k -tree by adding a new vertex adjacent to a clique of k vertices is a k -tree. *Partial k-trees* are subgraphs of k -trees. For results on variations of domination in k -trees and partial k -trees, see [3, 66, 95, 172, 180, 189, 190].

4 Interval Graphs

4.1 Interval Orderings of Interval Graphs

Recall that an interval graph is the intersection graph of a family of intervals in the real line.

In many papers, people design algorithms or prove theorems for interval graphs by using the interval models directly. This very often is accomplished by ordering the intervals according to some nondecreasing order of their right (or left) endpoints.

For instance, suppose $G = (V, E)$ is an interval graph with an interval model

$$\{I_i = [a_i, b_i] : 1 \leq i \leq n\},$$

where $b_1 \leq b_2 \leq \dots \leq b_n$. One can solve the domination problem for G by using exactly the same primal–dual algorithm for trees except replacing the 4th line of Algorithm DomTreePD by

let j be the largest index such that $v_j \in N[v_i]$.

In order to prove that the revised algorithm works for interval graphs, it is only necessary to show that S^* is a 2-stable set. Suppose to the contrary that S^* contains two vertices v_i and $v_{i'}$ with $i < i'$ and $d(v_i, v_{i'}) \leq 2$, say there is a vertex $v_k \in N[v_i] \cap N[v_{i'}]$. Consider the largest indexed vertex v_j of $N[v_i]$ as chosen in iteration i of the algorithm. Since $v_k \in N[v_i]$, $k \leq j$. Consider two cases:

Case 1. $j \leq i'$. In this case, $k \leq j \leq i'$. Then $b_k \leq b_j \leq b_{i'}$. Since $v_k \in N[v_{i'}]$, I_k intersects $I_{i'}$ and so $a_{i'} \leq b_k$. Therefore, $a_{i'} \leq b_j \leq b_{i'}$ and so I_j intersects $I_{i'}$.

Case 2. $i' < j$. In this case, $i < i' < j$. Then $b_i \leq b_{i'} \leq b_j$. Since $v_j \in N[v_i]$, I_i intersects I_j and so $a_j \leq b_i$. Therefore, $a_j \leq b_{i'} \leq b_j$ and so I_j intersects $I_{i'}$.

In any case, $v_j \in N[v_{i'}]$. As v_j is put into D^* in iteration i , when the algorithm processes $v_{i'}$, $N[v_{i'}] \cap D^* \neq \emptyset$ so that S^* does not contain $v_{i'}$, a contradiction. Therefore, S^* is a 2-stable set.

As one can see, the arguments in Cases 1 and 2 are quite similar. This is also true in many other proofs for interval graphs. One may expect that a unified property can be applied. This is in fact the so-called interval ordering in the following

theorem (see [175]). Notice that once property (IO) below holds, the conclusion $v_j \in N[v_i]$ above follows immediately.

Theorem 11 $G = (V, E)$ is an interval graph if and only if G has an interval ordering which is an ordering $[v_1, v_2, \dots, v_n]$ of V satisfying

$$i < j < k \text{ and } v_i v_k \in E \text{ imply } v_j v_k \in E. \quad (\text{IO})$$

Proof (\Rightarrow) Suppose G is the intersection graph of

$$\{I_i = [a_i, b_i] : 1 \leq i \leq n\}.$$

Without loss of generality, assume that $b_1 \leq b_2 \leq \dots \leq b_n$. Suppose $i < j < k$ and $v_i v_k \in E$. Then $b_i \leq b_j \leq b_k$. Since $v_i v_k \in E$, it is the case that $I_i \cap I_k \neq \emptyset$ which implies that $a_k \leq b_i$. Thus, $a_k \leq b_j \leq b_k$ and so $I_j \cap I_k \neq \emptyset$, that is, $v_j v_k \in E$.

(\Leftarrow) On the other hand, suppose (IO) holds. For any $v_i \in V$, let i^* be the minimum index such that $v_{i^*} \in N[v_i]$ and let interval $I_i = [i^*, i]$. If $v_i v_k \in E$ with $i < k$, then $k^* \leq i < k$ and so $I_i \cap I_k \neq \emptyset$. If $I_i \cap I_k \neq \emptyset$ with $i < k$, say $j \in I_i \cap I_k$, then $k^* \leq j \leq i < k$. Since $v_{k^*} v_k \in E$, by (IO), $v_i v_k \in E$. Therefore, G is an interval graph with

$$\{I_i = [i^*, i] : 1 \leq i \leq n\}$$

as an interval model. □

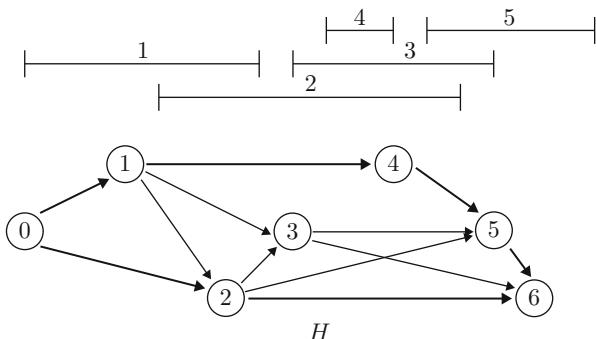
The problem of recognizing interval graphs and giving their interval ordering is a fundamental problem. Several linear-time recognition algorithms for interval graphs have been developed in the literature; see Booth and Leuker [26], Korte and Mohring [144], Hsu and his coauthors [123–125, 181], and Habib et al. [104], among many others. Note that a linear-time algorithm may not be easily implementable, and so there are still some efforts to search for new (simple) interval graph recognition algorithms and new ways to reconstruct an interval representation or just the interval ordering of a given interval graph. The 6-sweep LBFS algorithm by Corneil et al. [70] is a such one. It is believed that a 3-sweep LBFS algorithm is possible.

4.2 Domatic Numbers of Interval Graphs

Another interesting usage of the interval ordering is the following rewriting for the result on the domatic numbers of interval graphs obtained by Bertossi [24]. He transformed the domatic number problem on interval graphs to a network flow problem as follows. This is a typical example of the transformation method.

First, Bertossi's method is described as follows. Suppose $G = (V, E)$ is an interval graph, whose vertex set $V = \{1, 2, \dots, n\}$, with an interval model $\{[a_i, b_i] : 1 \leq i \leq n\}$. Without loss of generality, assume that

Fig. 8 The construction of H for an interval graph of domatic number 2



no two intervals share a common endpoint and $a_1 < a_2 < \dots < a_n$.

Two “dummy” vertices 0 and $n + 1$ are added to the graph with $b_0 < a_1$ and $b_n < a_{n+1}$. Then an acyclic directed network H is constructed as follows. The vertex set of H is $\{0, 1, 2, \dots, n, n + 1\}$, and there is a directed edge (i, j) in H if and only if $j \in P(i) \cup Q(i)$, where

$$P(i) = \{k : a_i < a_k < b_i < b_k\} \text{ and}$$

$$Q(i) = \{k : b_i < a_k \text{ and there is no } h \text{ with } b_i < a_h < b_h < a_k\}.$$

Figure 8 shows an example of H .

Bertossi then proved that any path from vertex 0 to vertex $n + 1$ in H corresponds to a proper dominating set of G and vice versa. His arguments have a flaw. In fact this statement is not true as $0, 2, 3, 5, 6$ is a path in the directed network H in Fig. 8, but its corresponding dominating set $\{2, 3, 5\}$ has a proper subset $\{2\}$ that is also a dominating set. To be precise, he only showed that

- (P1) a 0- $(n + 1)$ path in H corresponds to a dominating set of G , and
- (P2) a proper dominating set of G corresponds to a 0- $(n + 1)$ path in H .

Besides, the entire arguments can be treated in terms of the interval ordering as follows. Now assume that $[0, 1, 2, \dots, n, n + 1]$ is an interval ordering of the graph $G = (V, E)$ with two isolated vertices 0 and $n + 1$ added. Then construct a directed network H' with

vertex set $\{0, 1, 2, \dots, n, n + 1\}$ and

edge set $\{ij : i < j \text{ and } (i < h < j \text{ imply } ih \in E \text{ or } hj \in E)\}$.

Notice that H' is not the same as H . However, statements (P1) and (P2) remain true if H is replaced by H' . Also, there is a simpler proof using property (IO), which is different from their original proof using the endpoints of the intervals. The argument is as follows:

First, a 0- $(n + 1)$ path $P : 0 = i_0, i_1, \dots, i_r, i_{r+1} = n + 1$ in H certainly corresponds to a dominating set $D = \{i_0, i_1, \dots, i_r, i_{r+1}\}$ of G by the definition of the edge set of H' . This proves (P1). Conversely, for a proper dominating set D of G , consider the corresponding path P . For any $0 \leq s \leq r$, suppose $i_s < h < i_{s+1}$. Since D is a dominating set of G , there exists some $i_j \in D$ such that $hi_j \in E$.

Case 1. $j \leq s$. Then $i_s h \in E$ by (IO).

Case 2. $j = s + 1$. Then $hi_{s+1} \in E$.

Case 3. $j > s + 1$. If $N[i_{s+1}] \subseteq N[i_j]$, then $D \setminus \{v_{s+1}\}$ is a dominating set of G , violating that D is a proper dominating set. So, there is some vertex $k \in N[i_{s+1}] \setminus N[i_j]$. The case of $i_{s+1} < i_j < k$ or $h < k < i_j$ implies $k \in N[i_j]$ by (IO), a contraction. The case of $k < h < i_{s+1}$ implies $hi_{s+1} \in E$.

In any case, $i_s i_{s+1}$ is an arc in H . This proves (P2).

Primal–dual approaches were also used in [156, 187] to solve the domatic number problem in interval graphs. Manacher and Mankus [158] made it possible to get an $O(n)$ algorithm for the problem. Peng and Chang [168] used the primal–dual method to get a linear-time algorithm for the problem in strongly chordal graphs; see Sect. 6.3.

4.3 Weighted Independent Domination in Interval Graphs

There are many algorithms for variants of domination in interval graphs; see [16, 19, 48, 50, 53, 174, 175]. Among them, Ramalingam and Pandu Rangan [175] gave a unified approach to the weighted independent domination, the weighted domination, the weighted total domination, and the weighted connected domination problems in interval graphs by using the interval orderings. Their algorithms are demonstrated in this and the following subsections.

Now, suppose $G = (V, E)$ is an interval graph with vertex set $V = \{1, 2, \dots, n\}$, where $[1, 2, \dots, n]$ is an interval ordering of G . Assume that each vertex is associated with a real number as its weight. Notice that except for independent domination, according to Lemma 1, assume that the weights are nonnegative. Consider the following notation:

$V_i = \{1, 2, \dots, i\}$ and G_i denotes the subgraph $G[V_i]$ induced by V_i .

V_0 is the empty set.

$\text{low}(i) = \text{minimum element in } N[i]$.

$\text{maxlow}(i) = \max\{\text{low}(j) : \text{low}(i) \leq j \leq i\}$.

$L_i = \{\text{maxlow}(i), \text{maxlow}(i) + 1, \dots, i\}$.

$M_i = \{j : j > i \text{ and } j \text{ is adjacent to } i\}$.

For any family X of sets of vertices, $\min\{X\}$ denotes a minimum weighted set in X . If X is the empty set, then $\min\{X\}$ denotes a set of infinite weight.

Notice that the vertices in the set $\{1, 2, \dots, \text{low}(i) - 1\}$ are not adjacent to i and the vertices in $\{\text{low}(i), \text{low}(i) + 1, \dots, i\}$ are adjacent to i . The vertices in L_i form a maximal clique in the graph G_i . Let j be the vertex such that $\text{low}(i) \leq j \leq i$ and $\text{maxlow}(i) = \text{low}(j)$. Then, $\text{low}(i) \leq \text{low}(j) \leq j \leq i$. It can easily be seen that $N[j]$ is a subset of $L_i \cup M_i$. Furthermore, in G_i , j is adjacent only to the vertices in L_i .

Having all of these, it is ready to establish the solutions to weighted independent domination problem in interval graphs.

Let ID_i denote an independent dominating set of the graph G_i , and let MID_i denote the minimum weighted ID_i .

Notice that, in G_i , the set L_i is a maximal clique and that there is a vertex in L_i which is not adjacent to any vertex in $V_i \setminus L_i$. Hence, any ID_i contains exactly one vertex j in L_i . Furthermore, it is necessary and sufficient that $ID_i \setminus \{j\}$ dominates $V_{\text{low}(j)-1}$ and contains no vertex adjacent to j . Hence, $ID_i \setminus \{j\}$ is an independent dominating set of $G_{\text{low}(j)-1}$. In other words, a set is an ID_i if and only if it is of the form $ID_{\text{low}(j)-1} \cup \{j\}$ for some j in L_i .

These give the following lemma:

Lemma 4 (a) $MID_0 = \emptyset$. (b) For $1 \leq i \leq n$,

$$MID_i = \min\{MID_{\text{low}(j)-1} \cup \{j\} : j \in L_i\}.$$

A linear-time algorithm for the weighted independent domination problem in interval graphs then follows. The detailed description is omitted as it is easy. Similarly, in the following three subsections, only recursive formulas for variations of domination in interval graphs are presented.

4.4 Weighted Domination in Interval Graphs

Let D_i denote a subset of V that dominates V_i . Unlike independent domination, it is not necessary to restrict D_i as a subset of V_i . Let MD_i denote a minimum weighted D_i .

Since there is a vertex in L_i whose neighbors are all in $L_i \cup M_i$, the set D_i contains some vertex j in $L_i \cup M_i$. It is necessary and sufficient that the remaining set $D_i \setminus \{j\}$ dominates $V_{\text{low}(j)-1}$ since j dominates all vertices in $V_i \setminus V_{\text{low}(j)-1}$ and no vertex in $V_{\text{low}(j)-1}$. (Note that if $j \in L_i \cup M_i$, then $\text{low}(j) \leq i$). In other words, a set is a D_i if and only if it is of the form $D_{\text{low}(j)-1} \cup \{j\}$ for some j in $L_i \cup M_i$.

These give the following lemma:

Lemma 5 (a) $MD_0 = \emptyset$. (b) For $1 \leq i \leq n$,

$$MD_i = \min\{MD_{\text{low}(j)-1} \cup \{j\} : j \in L_i \cup M_i\}.$$

4.5 Weighted Total Domination in Interval Graphs

Let TD_i denote a subset of V that totally dominates V_i and let MTD_i be a minimum weighted TD_i . Let PD_i denote a subset of V that totally dominates $\{i\} \cup V_{\text{low}(i)-1}$, and let MPD_i be a minimum weighted PD_i .

As in domination, TD_i also includes some vertex j in $L_i \cup M_i$. If $j \in L_i$, then it is necessary and sufficient that the set $\text{TD}_i \setminus \{j\}$ totally dominates $V_{\text{low}(j)-1} \cup \{j\}$. If $j \in M_i$, then it is necessary and sufficient that the set $\text{TD}_i \setminus \{j\}$ totally dominates $V_{\text{low}(j)-1}$.

Similarly, any PD_i includes some vertex j adjacent to i . By the definition, $j \geq \text{low}(i)$. Hence, it is necessary and sufficient that the $\text{PD}_i \setminus \{j\}$ totally dominates $V_{\min\{\text{low}(i)-1, \text{low}(j)-1\}}$.

These give the following lemma:

Lemma 6 (a) $\text{MTD}_0 = \emptyset$. (b) For $1 \leq i \leq n$,

$$\text{MTD}_i = \min \left(\{ \text{MPD}_j \cup \{j\} : j \in L_i \} \bigcup \{ \text{MTD}_{\text{low}(j)-1} \cup \{j\} : j \in M_i \} \right).$$

$$\text{MPD}_i = \min \{ \text{MTD}_{\min\{\text{low}(j)-1, \text{low}(i)-1\}} \cup \{j\} : j \in N(i) \}.$$

Note that in the original paper by Ramalingam and Pandu Rangan [175], there is a typo that uses $j \in N[i]$ rather than $j \in N(i)$ in the formula for MPD_i .

4.6 Weighted Connected Domination in Interval Graphs

Let CD_i denote a connected dominating set of G_i that contains the vertex i , and let MCD_i denote a minimum weighted CD_i .

If $\text{low}(i) = 1$, then MCD_i is $\{i\}$ since all vertices have nonnegative weights. If $\text{low}(i) > 1$, then any CD_i contains vertices other than i and hence some vertex adjacent to i in G_i . Let j be the maximum vertex in $\text{CD}_i \setminus \{i\}$. Assume that $\text{low}(j) < \text{low}(i)$. Otherwise j is removed to get a CD_i of the same or lower weight. If $\text{low}(j) < \text{low}(i)$, then any other vertex of G_j adjacent to i is also adjacent to j . Thus, it is necessary and sufficient that $\text{CD}_i \setminus \{i\}$ is a CD_j .

These give the following lemma:

Lemma 7 (a) If $\text{low}(i) = 1$, then $\text{MCD}_i = \{i\}$. (b) For $\text{low}(i) > 1$,

$$\text{MCD}_i = \min \{ \text{MID}_j \cup \{i\} : j \in N[i] \text{ and } j < i \text{ and } \text{low}(j) < \text{low}(i) \}.$$

(c) $\min \{ \text{MCD}_n : i \in L_n \}$ is a minimum weighted connected dominating set of the graph G .

5 Chordal Graphs and NP-Completeness Results

5.1 Perfect Elimination Orderings of Chordal Graphs

It was seen in previous sections that the domination problem is well solved for trees and interval graphs. People then try to generalize the results for general graphs. However, the NP-completeness theory raised by Cook suggests that this is in general quite impossible. Garey and Johnson, in an unpublished paper (see [99]), pointed out that the dominating problem is NP-complete by transforming the vertex cover problem to it.

Vertex Cover

Instance: A graph $G = (V, E)$ and a positive integer $k \leq |V|$.

Question: Is there a subset $C \subseteq V$ of size at most k such that each edge xy of G has either $x \in C$ or $y \in C$?

Their idea can in fact be modified to prove many NP-complete results for the domination problem and its variations in many classes of graphs. Among these classes, the class of chordal graphs is most interesting in the study of many graph optimization problems. Chordal graphs are raised in the theory of perfect graphs; see [101]. It contains trees, interval graphs, directed path graphs, split graphs, undirected path graphs, etc., as subclasses.

Recall that a graph is chordal if every cycle of length at least four has a chord. The following property is an important characterization of chordal graphs. The proof presented here is from Theorem 4.3 in [102], except that the maximum cardinality search is not introduced explicitly.

Theorem 12 *A graph $G = (V, E)$ is chordal if and only if it has a perfect elimination ordering which is an ordering $[v_1, v_2, \dots, v_n]$ of V such that*

$$i < j < k \text{ and } v_i v_j, v_i v_k \in E \text{ imply } v_j v_k \in E. \quad (\text{PEO})$$

Proof (\Rightarrow) For any ordering $\sigma = [v_1, v_2, \dots, v_n]$, define the vector

$$d(\sigma) = (d_n, d_{n-1}, \dots, d_1),$$

where each $d_s = |\{t : t > s \text{ and } v_t \text{ is adjacent to } v_s\}|$. Choose an ordering σ such that $d(\sigma)$ is lexicographically largest.

Suppose $p < q < r$ and $v_r \in N(v_p) \setminus N(v_q)$. Consider the ordering σ' obtained from σ by interchanging v_p and v_q . Then $d'_s = d_s$ for all $s > q$ and

$$|\{t : t > q \text{ and } v_t \in N(v_p)\}| = d'_q \leq d_q = |\{t : t > q \text{ and } v_t \in N(v_q)\}|.$$

However, r is a vertex such that $r > q$ and $v_r \in N(v_p) \setminus N(v_q)$. Then, there exists some $s > q$ such that $v_s \in N(v_q) \setminus N(v_p)$. This gives

$p < q < r, v_r \in N(v_p) \setminus N(v_q)$ imply $v_s \in N(v_q) \setminus N(v_p)$ for some $s > q$. (*)

Next, (*) is used to prove the following claim:

Claim There does not exist any chordless path $P : v_{i_1}, v_{i_2}, \dots, v_{i_x}$ with $x \geq 3$ and $i_y < i_x < i_1$ for $1 < y < x$.

[Notice that the claim implies (PEO) as v_k, v_i, v_j is a chordless path with $i < j < k$ whenever $v_j v_k \notin E$.] Suppose to the contrary that such a path P exists. Choose one with a largest i_x . Since $i_2 < i_x < i_1$ and $v_{i_1} \in N(v_{i_2}) \setminus N(v_{i_x})$, by (*), there exists some $i_{x+1} > i_x$ such that $v_{i_{x+1}} \in N(v_{i_x}) \setminus N(v_{i_2})$. Let z be the minimum index such that $z \geq 2$ and $v_{i_{x+1}} v_z \in E$. Note that z exists and $z \geq 3$. For the case when $v_{i_1} v_{i_{x+1}} \notin E$, $P' : v_{i_1}, v_{i_2}, \dots, v_{i_{z-1}}, v_{i_z}, v_{i_{x+1}}$ (or its inverse) is a chordless path of length at least three with $i_y < i_{x+1} < i_1$ (or $i_y < i_1 < i_{x+1}$) for $1 < y \leq z$. In this case, $i_x < i_{x+1}$ is a contradiction to the choice of P . For the case when $v_{i_1} v_{i_{x+1}} \in E$, P' together with the edge $v_{i_1} v_{i_{x+1}}$ form a chordless cycle of length at least four, a contradiction to the fact that G is chordal.

(\Leftarrow) On the other hand, suppose (PEO) holds. For any cycle of length at least four, choose the vertex of the cycle with the least index. By (PEO), the two neighbors of this vertex in the cycle are adjacent. \square

5.2 NP-Completeness for Domination

This section first demonstrates Garey and Johnson's proof that the domination problem is NP-complete. The proof is adapted for split graphs, which are special chordal graphs. A *split graph* is a graph whose vertex set is the disjoint union of a clique C and a stable set S . Notice that a split graph is chordal as the ordering with the vertices in S first and the vertices in C next gives a perfect elimination ordering.

Theorem 13 *The domination problem is NP-complete for split graphs.*

Proof The proof is given by transforming the vertex cover problem in general graphs to the domination problem in split graphs. Given a graph $G = (V, E)$, construct the graph $G' = (V', E')$ with

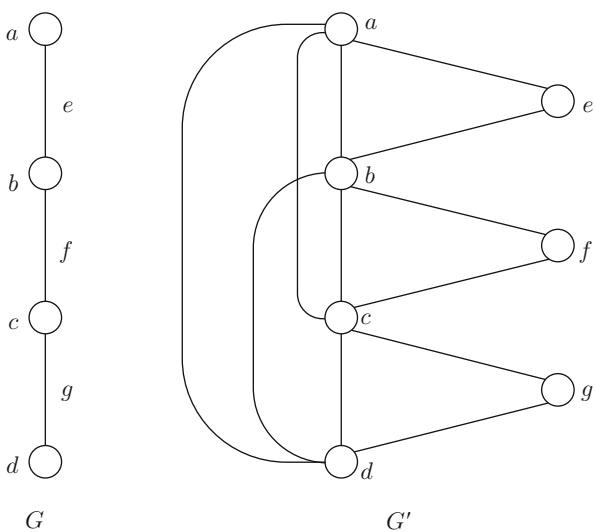
$$\text{vertex set } V' = V \cup E \text{ and}$$

$$\text{edge set } E' = \{v_1 v_2 : v_1 \neq v_2 \text{ in } V\} \cup \{ve : v \in e\}.$$

Notice that G' is a split graph whose vertex set V' is the disjoint union of the clique V and the stable set E (Fig. 9).

If G has a vertex cover C of size at most k , then C is a dominating set of G' of size at most k , by the definition of G' . On the other hand, suppose G' has a dominating set D of size at most k . If D contains any $e \in E$, say $e = uv$, then

Fig. 9 A transformation to a split graph



replace e with u to get a new dominating set of size at most k . In this way, assume that D is a subset of V . It is then clear that D is a vertex cover of G of size at most k .

Since the vertex cover problem is NP-complete, the domination problem is also NP-complete for split graphs. \square

Note that the dominating set of G' in the proof above in fact induces a connected subgraph.

Corollary 1 *The total and the connected domination problems are NP-complete for split graphs.*

In fact, the proof can be modified to get

Theorem 14 *The domination problem is NP-complete for bipartite graphs.*

Proof The vertex cover problem in general graphs is transformed to the domination problem in bipartite graphs as follows. Given a graph $G = (V, E)$, construct the graph $G' = (V', E')$ with

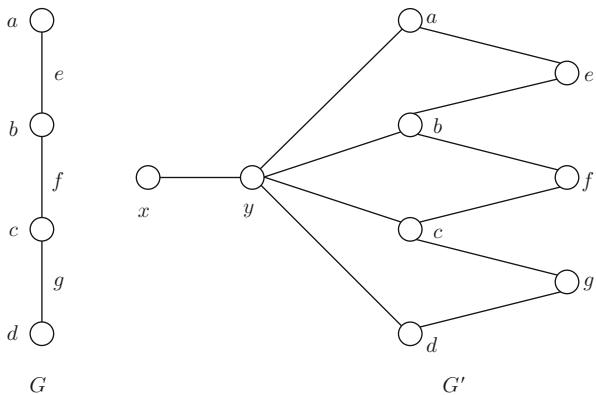
$$\text{vertex set } V' = \{x, y\} \cup V \cup E \text{ and}$$

$$\text{edge set } E' = \{xy\} \cup \{vv : v \in V\} \cup \{ve : v \in e\}.$$

Notice that G' is a bipartite graph whose vertex set V' is the disjoint union of two stable sets $\{x\} \cup V$ and $\{y\} \cup E$ (Fig. 10).

If G has a vertex cover C of size at most k , then $\{y\} \cup C$ is a dominating set of G' of size at most $k + 1$. On the other hand, suppose G' has a dominating set D of

Fig. 10 A transformation to a bipartite graph



size at most $k + 1$. Since $N_{G'}[x] = \{x, y\}$, D must contain x or y . One may assume that D contains y but not x , as $(D \setminus \{x\}) \cup \{y\}$ is also a dominating set of size at most $k + 1$. Since $y \in D$, if D contains any $e \in E$, say $e = uv$, one can replace e with u to get a new dominating set of size at most $k + 1$. In this way, one may assume that $D \setminus \{y\}$ is a subset of V . It is then clear that $D \setminus \{y\}$ is a vertex cover of G of size at most k .

Since the vertex cover problem is NP-complete, the domination problem is NP-complete for bipartite graphs. \square

Corollary 2 *The total and the connected domination problems are NP-complete for bipartite graphs.*

There are many other NP-complete results for variations of domination; see [12, 15, 25, 66, 76, 80, 96, 99, 114, 126, 131, 132, 165, 201, 202]. Most of the proofs are more or less similar to the above two. Only very few are proved by using different methods. As an example, Booth and Johnson [25] proved that the domination problem is NP-complete for undirected path graphs, which is another subclass of chordal graphs, by reducing the three-dimensional matching problem to it.

Theorem 15 *The domination problem is NP-complete for undirected path graphs.*

Proof Consider an instance of the three-dimensional matching problem, in which there are three disjoint sets W , X , and Y each of size q and a subset

$$M = \{m_i = (w_r, x_s, y_t) : w_r \in W, x_s \in X \text{ and } y_t \in Y \text{ for } 1 \leq i \leq p\}$$

of $W \times X \times Y$ having size p . The problem is to find a subset M' of M having size exactly q such that each $w_r \in W$, $x_s \in X$, and $y_t \in Y$ occurs in precisely one triple of M' .

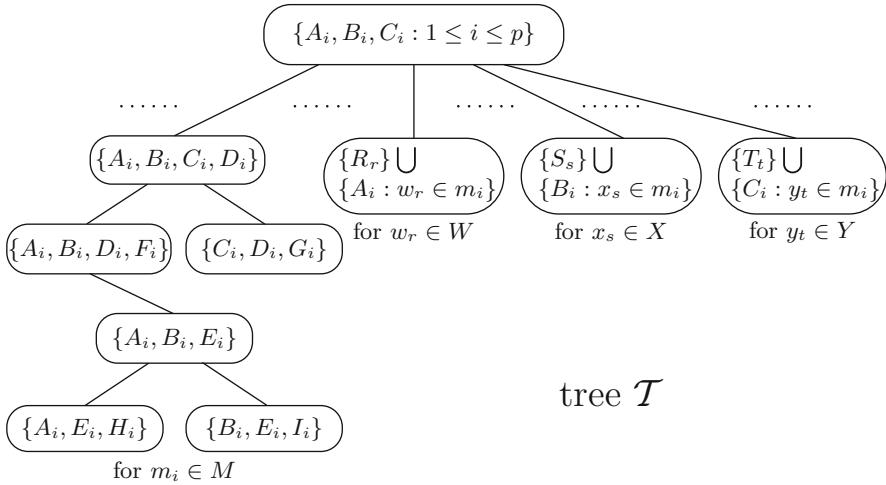


Fig. 11 A transformation to an undirected path graph

Given an instance of the three-dimensional matching problem, construct a tree \mathcal{T} having $6p + 3q + 1$ vertices from which an undirected path graph G is obtained. The vertices of the tree, which are represented by sets, are explained below.

For each triple $m_i \in M$, there are six vertices that depend only upon the triple itself and not upon the elements within the triple:

$$\begin{aligned}
 & \{A_i, B_i, C_i, D_i\} \\
 & \{A_i, B_i, D_i, F_i\} \\
 & \{C_i, D_i, G_i\} \\
 & \{A_i, B_i, E_i\} \\
 & \{A_i, E_i, H_i\} \\
 & \{B_i, E_i, I_i\} \quad \text{for } 1 \leq i \leq p.
 \end{aligned}$$

These six vertices form the subtree of \mathcal{T} corresponding to m_i , which is illustrated in Fig. 11. Next, there is a vertex for each element of W , X , and Y that depends upon the triples of M to which each respective element belongs:

$$\begin{aligned}
 & \{R_r\} \cup \{A_i : w_r \in m_i\} \quad \text{for } w_r \in W, \\
 & \{S_s\} \cup \{B_i : x_s \in m_i\} \quad \text{for } x_s \in X, \\
 & \{T_t\} \cup \{C_i : y_t \in m_i\} \quad \text{for } y_t \in Y.
 \end{aligned}$$

Finally, $\{A_i, B_i, C_i : 1 \leq i \leq p\}$ is the last vertex of \mathcal{T} . The arrangement of these vertices in the tree \mathcal{T} is shown in Fig. 11. This then results in an undirected path graph G with vertex set

$$\{A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i, I_i : 1 \leq i \leq p\} \cup \{R_j, S_j, T_j : 1 \leq j \leq q\}$$

of size $9p + 3q$, where the undirected path in \mathcal{T} corresponding to a vertex v of G consists of those vertices (sets) containing v in the tree \mathcal{T} .

The theorem then follows from the claim that G has a dominating set of size $2p + q$ if and only if the three-dimensional matching problem has a solution.

Suppose D is a dominating set of G of size $2p + q$. Observe that for any i , the only way to dominate the vertex set $\{A_i, B_i, C_i, D_i, E_i, F_i, G_i, H_i, I_i\}$ corresponding to m_i with two vertices is to choose D_i and E_i and that any larger dominating set might just as well consist of A_i, B_i , and C_i , since none of the other possible vertices dominate any vertex outside of the set. Consequently, D consists of A_i, B_i , and C_i for tm_i 's, and D_i and E_i for $p - t$ other m_i 's, and at least $\max\{3(q - t), 0\}$ R_r, S_s, T_t . Then,

$$2p + q = |D| \geq 3t + 2(p - t) + 3(q - t) = 2p + 3q - 2t$$

and so $t \geq q$. Picking q triples m_i for which A_i, B_i , and C_i are in D form a matching M' of size q .

Conversely, suppose the three-dimensional matching problem has a solution M' of size q . Let

$$D = \{A_i, B_i, C_i : m_i \in M'\} \cup \{D_i, E_i : m_i \in M \setminus M'\}.$$

It is straightforward to check that D is a dominating set of G of size $3q + 2(p - q) = 2p + q$. \square

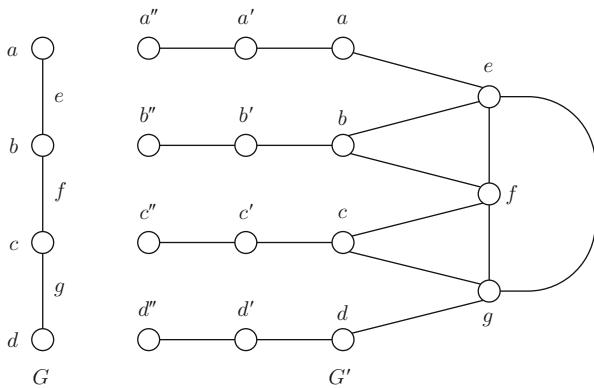
5.3 Independent Domination in Chordal Graphs

Farber [91] showed a surprising result that the independent domination problem is solvable by using a linear programming method. On the other hand, it is known [64] that the weighted independent domination problem is NP-complete. The proof has the same spirit of the proof of Theorem 13.

Theorem 16 *The weighted independent domination problem is NP-complete for chordal graphs.*

Proof The vertex cover problem in general graphs is transformed to the weighted independent domination problem in chordal graphs as follows. Given a graph $G = (V, E)$, construct the following chordal graph $G' = (V', E')$ with

Fig. 12 A transformation to a weighted chordal graph



vertex set $V' = \{v'', v' : v \in V\} \cup E$ and

edge set $E' = \{v''v', v'v : v \in V\} \cup \{ve : v \in e\} \cup \{e_1e_2 : e_1 \neq e_2 \text{ in } E\}$.

The weight of each $e \in E$ is $2|V| + 1$ and the weight of each vertex in V' is 1 (Fig. 12).

If G has a vertex cover C of size at most k , then $\{v'', v : v \in C\} \cup \{v' : v \in V \setminus C\}$ is an independent dominating set of G' with weight at most $|V| + k$. On the other hand, suppose G' has an independent dominating set D of weight at most $|V| + k$. As $k \leq |V|$, D contains no elements in E . Let $C = D \cap V$. It is clear that C is a vertex cover of G . Also, for each $v \in V$, the set D contains exactly one vertex in $\{v'', v'\}$. Thus, C is of size at most k .

Since the vertex cover problem is NP-complete, the weighted independent domination problem is NP-complete for chordal graphs. \square

Farber's algorithm for the independent domination problem in chordal graphs is by means of a linear programming method. Suppose $G = (V, E)$ is a chordal graph with a perfect elimination ordering $[v_1, v_2, \dots, v_n]$ and vertex weights w_1, w_2, \dots, w_n of real numbers. Write

$$i \sim j \text{ for } v_i \in N[v_j],$$

$$i \tilde{\sim} j \text{ for } i \sim j \text{ and } i \leq j,$$

$$i \tilde{\succ} j \text{ for } i \sim j \text{ and } i \geq j.$$

It follows from the definition of a perfect elimination ordering that each

$$C_j = \{v_i : i \tilde{\succ} j\}$$

is a clique. Thus, a set S of vertices is independent if and only if each C_j contains at most one vertex of S . Also, S is a dominating set if and only if for each j , S contains at least one vertex v_i with $i \sim j$. Consider the following linear problem:

$$\begin{aligned}
P_1(G, w) : \text{Minimize } & \sum_{i=1}^n w_i x_i, \\
\text{subject to } & \sum_{i \sim j} x_i \geq 1 \text{ for each } j, \\
& \sum_{i \sim j} x_i \leq 1 \text{ for each } j, \\
& x_i \geq 0 \quad \text{for each } i.
\end{aligned}$$

It follows from the above comments that there is a one-to-one correspondence between independent dominating sets of G and feasible 0–1 solutions to $P_1(G, w)$. Moreover, an optimal 0–1 solution to $P_1(G, w)$ corresponds to a minimum weighted independent dominating set in G . Notice that a set of vertices of G is an independent dominating set in G if and only if it is a maximal independent set in G . Consequently, there exist independent dominating sets, and $P_1(G, w)$ is a feasible linear program. It will follow from the algorithm presented below that if every vertex of G has a weight in $\{1, 0, -1, -2, \dots\}$, then $P_1(G, w)$ has an optimal 0–1 solution and that the following dual program has an integer optimal solution:

$$\begin{aligned}
D_1(G, w) : \text{Maximize } & \sum_{j=1}^n (y_j - z_j), \\
\text{subject to } & \sum_{j \sim i} y_j - \sum_{j \sim i} z_j \leq w_i \text{ for each } i, \\
& y_j, z_j \geq 0 \quad \text{for each } j.
\end{aligned}$$

If, however, not all vertex weights are in $\{1, 0, -1, -2, \dots\}$ (even though they are all integers), then it may be the case that neither $P_1(G, w)$ nor $D_1(G, w)$ has an integer solution. For the remainder of this section, assume that each vertex has a weight in $\{1, 0, -1, -2, \dots\}$.

First define two functions which simplify the presentation of the algorithm. For each i , let

$$\begin{aligned}
f(i) &= w_i + \sum_{j \sim i} z_j - \sum_{j \sim i} y_j, \\
g(i) &= w_i + \sum_{j \sim i} z_j - \sum_{j \sim i} y_j.
\end{aligned}$$

Note that $f(i)$ is the slack in the dual constraint associated with vertex v_i .

The algorithm to locate a minimum weighted dominating set in G has two stages. Stage 1 finds a feasible solution to $D_1(G, w)$ by scanning the vertices in the order v_1, v_2, \dots, v_n ; and stage 2 uses this solution to find a feasible 0–1 solution to $P_1(G, w)$ by scanning the vertices in the order v_n, v_{n-1}, \dots, v_1 . Initially, each $y_j = 0$, each $z_j = 0$, and each $x_i = 2$. (The interpretation of $x_i = 2$ is that x_i has not yet been assigned a value). If, at the time v_j is scanned in stage one, the dual constraint associated with v_j is violated, that is, if $f(j) < 0$, then add just enough to z_j to bring that constraint into feasible. Otherwise, add as much as possible to y_j without violating the dual constraint associated with v_j or with any previously scanned vertex. In stage two, if $x_i = 2$ and $g(i) = 0$ when v_i is scanned, then let $x_i = 1$ and let $x_j = 0$ for each v_j adjacent to v_i .

A formal description of the algorithm is given below.

Algorithm IndDomChordal. Determine $\gamma_i(G, w)$ for a chordal graph G with weights w_1, w_2, \dots, w_n in $\{1, 0, -1, -2, \dots\}$.

Input. A chordal graph G with a perfect elimination ordering $[v_1, v_2, \dots, v_n]$ and vertex weights w_1, w_2, \dots, w_n in $\{1, 0, -1, -2, \dots\}$.

Output. Optimal solutions to $P_1(G, w)$ and $D_1(G, w)$.

Method.

each $y_j \leftarrow 0$, each $z_j \leftarrow 0$ and each $x_i \leftarrow 2$;

Stage 1: **for** $j = 1$ **to** n **do**

if $f(j) < 0$ **then** $z_j \leftarrow -f(j)$
else $y_j \leftarrow \min\{f(k) : k \gtrsim j\}$;

Stage 2: **for** $i = n$ **to** 1 **by** -1 **do**

if $x_i = 2$ and $g(i) = 0$ **then**
 $x_i \leftarrow 1$;
 $x_j \leftarrow 0$ for each $v_j \in N(v_i)$;
end if.

The validity of the algorithm is established below.

Theorem 17 *Algorithm IndDomChordal finds a minimum weighted independent dominating set of a chordal graph with vertex weights in $\{1, 0, -1, -2, \dots\}$.*

Several lemmas are needed. Note that since the weights are integral, all $x_i, y_j, z_j, f(i), g(i)$ are integral at any time.

Lemma 8 *For each j , $f(j) \geq 0$ at all times after scanning v_j in stage 1.*

Proof The fact that $f(j) \geq 0$ immediately after scanning v_j in stage 1 follows from the choice of z_j and y_j . The fact that $f(j) \geq 0$ after scanning each v_k where $k > j$ follows from the choice of y_k . \square

Lemma 9 *At the end of stage one, $y_j \geq 0$ for any j .*

Proof This is an immediate consequence of Lemma 8. \square

Lemma 10 *At the end of stage 1, $0 \leq f(j) \leq g(j)$ for any j .*

Proof This follows from Lemmas 8 and 9. \square

Lemma 11 *At the end of stage 1, for each i , there is a $j \gtrsim i$ such that $g(j) = 0$.*

Proof According to Lemma 10, $g(i) \geq 0$. If $g(i) = 0$, then the lemma is true as j can be chosen to be i . Suppose $g(i) > 0$. Then $f(i)$ was positive immediately after scanning v_i , and so, by the choice of z_i and y_i , $z_i = 0$ and y_i was chosen to force $f(j)$ to be 0 for some $j \gtrsim i$. Thus, there is some $j \gtrsim i$ such that

$$\sum_{k \sim j, k \leq i} y_k - \sum_{k \sim j} z_j = w_j.$$

If $y_k = 0$ for each k such that $k \sim j$ and $k \leq i$, then $g(j) = 0$. Otherwise, choose k such that $y_k > 0$, $k \sim j$, and $k \leq i$. Then $i \sim k$ since $i \sim j$, $k \sim j$, and $[v_1, v_2, \dots, v_n]$ is a perfect elimination ordering. Hence, $k \sim i$ since $i \geq k$ and $i \sim k$. Since all vertex weights are integral, y_k is an integer. Thus, $y_k \geq w_i$, since $w_i \in \{1, 0, -1, -2, \dots\}$. Now, $g(i) > 0$, $y_k \geq w_i$, and $k \sim i$, and hence, there is some $\ell \sim i$ such that $z_\ell > 0$. By the choice of z_ℓ , it is the case that $g(\ell) = 0$. \square

Note that the proof of [Lemma 11](#) relies upon the fact that G has vertex weights in $\{1, 0, -1, -2, \dots\}$ to show that if $y_k > 0$, then $y_k \geq w_i$.

Lemma 12 *At the end of stage 2, for each j , if $g(j) = 0$, then $x_k = 1$ for some $k \sim j$.*

Proof It is clear from the instructions in stage two that, for each i , if $x_i = 1$ at any time during the algorithm, then $x_i = 1$ at the end of the algorithm. Suppose $g(j) = 0$. If x_j was 2 just prior to scanning v_j in stage two, then x_j was assigned the value of 1 when v_j was scanned, and hence, $x_j = 1$ at the end of the algorithm. In this case, choose $k = j$ as desired. Otherwise, x_j was 0 prior to scanning v_j . In that case, $x_j = 0$ by virtue of the fact that $x_k = 1$ for some previously scanned neighbor v_k of v_j , that is, $x_k = 1$ for some $k \sim j$. \square

Proof of Theorem 17. It is easy to see that Algorithm IndDomChordal halts after $O(|V| + |E|)$ operations. Next is to show that the final values of x_1, x_2, \dots, x_n and $y_1, y_2, \dots, y_n, z_1, z_2, \dots, z_n$ are feasible solutions to $P_1(G, w)$ and $D_1(G, w)$, respectively, and then to verify that these solutions satisfy the conditions of complementary slackness. It then follows that they are optimal.

- (i) **Feasibility of dual solution:** Clearly $z_j \geq 0$ for each j . By [Lemmas 8](#) and [9](#), $y_j \geq 0$ and $f(j) \geq 0$ for each j .
- (ii) **Feasibility of primal solution:** The instructions in stage two guarantee that if $x_i = x_j = 1$, then $v_i v_j \notin E$, and if $x_i = 0$, then $x_j = 1$ for some $j \sim i$. Since a 0–1 primal solution is feasible if and only if the set $\{v_j : x_j = 1\}$ is an independent dominating set in G , it suffices to show that each x_i is either 0 or 1. According to [Lemma 11](#), for each i , there is a $j \sim i$ such that $g(j) = 0$. According to [Lemma 12](#), there is a $k \sim j$ such that $x_k = 1$. Since $i \sim j$, $k \sim j$, and $[v_1, v_2, \dots, v_n]$ is a perfect elimination ordering, it follows that $i \sim k$. If $i = k$, then $x_i = 1$; otherwise $x_i = 0$.
- (iii) **Complementary slackness:** If $x_i > 0$, then $g(i) = 0$ by the choice of x_i , and hence, $f(i) = 0$, by [Lemma 10](#). Thus, $\sum_{j \sim i} y_j - \sum_{j \sim i} z_j = w_i$. If $z_j > 0$, then $g(j) = 0$ by the choice of z_j , and so $x_k = 1$ for some $k \sim j$, by [Lemma 12](#). Hence, $\sum_{i \sim j} x_i \geq 1$. Equality follows from the fact that the primal solution is feasible.

Suppose $y_j > 0$ but $x_i = x_k = 1$ for $i \sim j$ and $k \sim j$. Since $x_i = x_k = 1$, $g(i) = g(k) = 0$, and so $j \leq \min\{i, k\}$ by [Lemma 8](#), [Lemma 10](#), and the choice

of y_j . Thus, $i \tilde{>} j$ and $k \tilde{>} j$, which implies $i \sim k$ since $[v_1, v_2, \dots, v_n]$ is a perfect elimination ordering. On the other hand, $v_i v_k \notin E$ since $x_i = x_k = 1$. Consequently $i = k$. Hence, if $y_j > 0$, then $\sum_{i \sim j} x_i \leq 1$. Equality follows from the fact that the primal solution is feasible. \square

6 Strongly Chordal Graphs

6.1 Strong Elimination Orderings of Strongly Chordal Graphs

Strongly chordal graphs were introduced by several people [44, 93, 120] in the study of domination. In particular, most variations of the domination problem are solvable in this class of graphs. There are many equivalent ways to define them. This section adapts the notation from Farber's paper [93].

A graph $G = (V, E)$ is *strongly chordal* if it admits a *strong elimination ordering* which is an ordering $[v_1, v_2, \dots, v_n]$ of V such that the following two conditions hold:

- (a) If $i < j < k$ and $v_i v_j, v_i v_k \in E$, then $v_j v_k \in E$.
- (b) If $i < j < k < \ell$ and $v_i v_k, v_i v_\ell, v_j v_k \in E$, then $v_j v_\ell \in E$.

Notice that an ordering satisfying condition (a) is a perfect elimination ordering. Hence, every strong elimination ordering is a perfect elimination ordering, and every strongly chordal graph is chordal. The notion $i \sim j$ stands for $v_i \in N[v_j]$.

The following equivalent definition of the strong elimination ordering is more convenient in many arguments for strongly chordal graph.

Lemma 13 *An ordering $[v_1, v_2, \dots, v_n]$ of the vertices of G is a strong elimination ordering of G if and only if*

$$i \leq j, k \leq \ell, i \sim k, i \sim \ell \text{ and } j \sim k \text{ imply } j \sim \ell. \quad (\text{SEO})$$

Proof Suppose $[v_1, v_2, \dots, v_n]$ is a strong elimination ordering of G . Suppose that $i \leq j, k \leq \ell, i \sim k, i \sim \ell$, and $j \sim k$. If $i = j$ or $k = \ell$ or $j = \ell$, then clearly $j \sim \ell$. Suppose, on the other hand, that $i < j, k < \ell$, and $j \neq \ell$. By symmetry, assume that $i \leq k$. Now consider three cases:

- Case 1.* Suppose $j = k$. Then $i < j < \ell$ and $v_i v_j, v_i v_\ell \in E$, whence $v_j v_\ell \in E$, by (a) in the definition of a strong elimination ordering.
- Case 2.* Suppose $j < k$. Then $i < j < k < \ell$ and $v_i v_k, v_i v_\ell, v_j v_k \in E$, whence $v_j v_\ell \in E$, by (b) in the definition of a strong elimination ordering.
- Case 3.* Suppose $k < j$. If $i = k$, then $v_k v_\ell \in E$. Otherwise, $i < k < \ell$ and $v_i v_k, v_i v_\ell \in E$, whence $v_k v_\ell \in E$. Consequently, $v_k v_\ell, v_k v_j \in E$ and either $k < \ell < j$ or $k < j < \ell$. In either case, $v_j v_\ell \in E$.

This completes the proof of necessity. The proof of sufficiency is trivial and thus omitted. \square

6.2 Weighted Domination in Strongly Chordal Graphs

There are quite a few algorithms designed for variants of the domination problem in strongly chordal graphs; see [35, 38, 44, 45, 93, 120, 145, 194]. This section presents the linear algorithm, given by Farber [93], for locating a minimum weighted dominating set in a strongly chordal graph.

Let $G = (V, E)$ be a strongly chordal graph with a strong elimination ordering $[v_1, v_2, \dots, v_n]$ and vertex weights w_1, w_2, \dots, w_n . According to Lemma 1, assume that these vertex weights are nonnegative. Consider the following linear problem:

$$\begin{aligned} P_2(G, w) : & \text{Minimize } \sum_{i=1}^n w_i x_i, \\ & \text{subject to } \sum_{j \sim i} x_j \geq 1 \text{ for each } j, \\ & \quad x_i \geq 0 \quad \text{for each } i. \end{aligned}$$

By definition, a set S of vertices of G is a dominating set if and only if, for each j , S contains some vertex v_i such that $i \sim j$. Consequently, there is a one-to-one correspondence between feasible 0–1 solutions to $P_2(G, w)$ and dominating sets in G . Moreover, an optimal 0–1 solution to $P_2(G, w)$ corresponds to a minimum weighted dominating set in G . It follows from the algorithm below that $P_2(G, w)$ has an optimal 0–1 solution and that the following dual program has an optimal solution:

$$\begin{aligned} D_2(G, w) : & \text{Maximize } \sum_{j=1}^n y_j, \\ & \text{subject to } \sum_{j \sim i} y_j \leq w_i \text{ for each } i, \\ & \quad y_j \geq 0 \quad \text{for each } j. \end{aligned}$$

The algorithm presented below solves the linear programs $P_2(G, w)$ and $D_2(G, w)$. To simplify the presentation of the algorithm, define a function h and a family of sets. For each i , let

$$h(i) = w_i - \sum_{j \sim i} y_j \quad \text{and} \quad T_i = \{j : i \sim j \text{ and } y_j > 0\}.$$

Note that $h(i)$ is the slack in the dual constraint associated with vertex v_i , and T_i is the set of constraints in $P_2(G, w)$ containing x_i which must be at equality to satisfy the conditions of complementary slackness.

The algorithm has two stages. Stage 1 finds a feasible solution to $D_2(G, w)$ by scanning the vertices in the order v_1, v_2, \dots, v_n ; and stage 2 uses this solution to find a feasible 0–1 solution to $P_2(G, w)$ by scanning the vertices in the order v_n, v_{n-1}, \dots, v_1 . This algorithm uses a set T to assure that the conditions of complementary slackness are satisfied. Initially, $T = \{1, 2, \dots, n\}$, each $y_j = 0$, and each $x_i = 0$. When v_j is scanned in stage 1, y_j increases as much as possible without violating the dual constraint. In stage 2, if $h(i) = 0$ and $T_i \subseteq T$ when v_i is scanned, then let $x_i = 1$ and replace T by $T \setminus T_i$. Otherwise x_i remains 0. A more formal description of the algorithm follows.

Algorithm WDomSC. Determine $\gamma(G, w)$ for a strongly chordal graph G with nonnegative vertex weights w_1, w_2, \dots, w_n .

Input. A strongly chordal graph G with a strong elimination ordering $[v_1, v_2, \dots, v_n]$ and nonnegative vertex weights w_1, w_2, \dots, w_n .

Output. Optimal solutions to $P_2(G, w)$ and $D_2(G, w)$.

Method.

$T \leftarrow \{1, 2, \dots, n\}$; each $y_j \leftarrow 0$; each $x_i \leftarrow 0$;

Stage 1: **for** $j = 1$ **to** n **do**

$y_j \leftarrow \min\{h(k) : k \sim j\}$;

Stage 2: **for** $i = n$ **to** 1 **by** -1 **do**

if $(h(i) = 0 \text{ and } T_i \subseteq T)$ **then**

$x_i \leftarrow 1$;

$T \leftarrow T \setminus T_i$;

end if.

The algorithm is verified as follows.

Theorem 18 *Algorithm WDomSC finds a minimum weighted dominating set of a strongly chordal graph with nonnegative vertex weights in linear time provided that a strong elimination ordering is given.*

Proof It is easy to see that the algorithm halts after $O(|V| + |E|)$ operations. In order to show that the final values of x_1, x_2, \dots, x_n and $y_1, y_2, \dots, y_n, z_1, z_2, \dots, z_n$ are optimal solutions to $P_2(G, w)$ and $D_2(G, w)$, respectively, it suffices to show that these solutions are feasible and that they satisfy the conditions of complementary slackness.

- (i) **Feasibility of dual solution:** The instructions in stage 1 guarantee that $h(i) \geq 0$ for each i and $y_j \geq 0$ for each j .
- (ii) **Feasibility of primal solution:** Clearly, each x_i is either 0 or 1. Thus, it suffices to show that for each j , $x_i = 1$ for some $i \sim j$. By the choice of y_j , there is a $k \sim j$ such that $h(k) = 0$ and $\max T_k \leq j$. If $x_k = 1$, then the claim holds. Otherwise, by the algorithm, T_k was not contained in T when v_k was scanned in stage 2. Since, in stage 2, the vertices are scanned in the order v_n, v_{n-1}, \dots, v_1 , there is some $\ell > k$ such that $x_\ell = 1$ and $T_\ell \cap T_k \neq \emptyset$. Let $i \in T_\ell \cap T_k$. Then $i \leq j$ since $\max T_k \leq j$. Thus, $i \leq j$, $k < \ell$, $i \sim k$, $i \sim \ell$, and $j \sim k$, whence $\ell \sim j$ by Lemma 13, since $[v_1, v_2, \dots, v_n]$ is a strong elimination ordering. Hence, $\ell \sim j$ and $x_\ell = 1$. Consequently, the primal solution is feasible.
- (iii) **Complementary slackness:** If $x_i > 0$, then $x_1 = 1$ and so $h(i) = 0$, that is, $\sum_{j \sim i} y_j = w_i$.

Suppose $y_j > 0$. It is clear from the instructions that if $x_i = x_k = 1$, then $T_i \cap T_k = \emptyset$. Thus, $\sum_{i \sim j} x_i \leq 1$. Equality follows from the feasibility of the primal solution. \square

Farber in fact also gave an algorithm for the weighted independent domination problem with arbitrary real weights for strongly chordal graphs. The approach is similar to that for chordal graphs (with restricted weights) in Sect. 5.2. The only difference is that the function $g(i)$ is replaced by a set $S_i = \{j : i \sim j \text{ and } y_j > 0\}$ which is similar to T_i in this section. The development is similar to that in Sect. 5.2 and thus omitted.

6.3 Domatic Partition in Strongly Chordal Graphs

Peng and Chang [169] gave an elegant algorithm for the domatic partition problem in strongly chordal graphs.

Their algorithm uses a primal–dual approach. Suppose $[v_1, v_2, \dots, v_n]$ is a strong elimination ordering of $G = (V, E)$ with the minimum degree $\delta(G)$. Choose a vertex x of degree $\delta(G)$. As any dominating set D_i in a domatic partition of G contains at least one vertex v_i in $N[x]$ and two distinct D_i have different corresponding v_i , it is easy to see the following inequality.

Weak Duality Inequality: $d(G) \leq \delta(G) + 1$.

Their algorithm maintains $\delta(G) + 1$ disjoint sets. Initially, these sets are empty. The algorithm scans the vertices in the reverse order of the strong elimination ordering. A vertex is included in a set when it is scanned. When the algorithm terminates, these $\delta(G) + 1$ sets are dominating sets.

A vertex v is *completely dominated* if v is dominated by all of these $\delta(G) + 1$ dominating sets.

Algorithm DomaticSC. Determine a domatic partition of a strongly chordal graph G of size $\delta(G) + 1$.

Input. A strongly chordal graph $G = (V, E)$ with a strong elimination ordering $[v_1, v_2, \dots, v_n]$.

Output. A partition of V into $\delta(G) + 1$ disjoint dominating sets of G .

Method.

$S_i \leftarrow \emptyset$ for $1 \leq i \leq \delta(G) + 1$;

for $i = n$ **to** 1 **step** -1 **do**

find the largest $k \sim i$ such that v_k is not completely dominated;

let S_ℓ be a set that does not dominate v_k ;

$S_\ell \leftarrow \{v_i\} \cup S_\ell$;

if no such set exists then include v_i to an arbitrary S_ℓ ;

end do.

Before proving the correctness of the algorithm, two lemmas are needed.

Lemma 14 Assume $S_\ell \subseteq \{v_{i+1}, v_{i+2}, \dots, v_n\}$ and $k \sim i$, where $1 \leq i \leq n$. If S_ℓ does not dominate v_k , then S_ℓ does not dominate v_j for all $j \leq k$ with $j \sim i$.

Proof Suppose to the contrary that S_ℓ has a vertex v_p dominating v_j , that is, $i < p$ and $p \sim j$. Then $i < p$, $j \leq k$, $i \sim j$, $i \sim k$, and $p \sim j$ imply $p \sim k$ by (SEO), which contradicts that S_ℓ does not dominate v_k . \square

Let $r(v) = |\{x \in N[v] : x \text{ is not in any of the } \delta(G) + 1 \text{ sets}\}|$ and $\text{ndom}(v)$ be the number of sets that do not dominate v during the execution of Algorithm DomaticSC.

Lemma 15 *Algorithm DomaticSC maintains the following invariant:*

$$r(v_j) \geq \text{ndom}(v_j) \text{ for all } j \in \{1, 2, \dots, n\}.$$

Proof The lemma is proved by induction. Initially,

$$r(v_j) = \deg(v_j) + 1 \geq \delta(G) + 1 = \text{ndom}(v_j)$$

for all $v_j \in V$. During iteration i , only values of $r(v_j)$ and $\text{ndom}(v_j)$, where $j \sim i$, may be altered when v_i is included in a set S_ℓ . Notice that the algorithm determines the largest index $k \sim i$ such that v_k is not completely dominated. It then finds a set S_ℓ that does not dominate v_k (S_ℓ is chosen arbitrarily when v_k does not exist).

For any $j \sim i$ with $j \leq k$, by Lemma 14, v_j was not dominated by S_ℓ . Therefore, $r(v_j)$ and $\text{ndom}(v_j)$ are decremented by one after v_i is included in S_ℓ .

On the other hand, for any $j \sim i$ with $j > k$ (or nonexistence of such v_k), by the choice of the vertex v_k in the algorithm, vertex v_j is completely dominated, that is, $\text{ndom}(v_j) = 0$. Thus, the invariant is maintained. \square

Theorem 19 *Algorithm DomaticSC partitions the vertex set of a strongly chordal graph $G = (V, E)$ into $d(G) = \delta(G) + 1$ disjoint dominating sets in linear time provided that a strong elimination ordering is given.*

Proof Upon termination of the algorithm, $r(v_j) = 0$ for all $j \in \{1, 2, \dots, n\}$. According to Lemma 15, $\text{ndom}(v_j) = 0$ for all v_j in V . That is, these $\delta(G) + 1$ sets are dominating sets of G . The strong duality equality

$$d(G) = \delta(G) + 1$$

then follows from the weak duality inequality.

To implement that algorithm efficiently, each vertex v_i is associated with a variable $\text{ndom}(i)$ and an array $L(i)$ of size $\delta(G) + 1$. Initially, $\text{ndom}(i) = \delta(G) + 1$ and the values of entries in L_i are all zero. Thus, for each vertex it takes $O(d_i)$ time to test $\text{ndom}(i)$ to determine v_k , where d_i is the degree of v_i . It then takes

$O(\delta(G) + 1)$ time to decide which set v_i should go. Finally, for each $v_j \in N[v_i]$, it takes $O(1)$ time to update $\text{ndom}(j)$ and L_j . Therefore, the algorithm takes

$$O\left(\sum_{i=1}^n (d_i + \delta(G) + 1)\right) = O(|V| + |E|) \text{ time.} \quad \square$$

7 Permutation Graphs

Given a permutation $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ on the set $I_n = \{1, 2, \dots, n\}$, the *permutation graph* of π is the graph $G(\pi) = (I_n, E(\pi))$ with

$$E(\pi) = \{jk : (j - k)(\pi^{-1}(j) - \pi^{-1}(k)) < 0\}.$$

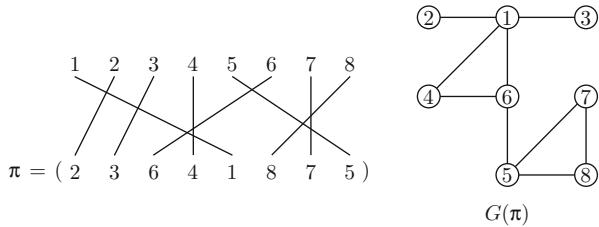
Note that $\pi^{-1}(j)$ is the position of j in the permutation π . [Figure 13](#) illustrates a permutation and its corresponding permutation graph. If a line between each integer i and its position in π is drawn, then n lines are created, each with an associated integer. In this way, two vertices j and k are adjacent in $G(\pi)$ if and only if their corresponding lines cross. That is, $G(\pi)$ is the intersection graph of these n lines. Notice that an independent set in $G(\pi)$ corresponds to an increasing subsequence of π , and a clique in $G(\pi)$ corresponds to a decreasing subsequence of π .

Permutation graphs were first introduced by Pnueli et al. in [87, 171]. Since that time quite a few polynomial-time algorithms have been constructed on permutation graphs. For example, Atallah et al. [8], Brandstädt and Kratsch [31]; Farber and Keil [94]; and Tsai and Hsu [191] have constructed polynomial domination and independent domination algorithms, Brandstädt and Kratsch [31] and Colbourn and Stewart [78] have constructed polynomial connected domination algorithms, and Brandstädt and Kratsch [31] and Corneil and Stewart [78] have constructed polynomial total domination algorithms.

This section presents a simple $O(n^2)$ -time algorithm, due to Brandstädt and Kratsch [32] for finding a minimum weighted independent dominating set in a permutation graph. Assume that the defining permutation π of the permutation graph is given as part of the input. Spinrad [186] has shown that π can be constructed in $O(n^2)$ time, given the graph G .

This algorithm takes advantage of the observation that a set is an independent dominating set if and only if it is a maximal independent set. Since maximal independent sets in permutation graphs correspond to maximal increasing subsequences in π , all that is necessary is to search for such a sequence in π of minimum weight. In particular, it determines, for every j , $1 \leq j < n$, the minimum weight $\gamma_i(j, w)$ of an independent dominating set in the subsequence $\pi(1), \pi(2), \dots, \pi(j)$, which contains $\pi(j)$ as the rightmost element. Let $w(j)$ denote the weight of vertex j .

Fig. 13 A permutation and its corresponding permutation graph



Algorithm WIndDomPer. Solve the weighted independent domination problem for permutation graphs.

Input. A permutation graph G with its corresponding permutation π on the set $\{1, 2, \dots, n\}$ and vertex weights $w(1), w(2), \dots, w(n)$ of real numbers.

Output. The weighted independent domination number $\gamma_i(G, w)$ of G .

Method.

```

for  $j = 1$  to  $n$  do
     $p(j) \leftarrow 0$ ;
     $\gamma_i(j) = w(\pi(j))$ ;
    end do;
for  $k = j - 1$  to  $1$  step  $-1$  do
    if  $(\pi(j) > \pi(k)$  and  $p(j) = 0)$  then
         $\gamma_i(j) \leftarrow w(\pi(j)) + \gamma_i(k)$ ;
         $p(j) \leftarrow \pi(k)$ ;
        end if;
    else if  $(\pi(j) > \pi(k) > p(j) > 0)$  then
         $\gamma_i(j) \leftarrow \min\{\gamma_i(j), w(\pi(j)) + \gamma_i(k)\}$ ;
         $p(j) \leftarrow \pi(k)$ ;
        end if
    end do;
 $m \leftarrow p(n)$ ;
 $\gamma_i(G, w) \leftarrow \gamma_i(n)$ ;
for  $j = n - 1$  to  $1$  step  $-1$  do
    if  $\pi(j) > m$  then
         $\gamma_i(G, w) \leftarrow \min\{\gamma_i(G, w), \gamma_i(j)\}$ ;
         $m \leftarrow \pi(j)$ ;
        end if.

```

The algorithm is illustrated by the permutation graph $G(\pi)$ in Fig. 13, where $\pi = (2 3 6 4 1 8 7 5)$ and all weights are equal to 1: $\gamma_i(j) = (1 2 3 3 1 2 2 2)$, and thus, the minimum size of an independent dominating set is 2, for example, the set $\{1, 5\}$.

8 Cocomparability Graphs

A graph $G = (V, E)$ is a *comparability graph* if G has an orientation $H = (V, F)$ such that $xy, yz \in F$ imply $xz \in F$. In other words, G has a *comparability ordering* which is an ordering $[v_1, v_2, \dots, v_n]$ of V satisfying

$$i < j < k \text{ and } v_i v_j, v_j v_k \in E \text{ imply } v_i v_k \in E. \quad (\text{CO})$$

A graph $G = (V, E)$ is a *cocomparability graph* if its complement \overline{G} is a comparability graph or, equivalently, G has a *cocomparability ordering* which is an ordering $[v_1, v_2, \dots, v_n]$ of V satisfying

$$i < j < k \text{ and } v_i v_k \in E \text{ imply } v_i v_j \in E \text{ or } v_j v_k \in E. \quad (\text{CCO})$$

There is an $O(n^{2.376})$ -time recognition algorithm for comparability graphs and thus for cocomparability graphs [186]. This has been improved by McConnell and Spinrad who gave an $O(n + m)$ -time algorithm constructing an orientation of any given graph G such that the orientation is a transitive orientation of G if and only if G has a transitive orientation [160]. Unfortunately, the best algorithm for testing whether the orientation is indeed transitive has running time $O(n^{2.376})$.

The class of cocomparability graphs is a well-studied superclass of the classes of interval graphs, permutation graphs, and trapezoid graphs. Domination problems on cocomparability graphs were considered for the first time by Kratsch and Stewart [148]. They obtained polynomial-time algorithms for the domination/total domination/connected domination and the weighted independent domination problems in cocomparability graphs. These algorithms are designed by dynamic programming using cocomparability orderings. Breu and Kirkpatrick [34] (see [4]) improved this by giving $O(nm^2)$ -time algorithms for the domination and the total domination problems and an $O(n^{2.376})$ -time algorithm for the weighted independent domination problem in cocomparability graphs.

On the other hand, the weighted domination, weighted total domination, and weighted connected domination problems are NP-complete in cocomparability graphs [49]. Also, the problem “Given a cocomparability graph G , does G have a dominating clique?” is NP-complete [148].

An $O(n^3)$ -time algorithm computing a minimum cardinality connected dominating set of a connected cocomparability graph has been given in [148]. The following is the algorithm for this problem given by Breu and Kirkpatrick [34] (see also [4]).

Let $[v_1, v_2, \dots, v_n]$ be a cocomparability ordering of a cocomparability graph $G = (V, E)$. For vertices $u, w \in V$, write $u < w$ if u appears before w in the ordering, that is, $u = v_i$ and $w = v_j$ implies $i < j$. For $i \leq j$ the set $\{v_k : i \leq k \leq j\}$ is denoted by $[v_i, v_j]$. Then $v_i v_j \in E$ implies that every vertex v_k with $i < k < j$ is adjacent to v_i or to v_j ; thus, $\{v_i, v_j\}$ dominates $[v_i, v_j]$. This can be generalized as follows: Let $S \subseteq V$ where $G[S]$ is connected. Then S dominates

$[\min(S), \max(S)]$ where $\min(S)$ (respectively, $\max(S)$) is the vertex of S with the smallest (respectively, largest) index in the ordering.

The following theorem and lemma are given in [148].

Theorem 20 *Any connected cocomparability graph G has a minimum connected dominating set S such that the induced subgraph $G[S]$ is a chordless path p_1, p_2, \dots, p_k .*

Lemma 16 *Suppose $S \subseteq V$ is a minimum connected dominating set of a cocomparability graph $G = (V, E)$ with a cocomparability ordering $[v_1, v_2, \dots, v_n]$. If $G[S]$ is a chordless path p_1, p_2, \dots, p_k , then every vertex $x < \min(S)$ is dominated by $\{p_1, p_2\}$ and every vertex $y > \max(S)$ is dominated by $\{p_{k-1}, p_k\}$.*

The following approach enables an elegant way of locating a chordless path of minimum size that dominates the cocomparability graph. A *source vertex* is a vertex v_i such that $v_k v_i \in E$ for all $k < i$, and a *sink vertex* is a vertex v_j such that $v_j v_k \in E$ for all $k > j$. Then $[v_1, v_2, \dots, v_n]$ is a *canonical cocomparability ordering* if $[v_1, v_2, \dots, v_r]$, $1 \leq r < n$, are the source vertices and v_s, v_{s+1}, \dots, v_n , $1 < s \leq n$, are the sink vertices. Note that every cocomparability graph G has a canonical cocomparability ordering. Furthermore, given any cocomparability ordering, a canonical one can be computed in time $O(n + m)$.

From now on, assume that $[v_1, v_2, \dots, v_n]$ is a canonical cocomparability ordering. Since the source vertices of G form a clique, any source vertex v_i dominates $[v_1, v_i]$. Analogously, since the sink vertices of G form a clique, any sink vertex v_j dominates $[v_j, v_n]$. Therefore, the vertex set of every path between a source vertex and a sink vertex is dominating.

The following theorem given in [34] enlightens the key property.

Theorem 21 *Every connected cocomparability graph $G = (V, E)$ satisfying $\gamma_c(G) \geq 3$ has a minimum connected dominating set which is the vertex set of a shortest path between a source and a sink vertex of G .*

Proof Let $[v_1, v_2, \dots, v_n]$ be a canonical cocomparability ordering of G . According to **Theorem 20**, there is a minimum connected dominating set S of G such that $G[S]$ is a chordless path $P : p_1, p_2, \dots, p_k$, $k \geq 3$. Construct below a chordless path P'' between a source vertex and a sink vertex of G that has the same number of vertices as the path P .

Let $p_1 = v_i$ and $p_2 = v_j$. First observe that $p_2 = v_j$ cannot be a source vertex, otherwise $N[p_2] \supseteq [v_1, v_j]$ implying that $\{p_2, p_3, \dots, p_k\}$ is also a connected dominating set of G , a contradiction. If p_1 is a source vertex, then P starts at a source vertex. In this case, proceed with the path $P' = P$ (possibly) rearranging p_{k-1}, p_k .

Suppose $p_1 = v_i$ is not a source vertex. Then there is a source vertex u of G with $u p_1 \notin E$. Since $[v_1, v_2, \dots, v_n]$ is a canonical cocomparability ordering and since

p_1 and p_2 are not source vertices, $u < p_1$ and $u < p_2$. Since $v_i v_j \in E$ and by Lemma 16, $\{v_i, v_j\}$ dominates $[1, \max\{v_i, v_j\}]$. Consequently $u p_2 \in E$.

Consider the set $S' = \{u, p_2, \dots, p_k\}$. Since $\{u, p_2\}$ dominates $[v_1, v_j]$, S' is a dominating set. Since $P : p_1, p_2, \dots, p_k$ is a chordless path, $t \geq 3$ implies $p_t u \notin E$. Thus, S' induces the chordless path $P' : u, p_2, \dots, p_k$.

Similarly, starting from P' the vertex p_k can be replaced, if necessary. Vertex p_{k-1} is not a sink vertex. If p_k is a sink vertex, then $S'' = S'$ and $P'' = P'$. Otherwise, replace p_k by a sink vertex v satisfying $v p_k \notin E$ to obtain S'' and P'' .

$G[S'']$ induces a chordless path between a sink and a source vertex. The vertex set of any path between a sink and a source vertex is a dominating set. By construction S'' is a minimum connected dominating set. Consequently S'' is the vertex set of a shortest path between a source vertex and a sink vertex of G . \square

According to Theorem 21, when $\gamma_c(G) \geq 3$, computing a minimum connected dominating set of a connected cocomparability graph G reduces to computing a shortest path between a source and a sink vertex of G .

Algorithm ConDomCC. Solve the connected domination problem in cocomparability graphs.

Input: A connected cocomparability graph $G = (V, E)$ and a canonical cocomparability ordering $[v_1, v_2, \dots, v_n]$ of G .

Output: A minimum connected dominating set of G .

Methods.

1. Check whether G has a minimum connected dominating set D of size at most 2. If so, output D and stop.
2. Construct a new graph G' by adding two new vertices s and t to G such that s is adjacent exactly to the source vertices of G and t is adjacent exactly to the sink vertices of G .
3. Compute a shortest path $P : s, p_1, p_2, \dots, p_k, t$ between s and t in G' by the breadth-first search.
4. Output $\{p_1, p_2, \dots, p_k\}$.

The correctness of Algorithm ConDomCC follows immediately from Theorem 21. The “almost linear” running time of the algorithm follows from the well-known fact that breadth-first search is a linear-time procedure.

Theorem 22 *For any connected cocomparability graph $G = (V, E)$ with a canonical cocomparability ordering $[v_1, v_2, \dots, v_n]$, Algorithm ConDomCC outputs a minimum connected dominating set of G in $O(nm)$ time. In fact, all parts of the algorithm except checking for a connected dominating set of size two can be done in time $O(n + m)$.*

It is clearly unsatisfactory that the straightforward test for a connected dominating set of size two dominates the overall running time. The crux is that there are even permutation graphs for which each minimum connected dominating set of size two contains neither a source nor a sink vertex (see [127, 142]). It seems that minimum

dominating sets of this type cannot be found by a shortest path approach. It is an open question whether step 1 of Algorithm ConDomCC can be implemented in a more efficient way.

Notice that the $O(n)$ -time algorithms computing a minimum connected dominating set for permutation graphs [127] and trapezoid graphs [142] both rely on [Theorem 21](#).

Corneil et al. have done a lot of research on asteroidal triple-free graphs, usually called AT-free graphs [72, 75]. They are defined as those graphs not containing an asteroidal triple, that is, a set of three vertices such that between any two of the vertices, there is a path avoiding the neighborhood of the third.

AT-free graphs form a superclass of the cocomparability graphs. They are a “large class of graphs” with nice structural properties, and some of them are related to domination. One of the major theorems on the structure of AT-free graphs states that every connected AT-free graph has a dominating pair, that is, a pair of vertices u, v such that the vertex set of each path between u and v is a dominating set.

An $O(n + m)$ algorithm computing a dominating pair for a given connected AT-free graph has been presented in [75]. This can be used to obtain an $O(n + m)$ algorithm computing a dominating path for connected AT-free graphs (see also [73]). An $O(n^3)$ algorithm computing a minimum connected dominating set for connected AT-free graphs is given in [10]. An $O(n + m)$ algorithm computing a minimum connected dominating set in connected AT-free graphs with diameter greater than three is given in [75].

9 Distance-Hereditary Graphs

9.1 Hangings of Distance-Hereditary Graphs

A graph is *distance hereditary* if every two vertices have the same distance in every connected induced subgraph. Distance-hereditary graphs were introduced by Howorka [121]. The characterization and recognition of distance-hereditary graphs have been studied in [11, 81, 82, 105, 121]. Distance-hereditary graphs are parity graphs [36] and include all cographs [68, 77].

The *hanging* h_u of a connected graph $G = (V, E)$ at a vertex $u \in V$ is the collection of sets $L_0(u), L_1(u), \dots, L_t(u)$ (or L_0, L_1, \dots, L_t if there is no ambiguity), where $t = \max_{v \in V} d_G(u, v)$ and

$$L_i(u) = \{v \in V : d_G(u, v) = i\}$$

for $0 \leq i \leq t$. For any $1 \leq i \leq t$ and any vertex $v \in L_i$, let $N'(v) = N(v) \cap L_{i-1}$. A vertex $v \in L_i$ with $1 \leq i \leq t$ is said to have a *minimal neighborhood* in L_{i-1} if $N'(x)$ is not a proper subset of $N'(v)$ for any $x \in L_i$. Such a vertex v certainly exists.

Theorem 23 ([11, 81, 82]) A connected graph $G = (V, E)$ is distance hereditary if and only if for every hanging $h_u = (L_0, L_1, \dots, L_t)$ of G and every pair of vertices $x, y \in L_i$ ($1 \leq i \leq t$) that are in the same component of $G - L_{i-1}$, we have $N'(x) = N'(y)$.

Theorem 24 ([11]) Suppose $h_u = (L_0, L_1, \dots, L_t)$ is a hanging of a connected distance-hereditary graph at u . For any two vertices $x, y \in L_i$ with $i \geq 1$, $N'(x) \cap N'(y) = \emptyset$ or $N'(x) \subseteq N'(y)$ or $N'(x) \supseteq N'(y)$.

Theorem 25 (Fact 3.4 in [105]) Suppose $h_u = (L_0, L_1, \dots, L_t)$ is a hanging of a connected distance-hereditary graph at u . If vertex $v \in L_i$ with $1 \leq i \leq t$ has a minimal neighborhood in L_{i-1} , then $N_{V \setminus N'(v)}(x) = N_{V \setminus N'(v)}(y)$ for every pair of vertices x and y in $N'(v)$.

9.2 Weighted Connected Domination

D'Atri and Moscarini [81] gave $O(|V||E|)$ algorithms for connected domination and Steiner tree problems in distance-hereditary graphs. Brandstädt and Dragan [29] presented a linear-time algorithm for the connected r -domination and Steiner tree problems in distance-hereditary graphs.

This section presents a linear-time algorithm given by Yeh and Chang [200] for finding a minimum weighted connected dominating set of a connected distance-hereditary graph $G = (V, E)$ in which each vertex v has a weight $w(v)$ that is a real number. According to Lemma 1, assume that the vertex weights are nonnegative.

Lemma 17 Suppose $h_u = \{L_0, L_1, \dots, L_t\}$ is a hanging of a connected distance-hereditary graph at u . For any connected dominating set D and $v \in L_i$ with $2 \leq i \leq t$, $D \cap N'(v) \neq \emptyset$.

Proof Choose a vertex y in D that dominates v . Then $y \in L_{i-1} \cup L_i \cup L_{i+1}$. If $y \in L_{i-1}$, then $y \in D \cap N'(v)$. So, assume that $y \in L_i \cup L_{i+1}$. Choose a vertex $x \in D \cap (L_0 \cup L_1)$ and an x - y path

$$P : x = v_1, v_2, \dots, v_m = y$$

using vertices only in D . Let j be the smallest index such that $\{v_j, v_{j+1}, \dots, v_m\} \subseteq L_i \cup L_{i+1} \cup \dots \cup L_t$. Then $v_j \in L_i$, $v_{j-1} \in N'(v_j)$, and v and v_j are in the same component of $G - L_{i-1}$. By Theorem 23, $N'(v) = N'(v_j)$ and so $v_{j-1} \in D \cap N'(v)$. In any case, $D \cap N'(v) \neq \emptyset$. \square

Theorem 26 Suppose $G = (V, E)$ is a connected distance-hereditary graph with a nonnegative weight function w on its vertices. Let $h_u = \{L_0, L_1, \dots, L_t\}$ be a hanging at a vertex u of minimum weight. Consider the set $\mathcal{A} = \{N'(v) : v \in L_i \text{ with } 2 \leq i \leq t \text{ and } v \text{ has a minimal neighborhood in } L_{i-1}\}$. For each $N'(v)$ in \mathcal{A} ,

choose one vertex v^* in $N'(v)$ of minimum weight, and let D be the set of all such v^* . Then D or $D \cup \{u\}$ or some $\{v\}$ with $v \in V$ is a minimum weighted connected dominating set of G .

Proof For any $x \in L_i$ with $2 \leq i \leq t$, by [Theorem 24](#), $N'(x)$ includes some $N'(v)$ in \mathcal{A} . This gives [Claim 1](#).

Claim 1 For any $x \in L_i$ with $2 \leq i \leq t$, $x \in N[L_{i-1} \cap D]$.

Claim 2 $D \cup \{u\}$ is a connected dominating set of G .

Proof of Claim 2. By [Claim 1](#) and $N[u] = L_1 \cup \{u\}$, $D \cup \{u\}$ is a dominating set of G . Also, by [Claim 1](#), for any vertex x in $D \cup \{u\}$, there exists an x - u path using vertices only in $D \cup \{u\}$, that is, $G[D \cup \{u\}]$ is connected. \square

Suppose M is a minimum weighted connected dominating set of G . According to [Lemma 17](#), $M \cap N'(v) \neq \emptyset$ for each $N'(v) \in \mathcal{A}$, say $v^{**} \in M \cap N'(v)$. Since any two sets in \mathcal{A} are disjoint, $|M| \geq |\mathcal{A}| = |D|$.

Case 1. $|M| = 1$. The theorem is obvious in this case.

Case 2. $|M| > |D|$. In this case, there is at least one vertex x in M that is not a v^{**} . Then

$$w(M) \geq \sum_{v^{**}} w(v^{**}) + w(x) \geq \sum_{v^*} w(v^*) + w(u) = w(D \cup \{u\}).$$

This together with [Claim 2](#) gives that $D \cup \{u\}$ is a minimum weighted connected dominating set of G .

Case 3. $|M| = |D| \geq 2$. Since \mathcal{A} contains pairwise disjoint sets, $M = \{v^{**} : N'(v) \in \mathcal{A}\}$. Then $w(M) = \sum_{v^{**}} w(v^{**}) \geq \sum_{v^*} w(v^*) = w(D)$.

For any two vertices x^* and y^* in D , x^{**} and y^{**} are in M . Since $G[M]$ is connected, there is an x^{**} - y^{**} path in $G[M]$:

$$x^{**} = v_0^{**}, v_1^{**}, \dots, v_n^{**} = y^{**}.$$

For any $1 \leq i \leq n$, since v_i^* and v_i^{**} are both in $N'(v_i) \in \mathcal{A}$, by [Theorem 25](#), $N_{V \setminus N'(v_i)}(v_i^*) = N_{V \setminus N'(v_i)}(v_i^{**})$. But $v_{i-1}^{**} \in N_{V \setminus N'(v_i)}(v_i^{**})$. Therefore, $v_{i-1}^{**} \in N_{V \setminus N'(v_i)}(v_i^*)$ and $v_i^* \in N_{V \setminus N'(v_{i-1})}(v_{i-1}^{**})$. Also, that v_{i-1}^* and v_{i-1}^{**} are both in $N'(v_{i-1}) \in \mathcal{A}$ implies that $N_{V \setminus N'(v_{i-1})}(v_{i-1}^*) = N_{V \setminus N'(v_{i-1})}(v_{i-1}^{**})$. Then $v_i^* \in N_{V \setminus N'(v_{i-1})}(v_{i-1}^*)$. This proves that v_{i-1}^* is adjacent to v_i^* for $1 \leq i \leq n$, and then

$$x^* = v_0^*, v_1^*, \dots, v_n^* = y^*$$

is an x^* - y^* path in $G[D]$, that is, $G[D]$ is connected.

For any x in V , since M is a dominating set, $x \in N[v^{**}]$ for some $N'(v) \in \mathcal{A}$. Note that v^{**} and v^* are both in $N'(v)$. According to [Theorem 25](#), $N_{V \setminus N'(v)}(v^{**}) = N_{V \setminus N'(v)}(v^*)$. In the case of $x \notin N'(v)$, $x \in N[v^{**}]$ implies $x \in N[v^*]$, that is, D

dominates x . In the case of $x \in N'(v)$, $N_{V \setminus N'(v)}(v^*) = N_{V \setminus N'(v)}(x)$. Since $G[D]$ is connected and $|D| \geq 2$, v^* is adjacent to some $y^* \in D \setminus N'(v)$. Then x is also adjacent to y^* , that is, D dominates x . In any case, D is a dominating set. Therefore, D is a minimum weighted connected dominating set of G . \square

Lemma 1 and **Theorem 26** together give an efficient algorithm for the weighted connected domination problem in distance-hereditary graphs as follows. To implement the algorithm efficiently, the set \mathcal{A} is not actually found. Instead, the following step is performed for each $2 \leq i \leq t$. Sort the vertices in L_i such that

$$|N'(x_1)| \leq |N'(x_2)| \leq \dots \leq |N'(x_j)|.$$

Then process $N'(x_k)$ for k from 1 to j . At iteration k , if $N'(x_k) \cap D = \emptyset$, then $N'(x_k)$ is in \mathcal{A} , and choose a vertex of minimum weight to put it into D ; otherwise, $N'(x_k) \notin \mathcal{A}$ and do nothing.

Algorithm WConDomDH. Find a minimum weighted connected dominating set of a connected distance-hereditary graph.

Input: A connected distance-hereditary graph $G = (V, E)$ and a weight $w(v)$ of real number for each $v \in V$.

Output: A minimum weighted connected dominating set D of graph G .

Method.

```

 $D \leftarrow \emptyset;$ 
let  $V' = \{v \in V : w(v) < 0\}$ ;
 $w(v) \leftarrow 0$  for each  $v \in V'$ ;
let  $u$  be a vertex of minimum weight in  $V$ ;
determine the hanging  $h_u = (L_0, L_1, \dots, L_t)$  of  $G$  at  $u$ ;
for  $i = 2$  to  $t$  do
    let  $L_i = \{x_1, x_2, \dots, x_j\}$ ;
    sort  $L_i$  such that  $|N'(x_{i_1})| \leq |N'(x_{i_2})| \leq \dots \leq |N'(x_{i_j})|$ ;
    for  $k = 1$  to  $j$  do
        if  $N'(x_{i_k}) \cap D = \emptyset$  then  $D \leftarrow D \cup \{y\}$  where  $y$  is a vertex
            of minimum weight in  $N'(x_{i_k})$ ;
    end do;
if not  $(L_1 \subseteq N[D]$  and  $G[D]$  is connected) then  $D \leftarrow D \cup \{u\}$ ;
for  $v \in V$  that dominates  $V$  do
    if  $w(v) < w(D)$  then  $D \leftarrow \{v\}$ ;
 $D \leftarrow D \cup V'$ .

```

Theorem 27 Algorithm WConDomDH gives a minimum weighted connected dominating set of a connected distance-hereditary graph in linear time.

Proof The correctness of the algorithm follows from **Lemma 1** and **Theorem 26**. For each i , sort L_i by using a bucket sort. Then the algorithm runs in $O(|V| + |E|)$ time. \square

Cross-References

- [A Unified Approach for Domination Problems on Different Network Topologies](#)
- [Variations of Dominating Set Problem](#)

Recommended Reading

1. S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability—a survey. *Bit* **25**, 2–23 (1985)
2. S. Arnborg, S.T. Hedetniemi, A. Proskurowski (eds.), Efficient Algorithms and Partial k-trees. Special Issue of *Discrete Applied Math.* **54**, 97–99 (1994)
3. S. Arnborg, A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k -trees. *Discrete Appl. Math.* **23**, 11–24 (1989)
4. K. Arvind, H. Breu, M.S. Chang, D.G. Kirkpatrick, F.Y. Lee, Y.D. Liang, K. Madhukar, C. Pandu Rangan, A. Srinivasan, Efficient algorithms in cocomparability and trapezoid graphs. Manuscript, 1996
5. K. Arvind, C. Pandu Rangan, Efficient algorithms for domination problems on cocomparability graphs. Technical Report TR-TCS-909-18, Indian Institute of Technology, 1990
6. T. Asano, Dynamic programming on intervals. *Int. J. Comput. Geom. Appl.* **3**, 323–330 (1993)
7. M. J. Atallah, S.R. Kosaraju, An efficient algorithm for maxdominance, with applications. *Algorithmica* **4**, 221–236 (1989)
8. M.J. Atallah, G.K. Manacher, J. Urrutia, Finding a minimum independent dominating set in a permutation graph. *Discrete Appl. Math.* **21**, 177–183 (1988)
9. B.S. Baker, Approximation algorithms for NP-complete problems on planar graphs, in *24th Annual Symposium on the Foundations of Computer Science*, J. ACM. **41**, 153–180 (1994)
10. H. Balakrishnan, A. Rajaraman, C. Pandu Rangan, Connected domination and Steiner set on asteroidal triple-free graphs, in *Proc. Workshop on Algorithms and Data Structures (WADS'93)*, vol. 709, Montreal, ed. by F. Dehne, J.R. Sack, N. Santoro, S. Whitesides (Springer, Berlin, 1993), pp. 131–141
11. H.J. Bandelt, H.M. Mulder, Distance-hereditary graphs. *J. Comb. Theory Ser. B* **41**, 182–208 (1986)
12. D.W. Bange, A.E. Barkauskas, P.J. Slater, Efficient dominating sets in graphs, in *Applications of Discrete Mathematics*, ed. by R.D. Ringeisen, F.S. Roberts (SIAM, Philadelphia, 1988), pp. 189–199
13. R. Bar-Yehuda, U. Vishkin, Complexity of finding k -path-free dominating sets in graphs. *Inform. Process. Lett.* **14**, 228–232 (1982)
14. R.E. Bellman, S.E. Dreyfus, *Applied Dynamic Programming* (Princeton University Press, Princeton, 1962)
15. A.A. Bertossi, Dominating sets for split and bipartite graphs. *Inform. Process. Lett.* **19**, 37–40 (1984)
16. A.A. Bertossi, Total domination in interval graphs. *Inform. Process. Lett.* **23**, 131–134 (1986)
17. A.A. Bertossi, On the domatic number of interval graphs. *Inform. Process. Lett.* **28**, 275–280 (1988)
18. A.A. Bertossi, M.A. Bonuccelli, Some parallel algorithms on interval graphs. *Discrete Appl. Math.* **16**, 101–111 (1987)
19. A.A. Bertossi, A. Gori, Total domination and irredundance in weighted interval graphs. *SIAM J. Discrete Math.* **1**, 317–327 (1988)
20. A.A. Bertossi, S. Moretti, Parallel algorithms on circular-arc graphs. *Inform. Process. Lett.* **33**, 275–281 (1990)

21. T.A. Beyer, A. Proskurowski, S.T. Hedetniemi, S. Mitchell, Independent domination in trees. *Congr. Numer.* **19**, 321–328 (1977)
22. N.L. Biggs, E.K. Lloyd, R.J. Wilson, *Graph Theory 1736–1936* (Clarendon Press, Oxford, 1986)
23. J.R.S. Blair, W. Goddard, S.T. Hedetniemi, S. Horton, P. Jones, G. Kubicki. On domination and reinforcement numbers in trees. *Discrete Math.* **308**(7), 1165–1175 (2008)
24. H.L. Bodlaender, Dynamic programming on graphs with bounded treewidth, in *Proc. 15th Internat. Colloq. on Automata, Languages and Programming*, ed. by T. Lepistö, A. Salomaa. Lecture Notes in Computer Science, vol. 317 (Springer, Heidelberg, 1988), pp. 105–118
25. K.S. Booth, J.H. Johnson, Dominating sets in chordal graphs. *SIAM J. Comput.* **11**, 191–199 (1982)
26. S. Booth, S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.* **13**, 335–379 (1976)
27. A. Brandstädt, The computational complexity of feedback vertex set, Hamiltonian circuit, dominating set, Steiner tree and bandwidth on special perfect graphs. *J. Inform. Process. Cybern.* **23**, 471–477 (1987)
28. A. Brandstädt, V.D. Chepoi, F.F. Dragan, The algorithmic use of hypertree structure and maximum neighbourhood orderings, in *20th Internat. Workshop Graph-Theoretic Concepts in Computer Science (WG'94)*, ed. by E.W. Mayr, G. Schmidt, G. Tinhofer. Lecture Notes in Computer Science, vol. 903 (Springer, Berlin, 1995), pp. 65–80
29. A. Brandstädt, F.F. Dragan, A linear-time algorithm for connected r -domination and Steiner tree on distance-hereditary graphs. Technical Report SM-DU-261, Univ. Duisburg, 1994
30. A. Brandstädt, F.F. Dragan, V.D. Chepoi, V.I. Voloshin, Dually chordal graphs, in *19th Internat. Workshop Graph-Theoretic Concepts in Computer Science (WG'93)*. Lecture Notes in Computer Science, vol. 790 (Springer, Berlin, 1993), pp. 237–251
31. A. Brandstädt, D. Kratsch, On the restriction of some NP-complete graph problems to permutation graphs, in *Proc. FCT'85*, ed. by L. Budach Lecture Notes in Computer Science, vol. 199 (Springer, Berlin, 1985), pp. 53–62
32. A. Brandstädt, D. Kratsch, On domination problems on permutation and other graphs. *Theor. Comput. Sci.* **54**, 181–198 (1987)
33. B. Bresar, M.A. Henning, D.F. Rall, Rainbow domination in graphs. *Taiwan. J. Math.* **12**(1), 213–225 (2008)
34. H. Breu, D.G. Kirkpatrick, Algorithms for dominating and Steiner set problems in cocomparability graphs. Manuscript, 1993
35. M.W. Broin, T.J. Lowe, A dynamic programming algorithm for covering problems with greedy totally balanced constraint matrices. *SIAM J. Algebraic Discrete Methods* **7**, 348–357 (1986)
36. M. Burlet, J.P. Uhry, Parity graphs. *Ann. Discrete Math.* **21**, 253–277 (1984)
37. D.I. Carson, Computational aspects of some generalized domination parameters. Ph.D. Thesis, Univ. Natal, 1995
38. G.J. Chang, Labeling algorithms for domination problems in sun-free chordal graphs. *Discrete Appl. Math.* **22**, 21–34 (1988/1989)
39. G.J. Chang, Total domination in block graphs. *Oper. Res. Lett.* **8**, 53–57 (1989)
40. G.J. Chang, The domatic number problem. *Discrete Math.* **125**, 115–122 (1994)
41. G.J. Chang, The weighted independent domination problem is NP-complete for chordal graphs. *Discrete Appl. Math.* **143**, 351–352 (2004)
42. G.J. Chang, G.L. Nemhauser, The k -domination and k -stability problems on graphs. Technical Report TR-540, School Oper. Res. Industrial Eng., Cornell Univ., 1982
43. G.J. Chang, G.L. Nemhauser, R -domination of block graphs. *Oper. Res. Lett.* **1**, 214–218 (1982)
44. G.J. Chang, G.L. Nemhauser, The k -domination and k -stability on sun-free chordal graphs. *SIAM J. Algebraic Discrete Methods* **5**, 332–345 (1984)
45. G.J. Chang, G.L. Nemhauser, Covering, packing and generalized perfection. *SIAM J. Algebraic Discrete Methods* **6**, 109–132 (1985)

46. G.J. Chang, C. Pandu Rangan, S.R. Coorg, Weighted independent perfect domination on cocomparability graphs. *Discrete Appl. Math.* **63**, 215–222 (1995)
47. G.J. Chang, J. Wu, X. Zhu, Rainbow domination on trees. *Discrete Appl. Math.* **158**(1), 8–12 (2010)
48. M.-S. Chang, Efficient algorithms for the domination problems on interval graphs and circular-arc graphs, in *Proc. IFIP 12th World Congress. IFIP Transactions A-12*, SIAM J. Comput. **27**, 1671–1694 (1998)
49. M.-S. Chang, Weighted domination on cocomparability graphs, in *Proc. ISAAC'95. Lecture Notes in Computer Science*, vol. 1004 (Springer, Berlin, 1995), pp. 121–131
50. M.-S. Chang, F.-H. Hsing, S.-L. Peng, Irredundance in weighted interval graphs, in *Proc. National Computer Symp.*, Taipei, Taiwan, 1993, pp. 128–137
51. M.-S. Chang, C.-C. Hsu, On minimum intersection of two minimum dominating sets of interval graphs. *Discrete Appl. Math.* **78**(1–3), 41–50 (1997)
52. M.-S. Chang, Y.-C. Liu, Polynomial algorithms for the weighted perfect domination problems on chordal and split graphs. *Inform. Process. Lett.* **48**, 205–210 (1993)
53. M.-S. Chang, Y.-C. Liu, Polynomial algorithms for weighted perfect domination problems on interval and circular-arc graphs. *J. Inform. Sci. Eng.* **10**, 549–568 (1994)
54. M.-S. Chang, S. Wu, G.J. Chang, H.-G. Yeh, Domination in distance-hereditary graphs. *Discrete Appl. Math.* **116**(1–2), 103–113 (2002)
55. G.A. Cheston, G.H. Fricke, S.T. Hedetniemi, D.P. Jacobs, On the computational complexity of upper fractional domination. *Discrete Appl. Math.* **27**, 195–207 (1990)
56. E. Cockayne, S. Goodman, S. Hedetniemi, A linear algorithm for the domination number of a tree. *Inform. Process. Lett.* **4**(2), 41–44 (1975)
57. E.J. Cockayne, S.T. Hedetniemi, A linear algorithm for the maximum weight of an independent set in a tree, in *Proc. Seventh Southeastern Conf. on Combinatorics, Graph Theory and Computing*, Utilitas Math., Winnipeg, 1976, pp 217–228
58. E.J. Cockayne, G. MacGillivray, C.M. Mynhardt, A linear algorithm for 0-1 universal minimal dominating functions of trees. *J. Combin. Math. Combin. Comput.* **10**, 23–31 (1991)
59. E.J. Cockayne, F.D.K. Roberts, Computation of minimum independent dominating sets in graphs. Technical Report DM-76-IR, Dept. Math., Univ. Victoria, 1974
60. E.J. Cockayne, F.D.K. Roberts, Computation of dominating partitions. *Infor.* **15**, 94–106 (1977)
61. C.J. Colbourn, J.M. Keil, L.K. Stewart, Finding minimum dominating cycles in permutation graphs. *Oper. Res. Lett.* **4**, 13–17 (1985)
62. C.J. Colbourn, L.K. Stewart, Permutation graphs: connected domination and Steiner trees. *Discrete Math.* **86**, 179–189 (1990)
63. C. Cooper, M. Zito, An analysis of the size of the minimum dominating sets in random recursive trees, using the Cockayne-Goodman-Hedetniemi algorithm. *Discrete Appl. Math.* **157**(9), 2010–2014 (2009)
64. D.G. Corneil, Lexicographic breadth first search—a survey. *Lect. Notes Comput. Sci.* **3353**, 1–19 (2004)
65. D.G. Corneil, A simple 3-sweep LBFS algorithm for the recognition of unit interval graphs. *Discrete Appl. Math.* **138**, 371–379 (2004)
66. D.G. Corneil, J.M. Keil, A dynamic programming approach to the dominating set problem on k -trees. *SIAM J. Algebraic Discrete Methods* **8**, 535–543 (1987)
67. D.G. Corneil, E. Kohler, J.-M. Lanlignel, On end-vertices of Lexicographic Breadth First Search. *Discrete Appl. Math.* **158**, 434–443 (2010)
68. D.G. Corneil, H. Lerchs, L. Stewart, Complement reducible graphs. *Discrete Appl. Math.* **3**, 163–174 (1981)
69. D.G. Corneil, S. Olariu, L. Stewart, The ultimate interval graph recognition algorithm? (extended abstract), in: *Proc. Ninth Annual ACM-SIAM Symp. Discrete Alg.*, ACM, New York (SIAM, Philadelphia, 1998), pp. 175–180
70. D.G. Corneil, S. Olariu, L. Stewart, The LBFS structure and recognition of interval graphs. *SIAM J. Discrete Math.* **23**, 1905–1953 (2009)

71. D.G. Corneil, S. Olariu, L. Stewart, A linear time algorithm to compute a dominating path in an AT-free graph. *Inform. Process. Lett.* **54**, 253–258 (1995)
72. D.G. Corneil, S. Olariu, L. Stewart, Asteroidal triple-free graphs. *SIAM J. Discrete Math.* **790**, 211–224 (1994)
73. D.G. Corneil, S. Olariu, L. Stewart, Computing a dominating pair in an asteroidal triple-free graph in linear time, in *Proc. 4th Algorithms and Data Structures Workshop. LNCS*, vol. 955 (Springer, Berlin, 1995), pp. 358–368
74. D.G. Corneil, S. Olariu, L. Stewart, A linear time algorithm to compute dominating pairs in asteroidal triple-free graphs, in *Proc. 22nd Internat. Colloq. on Automata, Languages and Programming (ICALP'95)*. Lecture Notes in Computer Science, vol. 994 (Springer, Berlin, 1995), pp. 292–302
75. D.G. Corneil, S. Olariu, L. Stewart, Linear time algorithms for dominating pairs in asteroidal triple-free graphs. *SIAM J. Comput.* **944**, 292–302 (1995)
76. D.G. Corneil, Y. Perl, Clustering and domination in perfect graphs. *Discrete Appl. Math.* **9**, 27–39 (1984)
77. D.G. Corneil, Y. Perl, L. Stewart Burlingham, A linear recognition algorithm for cographs. *SIAM J. Comput.* **14**, 926–934 (1985)
78. D.G. Corneil, L.K. Stewart, Dominating sets in perfect graphs. *Discrete Math.* **86**, 145–164 (1990)
79. J. Dabney, B.C. Dean, S.T. Hedetniemi, A Linear-Time algorithm for broadcast domination in a tree. *Networks* **53**(2), 160–169 (2009)
80. P. Damaschke, H. Müller, D. Kratsch, Domination in convex and chordal bipartite graphs. *Inform. Process. Lett.* **36**, 231–236 (1990)
81. A. D'Atri, M. Moscarini, Distance-hereditary graphs, Steiner trees, and connected domination. *SIAM J. Comput.* **17**, 521–538 (1988)
82. D.P. Day, O.R. Oellermann, H.C. Swart, Steiner distance-hereditary graphs. *SIAM J. Discrete Math.* **7**, 437–442 (1994)
83. C.F. de Jaenisch, *Traite des Applications de l'Analyse Mathematiques au jeu des Echecs*. (Leningrad, 1862)
84. G.S. Domke, J.H. Hattingh, S.T. Hedetniemi, R.C. Laskar, L.R. Markus, Restrained domination in graphs. *Discrete Math.* **203**(1–3), 61–69 (1999)
85. S.E. Dreyfus, A.M. Law, *The Art and Theory of Dynamic Programming* (Academic, New York, 1977)
86. J.E. Dunbar, W. Goddard, S. Hedetniemi, A.A. McRae, M.A. Henning, The algorithmic complexity of minus domination in graphs. *Discrete Appl. Math.* **68**(1–2), 73–84 (1996)
87. S. Even, A. Pnueli, A. Lempel, Permutation graphs and transitive graphs. *J. Assoc. Comput. Mach.* **19**(3), 400–410 (1972)
88. L. Euler, Solutio problematis ad geometriam situs pertinentis. *Acad. Sci. Imp. Petropol.* **8**, 128–140 (1736)
89. M. Farber, Applications of linear programming duality to problems involving independence and domination. Ph.D. Thesis, Simon Fraser Univ., 1981
90. M. Farber, Domination and duality in weighted trees. *Congr. Numer.* **33**, 3–13 (1981)
91. M. Farber, Independent domination in chordal graphs. *Oper. Res. Lett.* **1**, 134–138 (1982)
92. M. Farber, Characterizations of strongly chordal graphs. *Discrete Math.* **43**, 173–189 (1983)
93. M. Farber, Domination, independent domination and duality in strongly chordal graphs. *Discrete Appl. Math.* **7**, 115–130 (1984)
94. M. Farber, J.M. Keil, Domination in permutation graphs. *J. Algorithms* **6**, 309–321 (1985)
95. A.M. Farley, S.T. Hedetniemi, A. Proskurowski, Partitioning trees: matching, domination and maximum diameter. *Int. J. Comput. Inform. Sci.* **10**, 55–61 (1981)
96. M.R. Fellows, M.N. Hoover, Perfect domination. *Australas. J. Combin.* **3**, 141–150 (1991)
97. C.M.H. de Figueiredo, J. Meidanis, C.P. de Mello, A linear-time algorithm for proper interval graph recognition. *Inform. Process. Lett.* **56**, 179–184 (1995)

98. G.H. Fricke, M.A. Henning, O.R. Oellermann, H.C. Swart, An efficient algorithm to compute the sum of two distance domination parameters. *Discrete Appl. Math.* **68**, 85–91 (1996)
99. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, New York, 1979)
100. F. Gavril, Algorithms for minimum colorings, maximum clique, minimum coverings by cliques and maximum independent set of a chordal graph. *SIAM J. Comput.* **1**, 180–187 (1972)
101. M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs* (Academic, New York, 1980)
102. M.C. Golumbic, Algorithmic aspect of perfect graphs. *Ann. Discrete Math.* **21**, 301–323 (1984)
103. J. Guo, R. Niedermeier, D. Raible, Improved algorithms and complexity results for power domination in graphs. *Algorithmica* **52**(2), 177–202 (2008)
104. M. Habib, R. McConnell, C. Paul, L. Viennot, Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.* **234**, 59–84 (2000)
105. P.H. Hammer, F. Maffray, Completely separable graphs. *Discrete Appl. Math.* **27**, 85–99 (1990)
106. E.O. Hare, S.T. Hedetniemi, A linear algorithm for computing the knight's domination number of a $K \times N$ chessboard. *Congr. Numer.* **59**, 115–130 (1987)
107. J.H. Hattingh, M.A. Henning, The complexity of upper distance irredundance. *Congr. Numer.* **91**, 107–115 (1992)
108. J.H. Hattingh, M.A. Henning, P.J. Slater, On the algorithmic complexity of signed domination in graphs. *Australas. J. Combin.* **12**, 101–112 (1995)
109. J.H. Hattingh, M.A. Henning, J.L. Walters, On the computational complexity of upper distance fractional domination. *Australas. J. Combin.* **7**, 133–144 (1993)
110. J.H. Hattingh, R.C. Laskar, On weak domination in graphs. *Ars Combin.* **49**, 205–216 (1998)
111. T.W. Haynes, S.M. Hedetniemi, S.T. Hedetniemi, M.A. Henning, Domination in graphs applied to electric power networks. *SIAM J. Discrete Math.* **15**(4), 519–529 (2002)
112. T.W. Haynes, S.T. Hedetniemi, P.J. Slater, *Fundamentals of Domination in Graphs* (Marcel Dekker, New York, 1997)
113. T.W. Haynes, S.T. Hedetniemi, P.J. Slater (eds.), *Domination in Graphs: Advanced Topics* (Marcel Dekker, New York, 1997)
114. S.M. Hedetniemi, S.T. Hedetniemi, D.P. Jacobs, Private domination: theory and algorithms. *Congr. Numer.* **79**, 147–157 (1990)
115. S.M. Hedetniemi, S.T. Hedetniemi, R.C. Laskar, Domination in trees: models and algorithms, in *Graph Theory with Applications to Algorithms and Computer Science*, ed. by Y. Alavi, G. Chartrand, L. Lesniak, D.R. Lick, C.E. Wall (Wiley, New York, 1985), pp. 423–442
116. S.M. Hedetniemi, A.A. McRae, D.A. Parks, Complexity results. in *Domination in Graphs: Advanced Topics*, Chapter 9, ed. by T.W. Haynes, S.T. Hedetniemi, P.J. Slater (Marcel Dekker, New York, 1997)
117. S.T. Hedetniemi, R.C. Laskar, Bibliography on domination in graphs and some basic definitions of domination parameters. *Discrete Math.* **86**, 257–277 (1990)
118. S.T. Hedetniemi, R.C. Laskar, J. Pfaff, A linear algorithm for finding a minimum dominating set in a cactus. *Discrete Appl. Math.* **13**, 287–292 (1986)
119. P. Hell, J. Huang, Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs. *SIAM J. Discrete Math.* **18**, 55–570 (2005)
120. A.J. Hoffman, A.W.J. Kolen, M. Sakarovitch, Totally-balanced and greedy matrices. *SIAM J. Algebraic Discrete Methods* **6**, 721–730 (1985)
121. E. Howorka, A characterization of distance-hereditary graphs. *Q. J. Math. Oxford Ser. 2* **28**, 417–420 (1977)
122. W. Hsu, The distance-domination numbers of trees. *Oper. Res. Lett.* **1**, 96–100 (1982)
123. W.-L. Hsu, A simple test for interval graphs. *Lect. Notes Comput. Sci.* **657**, 11–16 (1993)

124. W.-L. Hsu, T.-H. Ma, Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM J. Comput.* **28**, 1004–1020 (1999)
125. W.-L. Hsu, R.M. McConnell, PC trees and circular-ones arrangements. *Theor. Comput. Sci.* **296**, 59–74 (2003)
126. S.F. Hwang, G.J. Chang, The k -neighbor domination problem in block graphs. *Eur. J. Oper. Res.* **52**, 373–377 (1991)
127. O.H. Ibarra, Q. Zheng, Some efficient algorithms for permutation graphs. *J. Algorithms* **16**, 453–469 (1994)
128. R.W. Irving, On approximating the minimum independent dominating set. *Inform. Process. Lett.* **37**(4), 197–200 (1991)
129. M.S. Jacobson, K. Peters, Complexity questions for n -domination and related parameters. *Congr. Numer.* **68**, 7–22 (1989)
130. T.S. Jayaram, G. Sri Krishna, C. Pandu Rangan, A unified approach to solving domination problems on block graphs. Report TR-TCS-90-09, Dept. of Computer Science and Eng., Indian Inst. of Technology, 1990
131. D.S. Johnson, The NP-completeness column: an ongoing guide. *J. Algorithms* **5**, 147–160 (1984)
132. D.S. Johnson, The NP-completeness column: an ongoing guide. *J. Algorithms* **6**, 291–305, 434–451 (1985)
133. V. Kamakoti, C. Pandu Rangan, Efficient transitive reduction of permutation graphs and its applications. *J. Comput. Sci. Inform. Comput. S. India*, **23**(3), 52–59 (1993)
134. L. Kang, M.Y. Sohn, T.C.E. Cheng, Paired-domination in inflated graphs. *Theor. Comput. Sci.* **320**(2–3), 485–494 (2004)
135. O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems I: the p -centers. *SIAM J. Appl. Math.* **37**, 513–538 (1979)
136. O. Kariv, S.L. Hakimi, An algorithmic approach to network location problems II: the p -medians. *SIAM J. Appl. Math.* **37**, 539–560 (1979)
137. J.M. Keil, Total domination in interval graphs. *Inform. Process. Lett.* **22**, 171–174 (1986)
138. J.M. Keil, The complexity of domination problems in circle graphs. *Discrete Appl. Math.* **42**, 51–63 (1993)
139. J.M. Keil, D. Schaefer, An optimal algorithm for finding dominating cycles in circular-arc graphs. *Discrete Appl. Math.* **36**, 25–34 (1992)
140. T. Kikuno, N. Yoshida, Y. Kakuda, The NP-completeness of the dominating set problem in cubic planar graphs. *Trans. IEEE*, **E63-E(6)**:443–444 (1980)
141. T. Kikuno, N. Yoshida, Y. Kakuda, A linear algorithm for the domination number of a series-parallel graph. *Discrete Appl. Math.* **5**, 299–311 (1983)
142. E. Köhler, Connected domination and dominating clique in trapezoid graphs. *Discrete Appl. Math.* **99**, 91–110 (2000)
143. A. Kolen, Solving covering problems and the uncapacitated plant location problem on trees. *Eur. J. Oper. Res.* **12**, 266–278 (1983)
144. N. Korte, R.H. Möhring, An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.* **18**, 68–81 (1989)
145. D. Kratsch, Finding dominating cliques efficiently, in strongly chordal graphs and undirected path graphs. *Discrete Math.* **86**, 225–238 (1990)
146. D. Kratsch, Algorithms, in *Domination in Graphs: Advanced Topics*, Chapter 8, ed. by T.W. Haynes, S.T. Hedetniemi, P.J. Slater (Marcel Dekker, New York, 1997)
147. D. Kratsch, R.M. McConnell, K. Mehlhorn, J.P. Spinrad, Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM J. Discrete Math.* **36**, 326–353 (2006)
148. D. Kratsch, L. Stewart, Domination on cocomparability graphs. *SIAM J. Discrete Math.* **6**(3), 400–417 (1993)
149. R. Laskar, J. Pfaff, S.M. Hedetniemi, S.T. Hedetniemi, On the algorithmic complexity of total domination. *SIAM J. Algebraic Discrete Methods* **5**(3), 420–425 (1984)
150. E.L. Lawler, P.J. Slater, A linear time algorithm for finding an optimal dominating subforest of a tree, in *Graph Theory with Applications to Algorithms and Computer Science* (Wiley, New York, 1985), pp. 501–506

151. Y.D. Liang, C. Rhee, S.K. Dall, S. Lakshmivarahan, A new approach for the domination problem on permutation graphs. *Inform. Process. Lett.* **37**, 219–224 (1991)
152. C.-S. Liao, G.J. Chang, Algorithmic aspect of k -tuple domination in graphs. *Taiwan. J. Math.* **6**(3), 415–420 (2002)
153. M. Livingston, Q.F. Stout, Constant time computation of minimum dominating sets. *Congr. Numer.* **105**, 116–128 (1994)
154. P.J. Looges, S. Olariu, Optimal greedy algorithms for indifference graphs. *Comput. Math. Appl.* **25**, 15–25 (1993)
155. E. Loukakis, Two algorithms for determining a minimum independent dominating set. *Int. J. Comput. Math.* **15**, 213–229 (1984)
156. T.L. Lu, P.H. Ho, G.J. Chang, The domatic number problem in interval graphs. *SIAM J. Discrete Math.* **3**, 531–536 (1990)
157. K.L. Ma, C.W.H. Lam, Partition algorithm for the dominating set problem. *Congr. Numer.* **81**, 69–80 (1991)
158. G.K. Manacher, T.A. Mankus, Finding a domatic partition of an interval graph in time $O(n)$. *SIAM J. Discrete Math.* **9**, 167–172 (1996)
159. M.V. Marathe, H.B. Hunt III, S.S. Ravi, Efficient approximation algorithms for domatic partition and on-line coloring of circular arc graphs. *Discrete Appl. Math.* **64**, 135–149 (1996)
160. R.M. McConnell, J.P. Spinrad, Modular decomposition and transitive orientation. *Discrete Math.* **201**, 189–241 (1999)
161. A.A. McRae, S.T. Hedetniemi, Finding n -independent dominating sets. *Congr. Numer.* **85**, 235–244 (1991)
162. S.L. Mitchell, S.T. Hedetniemi, Edge domination in trees. *Congr. Numer.* **19**, 489–509 (1977)
163. S.L. Mitchell, S.T. Hedetniemi, Independent domination in trees. *Congr. Numer.* **29**, 639–656 (1979)
164. S.L. Mitchell, S.T. Hedetniemi, S. Goodman, Some linear algorithms on trees. *Congr. Numer.* **14**, 467–483 (1975)
165. H. Müller, A. Brandstädt, The NP-completeness of Steiner tree and dominating set for chordal bipartite graphs. *Theor. Comput. Sci.* **53**, 257–265 (1987)
166. K.S. Natarajan, L.J. White, Optimum domination in weighted trees. *Inform. Process. Lett.* **7**, 261–265 (1978)
167. G.L. Nemhauser, *Introduction to Dynamic Programming* (Wiley, New York, 1967)
168. S.L. Peng, M.S. Chang, A new approach for domatic number problem on interval graphs, in *Proc. National Comp. Symp.*, Taipei, Republic of China, 1991, pp. 236–241
169. S.L. Peng, M.S. Chang, A simple linear time algorithm for the domatic partition problem on strongly chordal graphs. *Inform. Process. Lett.* **43**, 297–300 (1992)
170. J. Pfaff, R. Laskar, S.T. Hedetniemi, Linear algorithms for independent domination and total domination in series-parallel graphs. *Congr. Numer.* **45**, 71–82 (1984)
171. A. Pnueli, A. Lempel, S. Even, Transitive orientation of graphs and identification of permutation graphs. *Can. J. Math.* **23**, 160–175 (1971)
172. A. Proskurowski, Minimum dominating cycles in 2-trees. *Int. J. Comput. Inform. Sci.* **8**, 405–417 (1979)
173. S. Rajbaum, Improved tree decomposition based algorithms for domination-like problems. *Lect. Notes Comut. Sci.* **2286**, 613–627 (2002)
174. G. Ramalingam, C. Pandu Rangan, Total domination in interval graphs revisited. *Inform. Process. Lett.* **27**, 17–21 (1988)
175. G. Ramalingam, C. Pandu Rangan, A unified approach to domination problems in interval graphs. *Inform. Process. Lett.* **27**, 271–274 (1988)
176. A. Raychaudhuri, On powers of interval and unit interval graphs. *Congr. Numer.* **59**, 235–242 (1987)
177. F.S. Roberts, On the compatibility between a graph and a simple order. *J. Combin. Theory B* **11**, 28–38 (1971)
178. J.S. Rohl, A faster lexicographic N queens algorithm. *Inform. Process. Lett.* **17**, 231–233 (1983)

179. D.J. Rose, R.E. Tarjan, G.S. Lueker, Algorithmic aspects of vertex elimination on graphs. *SIAM J. Discrete Math.* **5**, 266–283 (1976)
180. P. Scheffler, Linear-time algorithms for NP-complete problems restricted to partial k -trees. Technical Report 03/87, IMATH, Berlin, 1987
181. W.-K. Shih, W.-L. Hsu, A new planarity test. *Theor. Comput. Sci.* **223**, 179–191 (1999)
182. K. Simon, A new simple linear algorithm to recognize interval graphs. *Lect. Notes Comput. Sci.* **553**, 289–308 (1991)
183. P.J. Slater, R -domination in graphs. *J. Assoc. Comput. Mach.* **23**, 446–450 (1976)
184. P.J. Slater, Domination and location in acyclic graphs. *Networks* **17**, 55–64 (1987)
185. R. Sosic, J. Gu, A polynomial time algorithm for the N -queens problem. *SIGART Bull.* **2**(2), 7–11 (1990)
186. J. Spinrad, On comparability and permutation graphs. *SIAM J. Comput.* **14**, 658–670 (1985)
187. A. Srinivasa Rao, C. Pandu Rangan, Linear algorithm for domatic number problem on interval graphs. *Inform. Process. Lett.* **33**, 29–33 (1989/1990)
188. A. Srinivasan Rao, C. Pandu Rangan, Efficient algorithms for the minimum weighted dominating clique problem on permutation graphs. *Theor. Comput. Sci.* **91**, 1–21 (1991)
189. J.A. Telle, A. Proskurowski, Efficient sets in partial k -trees. *Discrete Appl. Math.* **44**, 109–117 (1993)
190. J.A. Telle, A. Proskurowski, Practical algorithms on partial k -trees with an application to domination-type problems, in F. Dehne, J.R. Sack, N. Santoro, S. Whitesides *Proc. Third Workshop on Algorithms and Data Structures (WADS'93)*. Lecture Notes in Computer Science, vol. 703, Montréal (Springer, Berlin, 1993), pp. 610–621
191. K.H. Tsai, W.L. Hsu, Fast algorithms for the dominating set problem on permutation graphs. *Algorithmica* **9**, 109–117 (1993)
192. C. Tsouros, M. Satratzemi, Tree search algorithms for the dominating vertex set problem. *Int. J. Comput. Math.* **47**, 127–133 (1993)
193. D.B. West, *Introduction to Graph Theory*, 2nd edn. (Prentice-Hall, Inc., Upper Saddle River, NJ, 2001)
194. K. White, M. Farber, W. Pulleyblank, Steiner trees, connected domination and strongly chordal graphs. *Networks* **15**, 109–124 (1985)
195. T.V. Wimer, Linear algorithms for the dominating cycle problems in series-parallel graphs, partial k -trees and Halin graphs. *Congr. Numer.* **56**, 289–298 (1986)
196. M. Yannakakis, F. Gavril, Edge dominating sets in graphs. *SIAM J. Appl. Math.* **38**(3), 364–372 (1980)
197. H.G. Yeh, Distance-hereditary graphs: combinatorial structures and algorithms. Ph.D. Thesis, Univ. Taiwan, 1997
198. H.G. Yeh, G.J. Chang, Algorithmic aspect of majority domination. *Taiwan. J. Math.* **1**, 343–350 (1997)
199. H.G. Yeh, G.J. Chang, The path-partition problem in bipartite distance-hereditary graphs. *Taiwanese J. Math.* **2**, 353–360 (1998)
200. H.G. Yeh, G.J. Chang, Weighted connected domination and Steiner trees in distance-hereditary graphs. *Discrete Appl. Math.* **87**, 245–253 (1998)
201. C. Yen, R.C.T. Lee, The weighted perfect domination problem. *Inform. Process. Lett.* **35**(6), 295–299 (1990)
202. C. Yen, R.C.T. Lee, The weighted perfect domination problem and its variants. *Discrete Appl. Math.* **66**, 147–160 (1996)
203. W.C.-K. Yen, The bottleneck independent domination on the classes of bipartite graphs and block graphs. *Inform. Sci.* **157**, 199–215 (2003)
204. W.C.-K. Yen, Bottleneck domination and bottleneck independent domination on graphs. *J. Inform. Sci. Eng.* **18**(2), 311–331 (2002)

Algorithms and Metaheuristics for Combinatorial Matrices

Ilias S. Kotsireas

Contents

1	Introduction.....	284
2	Preliminaries and Notations.....	284
3	Cyclotomy Algorithms.....	288
4	String Sorting Algorithms.....	289
4.1	Periodic Autocorrelation.....	290
4.2	Aperiodic Autocorrelation.....	292
4.3	A General String Sorting Algorithmic Scheme.....	293
5	Genetic Algorithms.....	294
5.1	A General GAs Algorithmic Scheme.....	295
6	Simulated Annealing.....	295
6.1	A General SA Algorithmic Scheme.....	296
7	Particle Swarm Optimization (PSO).....	297
8	Ant Colony Optimization (ACO).....	300
9	Combinatorial Optimization.....	303
10	Applications.....	304
11	Conclusion.....	304
	Cross-References.....	305
	Recommended Reading.....	305

Abstract

Combinatorial matrices are matrices that satisfy certain combinatorial properties and typically give rise to extremely challenging search problems with thousands of variables. In this chapter we present a survey of some recent algorithms to search for some kinds of combinatorial matrices, with an emphasis to algorithms within the realm of optimization and metaheuristics. It is to be noted that for most kinds of combinatorial matrices there are several known infinite classes

I.S. Kotsireas

Department of Physics and Computer Science, Wilfrid Laurier University, Waterloo, ON, Canada
e-mail: ikotsire@wlu.ca

in the literature, but these infinite classes do not suffice to cover the entire spectra of possible orders of these matrices, therefore it is necessary to resort to computational and meta-heuristic algorithms.

1 Introduction

The search for combinatorial matrices and their associated designs has been a fertile testing ground for algorithm designers for decades. These extremely challenging combinatorial problems have been tackled with algorithms using concepts and techniques from discrete mathematics, number theory, linear algebra, group theory, optimization, and metaheuristics. This chapter will survey some recent algorithms to search for combinatorial matrices, with an emphasis to algorithms within the realm of optimization and metaheuristics. The hope is that this chapter will arouse the interest of algorithm designers from various areas in order to invent new formalisms and new algorithms to search efficiently for combinatorial matrices. There are several open problems in the broad area of combinatorial matrices, and it is clear that new ideas are required to solve them. The existing algorithms continue to yield new results, especially in conjunction with the use of parallel programming techniques, but will still eventually reach a point of saturation, beyond which they will probably not be of much use. Therefore, cross-fertilization of research areas is necessary for producing new results in the search for new combinatorial matrices, at both the theoretical and the practical level.

Combinatorial matrices are defined as matrices that possess some combinatorial properties, such as prescribed row/column sums and special structure described in terms of circulant matrices. The research area of combinatorial matrices has been developed in a systematic manner over the last 50 years in the four books [10, 11, 48, 88]. It is worthwhile to point out that the unifying concept of combinatorial matrices includes well-known and widely studied categories of matrices, such as adjacency and incidence matrices of graphs, Hadamard matrices, and doubly stochastic matrices.

In addition, there is a number of more recent books that contain useful information and new developments in the area of combinatorial matrices and the closely related area of design theory. The book [52] (see also the update [53]) sheds considerable light in the cocyclic (i.e., group cohomological) aspects of certain kinds of combinatorial matrices and is the only systematic book-size treatment of this topic. The book [94] features a very readable exposition of design theory with emphasis on bent functions and coding theory aspects. The book [19] offers an alternative algebraic foundation of design theory. The book [54] places its emphasis on symmetric designs and their properties.

2 Preliminaries and Notations

A wide class of combinatorial matrices (or the relevant sequences) can be defined via the concepts of the periodic and aperiodic (or nonperiodic) autocorrelation functions associated to a finite sequence. These two concepts have been invented

and studied widely within the engineering community, and their importance has been recognized in the combinatorics community as well, especially in connection with several kinds of combinatorial matrices.

All sequences in this chapter will be finite and will have elements either from $\{-1, +1\}$ (binary sequences) or from $\{-1, 0, +1\}$ (ternary sequences).

Definition 1 The periodic autocorrelation function associated to a finite sequence $A = [a_1, \dots, a_n]$ of length n is defined as

$$P_A(i) = \sum_{k=1}^n a_k a_{k+i}, \quad i = 0, \dots, n-1,$$

where $k + i$ is taken modulo n , when $k + i > n$.

Definition 2 The aperiodic (or nonperiodic) autocorrelation function of a sequence $[a_1, \dots, a_n]$ of length n is defined as

$$N_A(i) = \sum_{k=1}^{n-i} a_k a_{k+i}, \quad i = 0, \dots, n-1,$$

where $k + i$ is taken modulo n , when $k + i > n$.

[Definitions 1](#) and [2](#) are best clarified with an example.

Example 1 For a finite sequence of length $n = 7$, $A = [a_1, \dots, a_7]$, we have

$$\begin{aligned} P_A(0) &= a_1^2 + a_2^2 + a_3^2 + a_4^2 + a_5^2 + a_6^2 + a_7^2 \\ P_A(1) &= a_1 a_2 + a_2 a_3 + a_3 a_4 + a_4 a_5 + a_5 a_6 + a_6 a_7 + a_7 a_1 \\ P_A(2) &= a_1 a_3 + a_2 a_4 + a_3 a_5 + a_4 a_6 + a_5 a_7 + a_6 a_1 + a_7 a_2 \\ P_A(3) &= a_1 a_4 + a_2 a_5 + a_3 a_6 + a_4 a_7 + a_5 a_1 + a_6 a_2 + a_7 a_3 \\ P_A(4) &= a_1 a_4 + a_2 a_5 + a_3 a_6 + a_4 a_7 + a_5 a_1 + a_6 a_2 + a_7 a_3 \\ P_A(5) &= a_1 a_3 + a_2 a_4 + a_3 a_5 + a_4 a_6 + a_5 a_7 + a_6 a_1 + a_7 a_2 \\ P_A(6) &= a_1 a_2 + a_2 a_3 + a_3 a_4 + a_4 a_5 + a_5 a_6 + a_6 a_7 + a_7 a_1 \\ N_A(0) &= a_1^2 + a_2^2 + a_3^2 + a_4^2 + a_5^2 + a_6^2 + a_7^2 \\ N_A(1) &= a_1 a_2 + a_2 a_3 + a_3 a_4 + a_4 a_5 + a_5 a_6 + a_6 a_7 \\ N_A(2) &= a_1 a_3 + a_2 a_4 + a_3 a_5 + a_4 a_6 + a_5 a_7 \\ N_A(3) &= a_1 a_4 + a_2 a_5 + a_3 a_6 + a_4 a_7 \\ N_A(4) &= a_1 a_5 + a_2 a_6 + a_3 a_7 \\ N_A(5) &= a_1 a_6 + a_2 a_7 \\ N_A(6) &= a_1 a_7. \end{aligned}$$

Definition 3 Two finite sequences $[a_1, \dots, a_n]$ and $[b_1, \dots, b_n]$ of length n each are said to have constant periodic autocorrelation if there is a constant c such that

$$P_A(i) + P_B(i) = c, \quad i = 1, \dots, n - 1.$$

Definition 4 Two sequences $[a_1, \dots, a_n]$ and $[b_1, \dots, b_n]$ of length n each are said to have constant aperiodic autocorrelation if there is a constant c such that

$$N_A(i) + N_B(i) = c, \quad i = 1, \dots, n - 1.$$

Note that the index $i = 0$ is omitted from [Definitions 3](#) and [4](#), because of the property $P_A(0) = N_A(0) = \sum_{j=1}^n a_j^2$. Also note that when writing out binary and ternary sequences, it is customary in the combinatorial literature to denote -1 by $-$, zero by 0 , and $+1$ by $+$, and this is the convention adopted in the current chapter. Also note that [Definitions 3](#) and [4](#) can be extended to more than two sequences. [Definitions 3](#) and [4](#) are best illustrated with an example.

Example 2 The following binary sequences with $n = 26$ have constant aperiodic autocorrelation with $c = 0$:

$$\begin{aligned} & + + + + - + + - - + - + - + - + + + - - + + + \\ & + + + + - + + - - + - + + + + + - + - - + + - - \end{aligned}$$

The following ternary sequences with $n = 30$ have constant aperiodic autocorrelation with $c = 0$:

$$\begin{aligned} & + + - - - - + - - 0000 + 0 + - + + + + - - + - + + \\ & + + - - - - - + - - + + - - 0 - 0000 - - + + - + - - \end{aligned}$$

The following properties of the periodic and aperiodic autocorrelation functions can be observed (and verified) in [Examples 1](#) and [2](#):

Property 1 (Symmetry)

$$P_A(i) = P_A(n - i), i = 0, 1, \dots, n.$$

Property 2 (Complementarity)

$$P_A(i) = N_A(i) + N_A(n - i), i = 0, 1, \dots, n.$$

Property 3 (Second Elementary Symmetric Function)

$$P_A(1) + \dots + P_A(n - 1) = 2 e_2(a_1, \dots, a_n)$$

$$N_A(1) + \dots + N_A(n - 1) = e_2(a_1, \dots, a_n)$$

where $e_2(a_1, \dots, a_n) = \sum_{1 \leq i < j \leq n} a_i a_j$ is the second elementary symmetric function in the n variables a_1, \dots, a_n .

The proof of the three properties stated above is a straightforward application of [Definitions 1](#) and [2](#) and is left to the reader.

The complementarity property of periodic and aperiodic autocorrelation functions implies that:

Property 4 If two sequences have constant aperiodic autocorrelation, then they must also have constant periodic autocorrelation.

Therefore, the two pairs of sequences in [Example 2](#) also have constant periodic autocorrelation with $c = 0$.

The converse of the above property does not hold, and counterexamples may be easily found.

The concept of a circulant matrix is also important in connection with several kinds of combinatorial matrices and sequences.

Definition 5 A $n \times n$ matrix $C(A)$ is called circulant if every row (except the first) is obtained by the previous row by a right cyclic shift by one. In particular,

$$C(A) = \begin{bmatrix} a_1 & a_2 & \dots & a_{n-1} & a_n \\ a_n & a_1 & \dots & a_{n-2} & a_{n-1} \\ \vdots & \vdots & \dots & \vdots & \vdots \\ a_3 & a_4 & \dots & a_1 & a_2 \\ a_2 & a_3 & \dots & a_n & a_1. \end{bmatrix}$$

The relationship between circulant matrices and periodic autocorrelation is described by the following property, whose proof is also a straightforward application of [Definitions 1](#) and [2](#).

Property 5 Consider a finite sequence $A = [a_1, \dots, a_n]$ of length n and the circulant matrix $C(A)$ whose first row is equal to A . Then $P_A(i)$ is the inner product of the first row of $C(A)$ and the $i + 1$ row of $C(A)$.

It turns out that the concepts of periodic and aperiodic autocorrelation can serve as the unifying concepts needed to define several kinds of combinatorial matrices and sequences simultaneously. We summarize some of these kinds of combinatorial matrices and sequences in the next table.

Note that in [Table 1](#), TCP stands for “ternary complementary pairs” and PCS stands for “periodic complementary sequences.” We refer the reader to the papers listed in [Table 1](#) for the complete definitions of these combinatorial objects.

Table 1 Combinatorial matrices and sequences

Number/type of sequences	Defining property	Name	References
2 binary	aper. autoc. 0	Golay sequences	[8, 22, 36, 40, 41]
2 binary	per. autoc. 0	Hadamard matrices	[60]
2 binary	per. autoc. 2	D-optimal matrices	[29, 61]
2 binary	per. autoc. -2	Hadamard matrices	[63]
2 ternary	aper. autoc. 0	TCP	[18, 43]
2 ternary	per. autoc. 0	Weighing matrices	[64, 65]
3 binary	aper. autoc. const.	Normal sequences	[27]
4 binary	aper. autoc. 0	Base sequences	[26]
4 binary	aper. autoc. 0	Turyn-type sequences	[57]
4 ternary	aper. autoc. 0	T-sequences	[57]
2 ... 12 binary	per. autoc. 0	PCS	[6, 28, 66]

In the remainder of this chapter, we focus on various kinds of algorithms that can be used to search efficiently for combinatorial matrices and sequences listed in [Table 1](#), as well as any other such combinatorial objects that can be defined via periodic and aperiodic autocorrelation. In fact, we will concentrate on general algorithmic schemes that are applicable to all such combinatorial objects and will provide high-level descriptions for these algorithmic schemes.

3 Cyclotomy Algorithms

Cyclotomy algorithms to search for combinatorial matrices and sequences are based on the premise that certain combinatorial concepts called “supplementary difference sets” (abbreviated as SDS) can be constructed by taking unions of cyclotomic classes (or cosets) or group action orbits of a suitable subgroup of the group of invertible elements of the ring of integers modulo n , where n is a parameter referring to the length of the sequences. This powerful theme has been exploited by several authors at the theoretical and practical level over more than two decades.

The crucial concept of an SDS has been introduced by Jennifer Seberry 40 years ago [101, 102].

Definition 6 Let n be a positive integer and let S_1, \dots, S_k be subsets of \mathbb{Z}_n with cardinalities n_1, \dots, n_k , respectively. Let $T_i = \{(s_1^i - s_2^i) \bmod n, (s_2^i - s_1^i) \bmod n, s_1^i, s_2^i \in S_i\}$ be the multiset of all pairwise differences (taken modulo n) of all elements in S_i , for $i = 1, \dots, k$. If the multiset $T_1 \cup \dots \cup T_k$ is equal to λ copies of $\{1, \dots, n-1\}$, then S_1, \dots, S_k form an SDS with parameters $k - \{n; n_1, \dots, n_k; \lambda\}$.

A simple counting argument shows that existence of an SDS with parameters $k - \{n; n_1, \dots, n_k; \lambda\}$ implies the following condition:

$$\sum_{i=1}^k n_i(n_i - 1) = \lambda(n - 1).$$

Here is a toy example, where the SDS property can be verified by hand.

Example 3 Let $n = 13$ and $t = 3$. Then there are 4 cyclotomic classes: $S_1 = [1, 3, 9]$, $S_2 = [2, 5, 6]$, $S_3 = [4, 10, 12]$ and $S_4 = [7, 8, 11]$. It turns out that S_1 and S_2 form an SDS with parameters $2 - \{13; 3, 3; 1\}$ because $(\text{mod } 13)$ we have

$$T_1 \cup T_2 = \{1-3, 1-9, 3-9, 3-1, 9-1, 9-3\} \cup \{2-5, 2-6, 5-6, 5-2, 6-2, 6-5\},$$

i.e.,

$$T_1 \cup T_2 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}.$$

Here is a non-toy example, where the SDS property needs to be verified by a computer program.

Example 4 Let $n = 323$ and $t = 3$. Then there are 4 cyclotomic classes C_1, C_2, C_3, C_4 with cardinalities $n_1 = 144$, $n_2 = 144$, $n_3 = 18$ and $n_4 = 16$. Then the sets $S_1 = C_1 \cup C_3$ and $S_2 = S_1$ form an SDS with parameters $2 - \{323; 162, 162; 162\}$.

The paper [103] states a general theorem that indicates conditions under which suitable unions of cyclotomic classes form an SDS with specific parameters. The paper [38] contains recent theoretical results and in particular gives two constructions of skew Hadamard difference sets in the additive groups of suitable finite fields by using unions of cyclotomic classes. The paper [39] gives two constructions of strongly regular Cayley graphs on finite fields by using unions of cyclotomic classes. The paper [97] explores the idea of blending a mathematical programming formalism with a cyclotomy-based approach; the resulting algorithm is quite promising and deserves further investigation. Chapter 3 of the PhD thesis [47] is a virtual treasure trove of cyclotomy-based algorithms and applications of such algorithms to several different kinds of combinatorial matrices and sequences. Some of the most spectacular successes of cyclotomy algorithms include the discovery of new orders of Hadamard and skew Hadamard matrices [21, 23–25] as well as D-optimal matrices [29]. These types of methods are bound to continue to produce new results.

4 String Sorting Algorithms

String sorting algorithms to search for combinatorial matrices and sequences are based on certain restrictions that the properties of constant periodic or aperiodic autocorrelation imply. These restrictions possess the important advantage that they

can be used to decouple the (quadratic) equations in [Definitions 3](#) and [4](#). In addition, one needs to carefully distinguish between periodic and aperiodic autocorrelation, because the corresponding restrictions are of a different nature.

4.1 Periodic Autocorrelation

The restrictions imposed by the constancy of periodic autocorrelation are described via the concept of power spectral density (PSD) associated to a finite sequence. We illustrate the main idea in the context of Hadamard matrices with two circulant cores, following [\[42, 63\]](#).

Let n be an odd positive integer and suppose that two binary sequences $A = [a_1, \dots, a_n]$ and $B = [b_1, \dots, b_n]$, both of length n , have constant periodic autocorrelation function -2 , i.e.,

$$P_A(s) + P_B(s) = -2, \text{ for } s = 1, \dots, \frac{n-1}{2}. \quad (1)$$

Definition 7 Let $\omega = e^{\frac{2\pi i}{n}}$ be a primitive n -th root of unity.

The discrete Fourier transform (DFT) of A is defined by

$$\text{DFT}_A(s) = \sum_{k=1}^n a_k \omega^{ks}, \text{ for } s = 0, \dots, n-1.$$

The power spectral density (PSD) of A is defined by

$$\text{PSD}_A(s) = |\text{DFT}_A(s)|^2, \text{ for } s = 0, \dots, n-1.$$

Therefore, the PSD values are the squared magnitudes of the DFT values, i.e., $\text{PSD}_A(s) = \text{Re}(\text{DFT}_A(s))^2 + \text{Im}(\text{DFT}_A(s))^2$. Note that the well-known properties $\omega^n = 1$ and $\text{DFT}_A(n-s) = \text{DFT}_A(s)$, $s = 0, 1, \dots, n-1$ are routinely used in order to perform DFT/PSD calculations efficiently.

It can be proved (see [\[42\]](#)) that property (Eq. 1) implies that

$$\text{PSD}_A(s) + \text{PSD}_B(s) = 2n + 2, \text{ for } s = 1, \dots, \frac{n-1}{2}. \quad (2)$$

Example 5 Let $n = 55$ and consider the two sequences from [\[13\]](#):

$$\begin{aligned} & -++-+++\cdots+-+--++-++-++-+--+--++-+--+--++-+--+--++-+--+-- \\ & +---++-+--+--++-+--+--++-+--+--++-+--+--++-+--+--++-+--+-- \end{aligned}$$

The above sequences have constant periodic autocorrelation -2 and can be used to construct a Hadamard matrix of order $2 \cdot 55 + 2 = 112$. The following table records the $\frac{55-1}{2} = 27$ PSD values for the above sequences, as well as their sums, which is equal to 112 for $s = 1, \dots, 27$ as expected.

Table 2 PSD values for binary sequences with $n = 55$

s	$\text{PSD}_A(s)$	$\text{PSD}_B(s)$	$\text{PSD}_A(s) + \text{PSD}_B(s)$
1	29.80737844157036985336002	82.19262155456297731046190	112
2	30.28944472490585995185313	81.71055527763585566645278	112
3	55.43998376085268510929794	56.56001623465057222621275	112
4	102.0320272601899733257176	9.967972741391460830458954	112
5	89.65677565516284911337367	22.34322434507586735097245	112
6	15.47369185908972215139240	96.52630813774760368151000	112
7	11.47686739850811259463546	100.5231326000467370829829	112
8	76.73731376355725457366036	35.26268623768915584169775	112
9	46.11225525013170242777989	65.88774474677226578858688	112
10	0.4008926865885775543776042	111.5991073136016861161094	112
11	38.11145618000341879780940	73.88854381999376144822824	112
12	17.46064389518599154629647	94.53935610256366875094971	112
13	34.87931253417465350073346	77.12068746596972533277206	112
14	82.62901343793391797610934	29.37098656225313882607487	112
15	93.37954082614625749821953	18.62045917360481914328922	112
16	60.59338387747255107616868	51.40661612647071355873921	112
17	92.66303952890268414319827	19.33696047140536416932549	112
18	106.5590520047682870981779	5.440947992051451856194138	112
19	105.3670134147016013500540	6.632986586404617855942329	112
20	79.93243752088631246467129	32.06756247928573785046143	112
21	87.07144169101030219209510	24.92855831006362765566586	112
22	73.88854381999762010973329	38.11145617999893554527716	112
23	11.57785413372743006711576	100.4221458663961229278652	112
24	38.63848171187360617293295	73.36151828958980649610761	112
25	16.63035331130720400371166	95.36964668866435905703407	112
26	108.5508478995728704749974	3.449152098190036553335345	112
27	6.640953414706377747434240	105.3590465799776576376609	112

It is instructive to examine [Table 2](#) in order to see how [Eq. \(2\)](#) are materialized for the particular solution of [Example 5](#).

4.1.1 The PSD Criterion

[Equation \(2\)](#) can be used to derive the so-called PSD criterion by first remarking that since the PSD values are nonnegative, if for some $s \in \{1, \dots, \frac{n-1}{2}\}$ it turns out that $\text{PSD}_A(s) > 2n + 2$ or $\text{PSD}_B(s) > 2n + 2$, then the corresponding A or B candidate sequence can be safely discarded from the search.

4.1.2 A Subtle Point: Integer PSD Values

There is a subtle point regarding the computation of the PSD values that occur in string sorting algorithms, as well as other filtering-based algorithms to search for

sequences using the PSD criterion. The subtlety comes from the fact that although in the vast majority of cases the PSD values are (positive) floating point numbers, there exist cases when the PSD values are (positive) integers. A particular case when this phenomenon occurs is described in the following result proved in [64].

Lemma 1 *Let n be an odd integer such that $n \equiv 0 \pmod{3}$ and let $m = \frac{n}{3}$. Let $\omega = e^{\frac{2\pi i}{n}} = \cos\left(\frac{2\pi}{n}\right) + i \sin\left(\frac{2\pi}{n}\right)$ the principal n -th root of unity. Let $[a_1, \dots, a_n]$ be a sequence with elements from $\{-1, 0, +1\}$. Then we have that $\text{DFT}([a_1, \dots, a_n], m)$ can be evaluated explicitly in closed form and $\text{PSD}([a_1, \dots, a_n], m)$ is a nonnegative integer. The explicit evaluations are given by*

$$\begin{aligned}\text{DFT}([a_1, \dots, a_n], m) &= \left(A_1 - \frac{1}{2}A_2 - \frac{1}{2}A_3 \right) + \left(\frac{\sqrt{3}}{2}A_2 - \frac{\sqrt{3}}{2}A_3 \right) i \\ \text{PSD}([a_1, \dots, a_n], m) &= A_1^2 + A_2^2 + A_3^2 - A_1A_2 - A_1A_3 - A_2A_3\end{aligned}$$

where

$$A_1 = \sum_{i=0}^{m-1} a_{3i+1}, \quad A_2 = \sum_{i=0}^{m-1} a_{3i+2}, \quad A_3 = \sum_{i=0}^{m-1} a_{3i+3}.$$

It is important to carefully account for the phenomenon described by Lemma 1 in any implementation of string sorting algorithms in order not to miss any solutions due to numerical round-off error. One way of doing this is to omit the $n/3$ -th value from the construction of the string.

4.1.3 Bracelets and Necklaces

The symmetry property of periodic autocorrelation and PSD values under cyclic shifting must also be taken into account in order to avoid redundant computations for sequences that have the same periodic autocorrelation and PSD values. It turns out that the right combinatorial structures to deal with this symmetry property are bracelets and necklaces. The papers [89] and [90] and subsequent work by J. Sawada and his collaborators provide constant amortized time (CAT) algorithms to generate bracelets and necklaces, as well as extremely efficient C implementations of these algorithms.

4.2 Aperiodic Autocorrelation

The restrictions imposed by the constancy of periodic autocorrelation are described via the concept of a certain infinite class of functions associated to a finite sequence. We illustrate the main idea in the context of Turyn-type sequences, following [57].

Let n be an even positive integer and suppose that four binary sequences $X = [x_1, \dots, x_n]$ and $Y = [y_1, \dots, y_n]$, $Z = [z_1, \dots, z_n]$ and $W = [w_1, \dots, w_{n-1}]$ of

lengths $n, n, n, n - 1$ respectively, have constant aperiodic autocorrelation function in the sense that

$$N_X(s) + N_Y(s) + 2N_Z(s) + 2N_W(s) = 0, \text{ for } s = 1, \dots, n - 1. \quad (3)$$

The sequences X, Y, Z, W satisfying (3) are called Turyn type.

We associate to a binary sequence X of length n , the periodic (of period 2π) nonnegative function

$$f_X(\theta) = N_X(0) + 2 \sum_{j=1}^{n-1} N_X(j) \cos j\theta.$$

It can be proved (see [57]) that property (Eq. 3) implies that

$$f_X(\theta) + f_Y(\theta) + 2f_Z(\theta) + 2f_W(\theta) = 6n - 2, \quad \forall \theta. \quad (4)$$

4.2.1 Aperiodic Version of the PSD Criterion

Equation (4) can be used to derive the analog (an aperiodic version) of the PSD criterion in the aperiodic case. If for some value of θ it turns out that $f_X(\theta) > 6n - 2$ or $f_Y(\theta) > 6n - 2$, or $f_Z(\theta) > 3n - 1$ or $f_W(\theta) > 3n - 1$, then the corresponding X, Y, Z, W candidate sequence can be safely discarded from the search. In addition, if for some value of θ it turns out that $f_X(\theta) + f_Y(\theta) > 6n - 2$, then the corresponding pair (X, Y) of candidates sequences can be safely discarded from the search. Similar conditions can be derived for other partial sums of the four values $f_X(\theta), f_Y(\theta), f_Z(\theta), f_W(\theta)$ for arbitrary (but fixed) values of θ . The preprocessing of the sequences based on this aperiodic version of the PSD criterion is often carried out via a finite set of θ values, such as $\left\{ \frac{j\pi}{500}, j = 1, \dots, 500 \right\}$.

4.3 A General String Sorting Algorithmic Scheme

Based on the PSD criterion (periodic and aperiodic versions), a general string sorting algorithmic scheme can be defined as follows.

INPUT: lengths and types (binary/ternary) of two sequences A, B and the autocorrelation (periodic/aperiodic) property satisfied, value of PSD constant (or its aperiodic analog)

OUTPUT: sequences satisfying the input requirements (if any are found)

- Preprocess each one of the two sequences separately using the applicable version of the PSD criterion. In particular, discard all candidate sequences that violate the applicable version of the PSD criterion. If the set of all possible candidate sequences is too large to preprocess in its entirety, then use randomized methods to generate a representative part of this set.
- Encode candidate A -sequences by a string, namely, the concatenation of the integer parts of its PSD values, or the $f(\theta)$ values, for a small set of θ values.

Encode candidate B -sequences by a similar string, but taking the difference of the PSD (or $f(\theta)$ value) from the PSD constant (or its aperiodic analog).

- Sort the files containing the string encodings corresponding to each A -sequence and B -sequences separately. When the files containing the string encodings are too large to be sorted directly, this phase involves a distributed sorting algorithm.
- Look for identical strings in the two sorted files and output the corresponding solutions.

String sorting algorithms have been used in [?, 64] to compute several new weighing matrices of various weights constructed from two circulant cores, using the PSD criterion for periodic autocorrelation.

The ternary sequences with $n = 30$ in [Example 2](#) have been computed by the author using a C implementation of a string sorting algorithm and the aperiodic version of the PSD criterion.

5 Genetic Algorithms

Genetic algorithms (GAs) form a powerful metaheuristic method that exploits algorithmic analogs of concepts from Darwin's theory of evolution to design efficient search methods. The theory of genetic algorithms is presented in a extremely readable way in the classical book [44]. Among the more recent systematic treatments of GAs, one can consult [87]. Genetic algorithms have been applied successfully to tackle a wide range of difficult problems across many application areas. A central concept in the theory of GAs is the “objective function.” A judicious choice of an objective function is necessary in order to be able to apply GAs to solve a specific problem. Another important ingredient in the theory of GAs is the encoding of the solution space as a set of binary vectors. We will exemplify below how to formalize a problem of searching for combinatorial matrices as a GA problem, using binary sequences with constant periodic autocorrelation -2 as a case study, and following the exposition in [59].

Let n be an odd positive integer and suppose that two binary sequences $A = [a_1, \dots, a_n]$ and $B = [b_1, \dots, b_n]$, both of length n , have constant periodic autocorrelation function -2 , as in (1). Since the $m = \frac{n-1}{2}$ [Eq. \(1\)](#) must be satisfied simultaneously, one can consider the following two types of objective functions arising naturally:

$$OF_1 = | P_A(1) + P_B(1) + 2 | + \dots + | P_A(m) + P_B(m) + 2 |$$

and

$$OF_2 = (P_A(1) + P_B(1) + 2)^2 + \dots + (P_A(m) + P_B(m) + 2)^2.$$

There are two potentially important differences between the objective functions OF_1 OF_2 . First, OF_1 is not a continuous function, while OF_2 is. Second, OF_1 takes on smaller values than OF_2 . The GA will seek to minimize OF_1 and OF_2 , i.e., to find $2n$ $\{\pm 1\}$ values of the $2n$ variables $a_1, \dots, a_n, b_1, \dots, b_n$ such that OF_1 and OF_2

attain the value 0. Evidently, a set of $2n \{\pm 1\}$ values with this property is also a solution of the original problem of finding two binary sequences with constant periodic autocorrelation function -2 .

The GAs requirement of encoding the solution space as a set of binary vectors is inherent in the problem of searching for binary sequences with constant periodic/aperiodic autocorrelation function.

5.1 A General GAs Algorithmic Scheme

INPUT: k binary sequences A_1, \dots, A_k , all of length n , and the autocorrelation (periodic/aperiodic) property satisfied

OUTPUT: k -tuples of sequences satisfying the input requirements (if any are found)

- Specify an initial randomly chosen population (first generation) of binary sequences of length n .
- Encode the required periodic/aperiodic autocorrelation property in the form of OF_1 or OF_2 .
- Set current generation to be the first generation.
- Evolve the current generation to the next generation, using a set of genetic operators. Commonly used such operators include reproduction, crossover, and mutation.
- Examine the next generation to see whether it contains k binary sequences with the required periodic/aperiodic autocorrelation property. If yes, then output this solution and exit. If no, then set current generation to be the next generation and iterate the previous step.

Note that popular termination criteria for GAs include a predetermined number of generations to evolve or a predetermined amount of execution time.

There are several papers that use GAs to solve design-theoretic and combinatorial problems. In [1] the authors devise a GA to search for cocyclic Hadamard matrices. In [2] the author explains how to use genetic algorithms to solve three design-theoretic problems of graph-theoretic flavor. In [50] the authors apply GAs to construct D-optimal designs. In [68] the authors use so-called competent GAs to search efficiently for weighing matrices. In [17] the authors use SGA (simple genetic algorithm) to construct Hadamard matrices of various orders. The thesis [84] is concerned with the application of GAs to construct D-optimal experimental designs. The thesis [46] contains GAs to search for normal and near-Yang sequences.

6 Simulated Annealing

Simulated annealing (SA) was presented in [58] as a new approach to approximate solutions of difficult optimization problems. This approach is inspired from statistical mechanics and is motivated by an analogy to the behavior of physical systems in the presence of heat. In the words of Kirkpatrick et al.,

there is a deep and useful connection between statistical mechanics (the behavior of system with many degrees of freedom in thermal equilibrium at a finite temperature) and multivariate or combinatorial optimization (finding the minimum of a given function depending on many parameters).

In particular, they found that the Metropolis algorithm was well suited to approximate the observed behavior of solids when subjected to an annealing process. Recall that a local optimization method starts with some initial solution and, at each iteration, searches its neighbors for a better solution until no better solution can be found. A weakness of the local optimization method is that the program may get trapped at some local minima instead of detecting a global minimum. Simulated annealing addresses this issue by using randomization to improve on the local optimization search process, and in particular, occasional uphill moves (i.e., less than optimal solutions) are accepted with the hope that by doing so, a better solution will be obtainable later on. There are several parameters in the simulated annealing process which can significantly impact the actual performance, and there do not seem to be general rules on how to choose these parameters for the specific problem at hand.

In the annealing process, the physical system being optimized is first “melted” at a high temperature. The temperature is then decreased slowly until the system “freezes,” and no further physical changes occur. When the system’s structure is “frozen,” this correspond to a minimum energy configuration. The physical analogy for simulated annealing is often described using the annealing method for growing a crystal. The rate at which the temperature is reduced is vitally important to the annealing process. If the cooling is done too quickly, widespread irregularities will form and the trapped energy level will be higher than what one would find in a perfectly structured crystal. It can be shown that the states of this physical system correspond to the solutions of an optimization problem. The energy of a state in the physical world corresponds to the cost of a solution in the optimization world. The minimum energy configuration that is obtained when a system is frozen corresponds to an optimal solution in an optimization problem.

6.1 A General SA Algorithmic Scheme

The basic ingredients needed to formulate a problem in a manner amenable to SA algorithms are:

1. A finite set S . This is the set of all possible solutions.
2. An objective function OF defined on S .
3. The set S^* is the set of global minima (i.e., the desired solutions) using the OF .
It is assumed that $S^* \subset S$, a proper subset of S .
4. For each $i \in S$, we define a set $S(i) \subset S - \{i\}$ to be the set of *neighbors* of i .
5. A nonincreasing function $\alpha(t)$ called the *cooling schedule*, which controls how the temperature is lowered.

With the previous terminology in place, a general SA algorithmic scheme looks like:

```

Randomly select initial solution  $s_0$ ;
Select an initial temperature  $t_0 > 0$ ;
Select a temperature reduction function  $\alpha$ ;
Repeat
    Repeat
        Randomly select a neighbor,  $s$  of  $s_0$ ;
         $\delta = \text{OF}(s) - \text{OF}(s_0)$ ;
        if( $\delta < 0$ )
            then  $s_0 = s$ ;
        else
            generate random  $x$  uniformly in the range  $(0, 1)$ ;
            if( $x < e^{-\delta/t}$ )
                then  $s_0 = s$ ;
    Until  $\text{iteration\_count} = nrep$ 
    Set  $t = \alpha(t)$ ;
Until stopping condition = true
 $s_0$  is the approximation to the optimal solution

```

Recall that the goal of simulated annealing is to improve on the local search strategies by allowing some uphill moves in a controlled manner. The above SA scheme accepts a solution at each iteration if it is determined to be better than the current “best” solution. However, we also see how a less than optimal solution can be accepted and that it is accepted with probability $e^{-\delta/t}$. Also note that for the purposes of the OF required by SA, one can use OF_1 and OF_2 as defined previously for GAs. Commonly used cooling schedules include $\alpha(t) = \alpha t$, where $\alpha < 1$ and $\alpha(t) = \frac{t}{1 + \beta t}$ where β is small.

The papers [12, 55, 70, 72, 77] use SA techniques to tackle design theory problems.

7 Particle Swarm Optimization (PSO)

Particle swarm Optimization (PSO) is a population-based metaheuristic algorithm. It was introduced by R.C. Eberhart and J. Kennedy in 1995 [35] primarily for solving numerical optimization problems, as an alternative to evolutionary algorithms (EAs) [3]. Its verified efficiency in challenging optimization problems as well as its easy implementation rapidly placed PSO in a salient position among the state-of-the-art algorithms. Today, PSO counts a vast number of applications in diverse scientific fields [4, 5, 78], as well as an extensive bibliography [14, 37, 56, 83], and published scientific software [100].

Putting it formally, consider the n -dimensional global optimization problem:

$$\min_{x \in X \subset \mathbb{R}^n} f(x).$$

PSO probes the search space by using a *swarm*, namely, a set of search points:

$$S = \{x_1, x_2, \dots, x_N\}, \quad x_i \in X, \quad i \in I = \{1, 2, \dots, N\}.$$

Each search point of S constitutes a *particle*, i.e., a search vector:

$$x_i = (x_{i1}, x_{i2}, \dots, x_{in})^\top \in X, \quad i \in I,$$

which is randomly initialized in X and allowed to iteratively move within it. Its motion is based on stochastically adapted position shifts, called *velocity*, while it retains in memory the *best position* it has ever visited. These quantities are denoted as

$$v_i = (v_{i1}, v_{i2}, \dots, v_{in})^\top, \quad p_i = (p_{i1}, p_{i2}, \dots, p_{in})^\top, \quad i \in I,$$

respectively. Thus, if t denotes the current iteration of the algorithm, it holds that

$$p_i(t) = \arg \min_{q \in \{0, 1, \dots, t\}} \{f(x_i(q))\}, \quad i \in I.$$

The best positions constitute a sort of experience for the particles, guiding them towards the most promising regions of the search space, i.e., regions that possess lower function values.

In order to avoid the premature convergence of the particles on local minimizers, the concept of *neighborhood* was introduced [96]. A neighborhood of the i -th particle is defined as a set,

$$NB_{i,s} = \{j_1, j_2, \dots, j_s\} \subseteq I, \quad i \in NB_{i,s},$$

and consists of the indices of all the particles with which it can exchange information. Then, the neighborhood's best position

$$p_{g_i} = \arg \min_{j \in NB_{i,s}} \{f(p_j)\} \tag{5}$$

is used along with p_i to update the i -th particle's velocity at each iteration. The parameter s defines the *neighborhood size*. In the special case where $s = N$, the whole swarm constitutes the neighborhood. This case defines the so-called *global* PSO model (denoted as *gbest*), while strictly smaller neighborhoods correspond to the *local* PSO model (denoted as *lbest*). The schemes that are used for determining the particles that constitute each neighborhood are called *neighborhood topologies*, and they can have a crucial impact on PSO's performance. A popular scheme is the *ring* topology, where each particle assumes as neighbors of the particles with its

adjacent indices, assuming that indices recycle after N . Based on the definitions above, the iterative scheme of PSO is defined as follows [15]:

$$v_{ij}(t+1) = \chi \left[v_{ij}(t) + c_1 \mathcal{R}_1(p_{ij}(t) - x_{ij}(t)) + c_2 \mathcal{R}_2(p_{g_i,j}(t) - x_{ij}(t)) \right], \quad (6)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad (7)$$

where $i = 1, 2, \dots, N$; $j = 1, 2, \dots, n$; the parameter χ is the *constriction coefficient*; acceleration constants c_1 and c_2 are called the *cognitive* and *social* parameter, respectively; and \mathcal{R}_1 and \mathcal{R}_2 , are random variables uniformly distributed in the range $[0, 1]$. It shall be noted that a different value of \mathcal{R}_1 and \mathcal{R}_2 is sampled for each i and j in (6) at each iteration. Also, the best position of each particle is updated at each iteration, as follows:

$$p_i(t+1) = \begin{cases} x_i(t+1), & \text{if } f(x_i(t+1)) < f(p_i(t)), \\ p_i(t), & \text{otherwise,} \end{cases} \quad i \in I. \quad (8)$$

The PSO variant described above was introduced by M. Clerc and J. Kennedy in [15], and it has gained increasing popularity, especially in interdisciplinary applications. This can be attributed to its simplicity, its efficiency, and its theoretical background.

Based on the stability analysis [15], the parameter set $\chi = 0.729$, $c_1 = c_2 = 2.05$, was determined as a satisfactory setting that produces a balanced convergence speed of the algorithm. Nevertheless, alternative settings have been introduced in relevant works [98]. Pseudocode of PSO is reported in [Algorithm 1](#).

Unified particle swarm optimization (UPSO) was proposed as a scheme that harnesses the lbest and gbest PSO models, combining their exploration and exploitation properties [79, 82]. Let $\mathcal{G}_i(t+1)$ be the gbest velocity update of x_i , while $\mathcal{L}_{ij}(t+1)$ be the corresponding lbest velocity update. Then, from [Eq. \(6\)](#), it holds that

$$\mathcal{G}_{ij}(t+1) = \chi \left[v_{ij}(t) + c_1 \mathcal{R}_1(p_{ij}(t) - x_{ij}(t)) + c_2 \mathcal{R}_2(p_{g_i,j}(t) - x_{ij}(t)) \right], \quad (9)$$

$$\mathcal{L}_{ij}(t+1) = \chi \left[v_{ij}(t) + c_1 \mathcal{R}_1(p_{ij}(t) - x_{ij}(t)) + c_2 \mathcal{R}_2(p_{g_i,j}(t) - x_{ij}(t)) \right], \quad (10)$$

where g denotes the overall best particle. The aggregation of these search directions defines the main UPSO scheme [79]:

$$\mathcal{U}_{ij}(t+1) = u \mathcal{G}_{ij}(t+1) + (1-u) \mathcal{L}_{ij}(t+1), \quad (11)$$

$$x_{ij}(t+1) = x_{ij}(t) + \mathcal{U}_{ij}(t+1). \quad (12)$$

The parameter $u \in [0, 1]$ is called the *unification factor* and determines the trade off between the two directions. Obviously, the standard gbest PSO is obtained by setting $u = 1$ in (11), while $u = 0$ corresponds to the standard lbest PSO. All intermediate

values of $u \in (0, 1)$ define composite UPSO variants that combine the exploration and exploitation properties of the global and local PSO variant.

Besides the basic UPSO scheme, a stochastic parameter can also be incorporated in Eq. (11) to enhance UPSO's exploration capabilities [79]. Thus, depending on which variant UPSO is mostly based, Eq. (11) becomes

$$\mathcal{U}_{ij}(t+1) = \mathcal{R}_3 u \mathcal{G}_{ij}(t+1) + (1-u) \mathcal{L}_{ij}(t+1), \quad (13)$$

which is mostly based on lbest, or,

$$\mathcal{U}_{ij}(t+1) = u \mathcal{G}_{ij}(t+1) + \mathcal{R}_3 (1-u) \mathcal{L}_{ij}(t+1), \quad (14)$$

which is mostly based on gbest. The random variable $\mathcal{R}_3 \sim \mathcal{N}(\mu, \sigma^2)$ is normally distributed, resembling the mutation operator in EAs. Yet, it is biased towards directions that are consistent with the PSO dynamic, instead of the pure random mutation used in EAs. Following the assumptions of Matyas [71], a proof of convergence in probability was derived for the UPSO variants of Eqs. (13) and (14) [79].

Although PSO and UPSO have been primarily designed for real-valued optimization problems, there are various applications also in problems with integer, mixed-integer, and binary variables (e.g., see Eqs. [62, 69, 80, 81, 85, 86]). The most common and easy way to achieve this is by rounding the corresponding variables to their nearest integers when evaluating the particles. Of course, problem-dependent techniques may be additionally needed to enhance performance.

8 Ant Colony Optimization (ACO)

Ant colony optimization (ACO) is a probabilistic algorithm, primarily for solving combinatorial optimization problems. The general algorithmic concept of ACO was introduced by M. Dorigo in 1992 [30], aiming at detecting optimal paths in graphs by imitating the foraging behavior of real ants. The first approaches proved to be very promising, leading to further developments and enhancements [7, 31, 33].

Today, there is a variety of ant-based algorithms. Perhaps the most popular variants are Ant System (AS) [34], Max–Min Ant System (MMAS) [95], and Ant Colony System (ACS) [32]. Setting aside some differences, most ant-based approaches follow the same procedure flow, which can be summarized in the following actions:

```

Procedure ACO
WHILE (termination condition is false)
Construct_Solutions()
Optional_Further_Actions()
Update_Pheromones()
END WHILE

```

Algorithm 1: Pseudocode of the standard PSO algorithm

Input : Objective function, $f : X \subset \mathbb{R}^n \rightarrow \mathbb{R}$; swarm size: N ; parameters: χ, c_1, c_2 .
Output: Best detected solution: $x^*, f(x^*)$.

```

1 // Initialization
2 t ← 0.
3 for i ← 1 to N do
4     Initialize  $x_i(t)$  and  $v_i(t)$ , randomly.
5      $p_i(t) \leftarrow x_i(t)$ . // Initialize best position
6     Evaluate  $f(x_i(t))$ . // Evaluate particle
7 end
8 // Main Iteration
9 while (termination criterion not met) do
10    // Position and Velocity Update
11    for i ← 1 to N do
12        Determine  $p_{gi}(t)$  from the  $i$ -th particle's neighborhood.
13        for j ← 1 to n do
14             $v_{ij}(t+1) \leftarrow \chi [v_{ij}(t) + c_1 \mathcal{R}_1(p_{ij}(t) - x_{ij}(t)) + c_2 \mathcal{R}_2(p_{gi,j}(t) - x_{ij}(t))]$ .
15             $x_{ij}(t+1) \leftarrow x_{ij}(t) + v_{ij}(t+1)$ .
16        end
17    end
18    // Best Positions Update
19    for i ← 1 to N do
20        Evaluate  $f(x_i(t+1))$ . // Evaluate new position
21         $p_i(t+1) \leftarrow \begin{cases} x_i(t+1), & \text{if } f(x_i(t+1)) < f(p_i(t)), \\ p_i(t), & \text{otherwise.} \end{cases}$ 
22    end
23    t ← t + 1.
24 end

```

In general, the algorithm assumes a number, K , of search agents, called the *ants*. Each ant constructs a solution component by component. The construction procedure is based on the probabilistic selection of each component's value from a discrete and finite set. For this purpose, a table of *pheromones* is retained and continuously updated. The pheromones play the role of quality weights, used for determining the corresponding selection probability of each possible value of a solution component.

Putting it formally, let

$$x = (x_1, x_2, \dots, x_n)^\top \in D,$$

be a candidate solution of the problem, with each component $x_i, i = 1, 2, \dots, n$, taking values from a discrete and finite set D_i . The construction of a solution begins from an initially empty partial solution, $x^p = \emptyset$. At each step, the next component of x^p is assigned one of the possible values from its corresponding discrete set. Naturally, the values assigned in previous components may affect the feasible set of

candidate values for the current one, prohibiting the use of some of them to retain feasibility. Assume that x^p is already built up to the $(i - 1)$ -th component, and let $D_i = \{d_{i1}, d_{i2}, \dots, d_{iM}\}$ be the set of candidate values for the i -th component. Each possible value d_{ij} is associated with a pheromone level τ_{ij} . The selection of a specific value for the i -th component is based on the probabilistic selection among the different candidate values, using the selection probabilities:

$$P(d_{ij} | x^p) = \frac{\tau_{ij}^a \eta(d_{ij})^b}{\sum_{d_{il} \in D_i} \tau_{il}^a \eta(d_{il})^b}, \quad j = 1, 2, \dots, M, \quad (15)$$

where $\eta(\cdot)$ is a *heuristic function* that offers a measure of the “desirability” of each component value in the solution (e.g., in TSP problems, it can be associated with the traveled distance). The parameters a and b are fixed, and they offer flexibility in tuning the trade off between pheromone and heuristic information.

Upon constructing a complete solution, it is evaluated with the objective value, and the pheromones are updated so that component values that appear in better solutions increase their pheromone levels and, consequently, their selection probabilities for subsequent iterations of the algorithm. Also, all pheromones undergo a standard reduction in their values, a procedure also known as *evaporation*. Thus, the pheromones are updated as follows:

$$\tau_{ij} = \begin{cases} (1 - \rho) \tau_{ij} + \rho \Delta\tau, & \text{if } d_{ij} \text{ appears in a good candidate solution,} \\ (1 - \rho) \tau_{ij}, & \text{otherwise,} \end{cases} \quad (16)$$

where $\rho \in (0, 1]$ is the *evaporation rate* and $\Delta\tau$ is an increment that can be either fixed or proportional to the quality of the corresponding candidate solution.

Different ant-based approaches can differ from the basic scheme described above. For example, in AS with K ants, the pheromone update takes place after all ants have constructed their solutions and they all contribute to the update:

$$\tau_{ij} = (1 - \rho) \tau_{ij} + \sum_{k=1}^K \Delta\tau_{ij}^{(k)},$$

where $\Delta\tau_{ij}^{(k)}$ is the contributed pheromone from the k -th ant and it is related to its quality. On the other hand, ACS uses a different rule for selecting component values, where the value of the i -th component is determined as

$$d_{ig} = \arg \max_{d_{ij} \in D_i} \left\{ \tau_{ij}^a \eta(d_{ij})^b \right\}, \quad \text{if } q \leq q_0,$$

where $q \in [0, 1]$ is a uniformly distributed random number and $q_0 \in [0, 1]$ is a user-defined parameter; otherwise, the scheme of Eq. (15) is used. Also, only the best ant contributes to the pheromones, instead of the whole swarm. Finally, MMAS employs pheromone bounds $[\tau_{\min}, \tau_{\max}]$. Each pheromone is initialized to

its maximum value and updated only from the best ant, either of the current iteration or overall.

There is a rich literature of applications of ant-based approaches in combinatorial optimization problems [7, 31, 33]. Recently, such approaches were used also for the solution of autocorrelation problems [69].

9 Combinatorial Optimization

Combinatorial optimization methods seem like a natural candidate of methods that should be employed to tackle extremely hard optimization problems such as the search for binary and ternary sequences that satisfy autocorrelation constraints. This is because combinatorial optimization methods routinely deal with problems featuring thousands of discrete variables. The missing link that allows one to formalize autocorrelation problems in a manner suitable for combinatorial optimization methods was provided in [66] and [61]. This formalism made it in fact possible to solve in [66] the entire Bömer-Antweiler diagram, which was proposed 20 years before, in [6]. This formalism is based on the fact that autocorrelation can be expressed in terms of certain symmetric matrices, which correspond to the matrices associated to autocorrelation viewed as *quadratic form*. We shall illustrate the formalism for D-optimal matrices, following the exposition in [61].

Let n be an odd positive integer and set $m = \frac{n-1}{2}$. D-optimal matrices correspond to two binary sequences of length n each, with constant periodic autocorrelation 2; see [61]. D-optimal matrices are $2n \times 2n \{-1, +1\}$ -matrices that attain Ehlich's determinant bound. See [29] for a state-of-the-art survey of existence questions for D-optimal matrices with $n < 200$. We reproduce the following definition and lemma from [66].

Definition 8 Let $a = [a_1, a_2, \dots, a_n]^T$ be a column $n \times 1$ vector, where $a_1, a_2, \dots, a_n \in \{-1, +1\}$ and consider the elements of the periodic autocorrelation function vector $P_A(1), \dots, P_A(m)$. Define the following m symmetric matrices (which are independent of the sequence a), for $i = 1, \dots, m$

$$M_i = (m_{jk}), \text{ s.t. } \begin{cases} m_{jk} = m_{kj} = \frac{1}{2}, & \text{when } a_j a_k \in P_A(i), \ j, k \in \{1, \dots, n\} \\ 0, & \text{otherwise} \end{cases}$$

Lemma The matrices M_i can be used to write equations involving autocorrelation in a matrix form:

- For n odd:

$$a^T M_i a = P_A(i), \quad i = 1, \dots, m.$$

- For n even:

$$a^T M_i a = P_A(i), \quad i = 1, \dots, m-1 \text{ and } a^T M_m a = \frac{1}{2} P_A(m).$$

Now one can formulate the D-optimal matrices problem as a binary feasibility problem.

Binary Feasibility version of the D-optimal matrices problem Find two sequences $a = [a_1, \dots, a_n]$, $b = [b_1, \dots, b_n]$, (viewed as $n \times 1$ column vectors) such that

$$a^T M_i a + b^T M_i b = 2, \quad i = 1, \dots, m,$$

where $a_i, b_i \in \{-1, +1\}$, $i = 1, \dots, n$.

It is now clear that one can use combinatorial optimization algorithms to search for D-optimal matrices, as well as several other combinatorial matrices and sequences referenced in [Table 1](#). [Definition 8](#) can easily be adapted for aperiodic autocorrelation as well.

Note that the D-optimal matrices problem also features certain Diophantine constraints, as proved in [\[61\]](#) and subsequently generalized in [\[29\]](#). It is not clear what is the right way to incorporate this kind of Diophantine constraints into the combinatorial optimization formalism described here, and this is certainly an issue that deserves to be investigated further.

The papers [\[73–76\]](#) demonstrate how to use various optimization techniques to find new designs.

10 Applications

There are several applications of combinatorial matrices and sequences described in [Table 1](#), in such areas as telecommunications, coding theory, and cryptography. The reader can consult [\[45\]](#) for applications in signal design for communications, radar, and cryptography applications. The book [\[53\]](#) describes applications in signal processing, coding theory, and cryptography. The book [\[104\]](#) describes applications in communications, signal processing, and image processing. The paper [\[99\]](#) discusses how partial Hadamard matrices (see [\[20\]](#)) are used in the area of compressed sensing. The paper [\[93\]](#) describes in detail several applications of Hadamard matrices on code division multiple access (CDMA) communication systems as well as in telecommunications. The 55-page paper [\[49\]](#) describes applications of Hadamard matrices in binary codes, information processing, maximum determinant problems, spectrometry, pattern recognition, and other areas. The papers [\[91\]](#) and [\[92\]](#) and the thesis [\[9\]](#) discuss cryptography applications of Hadamard matrices. Chapter 10 of the book [\[51\]](#) is devoted to several combinatorial problems that can be tackled with stochastic search methods.

11 Conclusion

Combinatorial matrices and the associated binary and ternary sequences provide a wide array of extremely challenging optimization problems that can be tackled with a variety of metaheuristic and combinatorial methods. We tried to summarize

some of these methods in the present chapter. Although there are no polynomial complexity algorithms to solve the problem of searching for such matrices and sequences, the combination of new insights, new algorithms, and new implementations will continue to produce new results. The second edition of the Handbook of Combinatorial Designs [16], edited by Charles J. Colbourn and Jeffrey H. Dinitz, is a very valuable resource regarding the state of the art on open cases for several types of such matrices and sequences. The on-line updated webpage maintained by the authors is also quite useful, as many open cases have been resolved since the publication of the handbook in 2007. We firmly believe that cross-fertilization of disciplines is a very important process, from which all involved disciplines benefit eventually. It seems reasonable to predict that the area of combinatorial matrices and their associated sequences will continue to experience strong interactions with other research areas, for many years to come.

Cross-References

- ▶ [Algorithms for the Satisfiability Problem](#)
 - ▶ [Binary Unconstrained Quadratic Optimization Problem](#)
-

Recommended Reading

1. V. Álvarez, J.A. Armario, M.D. Frau, P. Real, A genetic algorithm for cocyclic Hadamard matrices, in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*. Lecture Notes in Computer Science, vol. 3857 (Springer, Berlin, 2006), pp. 144–153
2. D. Ashlock, Finding designs with genetic algorithms, in *Computational and Constructive Design Theory*. Mathematics and Its Applications, vol. 368 (Kluwer, Dordrecht, 1996), pp. 49–65
3. T. Bäck, D. Fogel, Z. Michalewicz, *Handbook of Evolutionary Computation* (IOP Publishing/Oxford University Press, New York, 1997)
4. A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part i: background and development. *Nat. Comput.* **6**(4), 467–484 (2007)
5. A. Banks, J. Vincent, C. Anyakoha, A review of particle swarm optimization. Part ii: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Nat. Comput.* **7**(1), 109–124 (2008)
6. L. Bömer, M. Antweiler, Periodic complementary binary sequences. *IEEE Trans. Inf. Theory* **36**(6), 1487–1494 (1990)
7. E. Bonabeau, M. Dorigo, G. Théraulaz, *Swarm Intelligence: From Natural to Artificial Systems* (Oxford University Press, New York, 1999)
8. P.B. Borwein, R.A. Ferguson, A complete description of Golay pairs for lengths up to 100. *Math. Comput.* **73**(246), 967–985 (2004)
9. A. Braeken, *Cryptographic Properties of Functions and S-Boxes*. PhD thesis, Katholieke Universiteit Leuven, Belgium, 2006
10. R.A. Brualdi, *Combinatorial Matrix Classes*. Encyclopedia of Mathematics and Its Applications, vol. 108 (Cambridge University Press, Cambridge, 2006)
11. R.A. Brualdi, H.J. Ryser, *Combinatorial Matrix Theory*. Encyclopedia of Mathematics and Its Applications, vol. 39 (Cambridge University Press, Cambridge, 1991)

12. Y.W. Cheng, D.J. Street, W.H. Wilson, Two-stage generalized simulated annealing for the construction of change-over designs, in *Designs, 2002. Mathematics and Its Applications*, vol. 563 (Kluwer, Boston, 2003), pp. 69–79
13. M. Chiarandini, I.S. Kotsireas, C. Koukouvinos, L. Paquete, Heuristic algorithms for Hadamard matrices with two circulant cores. *Theor. Comput. Sci.* **407**(1–3), 274–277 (2008)
14. M. Clerc, *Particle Swarm Optimization* (ISTE Ltd, London/Newport Beach, 2006)
15. M. Clerc, J. Kennedy, The particle swarm—explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **6**(1), 58–73 (2002)
16. C.J. Colbourn, J.H. Dinitz (eds.), *Handbook of Combinatorial Designs*. Discrete Mathematics and Its Applications (Boca Raton), 2nd edn. (Chapman & Hall/CRC, Boca Raton, 2007)
17. J. Cousineau, I.S. Kotsireas, C. Koukouvinos, Genetic algorithms for orthogonal designs. *Australas. J. Comb.* **35**, 263–272 (2006)
18. R. Craigen, Boolean and ternary complementary pairs. *J. Comb. Theory Ser. A* **104**(1), 1–16 (2003)
19. W. de Launey, D. Flannery, *Algebraic Design Theory*. Mathematical Surveys and Monographs, vol. 175 (American Mathematical Society, Providence, 2011)
20. W. de Launey, D.A. Levin, A Fourier-analytic approach to counting partial Hadamard matrices. *Cryptogr. Commun.* **2**(2), 307–334 (2010)
21. D.Ž. Doković, Two Hadamard matrices of order 956 of Goethals-Seidel type. *Combinatorica* **14**(3), 375–377 (1994)
22. D.Ž. Doković, Equivalence classes and representatives of Golay sequences. *Discret. Math.* **189**(1–3), 79–93 (1998)
23. D.Ž. Doković, Hadamard matrices of order 764 exist. *Combinatorica* **28**(4), 487–489 (2008)
24. D.Ž. Doković, Skew-Hadamard matrices of orders 188 and 388 exist. *Int. Math. Forum* **3**(21–24), 1063–1068 (2008)
25. D.Ž. Doković, Skew-Hadamard matrices of orders 436, 580, and 988 exist. *J. Comb. Des.* **16**(6), 493–498 (2008)
26. D.Ž. Doković, On the base sequence conjecture. *Discret. Math.* **310**(13–14), 1956–1964 (2010)
27. D.Ž. Doković, Classification of normal sequences. *Int. J. Comb. Art. ID* 937941, 15 (2011)
28. D.Ž. Doković, Cyclic $(v; r, s; \lambda)$ difference families with two base blocks and $v \leq 50$. *Ann. Comb.* **15**(2), 233–254 (2011)
29. D.Ž. Doković, I.S. Kotsireas, New results on D-optimal matrices. *J. Comb. Des.* **20**(6), 278–289 (2012)
30. M. Dorigo, *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 2002
31. M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in *New Ideas in Optimization*, ed. by D. Corne, M. Dorigo, F. Glover (McGraw-Hill, London, 1999), pp. 11–32
32. M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1**(1), 53–66 (1997)
33. M. Dorigo, T. Stützle, *Ant Colony Optimization* (MIT Press, Cambridge, 2004)
34. M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B* **26**(1), 29–41 (1996)
35. R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in *Proceedings Sixth Symposium on Micro Machine and Human Science* (IEEE Service Center, Piscataway, 1995), pp. 39–43
36. S. Eliahou, M. Kervaire, B. Saffari, A new restriction on the lengths of Golay complementary sequences. *J. Comb. Theory Ser. A* **55**(1), 49–59 (1990)
37. A.P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence* (Wiley, Hoboken, 2006)
38. T. Feng, Q. Xiang, Cyclotomic constructions of skew Hadamard difference sets. *J. Comb. Theory Ser. A* **119**(1), 245–256 (2012)

39. T. Feng, Q. Xiang, Strongly regular graphs from unions of cyclotomic classes. *J. Comb. Theory Ser. B* **102**, 982–995 (2012)
40. F. Fiedler, J. Jedwab, M.G. Parker, A framework for the construction of Golay sequences. *IEEE Trans. Inf. Theory* **54**(7), 3114–3129 (2008)
41. F. Fiedler, J. Jedwab, M.G. Parker, A multi-dimensional approach to the construction and enumeration of Golay complementary sequences. *J. Comb. Theory Ser. A* **115**(5), 753–776 (2008)
42. R.J. Fletcher, M. Gysin, J. Seberry, Application of the discrete Fourier transform to the search for generalised Legendre pairs and Hadamard matrices. *Australas. J. Comb.* **23**, 75–86 (2001)
43. A. Gavish, A. Lempel, On ternary complementary sequences. *IEEE Trans. Inf. Theory* **40**(2), 522–526 (1994)
44. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, 1989)
45. S.W. Golomb, G. Gong, *Signal Design for Good Correlation* (Cambridge University Press, Cambridge, 2005). For wireless communication, cryptography, and radar
46. M.M. Gysin, *Algorithms for Searching for Normal and Near-Yang Sequences*. University of Wollongong, 1993. Thesis (MSc)–University of Wollongong
47. M.M. Gysin, *Combinatorial Designs, Sequences and Cryptography*. PhD thesis, University of Wollongong, Wollongong, NSW, Australia, 1997
48. M. Hall Jr., *Combinatorial Theory* (Blaisdell Publishing Co./Ginn and Co., Waltham/Toronto/London, 1967)
49. A. Hedayat, W.D. Wallis, Hadamard matrices and their applications. *Ann. Stat.* **6**(6), 1184–1238 (1978)
50. A. Heredia-Langner, W.M. Carlyle, D.C. Montgomery, C.M. Borror, G.C. Runger, Genetic algorithms for the construction of d-optimal designs. *J. Qual. Technol.* **35**(1), 28–46 (2003)
51. H.H. Hoos, T. Stützle, *Stochastic Local Search: Foundations & Applications* (Elsevier/Morgan Kaufmann, Oxford/San Francisco, 2004)
52. K.J. Horadam, *Hadamard Matrices and Their Applications* (Princeton University Press, Princeton, 2007)
53. K.J. Horadam, Hadamard matrices and their applications: progress 2007–2010. *Cryptogr. Commun.* **2**(2), 129–154 (2010)
54. Y.J. Ionin, M.S. Shrikhande, *Combinatorics of Symmetric Designs*. New Mathematical Monographs, vol. 5 (Cambridge University Press, Cambridge, 2006)
55. J.A. John, D. Whitaker, Construction of resolvable row-column designs using simulated annealing. *Aust. J. Stat.* **35**(2), 237–245 (1993)
56. J. Kennedy, R.C. Eberhart, *Swarm Intelligence* (Morgan Kaufmann Publishers, San Francisco, 2001)
57. H. Kharaghani, B. Tayfeh-Rezaie, A Hadamard matrix of order 428. *J. Comb. Des.* **13**(6), 435–440 (2005)
58. S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing. *Science* **220**(4598), 671–680 (1983)
59. I.S. Kotsireas, C. Koukouvinos, Genetic algorithms for the construction of Hadamard matrices with two circulant cores. *J. Discret. Math. Sci. Cryptogr.* **8**(2), 241–250 (2005)
60. I.S. Kotsireas, C. Koukouvinos, Hadamard ideals and Hadamard matrices from two circulant submatrices. *J. Comb. Math. Comb. Comput.* **61**, 97–110 (2007)
61. I.S. Kotsireas, P.M. Pardalos, D-optimal matrices via quadratic integer optimization. *J. Heuristics* (2012) (to appear)
62. I.S. Kotsireas, C. Koukouvinos, K.E. Parsopoulos, M.N. Vrahatis, Unified particle swarm optimization for Hadamard matrices of Williamson type, in *Proceedings of the 1st International Conference on Mathematical Aspects of Computer and Information Sciences (MACIS 2006)*, Beijing, China, 2006, pp. 113–121
63. I.S. Kotsireas, C. Koukouvinos, J. Seberry, Hadamard ideals and Hadamard matrices with two circulant cores. *Eur. J. Combin.* **27**(5), 658–668 (2006)

64. I.S. Kotsireas, C. Koukouvinos, J. Seberry, Weighing matrices and string sorting. *Ann. Comb.* **13**(3), 305–313 (2009)
65. I.S. Kotsireas, C. Koukouvinos, P.M. Pardalos, An efficient string sorting algorithm for weighing matrices of small weight. *Optim. Lett.* **4**(1), 29–36 (2010)
66. I.S. Kotsireas, C. Koukouvinos, P.M. Pardalos, O.V. Shylo, Periodic complementary binary sequences and combinatorial optimization algorithms. *J. Comb. Optim.* **20**(1), 63–75 (2010)
67. I.S. Kotsireas, C. Koukouvinos, P.M. Pardalos, A modified power spectral density test applied to weighing matrices with small weight. *J. Comb. Optim.* **22**(4), 873–881 (2011)
68. I.S. Kotsireas, C. Koukouvinos, P.M. Pardalos, D.E. Simos, Competent genetic algorithms for weighing matrices. *J. Comb. Optim.* **24**, 508–525 (2012)
69. I.S. Kotsireas, K.E. Parsopoulos, G.S. Piperagkas, M.N. Vrahatis, Ant-based approaches for solving autocorrelation problems, in *Eighth International Conference on Swarm Intelligence (ANTS 2012)*, Lecture Notes in Computer Science (LNCS), Brussels, Belgium, Vol. 7461, pp. 220–227 Springer (2012)
70. P.C. Li, Combining genetic algorithms and simulated annealing for constructing lotto designs. *J. Comb. Math. Comb. Comput.* **45**, 109–121 (2003)
71. J. Matyas, Random optimization. *Autom. Remote Control* **26**, 244–251 (1965)
72. R.K. Meyer, C.J. Nachtsheim, Constructing exact D -optimal experimental designs by simulated annealing. *Am. J. Math. Manag. Sci.* **8**(3–4), 329–359 (1988)
73. L.B. Morales, Constructing difference families through an optimization approach: six new BIBDs. *J. Comb. Des.* **8**(4), 261–273 (2000)
74. L.B. Morales, Constructing some PBIBD(2)s by tabu search algorithm. *J. Comb. Math. Comb. Comput.* **43**, 65–82 (2002)
75. L.B. Morales, Constructing cyclic PBIBD(2)s through an optimization approach: thirty-two new cyclic designs. *J. Comb. Des.* **13**(5), 377–387 (2005)
76. L.B. Morales, Constructing 1-rotational NRDFs through an optimization approach: new (46,9,8), (51,10,9) and (55,9,8)-NRBDs. *J. Stat. Plann. Inference* **139**(1), 62–68 (2009)
77. K.J. Nurmela, P.R.J. Östergård, Upper bounds for covering designs by simulated annealing, in *Proceedings of the Twenty-fourth Southeastern International Conference on Combinatorics, Graph Theory, and Computing*, Boca Raton, FL, 1993, vol. 96, pp. 93–111
78. K.E. Parsopoulos, M.N. Vrahatis, Recent approaches to global optimization problems through particle swarm optimization. *Nat. Comput.* **1**(2–3), 235–306 (2002)
79. K.E. Parsopoulos, M.N. Vrahatis, UPSO: a unified particle swarm optimization scheme, in *Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004)*. Lecture Series on Computer and Computational Sciences, vol. 1 (VSP International Science Publishers, Zeist, 2004), pp. 868–873
80. K.E. Parsopoulos, M.N. Vrahatis, Unified particle swarm optimization for tackling operations research problems, in *Proceedings of the IEEE 2005 Swarm Intelligence Symposium*, Pasadena, CA, USA, 2005, pp. 53–59
81. K.E. Parsopoulos, M.N. Vrahatis, Studying the performance of unified particle swarm optimization on the single machine total weighted tardiness problem, in ed. by A. Sattar, B.H. Kang Lecture Notes in Artificial Intelligence (LNAI), vol. 4304 (Springer, 2006), pp. 760–769
82. K.E. Parsopoulos, M.N. Vrahatis, Parameter selection and adaptation in unified particle swarm optimization. *Math. Comput. Model.* **46**(1–2), 198–213 (2007)
83. K.E. Parsopoulos, M.N. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications* (Information Science Publishing (IGI Global), Hershey, 2010)
84. C.B. Pheatt, *Construction of D-Optimal Experimental Designs Using Genetic Algorithms*. ProQuest LLC, Ann Arbor, MI, 1995. Thesis (Ph.D.)–Illinois Institute of Technology
85. G.S. Piperagkas, C. Voglis, V.A. Tatsis, K.E. Parsopoulos, K. Skouri, Applying PSO and DE on multi-item inventory problem with supplier selection, in *The 9th Metaheuristics International Conference (MIC 2011)*, Udine, Italy, 2011, pp. 359–368
86. G.S. Piperagkas, I. Konstantaras, K. Skouri, K.E. Parsopoulos, Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics. *Comput. Oper. Res.* **39**(7), 1555–1565 (2012)

87. C.R. Reeves, J.E. Rowe, *Genetic Algorithms: Principles and Perspectives*. Operations Research/Computer Science Interfaces Series, vol. 20 (Kluwer, Boston, 2003). A guide to GA theory
88. H.J. Ryser, *Combinatorial Mathematics*. The Carus Mathematical Monographs, vol. 14 (The Mathematical Association of America, Buffalo, 1963)
89. J. Sawada, Generating bracelets in constant amortized time. *SIAM J. Comput.* **31**(1), 259–268 (2001)
90. J. Sawada, A fast algorithm to generate necklaces with fixed content. *Theor. Comput. Sci.* **301**(1–3), 477–489 (2003)
91. J. Seberry, X. Zhang, Y. Zheng, Cryptographic Boolean functions via group Hadamard matrices. *Australas. J. Comb.* **10**, 131–145 (1994)
92. J. Seberry, X. Zhang, Y. Zheng, Pitfalls in designing substitution boxes (extended abstract), in *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '94* (Springer, London, 1994), pp. 383–396
93. J. Seberry, B.J. Wysocki, T.A. Wysocki, On some applications of Hadamard matrices. *Metrika* **62**(2–3), 221–239 (2005)
94. D.R. Stinson, *Combinatorial Designs, Constructions and Analysis* (Springer, New York, 2004)
95. T. Stützle, H.H. Hoos, MAX MIN ant system. *Future Gener. Comput. Syst.* **16**, 889–914 (2000)
96. P.N. Suganthan, Particle swarm optimizer with neighborhood operator, in *Proceedings of the IEEE Congress on Evolutionary Computation*, Washington, D.C., USA, 1999, pp. 1958–1961
97. M.N. Syed, I.S. Kotsireas, P.M. Pardalos, D-optimal designs: a mathematical programming approach using cyclotomic cosets. *Informatica* **22**(4), 577–587 (2011)
98. I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection. *Inf. Process. Lett.* **85**, 317–325 (2003)
99. Y. Tsaig, D.L. Donoho, Extensions of compressed sensing. *Signal Process.* **86**(3), 549–571 (2006)
100. C. Voglis, K.E. Parsopoulos, D.G. Papageorgiou, I.E. Lagaris, M.N. Vrahatis, MEMPSODE: a global optimization software based on hybridization of population-based algorithms and local searches. *Comput. Phys. Commun.* **183**(5), 1139–1154 (2012)
101. J. Wallis, On supplementary difference sets. *Aequ. Math.* **8**, 242–257 (1972)
102. J. Wallis, A note on supplementary difference sets. *Aequ. Math.* **10**, 46–49 (1974)
103. M. Yamada, Supplementary difference sets and Jacobi sums. *Discret. Math.* **103**(1), 75–90 (1992)
104. R.K. Yarlagadda, J.E. Hershey, *Hadamard Matrix Analysis and Synthesis*. The Kluwer International Series in Engineering and Computer Science, vol. 383 (Kluwer, Boston, 1997). With applications to communications and signal/image processing

Algorithms for the Satisfiability Problem

John Franco and Sean Weaver

Contents

1	Introduction	312
2	Logic	313
3	Representations and Structures	319
3.1	(0 ± 1) Matrix	319
3.2	Binary Decision Diagrams	320
3.3	Implication Graph	321
3.4	Propositional Connection Graph	322
3.5	Variable-Clause Matching Graph	322
3.6	Formula Digraph	322
3.7	Satisfiability Index	324
3.8	And/Inverter Graphs	324
4	Applications	324
4.1	Consistency Analysis in Scenario Projects	325
4.2	Testing of VLSI Circuits	328
4.3	Diagnosis of Circuit Faults	330
4.4	Functional Verification of Hardware Design	332
4.5	Bounded Model Checking	336
4.6	Combinational Equivalence Checking	338
4.7	Transformations to Satisfiability	340
4.8	Boolean Data Mining	343
5	General Algorithms	345
5.1	Efficient Transformation to CNF Formulas	345
5.2	Resolution	351
5.3	Extended Resolution	355
5.4	Davis-Putnam Resolution	355
5.5	Davis-Putnam Loveland Logemann Resolution	356
5.6	Conflict-Driven Clause Learning	360
5.7	Satisfiability Modulo Theories	364
5.8	Stochastic Local Search	371

J. Franco (✉)

School of Electronic and Computing Systems, University of Cincinnati, Cincinnati, OH, USA

S. Weaver

U.S. Department of Defense, Ft. George G. Meade, MD, USA

5.9	Binary Decision Diagrams.....	375
5.10	Decompositions.....	388
5.11	Branch and Bound.....	395
5.12	Algebraic Methods.....	397
6	Algorithms for Easy Classes of CNF Formulas.....	404
6.1	2-SAT.....	405
6.2	Horn Formulas.....	407
6.3	Renamable Horn Formulas.....	408
6.4	Linear Programming Relaxations.....	409
6.5	q-Horn Formulas.....	412
6.6	Matched Formulas.....	414
6.7	Generalized Matched Formulas.....	415
6.8	Nested and Extended Nested Satisfiability.....	415
6.9	Linear Autark Formulas.....	421
6.10	Minimally Unsatisfiable Formulas.....	425
6.11	Bounded Resolvent Length Resolution.....	432
6.12	Comparison of Classes.....	434
6.13	Probabilistic Comparison of Incomparable Classes.....	436
7	The k -SAT Problem.....	439
8	Other Topics.....	443
9	Conclusion.....	443
Glossary.....		443
Cross-References.....		447
Recommended Reading.....		447

Abstract

This chapter discusses advances in SAT algorithm design, including the use of SAT algorithms as theory drivers, classic implementations of SAT solvers, and some theoretical aspects of SAT. Some applications to which SAT solvers have been successfully applied are also presented. The intention is to assist someone interested in applying SAT technology in solving some stubborn class of combinatorial problems.

1 Introduction

The satisfiability problem (SAT) has gained considerable attention over the past decade for two reasons. First, the performance of competitive SAT solvers has improved enormously due to the implementation of new algorithmic concepts. Second, so many real-world problems that are not naturally expressed as instances of SAT can be transformed to SAT instances and solved relatively efficiently using one of the ever-improving SAT solvers or solvers based on SAT. This chapter attempts to clarify these successes by presenting the important advances in SAT algorithm design as well as the classic implementations of SAT solvers. Some applications to which SAT solvers have been successfully applied are also presented. Finally, for the sake of developing some intuition on the subject, some classic theoretical results are presented.

The next section of the chapter presents some logic background and notation that will be necessary to understand and express the algorithms presented. The third

section presents several representations for SAT instances in preparation for discussing the wide variety of SAT solver implementations that have been tried. The fourth section presents some applications of SAT. The next two sections present algorithms for SAT, including search and heuristic algorithms plus systems that use SAT to manage the logic of complex computations. The seventh section presents some theoretical results regarding k -SAT instances and relates these with the performance of practical SAT solvers. The last section summarizes the results of the chapter.

2 Logic

An instance of satisfiability is a propositional logic expression in conjunctive normal form (CNF), the meaning of which is described in the next few paragraphs. The most elementary object comprising a CNF expression is the Boolean variable, shortened to *variable* in the context of this chapter. A variable takes one of two values from the set $\{0, 1\}$. A variable v may be negated in which case it is denoted $\neg v$. The value of $\neg v$ is opposite that of v . A *literal* is either a variable or a negated variable. The term *positive literal* is used to refer to a variable and the term *negative literal* is used to refer to a negated variable. The *polarity* of a literal is positive or negative accordingly.

The building blocks of propositional expressions are binary Boolean operators. A *binary Boolean operator* is a function $\mathcal{O}_b : \{0, 1\} \times \{0, 1\} \mapsto \{0, 1\}$. Often, such functions are presented in tabular form, called *truth tables*, as illustrated in Fig. 10. There are 16 possible binary operators and the most common (and useful) are \vee (or), \wedge (and), \rightarrow (implies), \leftrightarrow (equivalent), and \oplus (xor, alternatively exclusive-or). Mappings defining the common operators are shown in Fig. 1. The only interesting *unary Boolean operator*, denoted \neg (negation), is a mapping from 0 to 1 and 1 to 0. If $\neg l$ is a literal, then $\neg\neg l$ is the same as l .

A *formula* is a propositional expression consisting of literals, parentheses, and operators which has some semantic content and whose syntax is described recursively as follows:

1. Any single variable is a formula.
2. If ψ is a formula, then so is $\neg\psi$.
3. If ψ_1 and ψ_2 are both formulas and \mathcal{O} is a Boolean binary operator, then $(\psi_1 \mathcal{O} \psi_2)$ is a formula, ψ_1 is called the *left operand* of \mathcal{O} , and ψ_2 is called the *right operand* of \mathcal{O} .

Formulas can be simplified by removing some or all parentheses. Parentheses around nestings involving the same associative operators such as \vee , \wedge , and \leftrightarrow may be removed. For example, $(\psi_1 \vee (\psi_2 \vee \psi_3))$, $((\psi_1 \vee \psi_2) \vee \psi_3)$, and $(\psi_1 \vee \psi_2 \vee \psi_3)$ are considered to be the same formula. In the case of non-associative operators such as \rightarrow , parentheses may be removed, but right associativity is then assumed. The following is an example of a simplified formula:

$$(\neg v_0 \vee v_1 \vee \neg v_7) \wedge (\neg v_2 \vee v_3) \wedge (v_0 \vee \neg v_6 \vee \neg v_7) \wedge (\neg v_4 \vee v_5 \vee v_9),$$

Fig. 1 A (0 ± 1) matrix representation and associated CNF formula where clauses are labeled c_0, c_1, \dots, c_7 , from left to right, top to bottom

$$\begin{array}{cccccccccc} v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 \\ \hline c_0 & \left(\begin{array}{cccccccccc} -1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \\ c_1 & \\ c_2 & \\ c_3 & \\ c_4 & \\ c_5 & \\ c_6 & \\ c_7 & \end{array}$$

$$((\neg v_0 \vee v_1 \vee \neg v_7) \wedge (\neg v_1 \vee v_2) \wedge (\neg v_2 \vee v_3) \wedge (\neg v_0 \vee \neg v_3 \vee v_8) \wedge (v_0 \vee \neg v_6 \vee \neg v_7) \wedge (\neg v_5 \vee v_6) \wedge (\neg v_4 \vee v_5 \vee v_9) \wedge (v_0 \vee v_4))$$

Table 1 The most common binary Boolean operators and their mappings

Operator	Symbol	Mapping
Or	\vee	$\{00 \mapsto 0; 10, 01, 11 \mapsto 1\}$
And	\wedge	$\{00, 01, 10 \mapsto 0; 11 \mapsto 1\}$
Implies	\rightarrow	$\{10 \mapsto 0; 00, 01, 11 \mapsto 1\}$
Equivalent	\leftrightarrow	$\{01, 10 \mapsto 0; 00, 11 \mapsto 1\}$
XOR	\oplus	$\{00, 11 \mapsto 0; 01, 10 \mapsto 1\}$

A useful parameter associated with a formula is depth. The *depth* of a formula is determined as follows:

1. The depth of a formula consisting of a single variable is 0.
2. The depth of a formula $\neg \psi$ is the depth of ψ plus 1.
3. The depth of a formula $(\psi_1 \mathcal{O} \psi_2)$ is the maximum of the depth of ψ_1 and the depth of ψ_2 plus 1.

A truth assignment or *assignment* is a set of variables all of whom have value 1. If M is an assignment and V is a set of variables, then, if $v \in V$ and $v \notin M$, v has value 0. Any assignment of values to the variables of a formula induces a value on the formula. A formula is evaluated from innermost \neg or parentheses out using mappings associated with the Boolean operators that are found in [Table 1](#).

Many algorithms that will be considered later iteratively build assignments, and it will be necessary to distinguish variables that have been assigned a value from those that have not been. In such cases, a variable will be allowed to hold a third value, denoted \perp , which means the variable is unassigned. This requires that the evaluation of operations be augmented to account for \perp as shown in [Table 2](#). If M is an assignment of values to a set of variables where at least one variable has value \perp , then M is said to be a *partial assignment*.

Formulas that are dealt with in this chapter often have many components of the same type which are called clauses. Two common special types of clauses are disjunctive and conjunctive clauses. A *disjunctive clause* is a formula consisting only of literals and the operator \vee . If all the literals of a clause are negative (positive), then the clause is called a *negative clause* (respectively, *positive clause*).

Table 2 Boolean operator mappings with \perp . Symbol? means 1 or 0

Operator	Symbol	Mapping
Or	\vee	$\{00 \mapsto 0; 10, 01, 11, 1\perp, \perp 1 \mapsto 1; 0\perp, \perp 0, \perp\perp \mapsto \perp\}$
And	\wedge	$\{00, 01, 10 \mapsto 0; 11 \mapsto 1; ?\perp, \perp?, \perp\perp \mapsto \perp\}$
Implies	\rightarrow	$\{10 \mapsto 0; 00, 01, 11, 0\perp \mapsto 1; 1\perp, \perp? \mapsto \perp\}$
Equivalent	\leftrightarrow	$\{01, 10 \mapsto 0; 00, 11 \mapsto 1; ?\perp, \perp?, \perp\perp \mapsto \perp\}$
XOR	\oplus	$\{00, 11 \mapsto 0; 01, 10 \mapsto 1; ?\perp, \perp?, \perp\perp \mapsto \perp\}$
Negate	\neg	$\{0 \mapsto 1; 1 \mapsto 0, \perp \mapsto \perp\}$

In this chapter disjunctive clauses will usually be represented as sets of literals. When it is understood that an object is a disjunctive clause, it will be referred to simply as a *clause*. The following two lines show the same formula of four clauses expressed, above, in conventional logic notation and, below, as a set of sets of literals:

$$(\neg v_0 \vee v_1 \vee \neg v_7) \wedge (\neg v_2 \vee v_3) \wedge (v_0 \vee \neg v_6 \vee \neg v_7) \wedge (\neg v_4 \vee v_5 \vee v_9)$$

$$\{\{\neg v_0, v_1, \neg v_7\}, \{\neg v_2, v_3\}, \{v_0, \neg v_6, \neg v_7\}, \{\neg v_4, v_5, v_9\}\}.$$

A *conjunctive clause* is a formula consisting only of literals and the operator \wedge . A conjunctive clause will also usually be represented as a set of literals and called a clause when it is unambiguous to do so. The number of literals in any clause is referred to as the *width* of the clause.

Often, formulas are expressed in some normal form. Four of the most frequently arising forms are defined as follows:

A *CNF formula* is a formula consisting of a conjunction of two or more disjunctive clauses.

Given CNF formula ψ and L_ψ , the set of all literals in ψ , a literal l is said to be a *pure literal* in ψ if $l \in L_\psi$ but $\neg l \notin L_\psi$. A clause $c \in \psi$ is said to be a *unit clause* if c has exactly one literal.

A k -*CNF formula*, k fixed, is a CNF formula restricted so that the width of each clause is exactly k .

A *Horn formula* is a CNF formula with the restriction that all clauses contain at most one positive literal. Observe that a clause $(\neg a \vee \neg b \vee \neg c \vee g)$ is functionally the same as $(a \wedge b \wedge c \rightarrow g)$, so Horn formulas are closely related to logic programming. In fact, logic programming was originally the study of Horn formulas.

A *DNF formula* is a formula consisting of a disjunction of two or more conjunctive clauses.

When discussing a CNF or DNF formula ψ , V_ψ is used to denote its variable set and C_ψ is used to denote its clause set. The subscripts are dropped when the context is clear.

As mentioned earlier, evaluation of a formula is from innermost parentheses out using the truth tables for each operator encountered and a given truth assignment. If the formula evaluates to 1, then the assignment is called a *satisfying assignment*, *model*, or a *solution*.

There are 2^n ways to assign values to n Boolean variables. Any subset of those assignments is a Boolean function on n variables. Thus, the number of such functions is 2^{2^n} . Any Boolean function f on n variables can be expressed as a CNF formula ψ where the set of assignments for which f has value 1 is identical to the set of assignments satisfying ψ [129]. However, k -CNF formulas express only a proper subset of Boolean functions: for example, since a width k clause eliminates the fraction 2^{-k} of potential models, any Boolean function comprising more than $2^n(1 - 2^{-k})$ assignments cannot be represented by a k -CNF formula. Similarly, all Boolean functions can be expressed by DNF formulas but not by k -DNF formulas [129].

The partial evaluation of a given formula ψ is possible when a subset of its variables are assigned values. A partial evaluation usually results in the replacement of ψ by a new formula which expresses exactly those assignments satisfying ψ under the given partial assignment. Write $\psi|_{v=1}$ to denote the formula resulting from the partial evaluation of ψ due to assigning value 1 to variable v . An obvious similar statement is used to express the partial evaluation of ψ when v is assigned value 0 or when some subset of variables is assigned values. For example,

$$(v_1 \vee \neg v_2) \wedge (\neg v_1 \vee v_3) \wedge (\neg v_2 \vee \neg v_3)|_{v_1=1} = (v_3) \wedge (\neg v_2 \vee \neg v_3)$$

since $(v_3) \wedge (\neg v_2 \vee \neg v_3)$ expresses all solutions to $(v_1 \vee \neg v_2) \wedge (\neg v_1 \vee v_3) \wedge (\neg v_2 \vee \neg v_3)$ given v_1 has value 1.

If an assignment M is such that all the literals of a disjunctive (conjunctive) clause have value 0 (respectively, 1) under M , then the clause is said to be *falsified* (respectively, *satisfied*) by M . If M is such that at least one literal of a disjunctive (conjunctive) clause has value 1 (respectively, 0) under M , then the clause is said to be *satisfied* (respectively, *falsified*) by M . If a clause evaluates to \perp , then it is neither satisfied nor falsified.

A formula ψ is *satisfiable* if there exists at least one assignment under which ψ has value 1. In particular, a CNF formula is *satisfiable* if there exists a truth assignment to its variables which satisfies all its clauses. Otherwise, the formula is *unsatisfiable*. Every nonempty DNF formula is satisfiable, but a DNF formula that is satisfied by *every* truth assignment to its variables is called a *tautology*. The negation of a DNF tautology is an unsatisfiable CNF formula.

Several assignments may satisfy a given formula. Any satisfying assignment containing the smallest number of variables of value 1 among all satisfying assignments is called a *minimal model* with respect to 1. Thus, consistent with our definition of model as a set of variables of value 1, a minimal model is a set of variables of least cardinality. The usual semantics for Horn formula logic programming is the minimal model semantics: the only model considered is the (unique) minimal one.¹

¹Each satisfiable set of Horn clauses has a unique minimal model with respect to 1, which can be computed in linear time by a well-known algorithm [53, 78] which is discussed in Sect. 6.2. Implications of the minimal model semantics may be found in Sect. 6.5, among others.

If a CNF formula is unsatisfiable but removal of any clause makes it satisfiable, then the formula is said to be *minimally unsatisfiable*. Minimally unsatisfiable formulas play an important role in understanding the difference between *easy* and *hard* formulas.

This section ends with a discussion of formula equivalence. Three types of equivalence are defined along with symbols that are used to represent them as follows:

1. *Equality of formulas* ($\psi_1 = \psi_2$): two formulas are *equal* if they are the same string of symbols. Also “=” is used for equality of Boolean values, for example, $v = 1$.
2. *Logical equivalence* ($\psi_1 \Leftrightarrow \psi_2$): two formulas ψ_1 and ψ_2 are said to be logically equivalent if, for every assignment M to the variables of ψ_1 and ψ_2 , M satisfies ψ_1 if and only if M satisfies ψ_2 . For example, in the following expression, the two leftmost clauses on each side of “ \Leftrightarrow ” force v_1 and v_3 to have the same value so $(v_2 \vee v_3)$ may be substituted for $(v_1 \vee v_2)$. Therefore, the expression on the left of “ \Leftrightarrow ” is logically equivalent to the expression on the right.

$$(\neg v_1 \vee v_3) \wedge (v_1 \vee \neg v_3) \wedge (v_1 \wedge v_2) \Leftrightarrow (\neg v_1 \vee v_3) \wedge (v_1 \vee \neg v_3) \wedge (v_2 \wedge v_3).$$

Another example is

$$(v_1 \vee \neg v_2) \wedge (\neg v_1 \vee v_3) \wedge (\neg v_2 \vee \neg v_3) |_{v_1=1} \Leftrightarrow (v_3) \wedge (\neg v_2 \vee \neg v_3).$$

In the second example, assigning $v_1 = 1$ has the effect of eliminating the leftmost clause and the literal $\neg v_1$. After doing so, equivalence is clearly established.

It is important to point out the difference between $\psi_1 \Leftrightarrow \psi_2$ and $\psi_1 \Rightarrow \psi_2$. The former is an assertion that ψ_1 and ψ_2 are logically equivalent. The latter is just a formula of formal logic upon which one can ask whether there exists a satisfying assignment. The symbol “ \Leftrightarrow ” may not be included in a formula, and it makes no sense to ask whether a given assignment satisfies $\psi_1 \Leftrightarrow \psi_2$. It is easy to show that $\psi_1 \Leftrightarrow \psi_2$ if and only if $\psi_1 \Rightarrow \psi_2$ is a tautology (i.e., it is satisfied by every assignment to the variables of ψ_1 and ψ_2).

Similarly, define ψ_1 logically implies ψ_2 ($\psi_1 \Rightarrow \psi_2$): for every assignment M to the variables in ψ_1 and ψ_2 , if M satisfies ψ_1 , then M also satisfies ψ_2 . Thus, $\psi_1 \Leftrightarrow \psi_2$ if and only if $\psi_1 \Rightarrow \psi_2$ and $\psi_2 \Rightarrow \psi_1$. Also, $\psi_1 \Rightarrow \psi_2$ if and only if $\psi_1 \rightarrow \psi_2$ is a tautology.

3. *Functional equivalence* ($\psi_1 \Leftarrow_V \psi_2$): two formulas ψ_1 and ψ_2 , with variable sets V_{ψ_1} and V_{ψ_2} , respectively, are said to be functionally equivalent with respect to *variable base set* $V \subseteq V_{\psi_1} \cap V_{\psi_2}$ if, for every assignment M_V to just the variables of V , either:
 - a. There is an assignment M_1 to $V_{\psi_1} \setminus V$ and an assignment M_2 to $V_{\psi_2} \setminus V$ such that $M_V \cup M_1$ satisfies ψ_1 and $M_V \cup M_2$ satisfies ψ_2 or
 - b. There is no assignment to $V_{\psi_1} \cup V_{\psi_2}$ which contains M_V as a subset and satisfies either ψ_1 or ψ_2 .

The assignments $M_V \cup M_1$ and $M_V \cup M_2$ are called *extensions* to the assignment M_V .

The notation $\psi_1 \leftrightharpoons_V \psi_2$ is used to represent the functional equivalence of formulas ψ_1 and ψ_2 with respect to base set V . The notation $\psi_1 \leftrightharpoons \psi_2$ is used if $V = V_{\psi_1} \cap V_{\psi_2}$. In this case logical equivalence and functional equivalence are the same.

For example, $(\neg a) \wedge (a \vee b) \leftrightharpoons_{\{\neg a\}} (\neg a) \wedge (a \vee c)$ because $\exists b : (\neg a) \wedge (a \vee b)$ if and only if $\exists c : (\neg a) \wedge (a \vee c)$. But $(\neg a) \wedge (a \vee b) \neq (\neg a) \wedge (a \vee c)$ since $a = 0, b = 1, c = 0$ satisfies $(\neg a) \wedge (a \vee b)$ but falsifies $(\neg a) \wedge (a \vee c)$.

Functional equivalence is useful when transforming a formula to a more useful formula. For example, consider the Tseitin transformation [139] (Sect. 5.3) which extends resolution: for CNF formula ψ containing variables from set V_ψ , any pair of variables $x, y \in V_\psi$, and variable $z \notin V_\psi$,

$$\psi \leftrightharpoons_{V_\psi} \psi \wedge (z \vee x) \wedge (z \vee y) \wedge (\neg z \vee \neg x \vee \neg y).$$

The term *functional equivalence* is somewhat of a misnomer. For any set V of variables, \leftrightharpoons_V is an equivalence relation but \leftrightharpoons is not transitive: for example, $a \vee c \leftrightharpoons a \vee b$ and $a \vee b \leftrightharpoons a \vee \neg c$ but $a \vee c \not\leftrightharpoons a \vee \neg c$.

The foundational problem considered in this chapter is the satisfiability problem, commonly called SAT. It is stated as follows:

Satisfiability (SAT)

Given: A Boolean formula ψ .

Question: Determine whether ψ is satisfied by some truth assignment to the variables of ψ .

For the class of CNF, even 3-CNF, formulas, SAT is \mathcal{NP} -hard. However, for the class of CNF formulas of maximum width 2, SAT can be solved in linear time using Algorithm 19 of Sect. 6.1. For the class of Horn formulas, SAT can be solved in linear time using Algorithm 20 of Sect. 6.2. If ψ is a CNF formula such that reversing the polarity of some subset of its variables results in a Horn formula, then ψ is *renamable Horn*. Satisfiability of a renamable Horn formula can be determined in linear time by Algorithm 21 of Sect. 6.4.

The problem of determining the satisfiability of k -CNF formulas is referred to as k -SAT. In that context, input formulas are said to be instances of k -SAT or k -SAT formulas.

A second problem of interest is a generalization of the common problem of determining a minimal model for a given formula. The question of minimal models is considered in Sect. 6.2.

Variable-Weighted Satisfiability

Given: A CNF formula ψ and a function $w : V \mapsto \mathbb{Z}^+$ where $w(v)$ is the weight of variable v .

Question: Determine whether ψ is satisfied by some truth assignment and, if so, determine the assignment M such that

$$\sum_{v \in M} w(v) \text{ is minimized.}$$

A third problem of interest is the fundamental optimization version of SAT. This problem is probably more important in terms of practical use than even SAT.

Maximum Satisfiability (MAX-SAT)

Given: A CNF formula ψ .

Question: Determine the assignment M that satisfies the maximum number of clauses in ψ .

The following is an important variant of MAX-SAT:

Weighted Maximum Satisfiability (Weighted MAX-SAT)

Given: A CNF formula ψ and a function $w : C_\psi \mapsto \mathbb{Z}^+$ where $w(c)$ is the weight of clause $c \in C_\psi$.

Question: Determine the assignment M such that

$$\sum_{c \in \psi} w(c) \cdot s_M(c) \text{ is maximized,}$$

where $s_M(c)$ is 1 if clause c is satisfied by M and is 0 otherwise.

3 Representations and Structures

Algorithmic concepts are usually easier to specify and understand if a formula or a particular part of a formula is suitably represented. Many representations are possible for Boolean formulas, particularly when expressed as CNF or DNF formulas. As mentioned earlier, CNF and DNF formulas will often be represented as sets of clauses and clauses as sets of literals. In this section additional representations are presented; these will later be used to express algorithms and explain algorithmic behavior. Some of these representations involve only a part of a given formula.

3.1 (0 ± 1) Matrix

A CNF formula of m clauses and n variables may be represented as an $m \times n$ (0 ± 1) -matrix \mathcal{M} where the rows are indexed on the clauses, the columns are indexed on the variables, and a cell $\mathcal{M}(i, j)$ has the value $+1$ if clause i contains variable j as a positive literal, the value -1 if clause i contains variable j as a negative literal, and the value 0 if clause i does not contain variable j as a positive or negative literal. [Figure 1](#) shows an example of a CNF formula and its (0 ± 1) matrix representation.

It is well known that the question of satisfiability of a given CNF formula ψ can be cast as an integer program as follows:

$$\mathcal{M}_\psi \alpha + b \geq \mathbf{Z}, \quad (1)$$

$$\alpha_i \in \{0, 1\}, \text{ for all } 0 \leq i < n,$$

where \mathcal{M}_ψ is the (0 ± 1) matrix representation of ψ , \mathbf{b} is an integer vector with b_i equal to the number of -1 entries in row i of \mathcal{M}_ψ , and \mathbf{Z} is a vector of 1s. A solution to this system of inequalities certifies ψ is satisfiable. In this case a model can be obtained directly from α as α_i is the value of variable v_i . If there is no solution to the system, then ψ is unsatisfiable.

In addition to the well-known matrix operations, two matrix operations that are relevant to SAT algorithms on (0 ± 1) matrix representations are applied in this chapter. The first is column scaling: a column may be multiplied or *scaled* by -1 which has the effect of reversing the polarity of a single variable. Every solution before scaling corresponds to a solution after scaling; the difference is that the values of variables associated with scaled columns are reversed. The second is row and column reordering: rows and columns may be *permuted* with the effect on a solution being only a possible relabeling of variables taking value 1.

3.2 Binary Decision Diagrams

Binary decision diagrams [6, 103] are a general, graphical representation for arbitrary Boolean functions. Various forms have been put into use, especially for solving VLSI design and verification problems. A canonical form [27, 28] has been shown to be quite useful for representing some particular, commonly occurring, Boolean functions. An important advantage of BDDs is that the complexity of binary and unary operations such as existential quantification, logical or, and logical and, among others, is efficient with respect to the size of the BDD operands. Typically, a given formula ψ is represented as a large collection of BDDs and operations such as those stated above are applied repeatedly to create a single BDD which expresses the models, if any, of ψ . Intermediate BDDs are created in the process. Unfortunately, the size of intermediate BDDs may become extraordinarily and impractically large even if the final BDD is small. So in some applications BDDs are useful and in some they are not.

A *binary decision diagram* (BDD) is a rooted, directed acyclic graph. A BDD is used to compactly represent the truth table, and therefore complete functional description, of a Boolean function. Vertices of a BDD are called *terminal* if they have no outgoing edges and are called *internal* otherwise. There is one internal vertex, called the *root*, which has no incoming edge. There is at least one terminal vertex, labeled 1, and at most two terminal vertices, labeled 0 and 1. Internal vertices are labeled to represent the variables of the corresponding Boolean function. An internal vertex has exactly two outgoing edges, labeled 1 and 0. The vertices incident to edges outgoing from vertex v are called *then*(v) and *else*(v), respectively. Associated with any internal vertex v is an attribute called *index*(v) which satisfies the properties $\text{index}(v) < \min\{\text{index}(\text{then}(v)), \text{index}(\text{else}(v))\}$ and $\text{index}(v) = \text{index}(w)$ if and only if vertices v and w have the same labeling (i.e., correspond to the same variable). Thus, the *index* attribute imposes a linear ordering on the variables of a BDD. An example of a formula and one of its BDD representations is given in Fig. 2.

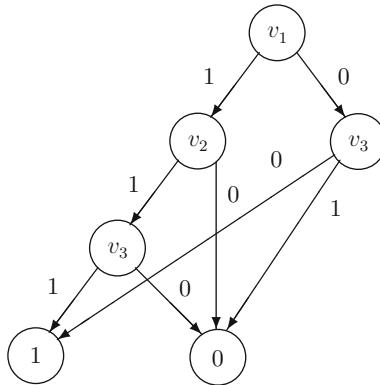


Fig. 2 The formula $(v_1 \vee \neg v_3) \wedge (\neg v_1 \vee v_2) \wedge (\neg v_1 \vee \neg v_2 \vee v_3)$ represented as a BDD. The topmost vertex is the root. The two bottom vertices are terminal vertices. Edges are directed from upper vertices to lower vertices. Vertex labels (variable names) are shown inside the vertices. The 0 branch out of a vertex labeled v means v takes the value 0. The 1 branch out of a vertex labeled v means v takes the value 1. The *index* of a vertex is, in this case, the subscript of the variable labeling that vertex

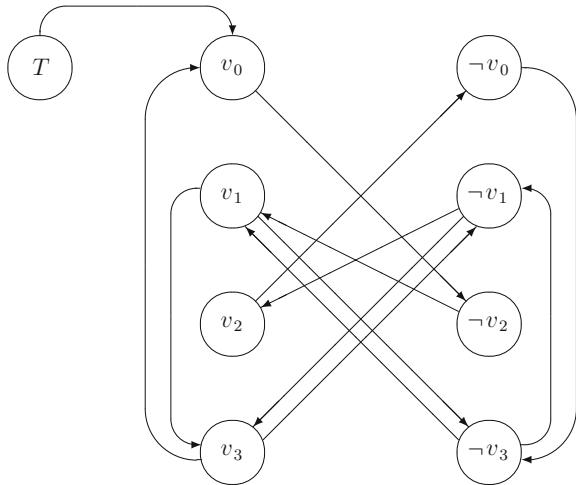
Clearly, there is no unique BDD for a given formula. In fact, for the same formula, one BDD might be extraordinarily large and another might be rather compact. It is usually advantageous to use the smallest BDD possible. At least one canonical form of BDD, called *reduced ordered BDD*, does this [27, 28]. The idea is to order the variables of a formula and construct a BDD such that (1) variables contained in a path from the root to any leaf respect that ordering and (2) each vertex is the root of a BDD that represents a Boolean function that is unique with respect to all other vertices. Two Boolean functions are equivalent if their reduced ordered BDDs are isomorphic. A more detailed explanation is given in Sect. 5.9.

3.3 Implication Graph

An *implication graph* of a CNF formula ψ is a directed graph $\vec{G}_\psi(V, \vec{E})$ where V consists of one special vertex T which corresponds to the value 1, other vertices which correspond to the literals of ψ , and the edge set \vec{E} such that there is an edge $\langle v_i, v_j \rangle \in \vec{E}$ if and only if there is a clause $(\neg v_i \vee v_j)$ in ψ and an edge $\langle T, v_i \rangle \in \vec{E}$ ($\langle T, \neg v_i \rangle \in \vec{E}$) if and only if there is a unit clause (v_i) (respectively, $(\neg v_i)$) in ψ . Figure 3 shows an example of an implication graph for a particular 2-CNF formula.

Implication graphs are most useful for, but not restricted to, 2-CNF formulas. In this role the meaning of an edge $\langle v_i, v_j \rangle$ is as follows: *if variable v_i is assigned the value 1 then variable v_j is inferred to have value 1, or else the clause $(\neg v_i \vee v_j)$ will be falsified.*

Fig. 3 An implication graph and associated 2-CNF formula



$$((v_0) \wedge (\neg v_0 \vee \neg v_2) \wedge (v_1 \vee v_2) \wedge (\neg v_1 \vee \neg v_3) \wedge \\ (\neg v_1 \vee v_3) \wedge (v_1 \vee v_3) \wedge (v_0 \vee \neg v_3))$$

3.4 Propositional Connection Graph

A *propositional connection graph* for a CNF formula ψ is an undirected graph $G_\psi(V, E)$ whose vertex set corresponds to the clauses of ψ and whose edge set is such that there is an edge $\{c_i, c_j\} \in E$ if and only if the clause in ψ represented by vertex c_i has a literal that appears negated in the clause represented by vertex c_j , and there is no other literal in c_i 's clause that appears negated in c_j 's clause. An example of a connection graph for a particular CNF formula is given in Fig. 4. Propositional connection graphs are a specialization of the first-order connection graphs developed by Kowalski [102].

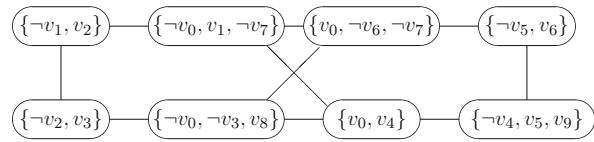
3.5 Variable-Clause Matching Graph

A *variable-clause matching graph* for a CNF formula is an undirected bipartite graph $G = (V_1, V_2, E)$ where V_1 vertices correspond to clauses and V_2 vertices correspond to variables and whose edge set contains an edge $\{v_i, v_j\}$ if and only if $v_i \in V_1$ corresponds to a variable that exists, either as positive or negative literal, in clause $v_j \in V_2$. An example of a variable-clause matching graph is shown in Fig. 5.

3.6 Formula Digraph

Formulas are defined recursively on Page 313. Any Boolean formula can be adapted to fit this definition with the suitable addition of parentheses, and its structure

Fig. 4 A propositional connection graph and associated CNF formula



$$((\neg v_0 \vee v_1 \vee \neg v_7) \wedge (\neg v_1 \vee v_2) \wedge (\neg v_2 \vee v_3) \wedge (\neg v_0 \vee \neg v_3 \vee v_8) \wedge \\ (v_0 \vee \neg v_6 \vee \neg v_7) \wedge (\neg v_5 \vee v_6) \wedge (\neg v_4 \vee v_5 \vee v_9) \wedge (v_0 \vee v_4))$$

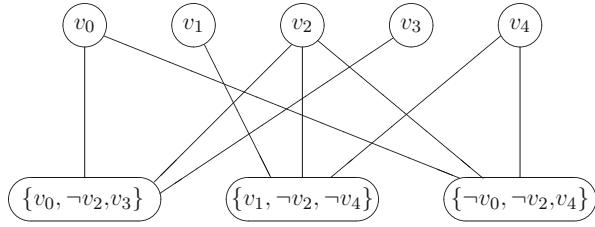


Fig. 5 A variable-clause matching graph and associated CNF formula

$$((v_0 \vee \neg v_2 \vee v_3) \wedge (v_1 \vee \neg v_2 \vee \neg v_4) \wedge (\neg v_0 \vee \neg v_2 \vee v_4))$$

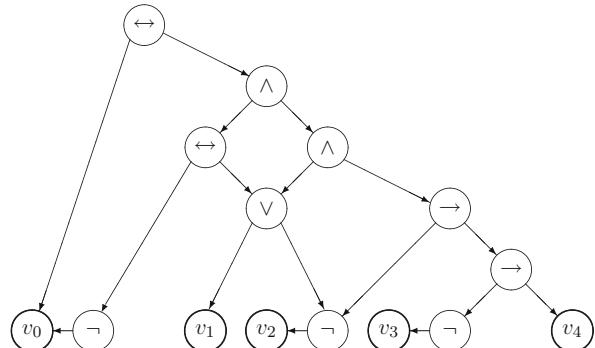


Fig. 6 A formula digraph and associated formula

$$v_0 \leftrightarrow ((\neg v_0 \leftrightarrow (v_1 \vee \neg v_2)) \wedge (v_1 \vee \neg v_2) \wedge (\neg v_2 \rightarrow \neg v_3 \rightarrow v_4))$$

(as opposed to its functionality) can be represented by a binary rooted acyclic digraph. In such a digraph, each internal vertex corresponds to a binary operator or a unary operator \neg that occurs in the formula. A vertex corresponding to a binary operator has two outward-oriented edges: the left edge corresponds to the left subformula and the right edge to the right subformula operated on. A vertex corresponding to \neg has one outward directed edge. The root represents the operator applied at top level. The leaves are variables. Call such a representation a *wff digraph*. An example is given in Fig. 6. An efficient algorithm for constructing a wff digraph is given in Sect. 5.1.

3.7 Satisfiability Index

Let ψ be a CNF formula and let \mathcal{M}_ψ be its (0 ± 1) matrix representation. Let $\alpha = (\alpha_0, \alpha_1, \dots, \alpha_{n-1})$ be an n -dimensional vector of real variables. Let z be a real variable and let $Z = (z, z, \dots, z)$ be an m dimensional vector where every component is the variable z . Finally, let $b = (b_0, b_1, \dots, b_{m-1})$ be an m dimensional vector such that for all $0 \leq i < m$, b_i is the number of negative literals in clause c_i . Form the system of inequalities

$$\mathcal{M}_\psi \alpha + b \leq Z, \quad (2)$$

$$0 \leq \alpha_i \leq 1 \text{ for all } 0 \leq i < n.$$

The satisfiability index of ψ is the minimum z for which no constraints of the system are violated. For example, the CNF formula

$$((v_1 \vee \neg v_2) \wedge (v_2 \vee \neg v_3 \vee v_5) \wedge (v_3 \vee \neg v_4 \vee \neg v_5) \wedge (v_4 \vee \neg v_1))$$

has satisfiability index of 5/4.

3.8 And/Inverter Graphs

An AIG is a directed acyclic graph where all *gate* vertices have in-degree 2, all *input* vertices have in-degree 0, all *output* vertices have in-degree 1, and edges are labeled as either *negated* or *not negated*. Any combinational circuit can be equivalently implemented as a circuit involving only 2-input *and* gates and *not* gates. Such a circuit has an AIG representation: *gate* vertices correspond directly to the *and* gates, negated edges correspond directly to the *not* gates, and inputs and outputs represent themselves directly. Negated edges are typically labeled by overlaying a white circle on the edge: this distinguishes them from non-negated edges which are unlabeled. Examples are shown in Fig. 7.

4 Applications

This section presents a sample of real-world problems that may be viewed as, or transformed to, instances of SAT. Of primary interest is to explore the thinking process required for setting up logic problems and gauging the complexity of the resulting systems. Since it is infeasible to meet this objective and thoroughly discuss a large number of known applications, a small but varied and interesting collection of problems are discussed.

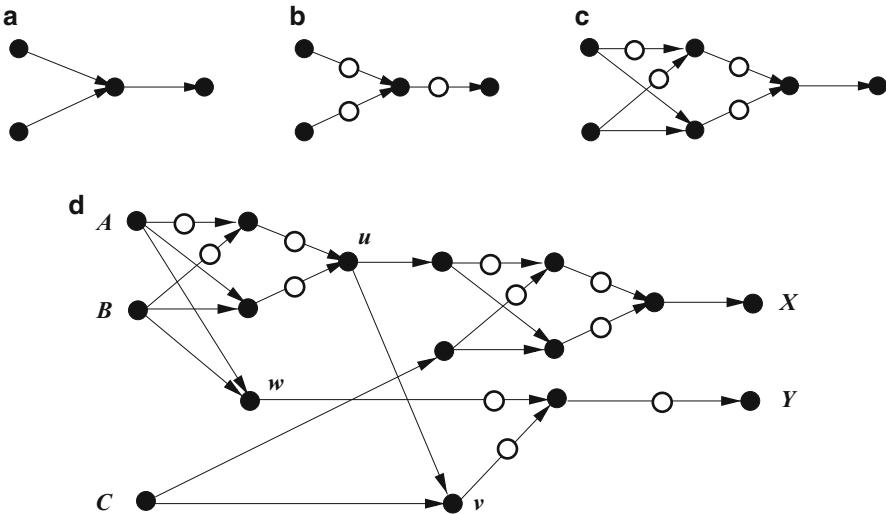


Fig. 7 Examples of AIGs. Edges with white circles are negated. Edges without a white circle are not negated. (a) and gate. (b) or gate. (c) xor gate. (d) 1-bit add circuit of Fig. 9

4.1 Consistency Analysis in Scenario Projects

This application, taken from the area of scenario management [8, 54, 63], is contributed by Feldmann and Sensen [57] of Burkhard Monien's old PC² group at Universität Paderborn, Germany. A *scenario* consists of (i) a progression of events from a known base situation to a possible terminal future situation and (ii) a means to evaluate its likelihood. Scenarios are used by managers and politicians to strategically plan the use of resources needed for solutions to environmental, social, economic, and other such problems.

A systematic approach to scenario management due to Gausemeier, Fink, and Schlake [62] involves the realization of *scenario projects* with the following properties:

1. There is a set S of *key factors*. Let the number of key factors be n .
2. For each key factor $s_i \in S$, there is a set $D_i = \{d_{i,1}, d_{i,2}, \dots, d_{i,m_i}\}$ of m_i possible *future developments*. In the language of databases, key factors are *attributes* and future developments are attribute *values*.
3. For all $1 \leq i \leq n$, $1 \leq k \leq m_i$, denote by $(s_i, d_{i,k})$ a feasible *projection* of development $d_{i,k} \in D_i$ from key factor s_i . For each pair of projections $(s_i, d_{i,k}), (s_j, d_{j,l})$, a *consistency value*, usually an integer ranging from 0 to 4, is defined. A consistency value of 0 typically means two projections are completely inconsistent, a value of 4 typically means the two projections support each other strongly, and the other values account for intermediate levels of support.

Consistency values may be organized in a $\sum_{i=1}^n m_i \times \sum_{i=1}^n m_i$ matrix with rows and columns indexed on projections.

4. Projections for all key factors may be *bundled* into a vector $x = (x_{s_1}, \dots, x_{s_n})$ where x_{s_i} is a future development of key factor s_i , $i = 1, 2, \dots, n$. In the language of databases, a bundle is a *tuple* which describes an assignment of values to each attribute.
5. The *consistency* of bundle x is the sum of the consistency values of all pairs $(s_i, x_{s_i}), (s_j, x_{s_j})$ of projections represented by x if no pair has consistency value of 0, and is 0 otherwise.

Bundles with greatest (positive) consistency are determined and clustered. Each cluster is a scenario.

To illustrate, consider a simplified example from [57] which is intended to develop likely scenarios for the German school system over the next 20 years. It was felt by experts that 20 key factors are needed for such forecasts; to keep the example small, only the first 10 are shown. The first five key factors and their associated future developments are as follows: $[s_1]$ *continuing education* ($[d_{1.1}]$ lifelong learning), $[s_2]$ *importance of education* ($[d_{2.1}]$ important, $[d_{2.2}]$ unimportant), $[s_3]$ *methods of learning* ($[d_{3.1}]$ distance learning, $[d_{3.2}]$ classroom learning), $[s_4]$ *organization and policies of universities* ($[d_{4.1}]$ enrollment selectivity, $[d_{4.2}]$ semester schedules), and $[s_5]$ *adequacy of trained people* ($[d_{5.1}]$ sufficiently many, $[d_{5.2}]$ not enough).

The table in Fig. 8, called a *consistency matrix*, shows the consistency values for all possible pairs of future developments. The s_i, s_j cell of the matrix shows the consistency values of pairs $\{(s_i, d_{i,x}), (s_j, d_{j,y}) : 1 \leq x \leq m_i, 1 \leq y \leq m_j\}$, with all pairs which include $(s_i, d_{i,x})$ on the x th row of the cell. For example, $|D_5| = 2$ and $|D_2| = 2$, so there are 4 numbers in cell s_5, s_2 and the consistency value of $(s_5, d_{5.2}), (s_2, d_{2.2})$ is the bottom right number of that cell. That number is 0 because experts have decided the combination of there being too few trained people ($d_{5.2}$) at the same time education is considered unimportant ($d_{2.2}$) is unlikely.

It is relatively easy to compute the consistency of a given bundle from this table. One possible bundle of future developments is $\{(s_i, d_{i,1}) : 1 \leq i \leq 10\}$. Since the consistency value of $\{(s_5, d_{5,1}), (s_2, d_{2,1})\}$ is 0, this bundle is inconsistent. Another possible bundle is

$$\begin{aligned} &\{(s_1, d_{1,1}), (s_2, d_{2,1}), (s_3, d_{3,1}), (s_4, d_{4,1}), (s_5, d_{5,2}), \\ &(s_6, d_{6,3}), (s_7, d_{7,1}), (s_8, d_{8,2}), (s_9, d_{9,1}), (s_{10}, d_{10,2})\}. \end{aligned}$$

Its consistency value is 120 (the sum of the 45 relevant consistency values from the table).

Finding good scenarios from a given consistency matrix requires efficient answers to the following questions:

1. Does a consistent bundle exist?
2. How many consistent bundles exist?
3. What is the bundle having the greatest consistency?

s_2	4 1										
s_3	2	2 3									
	3	2 3									
s_4	3	2 3	2 3								
	3	1 3	2 3								
s_5	2	0 4	0 4	2 3							
	3	4 0	4 0	2 3							
s_6	2	2 3	3 2	2 3	1 3						
	2	2 3	2 3	3 2	2 1						
	3	2 3	2 3	2 3	2 3						
s_7	3	4 0	3 0	2 3	0 4	3 0 3					
	1	0 4	0 3	2 3	4 0	1 3 2					
s_8	2	2 3	4 2	2 3	0 3	2 3 4	4 0				
	2	2 3	3 1	2 3	0 4	2 3 2	2 3				
	2	2 3	0 2	2 3	3 2	2 3 1	0 4				
s_9	3	2 3	2 3	2 3	1 3	2 3 2	4 0	4 2 1			
	3	2 3	2 3	2 3	1 4	2 3 2	4 1	3 2 1			
s_{10}	3	3 0	0 4	2 3	4 0	2 3 2	2 3	1 2 3	1 2		
	2	2 3	4 0	2 3	0 4	2 3 2	3 2	2 3 2	2 3		
	2	2 3	2 0	2 3	2 3	2 3 2	2 3	3 2 3	2 3		
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9		

Fig. 8 A consistency matrix for a scenario project**Table 3** Formula to determine existence of a consistent bundle

Clause of $\psi^{S,D}$	Subscript range	Meaning
$(\neg v_{i,j} \vee \neg v_{i,k})$	$1 \leq i \leq n, 1 \leq j < k \leq m_i$	≤ 1 development/key factor
$(v_{i,1} \vee \dots \vee v_{i,m_i})$	$1 \leq i \leq n$	≥ 1 development/key factor
$(\neg v_{i,k} \vee \neg v_{j,l})$	$i, j, k, l : C_{i,k,j,l} = 0$	Consistent developments only

Finding answers to these questions is \mathcal{NP} -hard [57]. But transforming to CNF formulas, in some cases with weights on proposition letters, and solving a variant of SAT are sometimes reasonable ways to tackle such problems. This context provides the opportunity to use our vast knowledge of SAT structures and analysis to apply an algorithm that has a reasonable chance of solving the consistency problems efficiently. It is next shown how to construct representative formulas so that, often, a large subset of clauses is polynomial-time solvable and the whole formula is relatively easy to solve.

Consider, first, the question whether a consistent bundle exists for a given consistency matrix of n key factors with m_i projections for factor i , $1 \leq i \leq n$. Let $C_{i,k,j,l}$ denote the consistency of the pair $(s_i, d_{i,k})(s_j, d_{j,l})$ and let $D = \cup_{i=1}^n D_i$. For each future development $d_{i,j}$, define variable $v_{i,j}$ which is intended to take the value 1 if and only if $d_{i,j}$ is a future development for key attribute s_i . The CNF formula $\psi^{S,D}$ with the clauses described in Table 3 then says that there is a consistent bundle. That is, the formula is satisfiable if and only if there is a consistent bundle.

The second question to be considered is how many consistent bundles exist for a given consistency matrix? This is the same as asking how many satisfying truth assignments there are for $\psi^{S,D}$. A simple inclusion-exclusion algorithm exhibits very good performance for some actual problems of this sort [57].

The third question is which bundle has the greatest consistency value? This question can be transformed to an instance of the variable-weighted satisfiability problem (defined on Page 318). The transformed formula consists of $\psi^{S,D}$ plus some additional clauses as follows. For each pair $(s_i, d_{i,k})(s_j, d_{j,l})$, $i \neq j$, of projections such that $C_{i,k,j,l} > 0$, create a new Boolean variable $p_{i,k,j,l}$ of weight $C_{i,k,j,l}$ and add the following subexpression to $\psi^{S,D}$:

$$(\neg v_{i,k} \vee \neg v_{j,l} \vee p_{i,k,j,l}) \wedge (v_{i,k} \vee \neg p_{i,k,j,l}) \wedge (v_{j,l} \vee \neg p_{i,k,j,l}).$$

Observe that a satisfying assignment requires $p_{i,j,k,l}$ to have value 1 if and only if $v_{i,k}$ and $v_{j,l}$ both have value 1, that is, if and only if the consistency value of $\{(s_i, d_{i,k}), (s_j, d_{j,l})\}$ is included in the calculation of the consistency value of the bundle.

If weight 0 is assigned to all variables other than the $p_{i,k,j,l}$'s, a maximum weight solution specifies a maximum consistency bundle. It should be pointed out that, although the number of clauses added to $\psi^{S,D}$ might be significant compared to the number of clauses originally in $\psi^{S,D}$, the total number of clauses will be linearly related to the size of the consistency matrix. Moreover, the set of additional clauses is a Horn subformula.² A maximum weight solution can be found by means of a branch-and-bound algorithm such as that discussed in Sect. 5.11.

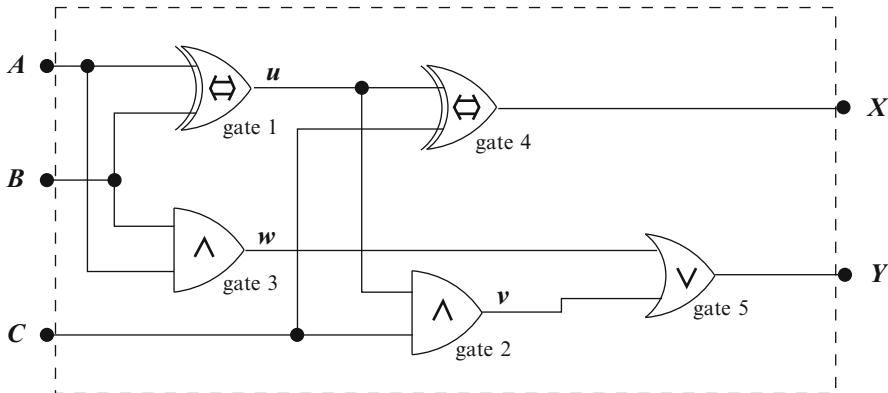
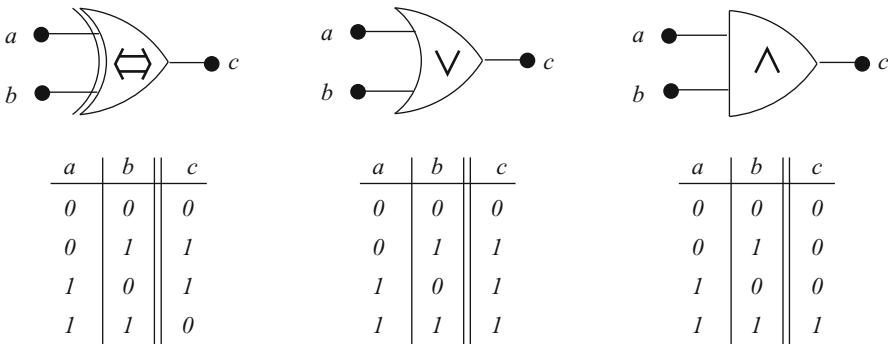
4.2 Testing of VLSI Circuits

A classic application is the design of test vectors for VLSI circuits. At the specification level, a combinational VLSI circuit is regarded to be a function mapping n 0-1 inputs to m 0-1 outputs.³ At the design level, the interconnection of numerous 0-1 logic gates are required to implement the function. Each connection entails an actual interconnect point that can fail during or soon after manufacture. Failure of an interconnect point usually causes the point to become *stuck-at* value 0 or 1.

Traditionally, VLSI circuit testing includes testing all interconnect points for stuck-at faults. This task is difficult because interconnect points are encased in plastic and are therefore inaccessible directly. The solution is to apply an input pattern to the circuit which excites the point under test *and* sensitizes a path through the circuit from the test point to some output so that the correct value at the test point can be determined at the output.

²Horn formulas are solved efficiently (see Sect. 6.2).

³Actual circuit voltage levels are abstracted to the values 0 and 1.

**Fig. 9** A 1-bit full adder circuit**Fig. 10** Truth tables for logic elements of a 1-bit full adder

The following example illustrates how such an input pattern and output is found for one internal point of a *1-bit full adder*: an elementary but ubiquitous functional hardware block that is depicted in Fig. 9. For the sake of discussion, assume a given circuit will have at most one stuck-at failure. The 1-bit full adder uses logic gates that behave according to the truth tables in Fig. 10 where a and b are gate inputs and c is a gate output. Suppose none of the interconnect points enclosed by the dashed line of Fig. 9 are directly accessible, and suppose it is desired to develop a test pattern to determine whether point w is stuck at 0. Then inputs must be set to give point w the value 1. This is accomplished by means of the Boolean expression $\psi_1 = (A \wedge B)$. The value of point w can only be observed at output Y . But this requires point v be set to 0. This can be accomplished if either the value of C is 0 or u is 0, and u is 0 if and only if A and B have the same value. Therefore, the Boolean expression representing sensitization of a path from w to Y is $\psi_2 = (\neg C \vee (A \wedge B) \vee (\neg A \wedge \neg B))$. The conjunction $\psi_1 \wedge \psi_2$ is the CNF expression $(A \wedge B)$. This expression is satisfied if and only if A and B are set to 1. Such an

input will cause output Y to have value 1 if w is not stuck at 0 and value 0 if w is stuck at 0, assuming no other interconnect point is stuck at some value.

Test patterns must be generated for all internal interconnect points of a VLSI circuit. There could be millions of these and the corresponding expressions could be considerably more complex than that of the example above. Moreover, testing is complicated by the fact that most circuits are not combinational: that is, they contain feedback loops. This last case is mitigated by adding circuitry for testing purposes only. The magnitude of the testing problem, although seemingly daunting, is not great enough to cause major concern at this time because it seems that SAT problems arising in this domain are usually easy. Thus, at the moment, the VLSI testing problem is considered to be under control. However, this may change in the near future since the number of internal points in a dense circuit is expected to continue to increase dramatically.

4.3 Diagnosis of Circuit Faults

A natural extension of the test design discussed in Sect. 4.2 is finding a way to automate the diagnosis of, say, bad chips, starting with descriptions of their bad outputs. How can one reason backwards to identify likely causes of the malfunction? Also, given knowledge that some components are more likely to fail than others, how can the diagnosis system be tailored to suggest the most likely causes first? The first of these questions is discussed in this section.

The first step is to write the Boolean expression representing both the normal and abnormal behavior of the analyzed circuit. This is illustrated using the circuit of Fig. 9. The expression is assembled in stages, one for each gate, starting with gate 3 of Fig. 9, which is an *and* gate. The behavior of a correctly functioning *and* gate is specified in the right-hand truth table in Fig. 10. The following formula expresses the truth table:

$$(a \wedge b \wedge c) \vee (\neg a \wedge \neg b \wedge c) \vee (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge \neg c).$$

If gate 3 is possibly stuck at 0, its functionality can be described by adding variable Ab_3 (for *gate 3 is abnormal*) and substituting A, B , and w for a, b , and c , to get the following abnormality expression:

$$\begin{aligned} & (A \wedge B \wedge w \wedge \neg Ab_3) \vee (\neg A \wedge \neg B \wedge w \wedge \neg Ab_3) \vee \\ & (\neg A \wedge B \wedge \neg w \wedge \neg Ab_3) \vee (\neg A \wedge \neg B \wedge \neg w \wedge \neg Ab_3) \vee (\neg w \wedge Ab_3) \end{aligned}$$

which has value 1 if and only if gate 3 is functioning normally for inputs A and B or it is functioning abnormally and w is stuck at 0. The extra variable may be regarded as a switch which allows toggling between abnormal and normal states for gate 3. Similarly, switches Ab_1, Ab_2, Ab_4 , and Ab_5 may be added for all the other gates in

Table 4 Possible stuck-at-0 failures of 1-bit adder gates (see Fig. 9) assuming given inputs are $A = 0$ and $B, C = 1$ and observed outputs are $X, Y = 0$. This is the list of assignments to Δ which satisfy the abnormality predicates for the 1-bit adder. A “1” in the column for Ab_i means gate i is stuck at 0

IDs	Ab_1	Ab_2	Ab_3	Ab_4	Ab_5
1–16	*	*	*	*	1
17–24	*	1	*	*	0
25–26	1	0	*	1	0

The symbol “*” means either “0” or “1”

the circuit, and corresponding expressions may be constructed using those switches. Then the set

$$\Delta = \{Ab_1, Ab_2, Ab_3, Ab_4, Ab_5\}$$

represents all possible explanations for stuck-at-0 malfunctions. The list of assignments to the variables of Δ which satisfy the collection of abnormality expressions, given particular inputs and observations, determines all possible combinations of stuck-at-0 failures in the circuit. The next task is to choose the most likely failure combination from the list. This requires some assumptions which are motivated by the following examples.

Suppose that, during some test, inputs are set to $A = 0$ and $B, C = 1$ and the observed output values are $Y = 0$ and $X = 1$ whereas $Y = 1$ and $X = 0$ are the correct outputs. One can reason backwards to try to determine which gates are stuck at 0. Gate 4 cannot be stuck at 0 since its output is 1. Suppose gate 4 is working correctly. Since the only gate that gate 4 depends on is gate 1, that gate must be stuck. One cannot tell whether gates 2, 3, and 5 are functioning; normally it would be assumed that they are functioning correctly until evidence to the contrary is obtained. Thus, the natural diagnosis is gate 1 is defective (only Ab_1 has value 1).

Alternatively, suppose under the same inputs it is observed that $X, Y = 0$. Possibly, gates 5 and 4 are malfunctioning. If so, all other combinations of gate outputs will lead to the same observable outputs. If gate 5 is defective but gate 4 is good, then $u = 1$ so gate 1 is good, and any possible combinations of w and v lead to the same observable outputs. If gate 5 is good and gate 4 is defective, the bad Y value may be caused by a defective gate 2. In that case gates 1 and 3 conditions do not affect the observed outputs. But if gate 2 is not defective, the culprit must be gate 1. If gates 4 and 5 are good, then $u = 1$ so gate 2 is defective. Nothing can be determined about the condition of gate 3 through this test. The results of this paragraph lead to 26 abnormal Δ values that witness the observed outputs: these are summarized in Table 4, grouped into three cases. In the first two of these cases, the minimum number of gates stuck at 0 is 1.

As before, it is assumed that the set of malfunctioning gates is as small as possible, so only those two diagnoses are considered: that is, either (i) gate 5 or (ii) gate 2 is defective. In general, it is argued, common sense leads us to consider

only *minimal* sets of abnormalities: sets, like gate 2 above, where no proper subset is consistent with the observations. This is Reiter's *principle of parsimony* [117]:

A diagnosis is a conjecture that some minimal set of components are faulty.

This sort of inference is called *non-monotonic* because it is inferred, above, that gate 3 was functioning correctly, since there is no evidence it was not. Later evidence may cause that inference to be withdrawn.

Yet a further feature of non-monotonic logic may be figured into such systems; the following illustrates the idea. Suppose it is known that one component, say, gate 5, is the least likely to fail. Then, if there are any diagnoses in which gate 5 does not fail, it will report only such diagnoses. If gate 5 fails in all diagnoses, then it will report all the diagnoses. Essentially, this reflects a kind of preference relationship among diagnoses.

There is now software which automates this diagnosis process (e.g., [66]). Although worst-case performance of such systems is provably bad, such a system can be useful in many circumstances. Unfortunately, implementations of non-monotonic inference are new enough that there is not yet a sufficiently large body of standard benchmark examples.

4.4 Functional Verification of Hardware Design

Proving correctness of the design of a given block of hardware has become a major concern due to the complexity of present day hardware systems and the economics of product delivery time. Prototyping is no longer feasible since it takes too much time and fabrication costs are high. Breadboarding no longer gives reliable results because of the electrical differences between integrated circuits and discrete components. Simulation-based methodologies are generally fast but do not completely validate a design since there are many cases left unconsidered. Formal verification methods can give better results, where applicable, since they will catch design errors that may go undetected by a simulation.

Formal verification methods are used to check correctness by detecting errors in translation between abstract levels of the design hierarchy. Design hierarchies are used because it is impractical to design a VLSI circuit involving millions of components at the substrate, or lowest, level of abstraction. Instead, it is more reasonable to design at the specification or highest level of abstraction and use software tools to translate the design, through some intermediate stages such as the logic-gate level, to the substrate level. The functionality between a pair of levels may be compared. In this case, the more abstract level of the pair is said to be the *specification* and the other level is the *implementation* level of the pair. If functionality is equivalent between all adjacent pairs of levels, the design is said to be *verified*.

Determining functional equivalence (defined on Page 317) between levels amounts to proving a theorem of the form *implementation I realizes specification S* in a particular, suitable formal proof system. For illustration purposes only, consider the 1-bit full adder of Fig. 9. Inputs *A* and *B* represent a particular bit position of

two different binary addends. Input C is the carry due to the addition at the next lower valued bit position. Output X is the value of the same bit position of the sum and output Y is the carry to the next higher valued bit position. The output X must have value 1 if and only if all inputs have value 1 or exactly one input has value 1. The output Y has value 1 if and only if at least two out of three inputs have value 1. Therefore, the following simple Boolean expression offers a reasonable specification of any 1-bit full adder:

$$(X \Leftrightarrow (A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) \vee (A \wedge B \wedge C)) \wedge \\ (Y \Leftrightarrow (A \wedge B) \vee (A \wedge C) \vee (B \wedge C)).$$

A proposed implementation of this specification is given in the dotted region of Fig. 9. Its behavior may be described by a Boolean expression that equates each gate output to the corresponding logical function applied to its inputs. The following is such an expression:

$$(u \Leftrightarrow (A \wedge \neg B) \vee (\neg A \wedge B)) \wedge \\ (v \Leftrightarrow u \wedge C) \wedge \\ (w \Leftrightarrow A \wedge B) \wedge \\ (X \Leftrightarrow (u \wedge \neg C) \vee (\neg u \wedge C)) \wedge \\ (Y \Leftrightarrow w \vee v).$$

Designate these formulas $\psi_S(A, B, C, X, Y)$ and $\psi_I(A, B, C, X, Y, u, v, w)$, respectively. The adder correctly implements the specification if, for all possible inputs $A, B, C \in \{0, 1\}$, the output of the adder matches the specified output. For any individual inputs, A, B, C , that entails checking whether $\psi_I(A, B, C, X, Y, u, v, w)$ has value 1 for the appropriate u, v, w ,

$$\psi_S(A, B, C, X, Y) \Leftrightarrow \exists u, v, w \psi_I(A, B, C, X, Y, u, v, w),$$

where the quantification $\exists u, v, w$ is over *Boolean values* u, v, w . For *specific* A, B, C , the problem is a satisfiability problem: can values for u, v, w which make the formula have value 1 be found? (Of course, it is an easy satisfiability problem in this case.) Thus, the question of whether the adder correctly implements the specification for all 32 possible input sequences is answered using the formula:

$$\forall A, B, C, X, Y (\psi_S(A, B, C, X, Y) \Leftrightarrow \exists u, v, w \psi_I(A, B, C, X, Y, u, v, w)),$$

which has a second level of Boolean quantification. Such formulas are called *quantified Boolean formulas*.

The example of the 1-bit full adder shows how combinational circuits can be verified. A characteristic of combinational circuits is that output behavior is strictly a function of the current values of inputs and does not depend on past history.

Table 5 The operators of temporal logic

Op. name	$(S, s_i) \models$	If and only if
	p (p a variable)	$s_i(p) = 1$
<i>Not</i>	$\neg\psi_1$	$(S, s_i) \not\models \psi_1$
<i>And</i>	$\psi_1 \wedge \psi_2$	$(S, s_i) \models \psi_1$ and $(S, s_i) \models \psi_2$
<i>Or</i>	$\psi_1 \vee \psi_2$	$(S, s_i) \models \psi_1$ or $(S, s_i) \models \psi_2$
<i>Henceforth</i>	$\Box\psi_1$	$(S, s_j) \models \psi_1$ for all states s_j , $j \geq i$
<i>Eventually</i>	$\Diamond\psi_1$	$(S, s_j) \models \psi_1$ for some state s_j , $j \geq i$
<i>Next</i>	$\circ\psi_1$	$(S, s_{i+1}) \models \psi_1$
		For some $j \geq i$,
<i>Until</i>	$\psi_1 \mathcal{U} \psi_2$	$(S, s_i), (S, s_{i+1}), \dots, (S, s_{j-1}) \models \psi_1$, and $(S, s_j) \models \psi_2$

However, circuits frequently contain components, such as registers, which exhibit some form of time dependency. Such effects may be modeled by some form of *propositional temporal logic*.

Systems of temporal logic have been applied successfully to the verification of some sequential circuits including microprocessors. One may think of a sequential circuit as possessing one of a finite number of valid *states* at any one time. The current state of such a circuit embodies the complete electrical signal history of the circuit beginning with some distinguished initial state. A change in the electrical properties of a sequential circuit at a particular moment in time is represented as a fully deterministic movement from one state to another based on the current state and a change in some subset of input values only. Such a change in state is accompanied by a change in output values.

A description of several temporal logics can be found in [149]. For illustrative purposes, one of these is discussed below, namely, the linear time temporal logic (LTTL). LTTL formulas take value 1 with respect to an infinite sequence of states $S = \{s_0, s_1, s_2, \dots\}$. States of S obey the following: state s_0 is a *legal* initial state of the system, state s_i is a *legal* state of the system at time step i , and every pair s_i, s_{i+1} must be a *legal* pair of states. Legal pairs of states are forced by some of the components of the formula itself (the latch example at the end of this section illustrates this). Each state is just an interpretation of an assignment of values to a set of Boolean variables.

LTTL is an extension of the propositional calculus that adds one binary and three unary *temporal* operators which are described below and whose semantics are outlined in Table 5 along with the standard propositional operators \neg , \wedge , and \vee (the definition of $(S, s_i) \models$ is given below). The syntax of LTTL formulas is the same as for propositional logic except for the additional operators.

Let ψ be a LTTL expression that includes a component representing satisfaction of some desired property in a given circuit. An example of a property for a potential JK flip-flop might be that *it is eventually possible to have a value of 1 for output Q while input J has value 1* which has a corresponding formula $\Diamond(J \wedge Q)$. The event that ψ has value 1 for state s_i in S is denoted by

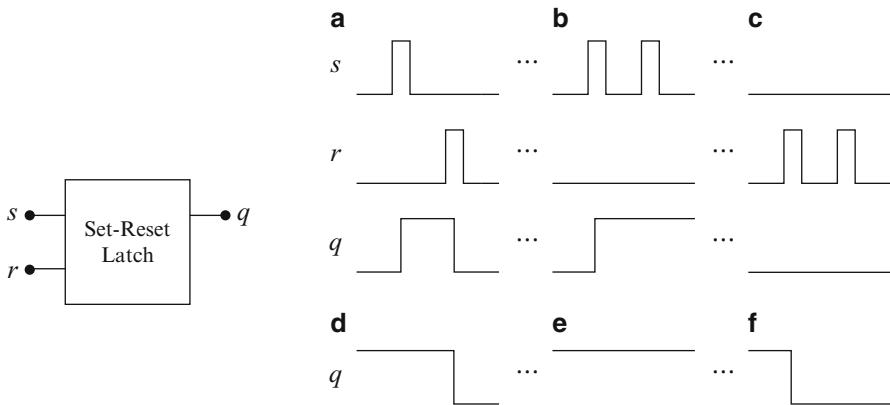


Fig. 11 A set-reset latch and complete description of signal behavior. Inputs are s and r , output is q . The horizontal axis represents time. The vertical axis represents signal value for both inputs and output. Values of all signals are assumed to be either 0 (low) or 1 (high) at any particular time. Two rows for q values are used to differentiate between the cases where the initial value of q is 0 or 1

$$(S, s_i) \models \psi.$$

Also, say

$$S \models \psi \text{ if and only if } (S, s_0) \models \psi.$$

Finally, two LTTL formulas ψ_1 and ψ_2 are *equivalent* if, for all sequences S ,

$$S \models \psi_1 \text{ if and only if } S \models \psi_2.$$

Our example concerns a hardware device of two inputs and one output called a *set-reset latch*. The electrical behavior of such a device is depicted in Fig. 11 as a set of six waveforms which show the value of the output q in terms of the history of the values of inputs r and s . For example, consider waveform (a) in the Fig. 11. This shows the value of the output q , as a function of time, if initially q , r , and s have value 0 and then s is *pulsed* or raised to value 1 then some time later dropped to value 0. The waveform shows the value of q rises to 1 some time after s does and stays at value 1 after the value of s drops. The waveform (a) also shows that if q has value 1 and r and s have value 0 and then r is pulsed, the value of q drops to 0. Observe the two cases where (i) r and s have value 1 at the same moment and (ii) q changes value *after* s or r pulses are not allowed. The six waveforms are enough to specify the behavior of the latch because the device is simple enough that only *recent* history matters.

The specification of this behavior is given by the LTTL formula of Table 6. Observe that the first three expressions of Table 6 represent assumptions needed for the latch to work correctly and do not necessarily reflect requirements that can

Table 6 Temporal logic formula for a set-reset latch

Expressions	Comments
$\square \neg(s \wedge r)$	No two inputs have value 1 simultaneously
$\square((s \wedge \neg q) \rightarrow ((s \mathcal{U} q) \vee \square s))$	Input s cannot change if s is 1 and q is 0
$\square((r \wedge q) \rightarrow ((r \mathcal{U} \neg q) \vee \square r))$	Input r cannot change if r is 1 and q is 1
$\square(s \rightarrow \diamond q)$	If s is 1, q will eventually be 1
$\square(r \rightarrow \diamond \neg q)$	If r is 1, q will eventually be 0
$\square((\neg q \rightarrow ((\neg q \mathcal{U} s) \vee \square \neg q)))$	Output q rises to 1 only if s becomes 1
$\square((q \rightarrow ((q \mathcal{U} r) \vee \square q)))$	Output q drops to 0 only if r becomes 1

be realized within the circuitry of the latch itself. Care must be taken to insure that the circuitry in which a latch is placed meets those requirements.

Latch states are triples representing values of s , r , and q , respectively. Some examples, corresponding to state sequences depicted by waveforms (a)–(f) in Fig. 11, that satisfy the formula of Table 6 are as follows:

$$\begin{aligned} S_a &: (\langle 000 \rangle, \langle 100 \rangle, \langle 101 \rangle, \langle 001 \rangle, \langle 011 \rangle, \langle 010 \rangle, \langle 000 \rangle, \dots) \\ S_b &: (\langle 000 \rangle, \langle 100 \rangle, \langle 101 \rangle, \langle 001 \rangle, \langle 101 \rangle, \langle 001 \rangle, \dots) \\ S_c &: (\langle 000 \rangle, \langle 010 \rangle, \langle 000 \rangle, \langle 010 \rangle, \langle 000 \rangle, \dots) \\ S_d &: (\langle 001 \rangle, \langle 101 \rangle, \langle 001 \rangle, \langle 011 \rangle, \langle 010 \rangle, \langle 000 \rangle, \dots) \\ S_e &: (\langle 001 \rangle, \langle 101 \rangle, \langle 001 \rangle, \langle 101 \rangle, \langle 001 \rangle, \dots) \\ S_f &: (\langle 001 \rangle, \langle 011 \rangle, \langle 010 \rangle, \langle 000 \rangle, \langle 010 \rangle, \langle 000 \rangle, \dots) \end{aligned}$$

Clearly, infinitely many sequences satisfy the formula of Table 6, so the problem of verifying functionality for sequential circuits appears daunting. However, by means of careful algorithm design, it is sometimes possible to produce such verifications and successes have been reported. In addition, other successful temporal logic systems such as computation tree logic and interval temporal logic have been introduced, along with algorithms for proving theorems in these logics. The reader is referred to [149], ►Combinatorial Optimization Techniques for Network-Based Data Mining for details and citations.

4.5 Bounded Model Checking

Section 4.4 considered solving verification problems of the form $S \models \psi_1 \equiv S \models \psi_2$. If it is desired instead to determine whether there exists an S such that $S \models \psi$ temporal operators can be traded for Boolean variables, the sentence can be expressed as a propositional formula, and a SAT solver applied. The propositional formula must have the following parts:

1. Components which force the property or properties of the time-dependent expression to hold
2. Components which establish the starting state

3. Components which force legal state transitions to occur

In order for the Boolean expression to remain of reasonable size, it is generally necessary to bound the number of time steps in which the time-dependent expression is to be verified, thus the name *bounded model checking*.

As an example, consider a simple 2-bit counter whose outputs are represented by variables v_1 (LSB) and v_2 (MSB). Introduce variables v_1^i and v_2^i whose values are intended to be the same as those of variables v_1 and v_2 , respectively, on the i th time step. Suppose the starting state is the case where both v_1^0 and v_2^0 have value 0. The transition relation is

Current output		Next output
00	:	01
01	:	10
10	:	11
11	:	00

the i th line of which can be expressed as the following Boolean function:

$$(v_1^{i+1} \leftrightarrow \neg v_1^i) \wedge (v_2^{i+1} \leftrightarrow v_1^i \oplus v_2^i).$$

Suppose the time-dependent expression to be proved is as follows:

Can the two-bit counter reach a count of 11 in exactly three time steps?

Assemble the propositional formula having value 1 if and only if the above query holds as the conjunction of the following three parts:

1. *Force the property to hold:*

$$(\neg(v_1^0 \wedge v_2^0) \wedge \neg(v_1^1 \wedge v_2^1) \wedge \neg(v_1^2 \wedge v_2^2) \wedge (v_1^3 \wedge v_2^3)).$$

2. *Express the starting state:*

$$(\neg v_1^0 \wedge \neg v_2^0).$$

3. *Force legal transitions (repetitions of the transition relation):*

$$\begin{aligned} & (v_1^1 \leftrightarrow \neg v_1^0) \wedge (v_2^1 \leftrightarrow v_1^0 \oplus v_2^0) \wedge \\ & (v_1^2 \leftrightarrow \neg v_1^1) \wedge (v_2^2 \leftrightarrow v_1^1 \oplus v_2^1) \wedge \\ & (v_1^3 \leftrightarrow \neg v_1^2) \wedge (v_2^3 \leftrightarrow v_1^2 \oplus v_2^2). \end{aligned}$$

Since $(a \leftrightarrow b) \Leftrightarrow (a \vee \neg b) \wedge (\neg a \vee b)$, the last expression can be directly turned into a CNF expression. Therefore, the entire formula can be turned into a CNF expression and solved with an off-the-shelf SAT solver.

The reader may check that the following assignment satisfies the above expressions:

$$v_1^0 = 0, v_2^0 = 0, v_1^1 = 1, v_2^1 = 0, v_1^2 = 0, v_2^2 = 1, v_1^3 = 1, v_2^3 = 1.$$

It may also be verified that no other assignment of values to v_1^i and v_2^i , $0 \leq i \leq 3$, satisfies the above expressions. Information on the use and success of bounded model checking may be found in [18, 36].

4.6 Combinational Equivalence Checking

The power of bounded model checking is not needed to solve the combinational equivalence checking (CEC) problem which is to verify that two given *combinational circuit* implementations are functionally equivalent. CEC problems are easier, in general, because they carry no time dependency and there are no feedback loops in combinational circuits. It has recently been discovered that CEC can be solved very efficiently, in general [92, 94, 95], by incrementally building a single-output and/inverter graph (AIG), representing the miter [26] of both input circuits, and checking whether the output has value 0 for all combinations of input values. The AIG is built one vertex at a time, working from input to output. Vertices are merged if they are found to be *functionally equivalent*. Vertices can be found functionally equivalent in two ways: (1) candidates are determined by random simulation [26, 93] and then checked by a SAT solver for functional equivalence, and (2) candidates are hashed to the same location in the data structure representing vertices of the AIG (for the address of a vertex to represent function, it must depend solely on the opposite endpoints of the vertex's incident edges, and therefore, an address change typically takes place on a merge). AIG construction continues until all vertices have been placed into the AIG and no merging is possible. The technique exploits the fact that checking the equivalence of two *topologically similar* circuits is relatively easy [65] and avoids testing all possible input-output combinations, which is CoNP -hard.

In CEC the AIG is incrementally developed from gate level representations. For example, Fig. 7a shows the AIG for an *and* gate, Fig. 7b shows the AIG for an *or* gate, Fig. 7c shows the AIG for *exclusive-or*, and Fig. 7d shows the AIG for the adder circuit of Fig. 9. Vertices may take 0-1 values. Values are assigned independently to input vertices. The value of a gate vertex (a dependent vertex) is the product $x_1 x_2$ where x_i is either the value of the non-arrow side endpoint of incoming edge i , $1 \leq i \leq 2$, if the edge is not negated or 1 minus the value of the endpoint if it is negated.

CEC begins with the construction of the AIG for one of the circuits as exemplified in Fig. 12a which shows an AIG and a circuit it is to be compared against. Working from input to output, a node of the circuit is determined to be functionally equivalent to an AIG vertex and is merged with the vertex. The output lines from the merged node become AIG edges outgoing from the vertex involved in the merge. Figure 12b shows the first two nodes of the circuit being merged with the AIG and Fig. 12c shows the completed AIG.

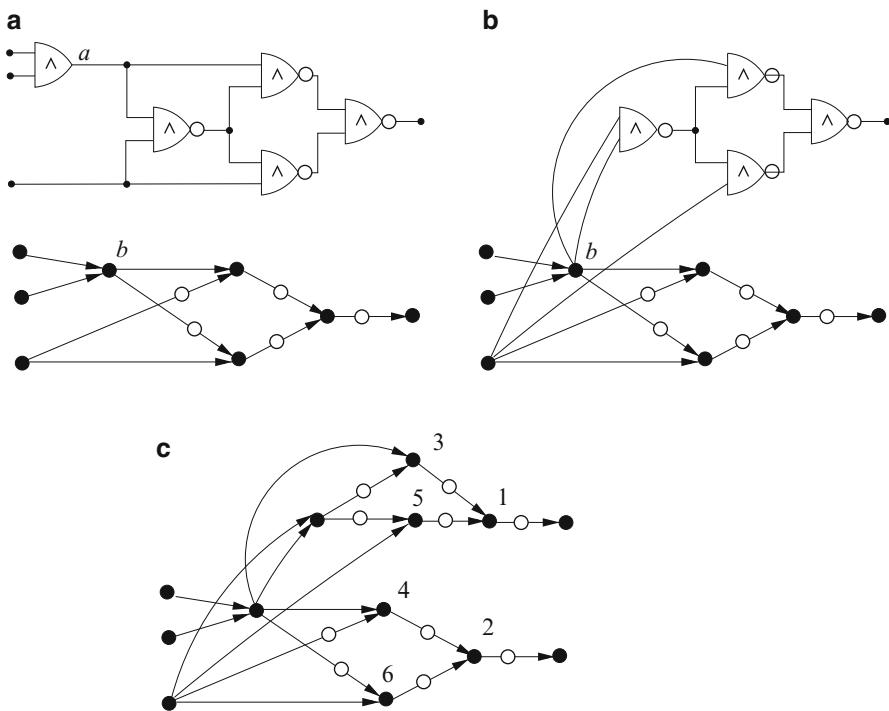


Fig. 12 Creating the AIG. (a) The beginning of the equivalency check – one circuit has been transformed to an AIG. (b) Node a of the circuit (Fig. 12a) has been merged with vertex b of the AIG. (c) The completed AIG

CEC proceeds with the application of many random input vectors to the input vertices of the AIG for the purpose of partitioning the vertices into potential equivalence classes (two vertices are in the same equivalence class if they take the same value under all random input vectors). In the case of Fig. 12c, suppose the potential equivalence classes are $\{1, 2\}$, $\{3, 4\}$, and $\{5, 6\}$.

The next step is to use a SAT solver to verify that the equivalences actually hold. Those that do are merged, resulting in a smaller AIG. The cycle repeats until merging is no longer possible. If the output vertices hash to the same address, the circuits are equivalent. Alternatively, the circuits are equivalent if the AIG is augmented by adding a vertex corresponding to an xor gate with incident edges connecting to the two output vertices of the AIG, and the resulting graph represents an unsatisfiable formula, determined by using a SAT solver.

Above, CEC has been shown to check the equivalence of two combinational circuits, but it can also be used for checking a specification against a circuit implementation or for reverse engineering a circuit.

4.7 Transformations to Satisfiability

It is routine in the operations research community to transform a given optimization problem into another, solve the new problem, and use the solution to construct a solution or an approximately optimal solution for the given problem. Usual targets of transformations are linear programming and network flows. In some cases, where the given problem is \mathcal{NP} -complete, it may be more efficient to obtain a solution or approximate solution by transformation to a satisfiability problem than by solving directly. However, care must be taken to choose a transformation that keeps running time down *and* supports low error rates. In this section a successful transformation from network Steiner tree problems to weighted MAX-SAT problems is considered (taken from [87]).

The network Steiner tree problem originates from the following important network cost problem (see, e.g., [4]). Suppose a potential provider of communications services wants to offer private intercity service for all of its customers. Each customer specifies a collection of cities it needs to have connected in its own private network. Exploiting the extraordinary bandwidth of fiber-optic cables, the provider intends to save cable costs by *piggy-backing* the traffic of several customers on single cables when possible. Assume there is no practical limit on the number of customers piggybacked to a single cable. The provider wants an answer to the following question: *Through what cities should the cables be laid to meet all customer connectivity requirements and minimize total cabling cost?*

This problem can be formalized as follows. Let $G(V, E)$ be a graph whose vertex set $V = \{c_1, c_2, c_3, \dots\}$ represents all cities and whose edge set E represents all possible connections between pairs of cities. Let $w : E \mapsto \mathbb{Z}^+$ be such that $w(\{c_i, c_j\})$ is the cost of laying fiber-optic cable between cities c_i and c_j . Let R be a given set of vertex-pairs $\{c_i, c_j\}$ representing pairs of cities that must be able to communicate with each other due to at least one customer's requirement. The problem is to find a minimum total weight subgraph of G such that there is a path between every vertex-pair of R .

Consider the special case of this problem in which the set T of all vertices occurring in at least one vertex-pair of R is a proper subset of all vertices of G (i.e., the provider has the freedom to use as connection points cities not containing customers' offices) and R requires that all vertices of T be connected. This is known as the network Steiner tree problem. An example and its optimal solution are given in Fig. 13. This problem is one of the first shown to be \mathcal{NP} -complete [85]. The problem appears in many applications and has been extensively studied. Many enumeration algorithms, heuristics, and approximation algorithms are known (see [4] for a list of examples).

A feasible solution to an instance (G, T, w) of the network Steiner tree problem is a tree spanning all vertices of a subgraph of G which includes T . Such a subgraph is called a *Steiner tree*. A *transformation from (G, T, w) to an instance of satisfiability* is a uniform method of encoding of G , T , and w by a CNF formula ψ with nonnegative weights on its clauses. A necessary (feasibility) property of any transformation is that a truth assignment to the variables of ψ which maximizes

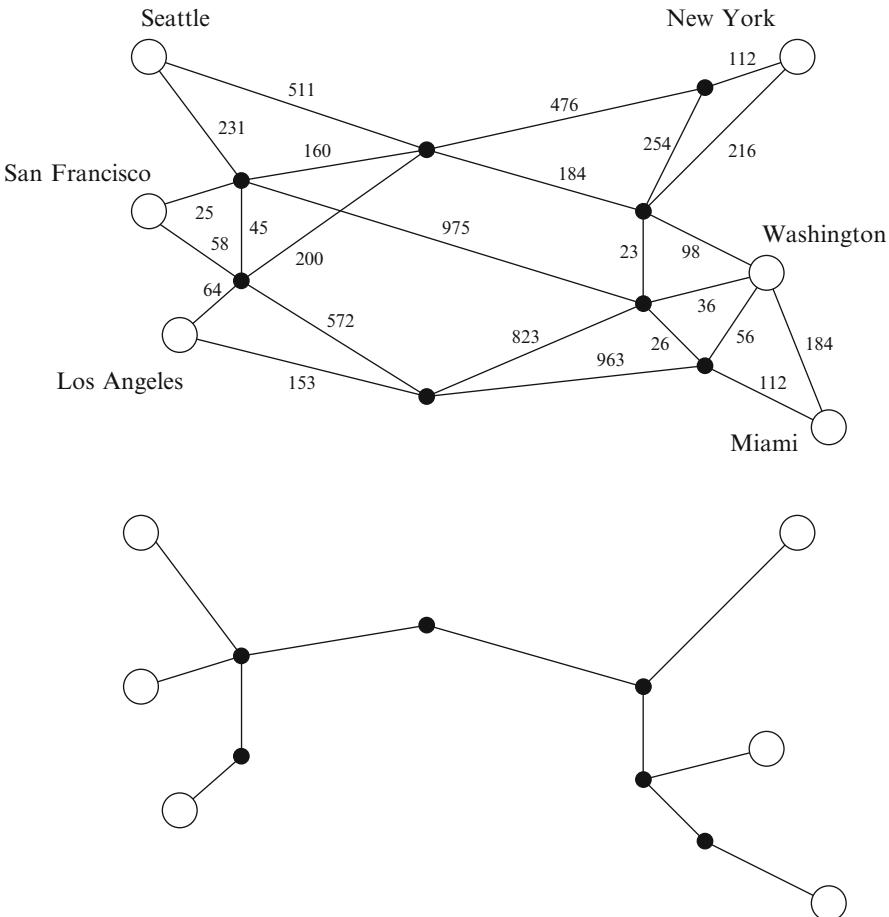


Fig. 13 An example of a network Steiner tree problem (*top*) and its optimal solution (*bottom*). The white nodes are terminals and the numbers represent hypothetical costs of laying fiber-optic cables between pairs of cities

the total weight of all satisfied clauses specifies a feasible solution for (G, T, w) . A desired (optimality) property is, in addition, that feasible solution have minimum total weight. Unfortunately, known transformations that satisfy both properties produce formulas of size that is superlinearly related to the number of edges and vertices in G . Such encodings are useless for large graphs. More practical linear transformations satisfying the feasibility property are possible but these do not satisfy the optimality property. However, it has been demonstrated that, using a carefully defined linear transformation, one can achieve close to optimal results.

As an example, consider the linear transformation introduced in [87]. This transformation has a positive integer parameter k which controls the quality of the

approximation. Without losing any interesting cases, assume that G is connected. It is also assumed that no two paths in G between the same nodes of T have the same weight; this can be made true, if necessary, by making very small adjustments to the weights of the edges.

Preprocessing

1. Define an auxiliary weighted graph $((T, E'), w')$ as follows:
Graph $G' = (T, E')$ is the complete graph on all vertices in T .
For edge $e' = \{c_i, c_j\}$ of G' , let $w'(e')$ be the total cost of the minimum cost path between c_i and c_j in G . (This can be found, e.g., by Dijkstra's algorithm).
2. Let W be a minimum-cost spanning tree of (G', w') . It is intended to choose, for each edge $\{c_i, c_j\}$ of W , (all the edges on) one entire path in (G, w) between c_i and c_j to be included in the Steiner tree. This will produce a Steiner tree for T, G , although perhaps not a minimum-cost tree, as long as no cycles in G are included.

For some prespecified, fixed k : Find the k minimum-cost paths in G between c_i and c_j (again, e.g., by a variation of Dijkstra's algorithm); call them $P_{i,j,1}, \dots, P_{i,j,k}$. Thus, the algorithm will choose one of these k paths between each pair of elements of W .

The Formula

The output of the preprocessing step is a tree W spanning all vertices of T . Each edge $\{c_i, c_j\} \in W$ represents one of the paths $P_{i,j,1}, \dots, P_{i,j,k}$ in G . The tree W , the list of edges in G , the lists of edges comprising each of the k shortest paths between pairs of vertices of W , and the number k are input to the transformation step. The path weights are not needed by the transformation step and are therefore not provided as an input. The transformation is then carried out as shown in [Table 7](#). In the table, E is the set of edges of G and I is any number greater than the sum of the weights of all edges of G .

A maximum weight solution to a formula of [Table 7](#) must satisfy all non-unit clauses because the weights assigned to those clauses are so high. Satisfying those clauses corresponds to choosing all the edges of at least one path between pairs of vertices in the list. Therefore, since the list represents a spanning tree of G' , a maximum weight solution specifies a connected subgraph of G which includes all vertices of T .

On the other hand, the subgraph must be a tree. If there is a cycle in the subgraph, then there is more than one path from a T vertex u to a non- T vertex v , so there is a shorter path that may be substituted for one of the paths going from u through v to some T vertex. Choosing a shorter path is always possible because it will still be one of the k shortest. Doing so removes at least one edge and cycle. This process may be repeated until all cycles are broken and the number of edges remaining is minimal for all T vertices to be connected. Due to the unit clauses, all edge variables whose values are not set to 1 by an assignment add their edge weights to that of the formula. Therefore, the maximum weight solution contains only edges forced by p variables and must specify a Steiner tree.

Table 7 Transformation from an instance of the network Steiner tree problem to a CNF formula. The instance has already been preprocessed to spanning tree W plus lists of k shortest paths $P_{i,j,x}$, $1 \leq x \leq k$, corresponding to edges $\{c_i, c_j\}$ of W . The edges of the original graph are given as set E . Boolean variables created expressly for the transformation are defined at the top and clauses are constructed according to the middle chart. Weights are assigned to clauses as shown at the bottom. The number I is chosen to be greater than the sum of all weights of edges in E

Variable	Subscript range	Meaning
$e_{i,j}$	Edge $\{c_i, c_j\} \in E$	$\{c_i, c_j\} \in$ the Steiner tree
$p_{i,j,l}$	$\{c_i, c_j\} \in W, 1 \leq l \leq k$	$P_{i,j,l} \subseteq$ the Steiner tree
Clause	Subscript range	Meaning
$(\neg e_{i,j})$	$\{c_i, c_j\} \in E$	$e_{i,j} \notin$ Steiner tree
$(p_{i,j,1} \vee \dots \vee p_{i,j,k})$	$\{c_i, c_j\} \in W$	Include at least one of k shortest paths between c_i, c_j
$(\neg p_{i,j,l} \vee e_{m,n})$	$\{c_m, c_n\} \in P_{i,j,l}$	Include all edges of that path
Clause	Weight	Significance
$(\neg e_{i,j})$	$w(\{i, j\})$	Maximize weights of edges <i>not</i> included
$(p_{i,j,1} \vee \dots \vee p_{i,j,k})$	I	Force connected subgraph of G containing all vertices of T
$(\neg p_{i,j,l} \vee e_{m,n})$	I	Force connected subgraph of G containing all vertices of T

A maximum weight solution specifies an optimal Steiner tree only if k is sufficiently large to admit all the shortest paths in an optimal solution. Generally this is not practical since too large a k will cause the transformation to be too large. However, good results can be obtained even with k values up to 30.

Once the transformation to satisfiability is made, an incomplete algorithm such as Walksat (see Sect. 5.8.1) or even a branch-and-bound variant (see Sect. 5.11) can be applied to obtain a solution. The reader is referred to [87] for empirical results showing speed and approximation quality.

4.8 Boolean Data Mining

The field of *data mining* is concerned with discovering *hidden* structural information in databases; it is a form of (or variant of) *machine learning*. In particular, it is concerned with finding hidden correlations among apparently weakly related data. For example, a credit card company might look for patterns of charging purchases that are frequently correlated with using stolen credit card numbers. Using this data, the company can look more closely at suspicious patterns in an attempt to discover thefts before they are reported by the card owners.

Many of the methods for data mining involve essentially numerical calculations. However, others work on Boolean data. Typically, the relevant part of the database consists of a set B of m Boolean n -tuples (plus keys to distinguish tuples, which presumably are unimportant here). Each of these n attributes might be the results of some monitoring equipment or experts' answers to questions.

Some of the tuples in B are known to have some property; the others are known not to have the property. Thus, what is known is a partially defined Boolean function f_B of the n variables. An important problem is to predict whether tuples to be added later will have the same property. This amounts to finding a completely defined Boolean function f which agrees with f_B at every value for which it is defined. Frequently, the goal is also to find such an f with, in some sense, a simple definition: such a definition, it is hoped, will reveal some interesting structural property or explanation of the data points.

For example, binary vectors of dimension two describing the condition of an automobile might have the following interpretation:

Vector	Value	Overheating	Coolant level
00	0	No	Low
01	0	No	Normal
10	0	Yes	Low
11	1	Yes	Normal

where the value associated with each vector is 1 if and only if the automobile's thermostat is defective.

One basic method is to search for such a function f with a relatively short disjunctive normal form (DNF) definition. An example formula in DNF is

$$(v_1 \wedge v_2 \wedge \neg v_3) \vee (\neg v_1 \wedge v_4) \vee (v_3 \wedge v_4 \wedge v_5 \wedge v_6) \vee (\neg v_3).$$

DNF formulas have the same form as that of CNF formulas except that the positions of the \wedge 's and the \vee 's are reversed. In this case the formula is in 4-DNF since each disjunct contains at most 4 conjuncts. Each *term*, for example, $(v_1 \wedge v_2 \wedge \neg v_3)$ is an *implicant*: whenever it is true, the property holds.

There is extensive research upon the circumstances under which, when some random sample points f_B from an actual Boolean function f are supplied, a program can, with high probability, *learn* a good approximation to f within reasonable time (e.g., [140]). Of course, if certain additional properties of f are known or assumed, more such functions f can be determined.

Applications are driving interest in the field. The problem has become more interesting due to the use of binarization which allows nonbinary data sets to be transformed to binary data sets [25]. Such transformations allow the application of special binary tools for cause-effect analysis that would otherwise be unavailable and may even sharpen *explainability*. For example, for a set of 290 data points containing 9 attributes related to Chinese labor productivity, it was observed in [24] that the short clause

Not in the northwest region *and* time **is later than 1987**,

where *time* ranges over the years 1985–1994, explains all the data and the clause

SOE is at least 71.44% and time is earlier than 1988,

where SOE means *state-owned enterprises*, explains 94 % of the data. The technique of logical analysis of data has been successfully applied to inherently nonbinary problems of oil exploration, psychometric analysis, and economics, among others.

5 General Algorithms

In this section some algorithms for solving various SAT problems are presented. Most of these algorithms operate on CNF formulas. Therefore, the following efficient transformation to CNF formulas is needed to demonstrate the general applicability of these algorithms.

5.1 Efficient Transformation to CNF Formulas

An algorithm due to Tseitin [139] efficiently transforms an arbitrary Boolean formula ϕ to a CNF formula ψ such that ψ has a model if and only if ϕ has a model and if a model for ψ exists, it is a model for ϕ . The transformation is important because it supports the general use of many of the algorithms which are described in following sections and require CNF formulas as input.

The transformation can best be visualized graphically. As discussed in Sect. 3.6, any Boolean formula ϕ can be represented as a binary rooted acyclic digraph W_ϕ where each internal vertex represents some operation on one or two operands (an example is given in Fig. 6). Associate with each internal vertex x of W_ϕ a new variable v_x not occurring in ϕ . If x represents a binary operator \mathcal{O}_x , let v_l and v_r be the variables associated with the left and right endpoints, respectively, of the two outward-oriented edges of x (v_l and v_r may be variables labeling internal or leaf vertices). If x represents the operator \neg , then call the endpoint of the outward-oriented edge v_g . For each internal vertex x of W_ϕ , write

$$\begin{aligned} v_x &\Leftrightarrow (v_l \mathcal{O}_x v_r) && \text{if } \mathcal{O}_x \text{ is binary or} \\ v_x &\Leftrightarrow \neg v_g && \text{if vertex } x \text{ represents } \neg. \end{aligned}$$

For each equivalence there is a short, functionally equivalent CNF expression. The table of Fig. 14 shows the equivalent CNF expression for all 16 possible binary operators where each bit pattern in the left column expresses the functionality of an operator for each of four assignments to v_l and v_r , in increasing order, from 00 to 11. The equivalent CNF expression for \neg is $(v_g \vee v_x) \wedge (\neg v_g \vee \neg v_x)$.

The target of the transformation is a CNF formula consisting of all clauses in every CNF expression from Fig. 14 that corresponds to an equivalence expressed at a non-leaf vertex of W_ϕ plus a unit clause that forces the root expression to evaluate to 1. For example, the expression of Fig. 6, namely,

$$v_0 \Leftrightarrow ((\neg v_0 \Leftrightarrow (v_1 \vee \neg v_2)) \wedge (v_1 \vee \neg v_2) \wedge (\neg v_2 \rightarrow \neg v_3 \rightarrow v_4)),$$

\mathcal{O}_x	Equivalent CNF Expression	Comment
0000	$(\neg v_x)$	$v_x \Leftrightarrow 0$
1111	(v_x)	$v_x \Leftrightarrow 1$
0011	$(v_l \vee \neg v_x) \wedge (\neg v_l \vee v_x)$	
1100	$(v_l \vee v_x) \wedge (\neg v_l \vee \neg v_x)$	$v_x \Leftrightarrow \neg v_l$
0101	$(v_r \vee \neg v_x) \wedge (\neg v_r \vee v_x)$	
1010	$(v_r \vee v_x) \wedge (\neg v_r \vee \neg v_x)$	
0001	$(v_l \vee \neg v_x) \wedge (v_r \vee \neg v_x) \wedge (\neg v_l \vee \neg v_r \vee v_x)$	$v_x \Leftrightarrow (v_l \wedge v_r)$
1110	$(v_l \vee v_x) \wedge (v_r \vee v_x) \wedge (\neg v_l \vee \neg v_r \vee \neg v_x)$	$v_x \Leftrightarrow (\neg v_l \vee \neg v_r)$
0010	$(\neg v_l \vee \neg v_x) \wedge (v_r \vee \neg v_x) \wedge (v_l \vee \neg v_r \vee v_x)$	
1101	$(v_l \vee v_x) \wedge (\neg v_r \vee v_x) \wedge (\neg v_l \vee v_r \vee \neg v_x)$	$v_x \Leftrightarrow (v_l \rightarrow v_r)$
0100	$(v_l \vee \neg v_x) \wedge (\neg v_r \vee \neg v_x) \wedge (\neg v_l \vee v_r \vee v_x)$	
1011	$(\neg v_l \vee v_x) \wedge (v_r \vee v_x) \wedge (v_l \vee \neg v_r \vee \neg v_x)$	$v_x \Leftrightarrow (v_l \leftarrow v_r)$
1000	$(\neg v_l \vee \neg v_x) \wedge (\neg v_r \vee \neg v_x) \wedge (v_l \vee v_r \vee v_x)$	$v_x \Leftrightarrow (\neg v_l \wedge \neg v_r)$
0111	$(\neg v_l \vee v_x) \wedge (\neg v_r \vee v_x) \wedge (v_l \vee v_r \vee \neg v_x)$	$v_x \Leftrightarrow (v_l \vee v_r)$
1001	$(v_l \vee \neg v_r \vee \neg v_x) \wedge (\neg v_l \vee v_r \vee \neg v_x) \wedge$ $(\neg v_l \vee \neg v_r \vee v_x) \wedge (v_l \vee v_r \vee v_x)$	$v_x \Leftrightarrow (v_l \leftrightarrow v_r)$
0110	$(\neg v_l \vee v_r \vee v_x) \wedge (v_l \vee \neg v_r \vee \neg v_x) \wedge$ $(v_l \vee v_r \vee \neg v_x) \wedge (\neg v_l \vee \neg v_r \vee \neg v_x)$	$v_x \Leftrightarrow (v_l \oplus v_r)$

Fig. 14 CNF expressions equivalent to $v_x \Leftrightarrow (v_l \mathcal{O}_x v_r)$ for \mathcal{O}_x as shown

transforms to

$$\begin{aligned}
 & (v_0 \vee v_{x_1}) \wedge (\neg v_0 \vee \neg v_{x_1}) \wedge \\
 & (v_2 \vee v_{x_2}) \wedge (\neg v_2 \vee \neg v_{x_2}) \wedge \\
 & (v_3 \vee v_{x_3}) \wedge (\neg v_3 \vee \neg v_{x_3}) \wedge \\
 & (\neg v_1 \vee v_{x_4}) \wedge (\neg v_{x_2} \vee v_{x_4}) \wedge (v_1 \vee v_{x_2} \vee \neg v_{x_4}) \wedge \\
 & (v_{x_3} \vee v_{x_5}) \wedge (\neg v_4 \vee v_{x_5}) \wedge (\neg v_{x_3} \vee v_4 \vee \neg v_{x_5}) \wedge \\
 & (v_{x_1} \vee \neg v_{x_4} \vee \neg v_{x_6}) \wedge (\neg v_{x_1} \vee v_{x_4} \vee \neg v_{x_6}) \wedge (\neg v_{x_1} \vee \neg v_{x_4} \vee v_{x_6}) \wedge \\
 & (v_{x_1} \vee v_{x_4} \vee v_{x_6}) \wedge (v_{x_2} \vee v_{x_7}) \wedge (\neg v_{x_5} \vee v_{x_7}) \wedge (\neg v_{x_2} \vee v_{x_5} \vee \neg v_{x_7}) \wedge \\
 & (v_{x_4} \vee \neg v_{x_8}) \wedge (v_{x_7} \vee \neg v_{x_8}) \wedge (\neg v_{x_4} \vee \neg v_{x_7} \vee v_{x_8}) \wedge \\
 & (v_{x_6} \vee \neg v_{x_9}) \wedge (v_{x_8} \vee \neg v_{x_9}) \wedge (\neg v_{x_6} \vee \neg v_{x_8} \vee v_{x_9}) \wedge \\
 & (v_0 \vee \neg v_{x_9} \vee \neg v_{x_{10}}) \wedge (\neg v_0 \vee v_{x_9} \vee \neg v_{x_{10}}) \wedge (\neg v_0 \vee \neg v_{x_9} \vee v_{x_{10}}) \wedge \\
 & (v_0 \vee v_{x_9} \vee v_{x_{10}}) \wedge (v_{x_{10}}).
 \end{aligned}$$

where each line except the last corresponds to an internal vertex x_i of W_ϕ of Fig. 6, i increasing from 1 in left-to-right and bottom-to-top order, and the new variables labeling those vertices are $v_{x_1}, \dots, v_{x_{10}}$ correspondingly. The last line forces the root expression to evaluate to 1. The algorithm of Fig. 15 expresses these ideas formally.

Algorithm 1.**Transformation to CNF (ϕ)**

```

/* Input  $\phi$ : formula consistent with syntax described on Page 2 */ 
/* Output  $\psi$ : CNF expression functionally equivalent to  $\phi$  */ 
/* Assume variables of  $\phi$  are labeled  $v_0, v_1, \dots, v_{n-1}$  */ 
/* Additional variables  $v_n, v_{n+1}, v_{n+2}, \dots$  are created as needed */ 
/* Locals: stack  $S$ , integer  $i$ , quoted expression  $\phi^q$ , set of clauses  $\phi^c$  */ 

Set  $\phi^q \leftarrow \text{"} \phi \text{"}.$ 
Next  $s \leftarrow \phi^q$ .
Repeat the following while  $\phi^q \neq \emptyset$ :
  If  $s$  is a variable and the symbol at the top of  $S$  is  $\neg$ ,
    Set  $i \leftarrow i + 1$ . // Evaluate ' $\neg s$ '.
    Replace the top of  $S$  with  $v_i$ .
    Append  $L \leftarrow \text{"} v_i \Leftrightarrow \neg s \text{"}$ .
    Next  $s \leftarrow \phi^q$ .
  Otherwise, if  $s$  is ')',
    Pop  $w \leftarrow S$ . // Evaluate '( $v \mathcal{O} w$ )'
    Pop  $\mathcal{O} \leftarrow S$ .
    Pop  $v \leftarrow S$ .
    Pop  $S$ .
    Set  $i \leftarrow i + 1$ .
    Append  $L \leftarrow \text{"} v_i \Leftrightarrow (v \mathcal{O} w) \text{"}$ .
    Set  $s \leftarrow v_i$ .
  Otherwise, // Push an operator, positive variable, or '(' onto  $S$ 
    Push  $S \leftarrow s$ .
    Next  $s \leftarrow \phi^q$ .
Set  $\psi \leftarrow \{\{\text{Pop } S\}\}$ 
Repeat the following while  $L \neq \emptyset$ :
  Pop  $\psi^q \leftarrow L$ .
  If  $\psi^q$  equals " $v_j \Leftrightarrow \neg v_g$ " for some  $v_j$  and  $v_g$ ,
    Set  $\psi \leftarrow \psi \cup \{\{v_j, v_g\}, \{\neg v_j, \neg v_g\}\}$ .
  Otherwise,  $\psi^q$  equals " $v_j \Leftrightarrow (v_l \mathcal{O} v_r)$ " so do the following:
    Build  $\phi^c$  based on  $\mathcal{O}$ ,  $v_j$ ,  $v_l$ ,  $v_r$  using the table of Figure 14.
    Set  $\psi \leftarrow \psi \cup \phi^c$ .
Output  $\psi$ .

```

□

Fig. 15 Algorithm for transforming a formula to a functionally equivalent CNF formula

For simplicity, it is assumed that the input is a formula with syntax consistent with that described on Page 313.

The next lemma and theorems show that the algorithm correctly transforms a given expression to a CNF formula and the size of the target formula is linearly related to the size of the original expression.

Lemma 1 Let \mathcal{O} be any binary Boolean operator except the two trivial ones that evaluate to 0 or 1 regardless of the value of their operands. Given formulas ψ_1 and ψ_2 , variables $v_1 \in V_{\psi_1} \setminus V_{\psi_2}$ and $v_2 \in V_{\psi_2} \setminus V_{\psi_1}$, and some base set V such that $\psi_1|_{v_1=1} \leftrightharpoons_V \psi_1|_{v_1=0}$ and $\psi_2|_{v_2=1} \leftrightharpoons_V \psi_2|_{v_2=0}$,

$$(v_1 \mathcal{O} v_2) \wedge \psi_1 \wedge \psi_2 \leftrightharpoons_V \neg(v_1 \mathcal{O} v_2) \wedge \psi_1 \wedge \psi_2.$$

Proof It must be shown that for every truth assignment M_V to V (1) if there is an extension that satisfies ψ_1 , ψ_2 , and $(v_1 \mathcal{O} v_2)$, then there is an extension that satisfies ψ_1 , ψ_2 , and $\neg(v_1 \mathcal{O} v_2)$ and (2) if there is an extension that satisfies ψ_1 , ψ_2 , and $\neg(v_1 \mathcal{O} v_2)$, then there is an extension that satisfies ψ_1 , ψ_2 , and $(v_1 \mathcal{O} v_2)$. It is only necessary to take care of (1) since a similar argument applies for (2).

Assume there is an extension M_1 that satisfies ψ_1 , ψ_2 , and $(v_1 \mathcal{O} v_2)$. From $\psi_1|_{v_1=1} \leftrightharpoons_V \psi_1|_{v_1=0}$ and $\psi_2|_{v_2=1} \leftrightharpoons_V \psi_2|_{v_2=0}$ and the fact that M_1 satisfies both ψ_1 and ψ_2 , all four extensions identical to M_1 except in v_1 and v_2 will satisfy ψ_1 and ψ_2 . One of those will also satisfy $\neg(v_1 \mathcal{O} v_2)$ since \mathcal{O} is nontrivial. Hence, if there is an extension to M_V that satisfies ψ_1 , ψ_2 , and $(v_1 \mathcal{O} v_2)$, there is also an extension that satisfies ψ_1 , ψ_2 , and $\neg(v_1 \mathcal{O} v_2)$. \square

Theorem 1 Let ϕ be a formula and $V_\phi = \{v_0, v_1, \dots, v_{n-1}\}$ the set of n variables contained in it. The output of [Algorithm 1](#) on input ϕ represents a CNF formula ψ , written (v_0) if ϕ has no operators and otherwise written $(v_i) \wedge \psi_\phi$, $n \leq i$, where clause (v_i) is due to the line “Set $\psi \leftarrow \{\{\text{Pop } S\}\}$ ” and ψ_ϕ is such that $\psi_\phi|_{v_i=1} \leftrightharpoons_{V_\phi} \psi_\phi|_{v_i=0}$.⁴ In addition, $\psi \leftrightharpoons_{V_\phi} \phi$. That is, any truth assignment $M_{V_\phi} \subset V_\phi$ is a model for ϕ if and only if there is a truth assignment $M_1 \subset \{v_n, v_{n+1}, \dots, v_i\}$ such that $M_{V_\phi} \cup M_1$ is a model for ψ .

Proof The output is a set of sets of variables and therefore represents a CNF formula. The line “Set $\psi \leftarrow \{\{\text{Pop } S\}\}$ ” is reached after all input symbols are read. This happens only after either the “Evaluate ‘ $\neg s$ ’” block or the “Push $S \leftarrow s$ ” line. In the former case, v_i is left at the top of the stack. In the latter case, s must be v_0 , in the case of no operators, or v_i from execution of the “Evaluate ‘ $(v \mathcal{O} w)$ ’” block immediately preceding execution of the “Push $S \leftarrow s$ ” line (otherwise, the input is not a formula). Therefore, either v_0 or v_i is the top symbol of S when “Set $\psi \leftarrow \{\{\text{Pop } S\}\}$ ” is executed so $\{v_0\}$ or $\{v_i\}$ is a clause of ψ .

Formulas ψ' and ψ may be shown to have the stated properties by induction on the depth of the input formula ϕ . For improved clarity, \leftrightharpoons is used instead of $\leftrightharpoons_{V_{\phi'}}$ below.

The base case is depth 0. In this case ϕ is the single variable v_0 and $n = 1$. The line “Push $S \leftarrow s$ ” is executed in the first Repeat block, then the line “Set $\psi \leftarrow \{\{\text{Pop } S\}\}$ ” is executed so $\psi = \{\{v_0\}\}$. Since $L = \emptyset$, execution terminates. Obviously, $\psi = \phi$ so both hypotheses are satisfied.

⁴The meaning of \leftrightharpoons is given on Page [318](#).

For the induction step, suppose ϕ is a formula of positive depth $k + 1$ and the hypotheses hold for all formulas of depth k or less. It is next shown that they also hold for ϕ . Consider first the case $\phi = \neg\phi'$ (ϕ' has depth k). [Algorithm 1](#) stacks \neg in the line “Push $S \leftarrow s$ ” and then, due to the very next line, proceeds as though it were operating on ϕ' . The algorithm reaches the same point it would have if ϕ' were input. However, in this case, \neg and a variable are on the stack. This requires one pass through the “Evaluate ‘ $\neg s$ ’” block which adds “ $v_i \Leftrightarrow \neg s$ ” to L and leaves v_i as the only symbol in S . Thus, upon termination,

$$\psi = (v_i) \wedge \psi_\phi = (v_i) \wedge (v_{i-1} \vee v_i) \wedge (\neg v_{i-1} \vee \neg v_i) \wedge \psi_{\phi'}$$

and v_i is not in $\psi_{\phi'}$. Hence,

$$\begin{aligned} \psi_\phi |_{v_i=1} &= ((v_{i-1} \vee v_i) \wedge (\neg v_{i-1} \vee \neg v_i)) |_{v_i=1} \wedge \psi_{\phi'} \\ &= (\neg v_{i-1}) \wedge \psi_{\phi'} \\ &\stackrel{\text{(by the induction hypothesis and Lemma 1, Page 348)}}{=} (v_{i-1}) \wedge \psi_{\phi'} \\ &= ((v_{i-1} \vee v_i) \wedge (\neg v_{i-1} \vee \neg v_i)) |_{v_i=0} \wedge \psi_{\phi'} \\ &= \psi_\phi |_{v_i=0}. \end{aligned}$$

Next, it is shown that $\psi \Leftarrow \phi$ for this case. The expression $(v_i) \wedge \psi_\phi$ can evaluate to 1 only when the value of v_i is 1. Therefore,

$$\begin{aligned} \psi &= (v_i) \wedge \psi_\phi \Leftarrow \psi_\phi |_{v_i=1} \\ &= (\neg v_{i-1}) \wedge \psi_{\phi'} \\ &\stackrel{\text{(from above)}}{=} (v_{i-1}) \wedge \psi_{\phi'} \\ &\stackrel{\text{(by } \psi \Leftarrow \phi \text{ induction hypothesis)}}{=} \phi' \\ &\stackrel{\text{defn}}{=} \phi. \end{aligned}$$

Finally, consider the case that $\phi = (\phi_l \mathcal{O} \phi_r)$ (ϕ_l and ϕ_r have depth at most k). The algorithm stacks a “(” then, by the inductive hypothesis and the recursive definition of a formula, completes operations on ϕ_a which results in ψ_{ϕ_l} in L . The line “Set $\psi \leftarrow \{\{\text{Pop } S\}\}$ ” is avoided because there are still unread input symbols. Thus, there are two symbols on the stack at this point: a “(” which was put there initially and a variable. The symbol \mathcal{O} is read next and pushed on S by the line “Push $S \leftarrow s$.” Then ψ_{ϕ_r} is put in L , but again the line “Set $\psi \leftarrow \{\{\text{Pop } S\}\}$ ” is avoided because there is still a “)” to be read. Thus, the stack now contains “(;” a variable, say, v_l ; an operator symbol \mathcal{O} ; and another variable, say, v_r . The final “)” is read and the “Evaluate ‘($v_l \mathcal{O} v_r$)’” section causes the stack to be popped, $v \Leftarrow (v_l \mathcal{O} v_r)$ to be added to L , and variable v_i to be put on S (in the final iteration of the first loop). Therefore, upon termination,

$$\psi = (v_i) \wedge \psi_\phi = (v_i) \wedge (v_i \Leftrightarrow (v_l \mathcal{O} v_r)) \wedge \psi_{\phi_l} \wedge \psi_{\phi_r},$$

where $(v_l) \wedge \psi_{\phi_l}$ is what the algorithm output represents on input ϕ_l and $(v_r) \wedge \psi_{\phi_r}$ is represented by the output on input ϕ_r (some clauses may be duplicated to exist both in ψ_{ϕ_l} and ψ_{ϕ_r}). Then,

$$\begin{aligned}\psi_\phi|_{v_i=1} &= (v_i \Leftrightarrow (v_l \mathcal{O} v_r))|_{v_i=1} \wedge \psi_{\phi_l} \wedge \psi_{\phi_r} \\ &= (v_l \mathcal{O} v_r) \wedge \psi_{\phi_l} \wedge \psi_{\phi_r} \\ &\Leftarrow \neg(v_l \mathcal{O} v_r) \wedge \psi_{\phi_l} \wedge \psi_{\phi_r} \quad (\text{induction hypothesis and Lemma 1}) \\ &= (v_i \Leftrightarrow (v_l \mathcal{O} v_r))|_{v_i=0} \wedge \psi_{\phi_l} \wedge \psi_{\phi_r} \\ &= \psi_\phi|_{v_i=0}.\end{aligned}$$

It remains to show $\psi \Leftarrow \phi$ for this case. By the induction hypothesis, $\psi_{\phi_l}|_{v_l=1} \Leftarrow \psi_{\phi_l}|_{v_l=0}$. Therefore, for a given truth assignment M_{ψ_ϕ} , if there is an extension such that v_l has value 1 (0) and ψ_{ϕ_l} has value 0, then there is always an extension such that v_l has value 0 (1), respectively, and ψ_{ϕ_l} has value 1. Since, by the induction hypothesis, $\phi_l \Leftarrow (v_l) \wedge \psi_{\phi_l}$, there is always an extension such that ψ_{ϕ_l} has value 1 and v_l has the same value as ϕ_l . The same holds for v_r and ϕ_r . Therefore,

$$\begin{aligned}\psi &= (v_i) \wedge \psi_\phi \Leftarrow \psi_\phi|_{v_i=1} \\ &= ((v_i) \wedge (v_i \Leftrightarrow (v_l \mathcal{O} v_r)))|_{v_i=1} \wedge \psi_{\phi_l} \wedge \psi_{\phi_r} \\ &\Leftarrow (v_l \mathcal{O} v_r) \wedge \psi_{\phi_l} \wedge \psi_{\phi_r} \\ &\Leftarrow (\phi_l \mathcal{O} \phi_r) \Leftarrow \phi.\end{aligned}$$

□

Theorem 2 *Algorithm 1* produces a representation using a number of symbols that is no greater than a constant times the number of symbols the input formula has if there are no double negations in the input formula.

Proof Each binary operator in the input string accounts for a pair of parentheses in the input string plus itself plus a literal. Hence, if there are m binary operators, the string has at least $4m$ symbols.

If there are m binary operators in the input string, m new variables associated with those operators will exist in ψ due to the Append of the “Evaluate ‘ $(v \mathcal{O} w)$ ’” block. For each, at most 12 symbols involving literals, 12 involving operators (both \wedge and \vee represented as “,”), and 8 involving parentheses (represented as “{” “}”) are added to ψ (see Fig. 14), for a total of at most $32m$ symbols. At most $m + n$ new variables associated with negations are added to ψ (both original variables and new variables can be negated) due to the Append of the “Evaluate ‘ $\neg s$ ’” block. For each, at most 4 literals, 4 operators, and 4 parentheses are added to ψ (from $\{v_i, v_j\}, \{\neg v_i, \neg v_j\}$), for a total of $12(m+n)$ symbols. Therefore, the formula output by *Algorithm 1* has at most $44m + 12n$ symbols.

Since $n - 1 \leq m$, $44m + 12n \leq 56m + 12$. Therefore, the ratio of symbols of ψ to ϕ is not greater than $(56m + 12)/4m = 14 + 3/m < 17$. \square

The next result shows that the transformation is computed efficiently.

Theorem 3 *Algorithm 1* has $O(m)$ worst-case complexity if input formulas do not have double negations, where m is the number of operators, and one symbol is used per parenthesis, operator, and variable.

Proof Consider the first Repeat block. Every right parenthesis is read once from the input, is never stacked, and results in an extra loop of the Repeat block due to the line “Set $s \leftarrow v_i$ ” in the first Otherwise block. Since all other symbols are read once during an iteration of the Repeat block, the number of iterations of this block is the number of input symbols plus the number of right parentheses. Since the number of operators (m) must be at least one fifth of the number of input symbols, the number of iterations of the Repeat block is no greater than $10m$. Since all lines of the Repeat block require unit time, the complexity of the block is $O(m)$.

The second Repeat block checks items in L , in order, and for each makes one of a fixed number of substitutions in which the number of variables is bounded by a constant. An item is appended to L every time a \neg or right parenthesis is encountered, and there is one right parenthesis for every binary operator. Thus, there are m items in L and the complexity of the second Repeat block is $O(m)$. Therefore, the complexity of the algorithm is $O(m)$. \square

If ϕ contains duplicate subformulas, the output will consist of more variables and CNF blocks than are needed to represent a CNF formula equivalent to ϕ . This is easily remedied by implementing L as a balanced binary search tree keyed on v_l , v_r , and \mathcal{O} and changing the Append to a tree insertion operation. If, before insertion in the “Evaluate ‘ $(v\mathcal{O}w)$ ’” block, it is discovered that “ $v_j \Leftrightarrow (v\mathcal{O}w)$ ” already exists in L for some v_j , then s can be set to v_j , i need not be incremented, and the insertion can be avoided. A similar change can be made for the “Evaluate ‘ $\neg s$ ’” block. With L implemented as a balanced binary search tree, each query and each insertion would take at most $\log(m)$ time since L will contain no more than m items. Hence, the complexity of the algorithm with the improved data structure would be $O(m \log(m))$.

As a final remark, there is no comparable, efficient transformation from a formula to a DNF formula.

5.2 Resolution

Resolution is a general procedure that is primarily used to certify that a given CNF formula is unsatisfiable, but can also be used to find a model, if one exists. The idea originated as *consensus* in [20] and [129] (ca. 1937) and was applied to DNF formulas in a form exemplified by the following:

$$(x \wedge y) \vee (\neg x \wedge z) \Leftrightarrow (x \wedge y) \vee (\neg x \wedge z) \vee (y \wedge z).$$

Consensus in DNF was rediscovered in [120] and [116] (ca. 1954) where it was given its name. A few years later its dual, as resolution in propositional logic, was applied to CNF (M. Davis and H. Putnam, 1958, Computational methods in the propositional calculus, unpublished report), for example,

$$(x \vee y) \wedge (\neg x \vee z) \Leftrightarrow (x \vee y) \wedge (\neg x \vee z) \wedge (y \vee z).$$

The famous contribution of Robinson [118] in 1965 was to lift resolution to first-order logic.

Let c_1 and c_2 be disjunctive clauses such that there is exactly one variable v that occurs negated in one clause and unnegated in the other. Then, the *resolvent* of c_1 and c_2 , denoted by $\mathcal{R}_{c_2}^{c_1}$, is a disjunctive clause which contains all the literals of c_1 and all the literals of c_2 except for v or its complement. The variable v is called a *pivot* variable. If c_1 and c_2 are treated as sets, their resolvent is $\{l : l \in c_1 \cup c_2 \setminus \{v, \neg v\}\}$.

The usefulness of resolvents derives from the following Lemma.

Lemma 2 *Let ψ be a CNF formula represented as a set of sets. Suppose there exists a pair $c_1, c_2 \in \psi$ of clauses such that $\mathcal{R}_{c_2}^{c_1} \notin \psi$ exists. Then $\psi \Leftrightarrow \psi \cup \{\mathcal{R}_{c_2}^{c_1}\}$.*

Proof Clearly, any assignment that does not satisfy CNF formula ψ cannot satisfy a CNF formula which includes a subset of clauses equal to ψ . Therefore, no satisfying assignment for ψ implies none for $\psi \cup \{\mathcal{R}_{c_2}^{c_1}\}$.

Now suppose M is a model for ψ . Let v be the pivot variable for c_1 and c_2 . Suppose $v \in M$. One of c_1 or c_2 contains $\neg v$. That clause must also contain a literal that has value 1 under M or else it is falsified by M . But that literal exists in the resolvent $\mathcal{R}_{c_2}^{c_1}$, by definition. Therefore, $\mathcal{R}_{c_2}^{c_1}$ is satisfied by M and so is $\psi \cup \{\mathcal{R}_{c_2}^{c_1}\}$. A similar argument applies if $v \notin M$. \square

The *resolution* method makes use of [Lemma 2](#) and the fact that a clause containing no literals cannot be satisfied by any truth assignment. A resolution algorithm for CNF formulas is presented in [Fig. 16](#). It uses the notion of pure literal (Page 315) to help find a model, if one exists. Recall, a literal is pure in formula ψ if it occurs in ψ , but its complement does not occur in ψ . If the algorithm outputs *unsatisfiable*, then the set of all resolvents generated by the algorithm is a *resolution refutation* for the input formula.

Theorem 4 *Let ψ be a CNF formula represented as a set of sets. The output of [Algorithm 2](#) on input ψ is unsatisfiable, if and only if ψ is unsatisfiable. If the output is a set of variables, then it is a model for ψ .*

Proof If the algorithm returns *unsatisfiable*, one resolvent is the empty clause. From [Lemma 2](#) and the fact that an empty clause cannot be satisfied, ψ is unsatisfiable.

If the algorithm does not return “unsatisfiable” the Otherwise block is entered because new resolvents cannot be generated from ψ . Next, a Repeat block to

Algorithm 2.**Resolution (ψ)**

```

/* Input  $\psi$  is a set of sets of literals representing a CNF formula */
/* Output is either “unsatisfiable” or a model for  $\psi$  */
/* Locals: set of variables  $M$  */
Set  $M \leftarrow \emptyset$ .
Repeat the following until some statement below outputs a value:
  If  $\emptyset \in \psi$ , Output “unsatisfiable.”
  If there are two clauses  $c_1, c_2 \in \psi$  such that  $\mathcal{R}_{c_2}^{c_1} \notin \psi$  exists,
    Set  $\psi \leftarrow \psi \cup \{\mathcal{R}_{c_2}^{c_1}\}$ .
  Otherwise,
    Repeat the following while there is a pure literal  $l$  in  $\psi$ :
      If  $l$  is a positive literal, Set  $M \leftarrow M \cup \{l\}$ .
      Set  $\psi \leftarrow \{c : c \in \psi, l \notin c\}$ .
    Arbitrarily index all variables in  $V_\psi$  as  $v_1, v_2, \dots, v_n$ .
    Repeat the following for  $i$  from 1 to  $n$ :
      If  $M_{1:i}$  falsifies some clause in  $\psi$ , Set  $M \leftarrow M \cup \{v_i\}$ .
    Output  $M$ .

```

□

Fig. 16 Resolution algorithm for CNF formulas

remove clauses containing pure literals is executed, followed by a final **Repeat** block which adds some variables to M .

Consider the result of the **Repeat** block on pure literals. All the clauses removed in this block are satisfied by M because all variables associated with negative pure literals are absent from M and all variables associated with positive pure literals are in M . Call the set of clauses remaining after this **Repeat** block ψ' . Now, consider the final **Repeat** block. If $\psi' = \emptyset$, then all clauses are satisfied by M from the previous **Repeat** block, and $M_{1:i}$ will never falsify a clause for any i . In that case M is returned and is a model for ψ . So suppose $\psi' \neq \emptyset$. The only way a clause can be falsified by $M_{1:i}$ is if it is $\{v_i\}$. In that case, the line “Set $M \leftarrow M \cup \{v_i\}$ ” changes M to satisfy that clause. The clause $\{\neg v_1\} \notin \psi'$ because otherwise it would have resolved with $\{v_1\}$ to give $\emptyset \in \psi$ which violates the hypothesis that the pure literal **Repeat** block was entered. Therefore, no clauses are falsified by $M_{1:i}$ at the beginning of the second iteration of the **Repeat** block when $M_{1:2}$ is considered.

Assume the general case that no clause is falsified for $M_{1:i-1}$ at the beginning of the i th iteration of the final **Repeat** block. A clause c_1 will be falsified by $M_{1:i}$ but not by $M_{1:i-1}$ if it contains literal v_i , which cannot be a pure literal that was processed in the previous **Repeat** block. Then, the line “Set $M \leftarrow M \cup \{v_i\}$ ” changes M to satisfy c_1 without affecting any previously satisfied clauses. However, it is possible that a clause that was not previously satisfied becomes falsified by the change. If there were such a clause c_2 , it must contain literal $\neg v_i$ and no other literal in c_2 would have a complement in c_1 ; otherwise, it would already have been satisfied by $M_{1:i-1}$. That means ψ , before the pure literal **Repeat** block, would contain $\mathcal{R}_{c_2}^{c_1}$. Moreover, $\mathcal{R}_{c_2}^{c_1}$ could not be satisfied by $M_{1:i-1}$ because such an assignment

would have satisfied c_1 and c_2 . But then $\mathcal{R}_{c_2}^{c_1}$ must be falsified by $M_{1:i-1}$ because it contains literals associated only with variables v_1, \dots, v_{i-1} . But this is impossible by the inductive hypothesis. It follows that the line “Set $M \leftarrow M \cup \{v_i\}$ ” does not cause any clauses to be falsified and that no clause is falsified for $M_{1:i}$. Since no clause is falsified by $M_{1:n}$, all clauses must be satisfied by M which is then a model. The statement of the theorem follows. \square

Resolution is a powerful tool for mechanizing the solution to variants of SAT. However, in the case of an unsatisfiable input formula, since the resolution algorithm offers complete freedom to choose which pair of clauses to resolve next, it can be difficult to control the total number of clauses resolved and therefore the running time of the algorithm. The situation in the case of satisfiable input formulas is far worse since the total number of resolvents generated can be quite large even when a certificate of satisfiability can be generated quite quickly using other methods.

It is easy to find examples of formulas on which the resolution algorithm may perform poorly or well, depending on the order of resolvents, and one infinite family is presented here. Let there be n *normal* variables $\{v_0, v_1, \dots, v_{n-1}\}$ and $n - 1$ *selector* variables $\{z_1, z_2, \dots, z_{n-1}\}$. Let each clause contain one normal variable and one or two selector variables. For each $0 \leq i \leq n - 1$, let there be one clause with v_i and one with $\neg v_i$. The selector variables allow the clauses to be *chained* together by resolution to form any n -literal clause consisting of normal variables. Therefore, the number of resolvents generated could be as high as $2^{O(n)}$ although the input length is $O(n)$. The family of input formulas is specified as follows:

$$\begin{aligned} & \{\{v_0, \neg z_1\}, \{z_1, v_1, \neg z_2\}, \{z_2, v_2, \neg z_3\}, \dots, \{z_{n-2}, v_{n-2}, \neg z_{n-1}\}, \{z_{n-1}, v_{n-1}\}, \\ & \{\neg v_0, \neg z_1\}, \{z_1, \neg v_1, \neg z_2\}, \dots, \{z_{n-2}, \neg v_{n-2}, \neg z_{n-1}\}, \{z_{n-1}, \neg v_{n-1}\}\}. \end{aligned}$$

The number of resolvents generated by the resolution algorithm greatly depends on the order in which clauses are chosen to be resolved. Resolving vertically, the 0th column adds resolvent $\{\neg z_1\}$. This resolves with the clauses of column 1 to add $\{v_1, \neg z_2\}$ and $\{\neg v_1, \neg z_2\}$, and these two add resolvent $\{\neg z_2\}$. Continuing, resolvents $\{\neg z_3\}, \dots, \{\neg z_{n-1}\}$ are added. Finally, $\{\neg z_{n-1}\}$ resolves with the two clauses of column $n - 1$ to generate resolvent \emptyset showing that the formula is unsatisfiable. The total number of resolutions executed is $O(n)$ in this case. On the other hand, resolving horizontally (resolve one clause of column 0 with a clause of column 1, then resolve the resolvent with a clause from column 2, and so on), all 2^n n -literal clauses of *normal* variables can be generated before \emptyset is.

The number of resolution steps executed on a satisfiable formula can be outrageous. Remove column $n - 1$ from the above formulas. They are then satisfiable. But that is not known until $2^{O(n)}$ resolutions fail to generate \emptyset .

This example may suggest that something is wrong with resolution and that it should be abandoned in favor of faster algorithms. While this is reasonable for satisfiable formulas, it may not be for unsatisfiable formulas. As illustrated by [Theorem 5](#), in the case of providing a certificate of unsatisfiability, the resolution algorithm can always *simulate* the operation of many other algorithms in time

bounded by a polynomial on input length. So the crucial problem for the resolution algorithm, when applied to unsatisfiable formulas, is determining the order in which clauses should be resolved to keep the number of resolvents generated at a minimum. A significant portion of this chapter deals with this question. However, first consider an extension to resolution that has shown improved ability to admit short refutations.

5.3 Extended Resolution

In the previous section it was shown that the size of a resolution refutation can vary enormously depending on the order in which resolvents are formed. That is, for at least one family of formulas, there are both exponentially long and linearly long refutations. But for some families of formulas, nothing shorter than exponential size resolution refutations is possible. However, the simple idea of *extended resolution* can sometimes yield short refutations in such cases.

Extended resolution is based on a result of Tseitin [139] who showed that, for any pair of variables v, w in a given CNF formula ψ , the following expression may be appended to ψ :

$$(z \vee v) \wedge (z \vee w) \wedge (\neg z \vee \neg v \vee \neg w), \quad (3)$$

where z is a variable that is not in ψ . From Fig. 14 this is equivalent to

$$z \leftrightarrow (\neg v \vee \neg w),$$

which means either v and w both have value 1 (then $z = 0$) or at least one of v or w has value 0 (then $z = 1$). Observe that as long as z is never used again, any of the expressions of Fig. 14 can be used in place of or in addition to (3). More generally,

$$z \leftrightarrow f(v_1, v_2, \dots, v_k)$$

can be used as well where f is any Boolean function of arbitrary arity. Judicious use of such extensions can result in polynomial size refutations for problems that have no polynomial size refutations without extension, a notable example being the pigeonhole formulas.

By adding variables not in ψ , one obtains, in linear time, a satisfiability-preserving translation from any propositional expression to CNF with at most a constant factor blowup in expression size as shown in Sect. 5.1.

5.4 Davis-Putnam Resolution

The Davis-Putnam procedure (DPP) [51] is presented here mainly for historical reasons. In DPP, a variable v is chosen and then all resolvents with v as pivot are

generated. When no more such resolvents can be generated, all clauses containing v or $\neg v$ are removed and the cycle of choosing a variable, generating resolvents, and eliminating clauses is repeated to exhaustion. The fact that it works is due to the following result.

Lemma 3 *Let ψ be a CNF formula. Perform the following operations:*

$$\psi_1 \leftarrow \psi.$$

Choose any variable v from ψ_1 .

Repeat the following until no new resolvents with v as pivot can be added to ψ_1 :

If there is a pair of clauses $c_1, c_2 \in \psi_1$ such that $\mathcal{R}_{c_2}^{c_1}$ with v as pivot exists and $\mathcal{R}_{c_2}^{c_1} \notin \psi_1$,

$$\psi_1 \leftarrow \psi_1 \cup \{\mathcal{R}_{c_2}^{c_1}\}.$$

$$\psi_2 \leftarrow \psi_1.$$

Repeat the following while there is a clause $c \in \psi_2$ such that $v \in c$ or $\neg v \in c$:

$$\psi_2 \leftarrow \psi_2 \setminus \{c\}.$$

Then ψ is satisfiable if and only if ψ_2 is satisfiable.

Proof By Lemma 2 ψ_1 is functionally equivalent to ψ . Since removing clauses cannot make a satisfiable formula unsatisfiable, if ψ and therefore ψ_1 is satisfiable, then so is ψ_2 .

Now, suppose ψ is unsatisfiable. Consider any pair of assignments M (without v) and $M \cup \{v\}$. Either both assignments falsify some clause not containing v or $\neg v$ or else all clauses not containing v or $\neg v$ are satisfied by both M and $M \cup \{v\}$. In the former case, some clause common to ψ and ψ_2 is falsified by M , so both formulas are falsified by M . In the latter case, M must falsify a clause $c_1 \in \psi$ containing v , and $M \cup \{v\}$ must falsify a clause containing $\neg v$. Then the resolvent $\mathcal{R}_{c_2}^{c_1} \in \psi_2$ is falsified by M . Therefore, since every assignment falsifies ψ , ψ_2 is unsatisfiable. \square

Despite the apparent improvement in the management of clauses over the resolution algorithm, DPP is not considered practical. However, it has spawned some other commonly used variants, especially the next algorithm to be discussed.

5.5 Davis-Putnam Loveland Logemann Resolution

Here a reasonable implementation of the key algorithmic idea in DPP is presented. The idea is to repeat the following: choose a variable, take care of all resolutions due to that variable, and then erase clauses containing it. The algorithm, called DPLL [52], is shown in Fig. 17. It was developed when Loveland and Logemann attempted to implement DPP but found that it used too much RAM. So they changed the way variables are eliminated by employing the splitting rule: assignments are

Algorithm 3.

```

DPLL ( $\psi$ )
/* Input  $\psi$  is a set of sets of literals representing a CNF formula */ 
/* Output is either “unsatisfiable” or a model for  $\psi$  */ 
/* Locals: integer  $d$ , variable stack  $V_P$ , list of formulas  $\psi_0, \psi_1, \dots$  */ 
/*           partial assignments  $M_{1:d}$ . */ 

Set  $d \leftarrow 0$ ; Set  $M_{1:0} \leftarrow \emptyset$ ; Set  $\psi_0 \leftarrow \psi$ ; Set  $V_P \leftarrow \emptyset$ .
Repeat the following until some statement outputs a value:
  If  $\psi_d = \emptyset$ , Output  $M_{1:d}$ .
  If  $\emptyset \in \psi_d$ ,
    Repeat the following until  $l$  is “tagged.” 
      If  $V_P = \emptyset$ , Output “unsatisfiable.”
      Pop  $l \leftarrow V_P$ .
      Set  $d \leftarrow d - 1$ .
      Push  $V_P \leftarrow \neg l$ . /*  $l$  and  $\neg l$  are not tagged */
      If  $l$  is a negative literal then  $M_{1:d+1} \leftarrow M_{1:d} \cup \{\neg l\}$ .
      Otherwise,  $M_{1:d+1} \leftarrow M_{1:d}$ .
      Set  $\psi_{d+1} \leftarrow \{c - \{\neg l\} : c \in \psi_d, l \notin c\}$ .
      Set  $d \leftarrow d + 1$ .
    Otherwise,
      If there exists a pure literal in  $\psi_d$ ,
        Choose a pure literal  $l$ .
      Otherwise, if there is a unit clause in  $\psi_d$ ,
        Choose a literal  $l$  in a unit clause and “tag”  $l$ .
      Otherwise,
        Choose a literal  $l$  in  $\psi_d$  and “tag”  $l$ .
      Push  $V_P \leftarrow l$ .
      If  $l$  is a positive literal,  $M_{1:d+1} \leftarrow M_{1:d} \cup \{l\}$ .
      Otherwise,  $M_{1:d+1} \leftarrow M_{1:d}$ .
      Set  $\psi_{d+1} \leftarrow \{c - \{\neg l\} : c \in \psi_d, l \notin c\}$ .
      Set  $d \leftarrow d + 1$ .

```

□

Fig. 17 DPLL algorithm for CNF formulas

recursively extended by one variable in both possible ways, looking for a satisfying assignment. Thus, DPLL is of the divide-and-conquer family of algorithms.

The DPLL algorithm of Fig. 17 is written iteratively instead of recursively for better comparison with other algorithms to be discussed later. Omitted is a formal proof of the fact that the output of DPLL on input ψ is *unsatisfiable* if and only if ψ is unsatisfiable, and if the output is a set of variables, then it is a model for ψ . The reader may consult [52] for details.

A common visualization of the flow of control of DPLL and similar algorithms involves a graph structure known as a search tree. Since search trees will be used several times in this chapter to assist in making some difficult points clear, this concept is explained here. A *search tree* is a rooted acyclic digraph where each

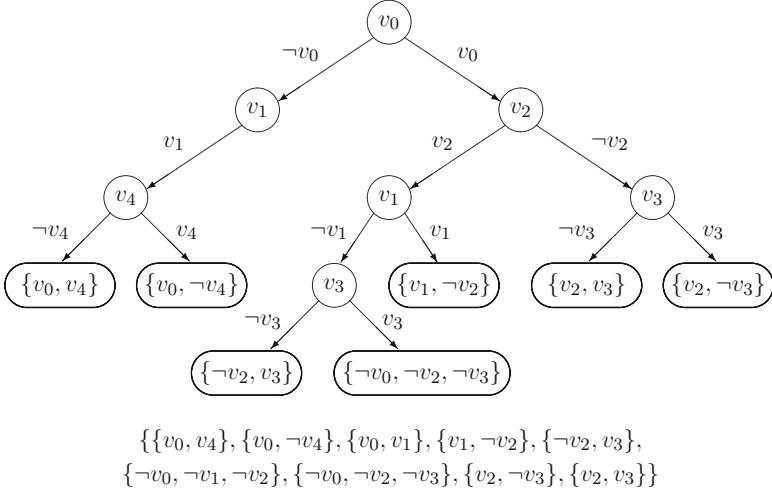


Fig. 18 A DPLL refutation tree for the above CNF formula

vertex has out-degree at most 2 and in-degree 1 except for the root. Each internal vertex represents some Boolean variable and each leaf may represent a clause or may have no affiliation. If an internal vertex represents variable v and it has two outward-oriented edges, then one is labeled v and the other is labeled $\neg v$; if it has one outward-oriented edge, that edge is labeled either v or $\neg v$. All vertices encountered on a path from root to a leaf represent distinct variables. The labels of edges on such a path represent a truth assignment to those variables: $\neg v$ means set the value of v to 0 and v means set the value of v to 1.

The remaining details are specific to a particular CNF formula ψ which is input to the algorithm modeled by the search tree. A leaf is such that the partial truth assignment represented by the path from the root either minimally satisfies all clauses of ψ or minimally falsifies at least one clause of ψ . In the latter case, one of the clauses falsified becomes the label for the leaf. If ψ is unsatisfiable, all leaves are labeled and the search tree is a *refutation tree*. A fully labeled refutation tree modeling one possible run of DPLL on a particular unsatisfiable CNF formula is shown in Fig. 18. With reference to Fig. 17, a path from the root to any vertex represents the state of V_P at some point in the run. For a vertex with out-degree 2, the left edge label is a *tagged literal*. For vertices with out-degree 1, the single edge label is a *pure literal*.

The DPLL algorithm is a performance compromise. Its strong point is that sets of resolvents are constructed incrementally, allowing some sets of clauses and resolvents to be entirely removed from consideration long before the algorithm terminates. But this is offset by the weakness that some freedom in choosing the order of forming resolvents is lost, so more resolvents may have to be generated. Applied to a satisfiable formula, DPLL can be a huge winner over the resolution

algorithm: if the right choices for variable elimination are made, models may be found in $O(n)$ time. However, for an unsatisfiable formula it is not completely clear which algorithm performs best generally. [Theorem 5](#) below shows that if there is a short DPLL refutation for a given unsatisfiable formula, then there must be a short resolution refutation for the same formula.⁵ But, in [7] a family of formulas for which a shortest resolution proof is exponentially smaller than the smallest DPLL refutation is presented. These facts seem to give the edge to resolution. On the other hand, it may actually be relatively easy to find a reasonably short DPLL refutation but very hard to produce an equally short or shorter resolution refutation. An important determining factor is the order in which variables or resolutions are chosen, and the best order is often sensitive to structural properties of the given formula. For this reason, quite a number of choice heuristics have been studied and some results on these are presented later in this chapter.

Many modern SAT solvers augment DPLL with other features such as restarts and conflict-driven clause learning. The refutation complexity of a solver so modified is then theoretically similar to that of resolution. More will be said about this in subsequent sections.

This section concludes with the following classic result.

Theorem 5 *Suppose DPLL is applied to a given unsatisfiable CNF formula ψ and suppose the two lines “Set $\psi_{d+1} \leftarrow \{c - \{\neg l\} : c \in \psi_d, l \notin c\}$ ” are together executed p times. Then there is a resolution refutation in which no more than $p/2$ resolvents are generated.*

Proof Run DPLL on ψ and, in parallel, generate resolvents from clauses of ψ and other resolvents. At most one resolvent will be generated every time the test “If $V_P = \emptyset$ ” succeeds or the line “Pop $l \leftarrow V_P$ ” is executed and l is neither tagged nor a pure literal. But this is at most half the number of times ψ_{d+1} is set. Hence, when DPLL terminates on an unsatisfiable formula, at most $p/2$ resolvents will be generated. Next, it is shown how to generate resolvents and show that \emptyset is a resolvent of two of them.

The idea can be visualized as a series of destructive operations on a DPLL refutation tree. Assume clauses in ψ label leaves as discussed on [Page 358](#). Repeat the following: if there are two leaf siblings, replace the subtree of the two siblings and their parent either with the resolvent of the leaf clauses, if they resolve, or with the leaf clause that is falsified by the assignment represented by the path from root to the parent (there must be one if ψ is unsatisfiable); otherwise, there is a pair of vertices consisting of one leaf and a parent, so replace this pair with the leaf. Replacement entails setting the edge originally pointing to the parent to point to the replacement vertex. Eventually, the tree consists of a root only and its label is the empty clause.

⁵See [39] for other results along these lines.

This visualization is implemented as follows. Define a stack S and set $S \leftarrow \emptyset$. The stack will hold clauses of ψ and resolvents. Run DPLL on ψ . In parallel, manipulate S as follows:

1. The test “If $\emptyset \in \psi_d$ ”:

Whenever this test succeeds, there is a clause $c \in \psi$ whose literals can only be complements of those in V_P so push $S \leftarrow c$.

2. The line “Pop $l \leftarrow V_P$ ”:

Immediately after execution of this line, if l is not *tagged* and is not a pure literal, then do the following. Pop $c_1 \leftarrow S$ and pop $c_2 \leftarrow S$. If c_1 and c_2 can resolve, push $S \leftarrow \mathcal{R}_{c_2}^{c_1}$; otherwise, at least one of c_1 or c_2 , say, c_1 , does not contain l or $\neg l$, so push $S \leftarrow c_1$.

3. The line “If $V_P = \emptyset$, Output ‘unsatisfiable.’”:

The algorithm terminates so pop $c_1 \leftarrow S$, pop $c_2 \leftarrow S$, and form the resolvent $c_t = \mathcal{R}_{c_2}^{c_1}$.

We claim that, when the algorithm terminates, $c_t = \emptyset$.

To prove this claim, it is shown that when c_1 and c_2 are popped in Step 2, c_2 contains literals that are only complementary to those of a subset of $V_P \cup \{l\}$ and c_1 contains literals that are only complementary to those of a subset of $V_P \cup \{\neg l\}$, and if l is a pure literal in the same step, then c_1 contains literals complementary to those of a subset of V_P . Since $V_P = \emptyset$ when c_1 and c_2 are popped to form the final resolvent, c_t , it follows that $c_t = \emptyset$.

By induction on the maximum depth of stacking of c_2 and c_1 . The base case has c_2 and c_1 as clauses of ψ . This can happen only if c_2 had been stacked in Step 1 then Step 2 was executed and then c_1 was stacked soon after in Step 1. It is straightforward to check that the hypothesis is satisfied in this case.

Suppose the hypothesis holds for maximum depth up to k and consider a situation where the maximum depth of stacking is $k + 1$. There are two cases. First, suppose l is a pure literal. Then neither l nor $\neg l$ can be in c_1 so, by the induction hypothesis, all of the literals of c_1 must be complementary to literals of V_P . Second, suppose l is not *tagged* and not a pure literal. Then c_1 and c_2 are popped. By the induction hypothesis, c_1 contains only literals complementary to some in $V_P \cup \{\neg l\}$ and c_2 contains only literals complementary to some in $V_P \cup \{l\}$. Therefore, if they resolve, the pivot must be l and the resolvent contains only literals complementary to some in V_P . If they do not resolve, by the induction hypothesis, one must not contain l or $\neg l$. That one has literals only complementary to those of V_P , so the hypothesis holds in this case. \square

5.6 Conflict-Driven Clause Learning

Early SAT solvers used refinements and enhancements to DPLL to solve SAT instances. Recently, these refinements and enhancements have been standardized and developed into a parameterized algorithm. This algorithm is far enough removed from DPLL that it is commonly referred to as conflict-driven clause learning (CDCL) [119, 130, 131].

The reasons for the increased solving power and efficiency of modern SAT solvers lie in specialized CDCL techniques tailored to support core CDCL operations such as binary constraint propagation (BCP) and heuristic computation. Each technique on its own is not very impressive but, when all of these techniques are fitted together in the DPLL framework, orders of magnitude improvement can be seen. For example, conflict clause learning helps to create a dynamic search space that is DAG-like instead of treelike as was the case for early DPLL-based algorithms [16]. This technique, when used in conjunction with dynamic restarts, is as strong as general resolution and exponentially more powerful than DPLL [16]. The combination of such techniques has allowed SAT solvers to be used practically in a wide range of fields such as bioinformatics [107], cryptography [110, 132], and planning [46, 151] as well as on many hard combinatorial problems such as van der Waerden numbers [100, 101]. Also, the recent integration of SAT-based techniques with theory solvers (this technology is named SMT [126] – details are given in Sect. 5.7) is enabling push-button solutions to industrial strength problems that previously required expert human interaction to solve (e.g., see [77]).

Two chapters in the Handbook of Satisfiability [19] cover CDCL extensively. The reader is referred to [50] and [131] for details. The remainder of this section highlights the main points.

5.6.1 Conflict Analysis

DPLL explores both sides of a choicepoint before backtracking over it. This is called *chronological backtracking* and is not generally conducive to good SAT solving because it is often the case that the same conflict exists on both sides of a choicepoint. Analyzing the conflict on one side may prevent re-computation of the same conflict on the other side, in which case the choicepoint can be *backjumped* over. Performing conflict analysis in the framework of DPLL enables nonchronological backtracking, which, in effect, works to counteract bad decisions made by the search heuristic.

Conflict analysis is often depicted in terms of an implication graph, though, in practice, the implication graph is not actually built. In modern SAT solvers, new clauses are generated by resolution steps ordered by the current search tree path. Specifically, a conflict is reached when two asserting clauses infer opposing literals during BCP. These two clauses can be resolved together to create a new clause called a *conflict clause*. This clause will either be the empty clause (in which case the formula is unsatisfiable) or it will, if evaluated under the current partial assignment, be falsified. Consider the most recently assigned literal in the conflict clause. If the literal was assigned by an asserting clause, the conflict clause and the asserting clause are resolved, creating a new conflict clause, and the process is repeated. If the literal was assigned by a choicepoint, the current search tree up to the literal is undone and the value assigned by the choicepoint is flipped. All new conflict clauses may be safely added to the original CNF formula (i.e., learned) because they are logical consequences of the original formula [130]. Learning conflict clauses helps to prune the search space by preventing re-exploration of shared search structure, especially when used in conjunction with restarts. Also, prior to adding

newly learned clauses, it may be beneficial to attempt to minimize the clauses by performing a restricted version of resolution with other asserting clauses [15, 143].

There is at least one special case to consider when learning conflict clauses. If a conflict clause contains only one literal at the current decision level, called a unique implication point (UIP), the search state can backjump to the second most recently assigned literal in the clause. This will cause the UIP literal to be naturally inferred via BCP [112] and has the effect of promoting those parts of the search tree relevant to the current conflict.

Conflict analysis is also used to generate resolution proofs of unsatisfiable instances. In the same way that an assignment to a satisfiable instance acts as a polynomial-time checkable certificate that an instance is satisfiable, a resolution proof acts as a certificate of unsatisfiability, checkable in linear time and space on the number of learned clauses (though the number of learned clauses can be exponential on the number of variables in the best case). More information on this topic can be found in [15, 150], and [142].

5.6.2 Conflict Clause Memory Management

Not all derived conflict clauses can be stored. Too many conflict clauses increase the overhead of finding an applicable one, and too few conflict clauses reduce the chance that a good one can be found to prune the search space. Therefore, the size of the conflict clause database, what goes into it, and how long entries should stay there have been well studied. The number of conceivable schemes is quite large. A good one was recently discovered and reported in [11]. In that paper the learned clause measure LBD (for literals blocks distance) is defined to be the number of different levels at which literals of a clause were assigned values as choicepoints up to the current assignment. A small LBD suggests a learned clause that was useful in generating a long sequence of unit resolutions. It was found in experiments on all SAT-Race '06 benchmarks that 40 % of unit resolutions occur in clauses of $LBD = 2$ versus 20 % on 2-literal clauses, and generally, the lower the LBD, the more significant the contribution to BCP. Due to this, the following *cleaning strategy* was proposed: remove half of the learned clauses, not including a class of easy discovered clauses of least LBD, every $20000 + 500 * x$ conflicts, where x is the number of times this operation has been performed before on the current input. Adding this strategy to existing solvers resulted in a significant improvement in performance.

Many other ideas for determining the life of conflict clauses have been proposed and tried. Several of these are mentioned in [131] near the end of the chapter.

5.6.3 Lazy Structures

An important aspect of CDCL associated with performance has to do with maintaining data structures that store solver state. The degree to which these structures are managed lazily has an impact on the optimal choice of other solver mechanisms. For example, lazy structures result in faster backtracking, so “the authors of Chaff proposed to always stop clause learning at the first UIP” and “always take the backtrack step” [131]. In addition, the implementation of BCP and conflict analysis

is influenced by the implementation of lazy structures. Using lazy structures, the fact that a clause becomes satisfied by a state change may not be reported to solver mechanisms immediately or even at all. This presents potential savings by preventing some unnecessary computation and does not affect the correctness of the solver. Next, in this section the classic notion of watched literals [112] is used to illustrate this point.

Suppose a clause is stored as a sequence of literals and suppose the number of literals is at least 2. For each clause, assign two pointers each pointing to one of the literals but both pointing to different literals of the clause. The literals pointed to are called *watched literals* and are distinguished in name as the *first* and *second* watched literal. The following properties are maintained always:

1. A clause is satisfied if and only if its first watched literal has value 1.
2. A clause is falsified if and only if its first and second watched literals have value 0.
3. A clause is *unit* if and only if one watched literal is unassigned and the other has value 0.
4. A clause is unassigned and not *unit* if and only if both watched literals are unassigned.

Initially, a clause is unassigned and the pointers are pointing to arbitrary literals. If a clause is unassigned, not *unit*, and a literal other than the second watched literal takes value 1, then that literal becomes the first watched literal. If the literal taking value 1 is the second watched literal, the first and second watched literals are swapped. If both watched literals are unassigned and one of them takes value 0, a check must be made to determine whether the clause has another unassigned literal. If it does not, the clause becomes a unit clause and the single unassigned literal takes value 1 and becomes the first watched literal swapping title, if necessary, with the watched literal of value 0. Otherwise, the pointer referencing the recently assigned watched literal gets set to point to the unassigned literal that was found during the check. The search for an unassigned literal can proceed in any direction among the literals of the clause. Observe that the watched literal pointers do not have to be reset on backtrack. Further savings are obtained by associating a *level* and *value* field with every literal: when a clause is satisfied or falsified, it is the level and value of the first watched literal that is set. Since watched literals do not change during backtrack, this information represents clause status as well. Earlier schemes for managing clause status during backtrack associate level and value fields with clauses and require searching the literals of a clause to determine status. Thus, implementing the watched literal concept results in much less backtracking overhead and significantly increases the number of backtracks handled per second.

As can be observed from the above discussion, watched literals are useful in reacting to single-variable inferences. But more advanced techniques look at multivariable inferences, for example, two-variable equivalence. Watched literals cannot help in this case. Additional clause references, of course, may be used to support these advances [141] at added expense. Another idea [106] is to maintain an order of assigned literals in a clause by nondecreasing level, while the unassigned literals remain watched as above. As a variable is assigned a value, it is *sifted* into

its proper place in the ordered list. The problem with this idea is that all literals in the ordered list may need to be visited on a backtrack.

5.6.4 CDCL Heuristics

Lazy data structures can drastically increase the speed of BCP, but since the current state of the search is not readily available, traditional heuristics such as MOMs (see [60] for a complete discussion), Jeroslow-Wang [80], and Johnson [81] cannot be used. The first CDCL heuristic is the variable state independent decaying sum (VSIDS) [112] heuristic. VSIDS maintains one counter per literal in the SAT instance, initialized to the number of clauses each literal occurs in. VSIDS branches on the literal with the highest count, though this requires sorting the counters, which is expensive, and so is only done periodically. When a new clause is learned, the counts for each literal in the clause are incremented. Also periodically, the counters are halved. This heuristic and many similar variants efficiently glean information from learned clauses to guide the solver towards unsatisfiable cores, enabling modern solvers to solve large SAT instances (millions of variables and clauses) that have small resolution proofs.

5.6.5 Restarts

Conflict-directed heuristics can sometimes cause a solver to focus too heavily on one particular region of the search space, leading to heavy-tailed runtime distribution (where progress seems to exponentially decrease the longer a solver runs). One way to positively modify this behavior [68] is to restart. A *dynamic restart* [76, 88] clears a solver’s current partial assignment; heuristic information and learned clauses are kept, and a short preprocessing phase where the learned clause database is garbage collected may typically follow a restart. As reported empirically in the literature, “it is advantageous to restart often” [50].

5.7 Satisfiability Modulo Theories

Despite highly significant and broad advances to SAT solver design, there are still many problems for which DPLL and CDCL variants are a challenge. For example, it is well known that verifying correctness of arithmetic circuits is generally difficult for SAT solvers. The problem is that datapath operations are over a fixed bit-width and, beyond modular arithmetic, some operations such as multiplication are not linear and particularly vexing for SAT solvers. An accepted method, called *bit-blasting*, for dealing with small bit-widths is to describe operations as propositional formulas over vectors of Boolean variables, convert to CNF using the results of Sect. 5.1, and solve with a SAT solver. But it is doubtful that bit-blasting is going to be universally practical for 64-bit or even 32-bit logic. As another, more universally occurring example, SAT solvers have a difficult time handling cardinality constraints and, more generally, constraints comparing numeric expressions. To mitigate these and other problems, a significant body of research has studied the use of a SAT solver to manage a collection of special purpose solvers for first order theories,

Fig. 19 An SMT instance with uninterpreted functions and quantifiers

```
(set-evidence! true)
(define f::(→ int int))
(define g::(→ int int))
(define a::int)
(assert (forall (x:int) (= (f x) x)))
(assert (forall (x:int) (= (g (g x)) x)))
(assert (/= (g (f (g a))) a))
(check)
```

Fig. 20 An SMT instance with bit-vector expressions

```
(set-evidence! true)
(define b1::(bitvector 8))
(define b2::(bitvector 8))
(define v1::(bitvector 8))
(assert (= v1 (mk-bv 8 39)))
(define v2::(bitvector 8))

(assert (= (bv-mul b1 (mk-bv 8 17)) b2))
(assert (= (bv-add (bv-shift-right0 b2 2) v2) v1))
(check)
```

including bit-vector arithmetic. This extension of SAT has found applications in hardware verification, software model checking and testing, and vulnerability detection, among other applications, and is called *Satisfiability Modulo Theories* or SMT.

A formal specification of SMT is not given here; the reader is invited to check [12] for the syntax and semantics of SMT-LIB, the official documented language of SMT solvers. Most importantly, with respect to SAT, an instance of SMT is a formula in first-order logic where uninterpreted functions are admitted and Boolean variables are used to tie predicates from disparate theories and allow a SAT solver to decide the consistency of the constraints of the instance. The scope of SMT is made apparent through the SMT examples shown in Figs. 19 and 20. The examples are expressed in a lisp-like language that is native to YICES [55], one of the leading available SMT solvers, and may actually be input to YICES.

In Fig. 19, f and g are uninterpreted functions: note that only their signatures, namely, a single `int` as input and an `int` as output, are defined in the second and third lines. However, the fifth and sixth lines force $f(x) = x$, $g(y) = x$, and $g(x) = y$ for all integers. This is inconsistent with the seventh line which requires, for any integer a , that $g(f(g(a)))$, which is $g(g(a))$ and therefore a from the above, is not equal to a . On this input YICES returns `unsat`. If $/=$ (not equal) is changed to $=$, YICES returns `unknown` with the following interpretation: $a = 2$, $g(2) = 1$, $f(1) = 1$, $g(1) = a2$. The first and eighth lines are YICES-specific directives.

Figure 20 shows constraints expressed in terms of bit-vector operations. This instance is satisfied by 8-bit numbers $b1$ and $v2$ such that multiplying $b1$ by 17, shifting right by 2 bits, and adding $v2$ result in 39.

YICES output on this example is as follows:

```
sat
(= v1 0b00100111)
(= b1 0b01110110)
(= b2 0b11010110)
(= v2 0b11110010)
```

That is, $b1$ is 118 and $v2$ is 242, which is one of many possible solutions.

Clearly, admitting bit-vector operations is important as it makes circuit design and verification problems easier to express and solve: trying to express the constraints of such problems as a CNF formula is unnatural and time-and-space consuming. A special solver for the theory of bit-vector arithmetic and logic may be developed to handle bit-vector predicates such as those in Fig. 20. Other theory solvers may be designed to handle other problems that are difficult for CNF solvers. Examples of such theories, including the theory of bit-vectors, are as follows:

1. *Linear arithmetic (LA)*: constraints are equations or inequalities involving addition and subtraction over rational variables, constants, or constants times variables.
2. *Uninterpreted functions (UF)*: all uninterpreted function, constant, and predicate symbols together with Leibniz's law that allows substitution of equals for equals.
3. *Difference arithmetic*: this is linear arithmetic restricted to equations or inequalities of the form $x - y \leq c$ where x and y are variables and c is a constant.
4. *Nonlinear arithmetic*: constraints are like linear arithmetic except that variables may be multiplied by other variables.
5. *Arrays*: array constraints ensure that the value of a memory location does not change unless a specific *write* operation on that location with a different value is executed and that a value written to a location may be accessed by a read operation. Symbols of this theory might be *select* and *store* with semantics given by the following:

```
select(store(v, i, e), j) = if  $i = j$  then  $e$  else select( $v, j$ )
store( $v, i, \text{select}(v, i)$ ) =  $v$ 
store(store( $v, i, e$ ),  $i, f$ ) = store( $v, i, f$ )
 $i \neq j \Rightarrow \text{store}(\text{store}(v, i, e), j, f) = \text{store}(\text{store}(v, j, f), i, e)$ ,
```

where *select*(v, j) means read the j th value from array v and *store*(v, i, e) means store the value e into the i th element of array v .

6. *Lists*: symbols of this theory might be *car*, *cdr*, *cons*, and *atom* with semantics given by the following:

```
car(cons( $x, y$ )) =  $x$ 
cdr(cons( $x, y$ )) =  $y$ 
\neg atom( $x$ ) \Rightarrow cons(car( $x$ ), cdr( $x$ )) =  $x$ 
\neg atom(cons( $x, y$ ))
```

7. *Fixed-width bit-vectors*: the main reason for including a bit-vector theory is to remove the burden of solving bit-level formulas from the SAT solver. As seen in Fig. 20, constraints are expressed at word level and may involve any machine-level operations on words.
8. *Recursive data types*: this is an abstraction for common data structures that may be defined recursively such as linked lists and for common data types that may be defined recursively such as a tree or the natural numbers.

It is not within the scope of this chapter to discuss the construction of decision procedures for theories. The interested reader may begin a search for such information by consulting [13]. The remainder of this section will show how a SAT solver can be made to interact with constraints from several theories.

Some definitions are given now to make the procedures to be discussed later easier to express and analyze. A *first-order formula* is an expression consisting of a set V of variables and symbols $(,)$, \neg , \vee , \wedge , \rightarrow , \exists , \forall ; a set \mathcal{P} of predicate symbols; a set \mathcal{F} of function symbols, each with associated positive arity; and a set $C \subset \mathcal{F}$ of constant symbols which are just 0-arity function symbols, for example, x or f . A *signature*, denoted Σ , is a subset of $\mathcal{F} \cup \mathcal{P}$. A *first-order structure* \mathcal{A} specifies a semantics for the above: that is, a domain A of elements, mappings of every n -ary function symbol $f \in \Sigma$ to a total function $f^{\mathcal{A}} : A^n \mapsto A$ and mappings of every n -ary predicate symbol $p \in \Sigma$ to a relation $p^{\mathcal{A}} \subseteq A^n$. The symbols \exists and \forall are quantifiers and the scope of one of these is the entire formula. A *sentence* is a formula all of whose variables are bound by a quantifier (i.e., the formula has no free variables) and therefore evaluates either to *true* or *false*, depending on the structure it is evaluated over. A formula is a *ground formula* if it has no variables. Observe that $x + 1 \leq y$ has no variables because x and y are constants in the theory of linear arithmetic and is therefore a ground formula. Denote $(\mathcal{A}, \alpha) \models \phi$ to mean the sentence ϕ evaluates to *true* in structure \mathcal{A} under variable assignment $\alpha : V \mapsto A^{|V|}$. Sentence ϕ is said to be *satisfiable* in \mathcal{A} if $(\mathcal{A}, \alpha) \models \phi$ for some α . Sentence ϕ is said to be *valid* in \mathcal{A} if $(\mathcal{A}, \alpha) \models \phi$ for every α . If \mathcal{A} is finite, the question $(\mathcal{A}, \alpha) \models \phi$ for some α and given ϕ is decidable. However, for infinite structures, the question may be undecidable.

A *theory* T is a set of first-order sentences expressed in the language of some signature Σ . Equality is implicit in the theories mentioned earlier so their signatures do not contain the symbol $=$. A *model* of T is a structure \mathcal{A} in which every sentence of T is valid. Denote by $T \models \phi$ the condition where \mathcal{A} is a model for T and $(\mathcal{A}, \alpha) \models \phi$ for all α . Then ϕ is said to be *T -valid*. Let \perp denote a formula that is not satisfied by any structure. Then ϕ is said to be *T -satisfiable* if $T \cup \{\phi\} \not\models \perp$, also written $T, \phi \not\models \perp$. T -validity and T -satisfiability are seen as dual concepts since $T, \neg\phi \models \perp$ if and only if $T \models \phi$.

One is generally interested in combining theories: that is, proving that statements with terms from more than one theory are valid over those combined theories. Suppose, for two theories T_1 of Σ_1 and T_2 of Σ_2 , there exist procedures P_1 and P_2 that determine whether a given formula ϕ_1 expressed in the language of Σ_1 is T_1 -valid and whether a given formula ϕ_2 expressed in the language of Σ_2 is T_2 -valid,

respectively. It is generally not possible to solve the problem of determining whether $\{\phi_1\} \cup \{\phi_2\}$ in the language of $\Sigma_1 \cup \Sigma_2$ is $(T_1 \cup T_2)$ -valid with P_1 and P_2 . However, it can be under some modest restrictions that usually do not get in the way in practice. These restrictions will be assumed for the rest of this section and are as follows: theories T_1 and T_2 are decidable, $\Sigma_1 \cap \Sigma_2 = \emptyset$, and T_1 and T_2 are stably infinite – let C be an infinite set of constants not in signature Σ and define a theory T of Σ to be *stably infinite* if every T -satisfiable ground formula of signature $\Sigma \cup C$ is satisfiable in a model of T with an infinite domain of elements.

By the duality noted above, the validity problem can be treated as a satisfiability problem where it is more conveniently solved. Suppose ϕ is a formula that is logically equivalent to a DNF formula with m conjuncts $\phi_{i,1} \wedge \phi_{i,2}$, $i \in \{1, 2, \dots, m\}$, where $\phi_{i,1}$ and $\phi_{i,2}$ are expressed in the language of $\Sigma_1 \cup C$ and $\Sigma_2 \cup C$, respectively, and C is a set of constants that are shared by $\phi_{i,1}$ and $\phi_{i,2}$. Terms $\phi_{i,1} \wedge \phi_{i,2}$ are said to be in pure form. Clearly, ϕ is $(T_1 \cup T_2)$ -unsatisfiable if and only if for all $i \in \{1, 2, \dots, m\}$, $\phi_{i,1} \wedge \phi_{i,2}$ is $(T_1 \cup T_2)$ -unsatisfiable. It is desirable to use P_1 and P_2 individually to determine this. By Craig's interpolation lemma [45], a pure-form term $\phi_{i,1} \wedge \phi_{i,2}$ is $(T_1 \cup T_2)$ -unsatisfiable if and only if there is a formula ψ , called an *interpolant*, whose free variables are a subset of the free variables contained in both $\phi_{i,1}$ and $\phi_{i,2}$ such that $T_1, \phi_{i,1} \models \psi$ and $T_2, \phi_{i,2}, \psi \models \perp$. Finding an interpolant is a primary goal of an SMT solver and the subject of much of the remainder of this section, the exposition of which is strongly influenced by Klaedtke [90]. Although combining only two theories is considered here, the process generalizes to more than two theories.

The first step in combining theories is to produce a pure-form formula from ϕ . This is accomplished by introducing variables to C which remove the dependence on objects from an *alien* theory through substitution. More specifically, suppose function $f \in \Sigma_1$ and one of its arguments t is a term, possibly a constant, expressed in Σ_2 . Then create a new w , set $C = C \cup \{w\}$, add a new constraint $w = t$, and replace t by w in the argument list of f . Similarly, for predicates $p \in \Sigma_1$, $p \in \Sigma_2$ and function $f \in \Sigma_2$. If there is a constraint $s = t$ where $s \in \Sigma_1$ and $t \in \Sigma_2$, then create a new w , and replace the constraint with $w = s$ and $w = t$. If there is a constraint $s \neq t$, then add a new w_1 and new w_2 and replace the constraint with $w_1 = s$, $w_2 = t$, and $w_1 \neq w_2$. For example, consider

$$\phi = \{x \leq y, y \leq x + \text{car}(\text{cons}(0, x)), p(h(x) - h(y)), \neg p(0)\},$$

which mixes uninterpreted functions, linear arithmetic, and lists. For this ϕ , $C = \{x, y\}$. The constraint on the left need not be touched because it only contains symbols from linear arithmetic. The subterm $h(x) - h(y)$ is mixed, so create new constants $g_1 = h(x)$ and $g_2 = h(y)$ and substitute $g_1 - g_2$ for $h(x) - h(y)$. The term $p(g_1 - g_2)$ is mixed, so create constant $g_3 = g_1 - g_2$ and substitute. The remaining term $p(g_3)$ has no *alien* subterms. The symbol 0 is alien to the theory of uninterpreted functions. Hence, create $g_4 = 0$ and substitute making the rightmost constraint $\neg p(g_4)$, and changing the list theory subterm to $\text{car}(\text{cons}(g_4, x))$, both

of which may be left alone. However, the second constraint from the left is still mixed, so create $g_5 = \text{car}(\text{cons}(g_4, x))$ and substitute. This leaves

$$\begin{aligned} C &= \{x, y, g_1, g_2, g_3, g_4, g_5\}, && (\text{Shared constants}) \\ \phi_1 &= \{x \leq y, y \leq x + g_5, g_3 = g_1 - g_2, g_4 = 0\}, && (\text{LA}) \\ \phi_2 &= \{g_1 = h(x), g_2 = h(y), p(g_4) = \text{false}, p(g_3) = \text{true}\}, && (\text{UF}) \\ \phi_3 &= \{g_5 = \text{car}(\text{cons}(g_4, x))\}, && (\text{Lists}) \end{aligned}$$

so ϕ has become a pure formula whose constraints are partitioned according to the theories of linear arithmetic, uninterpreted functions, and lists and are tied by newly created constants and equality constraints.

The next step is to propagate inferred equalities between the partitioned sections. First, this is illustrated using the above example, then a general procedure for propagation is presented. By the first axiom of the theory of lists, the constraint $g_5 = \text{car}(\text{cons}(g_4, x))$ infers $g_4 = g_5$. This propagates to section LA where $g_5 = 0$ is inferred. Then $y \leq x$ is inferred and $x = y$ is inferred from $x \leq y$ and $y \leq x$. The inferred constraint $x = y$ propagates to section UF where $g_1 = g_2$ is inferred. Propagating back to LA infers $g_3 = 0$ and finally $g_3 = g_4$. But this contradicts UF constraints which say $p(g_3) = \text{true}$ and $p(g_4) = \text{false}$. Hence, the conclusion is that ϕ is unsatisfiable in the combined theories of linear arithmetic, lists, and uninterpreted functions. Observe that ϕ_1 , ϕ_2 , and ϕ_3 are all satisfiable in their respective theories.

[Figure 21](#) shows an algorithm for determining satisfiability of a formula $\phi_1 \wedge \phi_2$ that is in pure form with ϕ_1 of signature $\Sigma_1 \cup C$, ϕ_2 of signature $\Sigma_2 \cup C$, C a set of shared constants, and $\Sigma_1 \cap \Sigma_2 = \emptyset$ and assumes there exist decision procedures P_1 and P_2 for theories T_1 and T_2 over Σ_1 and Σ_2 , respectively. It is straightforward to generalize this to any number of theories; it is also possible to use the generalized algorithm to decide any mixed theory formula by converting to DNF in pure form and applying the algorithm to the conjuncts separately. Step 1 applies decision procedures P_1 and P_2 to ϕ_1 and ϕ_2 . Both P_1 and P_2 not only decide whether their respective theories are consistent but also infer the maximum number of equalities from ϕ_1 and ϕ_2 within their respective theories. If no inconsistencies are found, these inferences are propagated out of their theories and the step is repeated; otherwise, *unsatisfiable* is returned. Step 2 does a case split in case a disjunction of equalities is inferred by one of the partitioned sets of constraints. Step 3 returns “satisfiable” if nothing else can be inferred and no inconsistencies between current sets ϕ_1 and ϕ_2 have been discovered. The algorithm always terminates since the number of shared constants is finite and the number of equalities added is directly related to that number. The algorithm closely follows the algorithm of the original Nelson-Oppen paper [113].

Completeness of [Algorithm 4](#) is not difficult to see. Soundness is proved by observing that the algorithm produces an interpolant for ϕ_1 and ϕ_2 : namely, the residue of $\phi_1 \wedge \phi_2$. The residue of a formula ϕ , denoted $\text{Res}(\phi)$, is the strongest collection of equalities between constants that are inferred by the formula. The important property of residues is that if ϕ_1 and ϕ_2 are formulas whose signatures

Algorithm 4.

```

CTSolver ( $\phi_1, \phi_2, P_1, P_2$ )
/* Input: formulas  $\phi_1$  over  $\Sigma_1 \cup C$ ,  $\phi_2$  over  $\Sigma_2 \cup C$ ,
   where  $\Sigma_1 \cap \Sigma_2 = \emptyset$ , and decision procedures
    $P_1, P_2$  for theories  $T_1$  over  $\Sigma_1$  and  $T_2$  over  $\Sigma_2$  */
/* Output: satisfiable if  $\phi_1 \wedge \phi_2$  is  $(T_1 \cup T_2)$ -satisfiable,
   otherwise unsatisfiable */

```

1. Apply P_1 to ϕ_1 , and apply P_2 to ϕ_2 collecting all inferences of equality made within T_1 and T_2 . If ϕ_1 or ϕ_2 is unsatisfiable, return unsatisfiable. Otherwise, if there is an inferred equality between constants $u = v$ that is not inferred by one of ϕ_1 or ϕ_2 , then add $u = v$ to the set that does not infer it and repeat this step.
2. If ϕ_1 or ϕ_2 infers a disjunction of equalities between constants without inferring any of the equalities alone, then, for every such equality $u = v$ inferred by ϕ_1 , call **CTSolver**($\phi_1, \phi_2 \cup \{u = v\}, P_1, P_2$) and for every such equality $u = v$ inferred by ϕ_2 , call **CTSolver**($\phi_1 \cup \{u = v\}, \phi_2, P_1, P_2$). If, for every inferred equality both calls to **CTSolver** return unsatisfiable, then return unsatisfiable, otherwise return satisfiable.
3. If neither ϕ_1 nor ϕ_2 infers an equality or a disjunction of equalities, then return satisfiable.

□

Fig. 21 Combined theory solver for two theories

overlap only in constants, then $\text{Res}(\phi_1 \wedge \phi_2)$ is logically equivalent to $\text{Res}(\phi_1) \wedge \text{Res}(\phi_2)$. Thus, the residue of the conjunction of all formulas derived within each theory is the conjunction of the residues of formulas for each theory. The algorithm returns “satisfiable” only if all theories are consistent and no additional equalities of constants can be inferred. In this case the residues of all theories are satisfiable. It may be observed that the conjunction of the residues is therefore satisfiable. Hence, the residue of the conjunction of theories is satisfiable. This means if the algorithm returns satisfiable, then $\phi_1 \wedge \phi_2$ is satisfiable.

The purpose of this section is to acquaint the reader with procedures that comprise SMT solver implementations. As such, much has been left out. This chapter has not considered elimination of the purification step by shifting the burden of handling alien terms to the theory solvers themselves, application of theory specific rewrites, Shostak’s theories and method, and how to lift the requirement of stably infinite theories for completeness by propagating cardinality constraints in addition to equalities. Nelson-Oppen only works for quantifier free formulas, but there are some techniques for allowing quantifiers which are not discussed here. For more information about these topics, the reader may start by consulting [13].

5.8 Stochastic Local Search

In DPLL and CDCL variants, a search for a satisfying assignment proceeds by extending a partial assignment as long as no constraint is falsified. In stochastic local search (SLS), there is a “current” complete assignment at every point in the search: if some clauses are not satisfied by the current assignment, one of a set of functions, each designed to create a new assignment from the given formula and current assignment, is applied according to some probability schedule to determine a new current assignment. This process continues until either all clauses are satisfied or until some function determines it is time to give up. Thus, SLS algorithms are generally *incomplete* and are not typically designed to provide a certificate for an unsatisfiable formula. A main strength of SLS algorithms is that they admit “long jumps” through the search space and thereby avoid the condition of becoming mired in an unnecessarily long exploration of a search subspace, particularly that for which all search trajectories lead to a local minimum number of satisfied clauses: deterministic solvers must take special measures to avoid that condition. On the other hand, it is difficult for SLS solvers to use the systematic tabulation and learning techniques that have made the deterministic variants successful.

5.8.1 Walksat

A successful early implementation of SLS for SAT is Walksat [127]. The Walksat algorithm is shown in Fig. 22. It is safe to assume that all clauses in the input formula ψ have at least two literals since elementary and prudent preprocessing of ψ would have assigned a value to any variable in a unit clause to satisfy that clause and would have propagated the effect of that change to the remainder of ψ .

This algorithm has a few interesting features worth pointing out. One is the probability p which is used to choose a variable whose value should be reversed (from now on the word *flipped* is used to mean just this): whether a random literal or the one whose flipped value results in the minimum number of satisfied clauses that become falsified. Experiments suggest the optimal value for p depends on and is sensitive to the particular class of problem the algorithm is applied to [89]. For example, it has been found that $p = 0.57$ is best for random 3-SAT formulas⁶ and this point sits at the minimum of an upward facing, cup-shaped curve with steeply increasing slopes on either side of the minimum. A second interesting feature is that a flip is chosen randomly *only if* there is not a variable in c which does not *break* any satisfied clauses. Thus, emphasizing positive movement to the goal of satisfying the most clauses as early as possible, Walksat has weakened some of the protection against loop stagnation that random flips provide. A third feature is the parameter mf which limits exploration of a local search space: if significant progress is not being made for a while, then exploration of this local space terminates and a long jump to a different local search space is made and search resumed. This *restart*

⁶Random 3-SAT formulas are defined in Sect. 7, Page 440.

Algorithm 5.

```

Walksat ( $\psi, mt, mf, p$ )
/* Input:  $\psi$  is a set of sets of literals representing a CNF formula, */
/*          mt is the maximum number of solution attempts,           */
/*          mf is the maximum number of assignment changes,         */
/*          p is a probability                                     */
/* Output is either a model for  $\psi$  or “give up”            */

    Repeat the following  $mt$  times:
        Set  $M \leftarrow$  a random truth assignment for  $\psi$ .
        Repeat the following  $mf$  times:
            If  $M$  is a model for  $\psi$ , Output  $M$ .
            Set  $c \leftarrow$  a random falsified clause of  $\psi$ .
            If  $\exists l \in c$  such that flipping  $var(l)$  does not
                falsify a satisfied clause, Set  $v \leftarrow var(l)$ .
            Otherwise, Set  $v \leftarrow var(l)$  where, with probability  $p$ ,  $l$  is
                chosen randomly from  $c$ ; and otherwise is the literal  $l \in c$ 
                for which the number of satisfied clauses that become
                falsified by flipping  $var(l)$  is minimum.
            Flip  $v$ .
        Output “give up”
    □

```

Fig. 22 Walksat

mechanism compensates for the possible problem nonrandom flips may get into as mentioned above. Walksat may be applied to MAX-SAT problems. A version for weighted MAX-SAT problems is available [86].

5.8.2 Novelty Family

Novelty [109] is a variant of Walksat that uses a more sophisticated variable selection algorithm. Once a falsified clause c is chosen, a score is computed for each variable v represented in c , based on the total number of satisfied clauses that would result if v were flipped. If the variable with highest score is not the variable that was most recently flipped, it is flipped now. Otherwise, it becomes the flipped variable with probability $1 - p$ or the second highest scoring variable becomes the flipped variable with probability p .

Although Novelty is an improvement over Walksat in many empirical cases, it has the problem that its inner loop may not terminate for a given M [73]. Novelty+ is a slight modification that eliminates this problem by introducing a second probability wp : a variable is randomly selected from c and flipped with probability wp and otherwise the Novelty flip heuristic is applied. It is reported that $wp = 0.01$ is sufficient to eliminate the non-termination problem [75] (i.e., a solution will be found with arbitrarily high probability, if one exists, even if $mt = 1$, as long as mf is set arbitrarily high).

Adaptive Novelty+ provides one more tweak to the Walksat/Novelty+ approach: the noise parameter p is allowed to vary during search. It has been mentioned above that algorithm performance is sensitive to p : thus, the optimal value for p depends significantly on the particular class of input formulas. Adaptive Novelty+ [74] attempts to mitigate this problem. If p is increased, the probability of inner loop stagnation is reduced. If p is decreased, the ability of the algorithm to follow a gradient to a solution is improved. These two counteracting conditions are thought to explain the sensitivity of p to problem domain. Adaptive Novelty+ begins at $p = 0$ and witnesses a rapid improvement in the number of clauses satisfied. When stagnation is detected, p is increased. Such increases continue periodically until stagnation is overcome and then p is reduced. The cycle is continued until termination. Stagnation is detected when it is noticed that an objective function has not changed over a certain number of steps, called the Stagnation Detection Interval. The usual objective function is the number of satisfied clauses. One specific adaptation strategy is the following:

$$\begin{aligned} \text{Stagnation Detection Interval: } & m/6, m = \text{number of clauses} \\ \text{Incremental increase in } p: & p = p + (1 - p) * 0.2 \\ \text{Incremental decrease in } p: & p = p - 0.4 * p \end{aligned}$$

The asymmetry between increases and decreases in p is motivated by the fact that detecting search stagnation is computationally more expensive than detecting search progress and that it is advantageous to approximate optimal noise levels from above rather than from below [74]. After p has been reset, the current objective function value is stored and becomes the basis for detecting stagnation. Observe that the factors 1/6, 0.4, and 0.2 are parameters that could be adjusted, but according to [74], performance appears to be less sensitive to these parameters than it is to p .

5.8.3 Discrete Lagrangian Methods

Discrete Lagrangian methods are another class of SLS algorithms which control the ability of a search to escape a subspace that contains a local minimum [128]. Let $\psi = \{c_1, c_2, \dots, c_m\}$ be a CNF formula with clauses c_1, \dots, c_m ; let $V = \{v : \exists c \in \psi \text{ s.t. } v \in c \text{ or } \neg v \in c\}$; and attach labels to variables to write $V = \{v_1, v_2, \dots, v_n\}$. Define for all $c_i \in \psi$ and $v_j \in V$ as follows:

$$Q_{i,j}(v_j) = \begin{cases} 1 - v_j & \text{if } v_j \in c_i \\ v_j & \text{if } \neg v_j \in c_i \\ 1 & \text{otherwise.} \end{cases}$$

Let x be an n -dimensional 0-1 vector where x_j holds the value of variable v_j , and let $C_i(x) = \prod_{j=1}^n Q_{i,j}(v_j)$. Define

$$N(x) = \sum_{i=1}^m C_i(x).$$

Then $N(x)$ is the number of unsatisfied clauses in ψ under the assignment x . Consider the following optimization problem:

$$\begin{aligned} \text{minimize over } x \in \{0, 1\}^n & N(x) = \sum_{i=1}^m C_i(x) \\ \text{subject to} & C_i(x) = 0 \quad \forall i \in \{1, 2, \dots, m\}. \end{aligned}$$

The objective function (the first line of the above) is enough to specify the SAT problem. The extra constraints are added to guide the search. In particular, the extra constraints help bring the search out of a local minimum.

The above overconstrained optimization problem may be converted to an unconstrained optimization problem by applying a Lagrange multiplier λ_i to each constraint $C_i(x)$ and adding the result to the objective function. Writing λ to represent the vector $\langle \lambda_1, \dots, \lambda_m \rangle$ of Lagrange multipliers and $\mathbf{C}(x)$ to represent the vector $\langle C_1(x), \dots, C_m(x) \rangle$, define

$$L(x, \lambda) = N(x) + \lambda^T \mathbf{C}(x),$$

where $x \in \{0, 1\}^n$ and λ_i can be real valued. A saddle point (x^*, λ^*) of $L(x, \lambda)$ is defined to satisfy the following condition:

$$L(x^*, \lambda) \leq L(x^*, \lambda^*) \leq L(x, \lambda^*)$$

for all λ sufficiently close to λ^* and for all x whose Hamming distance between x^* and x is 1. The *discrete Lagrangian method* (DLM) for solving CNF SAT can be defined as a set of equations which update x and λ :

$$\begin{aligned} x^{k+1} &= x^k - \Delta_x L(x^k, \lambda^k) \\ \lambda^{k+1} &= \lambda^k + \mathbf{C}(x^k), \end{aligned}$$

where $\Delta_x L(x, \lambda)$ is the *discrete gradient operator* with respect to x such that $\Delta_x L(x, \lambda) = \langle \delta_1, \delta_2, \dots, \delta_n \rangle$, $\forall 1 \leq i \leq n \delta_i \in \{-1, 0, 1\}$, $\sum_{1 \leq i \leq n} |\delta_i| = 1$, and $(x - \Delta_x L(x, \lambda)) \in \{0, 1\}^n$ (i.e., $\Delta_x L(x, \lambda)$ identifies a variable to be flipped and $x - \Delta_x L(x, \lambda)$ is a neighbor of x – i.e., at Hamming distance 1 from x). There are two ways to calculate $\Delta_x L(x, \lambda)$. One way, termed *greedy*, replaces the current assignment with a neighboring assignment leading to the maximum decrease in the Lagrangian function value. Another way, termed *hill climbing*, replaces the current assignment with a neighboring assignment that leads to a decrease in the value of the Lagrangian function. Both involve some local search in the neighborhood of the current assignment, but for every greedy update, the complexity of this search is $O(n)$, whereas the hill-climbing update generally is less expensive.

Algorithm 6.

```

DLM ( $\psi$ )
/* Input:  $\psi$  is a set of sets of literals representing a CNF formula, */
/* Output is either a model for  $\psi$  or “give up” */
Set  $x \leftarrow$  a random  $n$ -dimensional 0-1 vector.
Set  $\lambda \leftarrow \langle 0, 0, \dots, 0 \rangle$ .
Construct  $\mathbf{C}(x)$  from  $\psi$ .
Repeat the following while  $N(x) > 0$ :
    Set  $x \leftarrow x - \Delta_x L(x, \lambda)$ .
    If condition to update  $\lambda$  is satisfied, do this:
         $\lambda \leftarrow \lambda + \gamma \cdot \mathbf{C}(x)$ .
Output  $x$ 
□

```

Fig. 23 Discrete Lagrangian method

An algorithmic sketch of DLM is shown in Fig. 23. The parameter γ controls the magnitude of the changes to λ_i . A value of $\gamma = 1$ is reported to be sufficient for many benchmarks, but a smaller γ has resulted in improved performance for some difficult ones. Experiments have shown that λ should not be updated every time the current assignment is replaced. This is because the changes in $L(x, \lambda)$ are usually very small and relatively larger changes in λ can overcompensate causing the search to become lost. Therefore, the DLM algorithm contains a line stating that a condition must be satisfied before λ can be updated. One possibility is that λ is not updated until $\Delta_x L(x, \lambda) = 0$. When that happens, the local minimum is reached and the change in λ is needed to jump to a different section of the search space. This strategy may cause a problem if a search-space plateau is entered. In that case, it will be prudent to change λ sooner. For improved variants of DLM, see [147, 148].

5.9 Binary Decision Diagrams

Binary decision diagrams (BDDs) as graphs were discussed in Sect. 3.2. In this section the associated BDD data structure and efficient operations on that data structure are discussed. Attention is restricted to reduced ordered binary decision diagrams (ROBDDs) due to its compact, efficient, canonical properties.

The following is a short review of Sect. 3.2. A ROBDD is a BDD such that (1) there is no vertex v such that $\text{then}(v) = \text{else}(v)$ and (2) the subgraphs of two distinct vertices v and w are not isomorphic. A ROBDD represents a Boolean function uniquely in the following way (symbol v will represent both a vertex of a ROBDD and a variable labeling a vertex). Define $f(v)$ recursively as follows:

1. If v is the terminal vertex labeled 0, then $f(v) = 0$.
2. If v is the terminal vertex labeled 1, then $f(v) = 1$.
3. Otherwise, $f(v) = (v \wedge f(\text{then}(v))) \vee (\neg v \wedge f(\text{else}(v)))$.

Algorithm 7.

```

findOrCreateNode ( $v, t, e$ )
/* Input: variable label  $v$ , Node object indices  $t$  and  $e$  */  

/* Output: an array index of a Node object  $\langle v, t, e \rangle$  */  

  If  $t == e$  then Return  $t$ .  

  Set  $node \leftarrow lookup(\langle v, t, e \rangle)$ .  

  If  $node \neq null$  then Return  $node$ .  

  Set  $node \leftarrow createNode(\langle v, t, e \rangle)$ .  

   $insert(\langle v, t, e \rangle, node)$ .  

  Return  $node$ .

```

□

Fig. 24 Procedure for finding a node or creating and inserting it

Then $f(root(v))$ is the function represented by the ROBDD. A Boolean function has different ROBDD representations, depending on the variable order imposed by $index$, but there is only one ROBDD for each ordering. Thus, ROBDDs are known as a canonical representation of Boolean functions. From now on BDD will be used to refer to a ROBDD.

A data structure representing a BDD consists of an array of Node objects (or nodes), each corresponding to a BDD vertex and each containing three elements: a variable label v , $then(v)$, and $else(v)$, where the latter two elements are BDD array indices. The first two nodes in the BDD array correspond to the 0 and 1 terminal vertices of a BDD. For both, $then(..)$ and $else(..)$ are empty. Denote by $terminal(1)$ and $terminal(0)$ the BDD array locations of these nodes. All other nodes fill up the remainder of the BDD array in the order they are created. A node that is not in the BDD array can be created and added to the BDD array, and its array index returned in constant time. A hashtable, commonly referred to as a *unique table*, is maintained for the purpose of finding a node's array location given v , $then(v)$, $else(v)$. If there is no corresponding entry in the hashtable, a new node is created and recorded in the hashtable with key v , $then(v)$, $else(v)$. A single procedure called **findOrCreateNode** takes v , $then(v)$, $else(v)$ as arguments and returns the BDD array index of a node, either a newly created one or an existing one. This procedure is shown in Fig. 24.

The main BDD construction operation is to find and attach two descendant nodes ($then(v)$ and $else(v)$) to a parent node (v). The procedure **findOrCreateNode** is used to ensure that no two nodes in the final BDD data structure represent the same function. The procedure for building a BDD data structure is **buildBDD**, shown in Fig. 25. It is assumed that variable indices match the value of $index$ applied to that variable (thus, $i = index(v_i)$). The complexity of **buildBDD** is proportional to the number of nodes that must be created. In all interesting applications, many BDDs are constructed. But they may all share the BDD data structure above. Thus, a node may belong to many BDDs.

Algorithm 8.

```

buildBDD ( $f, i$ )
/* Input: Boolean function  $f$ , index  $i$  */          */
/* Output: root Node of BDD representing  $f$  */      */
  If  $f \Leftrightarrow 1$  return terminal(1).           */
  If  $f \Leftrightarrow 0$  return terminal(0).           */
  Set  $t \leftarrow \text{buildBDD}(f|_{v_i=1}, i+1)$ .    */
  Set  $e \leftarrow \text{buildBDD}(f|_{v_i=0}, i+1)$ .    */
  Return findOrCreateNode( $v_i, t, e$ ).           */

```

□

Fig. 25 Algorithm for building a BDD: invoked using **buildBDD**($f, 1$)**Algorithm 9.**

reduce₁ (v, f) /* Input: variable v , BDD f */ */ /* Output: reduced BDD */ */ If $\text{root}(f) == v$ then Return <i>then</i> ($\text{root}(f)$). Return f .	reduce₀ (v, f) /* Input: variable v , BDD f */ */ /* Output: reduced BDD */ */ If $\text{root}(f) == v$ then Return <i>else</i> ($\text{root}(f)$). Return f .
--	--

□

Fig. 26 Operations **reduce₁** and **reduce₀**

The operations **reduce₁** and **reduce₀**, shown in Fig. 26, are used to describe several important BDD operations in subsequent sections. Assuming v is the root of the BDD representing f , the operation **reduce₁**(v, f) returns f constrained by the assignment of 1 to variable v and **reduce₀**(v, f) returns f constrained by the assignment of 0 to the variable v .

Details on performing the common binary operations of \wedge and \vee on BDDs will be ignored here. The reader may refer to [9] for detailed descriptions. Here, it is only mentioned that, using a dynamic programming algorithm, the complexity of these operations is proportional to the product of the sizes of the operands and the size of the result of the operation can be that great as well. Therefore, using \wedge alone, for example (as so many problems would require), could lead to intermediate structures that are too large to be of value. This problem is mitigated somewhat by operations of the kind discussed in the next four subsections, particularly existential quantification.

The operations considered next are included not only because they assist BDD-oriented solutions but mainly because they can assist search-oriented solutions when used properly. For example, if inputs are expressed as a collection of BDDs, then they may be preprocessed to reveal information that may be exploited later,

Algorithm 10.

```

exQuant ( $f, v$ )
/* Input: BDD  $f$ , variable  $v$  */  

/* Output: BDD  $f$  with  $v$  existentially quantified away */  

  If  $\text{root}(f) == v$  then Return  $\text{then}(\text{root}(f)) \vee \text{else}(\text{root}(f))$ .  

  If  $\text{index}(v) > \text{index}(\text{root}(f))$  then Return 0. // If  $v$  is not in  $f$  do nothing  

  Set  $h_{f_1} \leftarrow \text{exQuant}(\text{then}(\text{root}(f)), v)$ .  

  Let  $h_{f_0} \leftarrow \text{exQuant}(\text{else}(\text{root}(f)), v)$ .  

  If  $h_{f_0} == h_{f_1}$  then Return  $h_{f_1}$ .  

  Return FindOrCreateNode( $\text{root}(f), h_{f_1}, h_{f_0}$ ).  

□

```

Fig. 27 Algorithm for existentially quantifying variable v away from BDD f . The \vee denotes the *or* of BDDs

during search. In particular, inferences⁷ may be determined and used to reduce input complexity. The discussion of the next four sections emphasizes this role.

5.9.1 Existential Quantification

A Boolean function which can be written

$$f(v, \vec{x}) = (v \wedge h_1(\vec{x})) \vee (\neg v \wedge h_2(\vec{x}))$$

can be replaced by

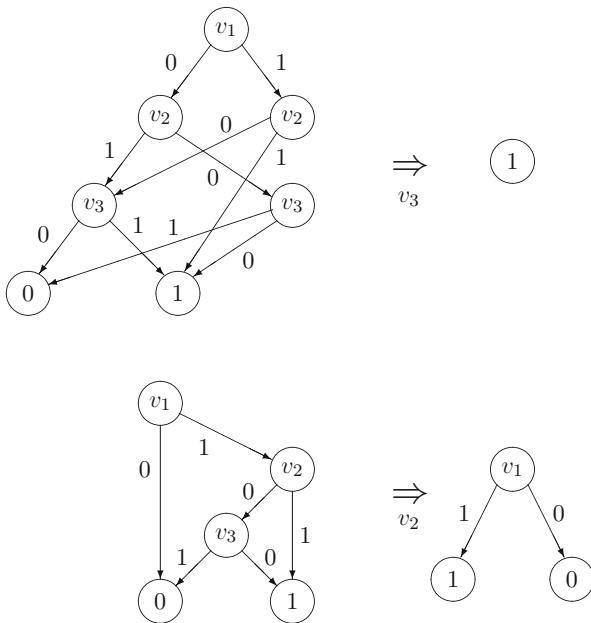
$$f(\vec{x}) = h_1(\vec{x}) \vee h_2(\vec{x}),$$

where \vec{x} is a list of one or more variables. There is a solution to $f(\vec{x})$ if and only if there is a solution to $f(v, \vec{x})$, so it is sufficient to solve $f(\vec{x})$ to get a solution to $f(v, \vec{x})$. Obtaining $f(\vec{x})$ from $f(v, \vec{x})$ is known as *existentially quantifying v away from $f(v, \vec{x})$* . This operation is efficiently handled if $f(v, \vec{x})$ is represented by a BDD. However, since most interesting BDD problems are formulated as a conjunction of functions and therefore as conjunctions of BDDs, existentially quantifying away a variable v succeeds easily only when just one of the input BDDs contains v . Thus, this operation is typically used together with other BDD operations for maximum effectiveness. The algorithm for existential quantification is shown in Fig. 27.

If inferences can be revealed in preprocessing, they can be applied immediately to reduce input size and therefore reduce search complexity. Although existential quantification can, by itself, uncover inferences (see, e.g., Fig. 28), those same inferences are revealed during BDD construction if inference lists for each node are built and maintained. Therefore, a more effective use of existential quantification is

⁷Finding inferences is referred to in the BDD literature as finding *essential* values to variables, and a set of inferences (a conjunction of literals) is referred to as a *cube*.

Fig. 28 Two examples of existentially quantifying a variable away from a function. Functions are represented as BDDs on the left. Variable v_3 is existentially quantified away from the top BDD leaving 1, meaning that regardless of assignments given to variables v_1 and v_2 , there is always an assignment to v_3 which satisfies the function. Variable v_2 is existentially quantified away from the bottom BDD leaving the inference $v_1 = 1$



in support of other operations, such as strengthening (see Sect. 5.9.5), to uncover those inferences that cannot be found during BDD construction or in tandem with \wedge to retard the growth of intermediate BDDs.

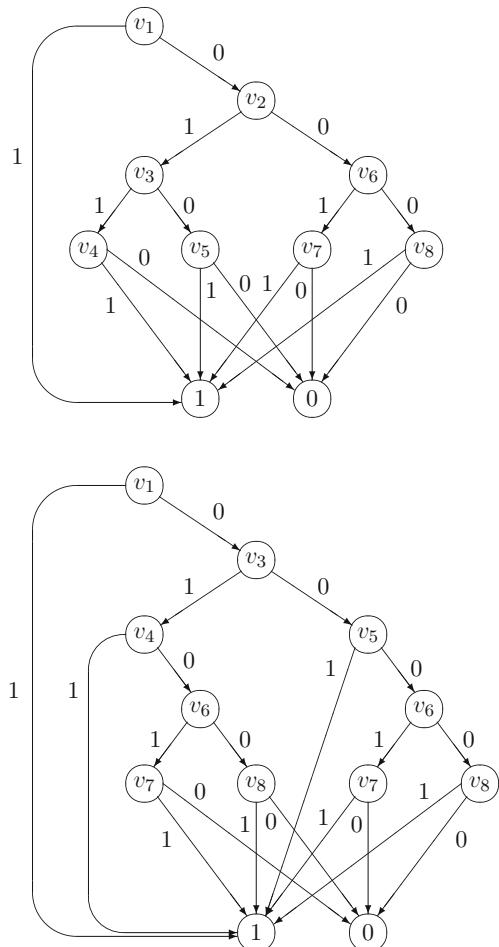
Existential quantification, if applied as a preprocessing step prior to search, can increase the number of choicepoints expanded per second but can increase the size of the search space. The increase in choicepoint speed is because existentially quantifying a variable away from the function has the same effect as branching from a choicepoint in both directions. Then overhead is reduced by avoiding heuristic computations. However, search-space size may increase since the elimination of a variable can cause subfunctions that had been linked only by that variable to become merged with the result that the distinction between the subfunctions becomes blurred. This is illustrated in Fig. 29. The speedup can overcome the lost intelligence but it is sometimes better to turn it off.

5.9.2 Reductions and Inferences

Consider the truth tables corresponding to two BDDs f and c over the union of variable sets of both f and c . Build a new BDD g with variable set no larger than the union of the variable sets of f and c and with a truth table such that on rows which c maps to 1, g maps to the same value that f maps to and on other rows g maps to any value, independent of f . It should be clear that $f \wedge c$ and $g \wedge c$ are identical so g can replace f in a collection of BDDs without changing its solution space.

There are at least three reasons why this might be done. The superficial reason is that g can be made smaller than f . A more important reason is that inferences can

Fig. 29 Existential quantification can cause blurring of functional relationships. The top function is seen to separate variables v_6 , v_7 , and v_8 from v_3 , v_4 , and v_5 if v_2 is chosen during search first. Existentially quantifying v_2 away from the top function before search results in the bottom function in which no such separation is immediately evident. Without existential quantification, the assignment $v_1 = 0$, $v_2 = 1$, $v_3 = 1$ reveals the inference $v_4 = 1$. With existential quantification the assignment must be augmented with $v_7 = 0$ and $v_8 = 0$ (but v_2 is no longer necessary) to get the same inference



be discovered. The third reason is that BDDs can be removed from the collection without loss. Consider, for example, BDDs representing functions

$$\begin{aligned} f &= (v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_1 \vee v_2) \text{ and} \\ c &= (v_1 \vee \neg v_2). \end{aligned}$$

Let a truth table row be represented by a 0-1 vector which reflects assignments of variables indexed in increasing order from left to right. Let g have the same truth table as f except for row $\langle 011 \rangle$ which c maps to 0 and g maps to 1. Then $g = (v_1 \leftrightarrow v_2)$ and $f \wedge c$ is the same as $g \wedge c$ but g is smaller than f . As an example of discovering inferences, consider

$$f = (v_1 \rightarrow v_2) \wedge (\neg v_1 \rightarrow (\neg v_3 \wedge v_4))$$

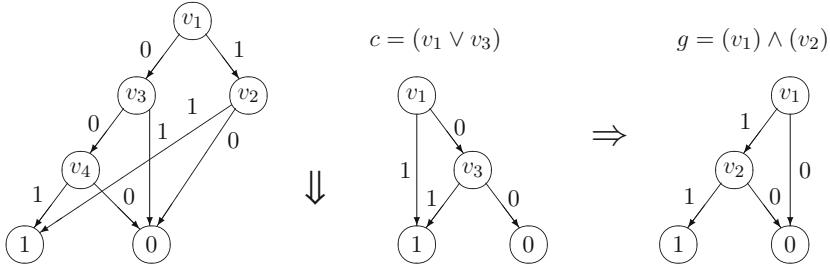


Fig. 30 A call to `restrict(f, c)` returns the BDD g shown on the *right*. In this case inferences $v_1 = 1$ and $v_2 = 1$ are revealed. The symbol \Downarrow denotes the operation

$$f = (v_1 \rightarrow v_2) \wedge (\neg v_1 \rightarrow (\neg v_3 \wedge v_4)) \text{ and}$$

$$c = (v_1 \vee v_3).$$

Let g have the same truth table as f except g maps rows $\langle 0001 \rangle$ and $\langle 0101 \rangle$ to 0, as does c . Then $g = (v_1) \wedge (v_2)$ which reveals two inferences. The BDDs for f , c , and g of this example are shown in Fig. 30. The example showing BDD elimination is deferred to Theorem 6, Sect. 5.9.4.

Clearly, there are numerous strategies for creating g from f and c and replacing f with g . An obvious one is to have g map to 0 all rows that c maps to 0. This strategy, which will be called zero-restrict in this chapter, turns out to have weaknesses. Its obvious dual, which has g map to 1 all rows that c maps to 0, is no better. For example, applying zero-restrict to f and c of Fig. 32 produces $g = \neg v_3 \wedge (v_1 \vee (\neg v_1 \wedge \neg v_2))$ instead of the inference $g = \neg v_3$ which is obtained from a more intelligent replacement strategy. An alternative approach, one of many possible ones, judiciously chooses some rows of g to map to 1 and others to map to 0 so that g 's truth table reflects a logic pattern that generates inferences. The truth table of c has many 0 rows and this is exploited. Specifically, c maps rows $\langle 010 \rangle$, $\langle 011 \rangle$, $\langle 101 \rangle$, and $\langle 111 \rangle$ to 0. The more intelligent strategy lets g map rows $\langle 011 \rangle$ and $\langle 111 \rangle$ to 0 and rows $\langle 010 \rangle$ and $\langle 101 \rangle$ to 1. Then $g = \neg v_3$.

Improved replacement strategies might target particular truth table patterns, for example, equivalences, or they might aim for inference discovery. Since there is more freedom to manipulate g if the truth table of c has many zeros, it is important to choose c as carefully as the replacement strategy. This is illustrated by the examples of Figs. 31 and 32 where, in the first case, no inference is generated but after f and c are swapped, an inference is generated. The next two subsections show two replacement strategies that are among the more commonly used.

5.9.3 Restrict

The original version of restrict is what is called zero-restrict above. That is, the original version of restrict is intended to remove paths to *terminal*(1) from f

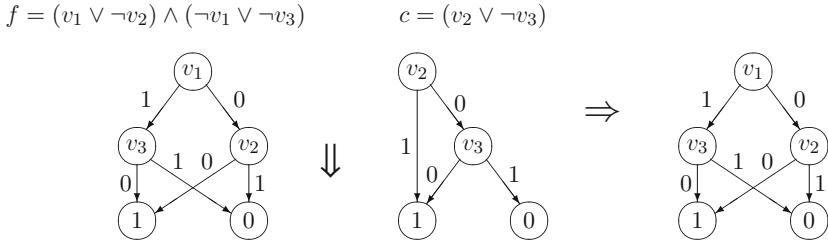


Fig. 31 A call to **restrict**(f, c) results in no change

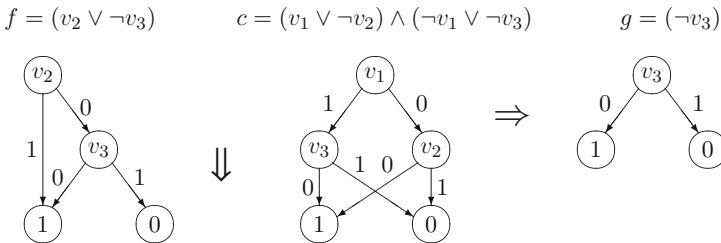


Fig. 32 Reversing the roles of f and c in Fig. 31, a call to **restrict**(f, c) results in the inference $g = \neg v_3$ as shown on the right. In this case, the large number of 0 truth table rows for c was exploited to advantage

that are made irrelevant by c . The idea was introduced in [43]. In this chapter an alternative version which is implemented as Algorithm 11 of Fig. 33 is considered. Use the symbol \Downarrow to denote the restrict operator. Then $g = f \Downarrow c$ is the result of zero-restrict after all variables in c that are not in f are existentially quantified away from c . Figures 30–32 show examples that were referenced in the previous subsection.

Procedure **restrict** is similar to a procedure called generalized cofactor (**gcf**) or constrain (see the next subsection for a description). Both **restrict**(f, c) and **gcf**(f, c) agree with f on interpretations where c is satisfied but are generally somehow simpler than f . Procedure **restrict** can be useful in preprocessing because the BDDs produced from it can never contain more variables than the BDDs they replace.

On the negative side, it can, in odd cases, cause a garbling of local information. Moreover, although **restrict** may reveal some of the inferences that strengthening would (see below), it can still cause the number of search choicepoints to increase. Both these issues are related: **restrict** can spread an inference that is evident in one BDD over multiple BDDs (see Fig. 34 for an example).

5.9.4 Generalized Cofactor

The *generalized cofactor* operation, also known as *constrain*, is denoted here by $|$ and implemented as **gcf** (Algorithm 12) in Fig. 35. It takes BDDs f and c as input

Algorithm 11.

```

restrict ( $f, c$ )
/* Input: BDD  $f$ , BDD  $c$  */  

/* Output: BDD  $f$  restricted by  $c$  */  

  If  $c$  or  $f$  is terminal(1) or if  $f$  is terminal(0) return  $f$ .
  If  $c == \neg f$  return terminal(0).
  If  $c == f$  return terminal(1).
  //  $f$  and  $c$  have a non-trivial relationship
  Set  $v_f \leftarrow \text{root}(f)$ . //  $v_f$  is a variable
  Set  $v_c \leftarrow \text{root}(c)$ . //  $v_c$  is a variable
  If  $\text{index}(v_f) > \text{index}(v_c)$  return restrict( $f$ , exQuant( $c, v_c$ )).
  If reduce0( $v_f, c$ ) is terminal(0) then
    Return restrict(reduce1( $v_f, f$ ), reduce1( $v_f, c$ )).  

  If reduce1( $v_f, c$ ) is terminal(0) then
    Return restrict(reduce0( $v_f, f$ ), reduce0( $v_f, c$ )).  

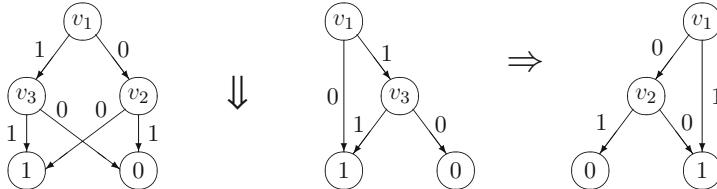
  Set  $h_{f_1} \leftarrow \text{restrict}(\text{reduce}_1(v_f, f), \text{reduce}_1(v_f, c))$ .
  Set  $h_{f_0} \leftarrow \text{restrict}(\text{reduce}_0(v_f, f), \text{reduce}_0(v_f, c))$ .
  If  $h_{f_1} == h_{f_0}$  then Return  $h_{f_1}$ .
  Return findOrCreateNode( $v_f, h_{f_1}, h_{f_0}$ ).  

□

```

Fig. 33 Algorithm for restricting a BDD f by a BDD c

$$f = (v_1 \vee \neg v_2) \wedge (\neg v_1 \vee v_3) \quad c = (\neg v_1 \vee v_3) \quad g = (v_1 \vee \neg v_2)$$

**Fig. 34** A call to **restrict**(f, c) spreads an inference that is evident in one BDD over multiple BDDs. If v_3 is assigned 0 in f , then $v_1 = 0$ and $v_2 = 0$ are inferred. After replacing f with $g = \text{restrict}(f, c)$, to get the inference $v_2 = 0$ from the choice $v_3 = 0$, visit c to get $v_1 = 0$ and then g to get $v_2 = 0$. Thus, restrict can increase work if not used properly. In this case, restricting in the reverse direction leads to a better result

and produces $g = f|c$ by *sibling substitution*. BDD g may be larger or smaller than f , but, more importantly, systematic use of this operation can result in the elimination of BDDs from a collection. Unfortunately, by definition, the result of this operation depends on the underlying BDD variable ordering, so it cannot be regarded as a logical operation. It was introduced in [42].

BDD g is a generalized cofactor of f and c if for any truth assignment t , $g(t)$ has the same value as $f(t')$ where t' is the *nearest* truth assignment to t that maps c to 1. The notion of *nearest* truth assignment depends on a permutation π of the numbers

Algorithm 12.

```

gcf ( $f, c$ )
/* Input: BDD  $f$ , BDD  $c$  */  

/* Output: greatest co-factor of  $f$  by  $c$  */  

If  $f == \text{terminal}(0)$  or  $c == \text{terminal}(0)$  return  $\text{terminal}(0)$ .
If  $c == \text{terminal}(1)$  or  $f == \text{terminal}(1)$  return  $f$ .
Set  $v_m \leftarrow \text{index}^{-1}(\min\{\text{index}(\text{root}(c)), \text{index}(\text{root}(f))\})$ .
//  $v_m$  is the top variable of  $f$  and  $c$ 
If  $\text{reduce}_0(v_m, c) == \text{terminal}(0)$  then
    Return gcf( $\text{reduce}_1(v_m, f)$ ,  $\text{reduce}_1(v_m, c)$ ).
If  $\text{reduce}_1(v_m, c) == \text{terminal}(0)$  then
    Return gcf( $\text{reduce}_0(v_c, f)$ ,  $\text{reduce}_0(v_c, c)$ ).
Set  $h_1 \leftarrow \text{gcf}(\text{reduce}_1(v_m, f), \text{reduce}_1(v_m, c))$ .
Set  $h_0 \leftarrow \text{gcf}(\text{reduce}_0(v_m, f), \text{reduce}_0(v_m, c))$ .
If  $h_1 == h_0$  then Return  $h_1$ .
Return FindOrCreateNode( $v_m, h_1, h_0$ ).
□

```

Fig. 35 Algorithm for finding a greatest common cofactor of a BDD

$1, 2, \dots, n$ which states the variable ordering of the input BDDs. Represent a truth assignment to n variables as a vector in $\{0, 1\}^n$ and, for truth assignment t , let t_i denote the i th bit of the vector representing t . Then the distance between two truth assignments t' and t'' is defined as $\sum_{i=1}^n 2^{n-i} (t'_{\pi_i} \oplus t''_{\pi_i})$. One pair of assignments is nearer to each other than another pair if the distance between that pair is less. It should be evident that distances between pairs are unique for each pair.

For example, Fig. 36 shows BDDs f and c under the variable ordering given by $\pi = \langle 1, 2, 3, 4 \rangle$. For assignment vectors $\langle * * 01 \rangle$, $\langle * * 10 \rangle$, and $\langle * * 11 \rangle$ (where $*$ is a wildcard meaning 0 or 1), $\text{gcf}(f, c)$, shown as the BDD at the bottom of Fig. 36, agrees with f since those assignments cause c to evaluate to 1. The closest assignment to $\langle 0000 \rangle$, $\langle 0100 \rangle$, $\langle 1000 \rangle$, and $\langle 1100 \rangle$ causing c to evaluate to 1 is $\langle 0001 \rangle$, $\langle 0101 \rangle$, $\langle 1001 \rangle$, and $\langle 1101 \rangle$, respectively. On all these inputs $\text{gcf}(f, c)$ has value 1, which the reader can check in Fig. 36.

The following expresses the main property of $|$ that makes it useful.

Theorem 6 Given BDDs f_1, \dots, f_k , for any $1 \leq i \leq k$, $f_1 \wedge f_2 \wedge \dots \wedge f_k$ is satisfiable if and only if $(f_1|f_i) \wedge \dots \wedge (f_{i-1}|f_i) \wedge (f_{i+1}|f_i) \wedge \dots \wedge (f_k|f_i)$ is satisfiable. Moreover, any assignment satisfying the latter can be mapped to an assignment that satisfies $f_1 \wedge \dots \wedge f_k$.

Proof If it can be shown that

$$(f_1|f_i) \wedge \dots \wedge (f_{i-1}|f_i) \wedge (f_{i+1}|f_i) \wedge \dots \wedge (f_k|f_i) \quad (4)$$

is satisfiable if and only if

$$(f_1|f_i) \wedge \dots \wedge (f_{i-1}|f_i) \wedge (f_{i+1}|f_i) \wedge \dots \wedge (f_k|f_i) \wedge f_i \quad (5)$$

is satisfiable, then, since (5) is equivalent to $f_1 \wedge \dots \wedge f_k$, the first part of the theorem will be proved. Suppose (5) is satisfied by truth assignment t . That t represents a truth table row that f_i maps to 1. Clearly that assignment also satisfies (4). Suppose no assignment satisfies (5). Then all assignments for which f_i maps to 1 do not satisfy (4) since otherwise (5) would be satisfied by any that do. Only truth assignments t which f_i maps to 0 need to be considered. Each $(f_j|f_i)$ in (4) and (5) maps to the same value that f_j maps the *nearest* truth assignment, say, r , to t that satisfies f_i . But r cannot satisfy (5) because it cannot satisfy (4) by the argument above. Hence, there is no truth assignment falsifying f_i but satisfying (4) so the first part is proved.

For the second part, observe that any truth assignment that satisfies (4) and (5) also satisfies $f_i \wedge \dots \wedge f_k$, so it is only necessary to consider assignments t that satisfy (4) but not (5). In that case, by construction of $(f_j|f_i)$, the assignment that is *nearest* to t and satisfies f_i also satisfies $(f_j|f_i)$. That assignment satisfies $f_1 \wedge \dots \wedge f_k$. \square

This means that, for the purposes of a solver, generalized cofactoring can be used to eliminate one of the BDDs among a given conjoined set of BDDs: the solver finds an assignment satisfying $\text{gcf}(f_1, f_i) \wedge \dots \wedge \text{gcf}(f_k, f_i)$ and then extends the assignment to satisfy f_i ; otherwise, the solver reports that the instance has no solution. However, unlike **restrict**, generalized co-factoring cannot by itself reduce the number of variables in a given collection of BDDs. Other properties of the **gcf** operation, all of which are easy to show, are:

1. $f = c \wedge \text{gcf}(f, c) \vee \neg c \wedge \text{gcf}(f, \neg c)$.
2. $\text{gcf}(\text{gcf}(f, g), c) = \text{gcf}(f, g \wedge c)$.
3. $\text{gcf}(f \wedge g, c) = \text{gcf}(f, c) \wedge \text{gcf}(g, c)$.
4. $\text{gcf}(f \wedge c, c) = \text{gcf}(f, c)$.
5. $\text{gcf}(f \wedge g, c) = \text{gcf}(f, c) \wedge \text{gcf}(g, c)$.
6. $\text{gcf}(f \vee g, c) = \text{gcf}(f, c) \vee \text{gcf}(g, c)$.
7. $\text{gcf}(f \vee \neg c, c) = \text{gcf}(f, c)$.
8. $\text{gcf}(\neg f, c) = \neg \text{gcf}(f, c)$.

9. If c and f have no variables in common and c is satisfiable, then $\text{gcf}(f, c) = f$.

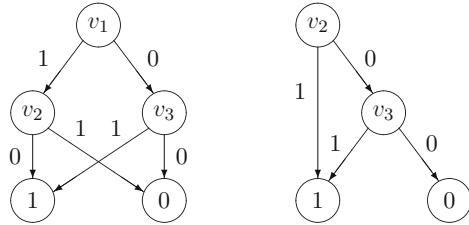
Care must be taken when cofactoring in *both* directions (exchanging f for c). For example, $f \wedge g \wedge h$ cannot be replaced by $(g|f) \wedge (f|g) \wedge h$ since the former may be unsatisfiable when the latter is satisfiable.

Examples of the application of **gcf** are shown in Figs. 36 and 37. Figure 36 illustrates the possibility of increasing BDD size. Figure 37 presents the same example after swapping v_1 and v_3 under the same variable ordering and shows that the result produced by **gcf** is sensitive to variable ordering. Observe that the functions produced by **gcf** in both figures have different values under the assignment $v_1 = 1$, $v_2 = 1$, and $v_3 = 0$. Thus, the function returned by **gcf** depends on the variable ordering as well.

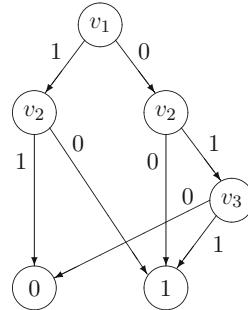
Fig. 36 Generalized cofactor operation on f and c as shown. In this case the result is more complicated than f . The variable ordering is $v_1 < v_2 < v_3$

$$f = (v_1 \rightarrow \neg v_2) \vee (\neg v_1 \rightarrow v_3)$$

$$c = (v_2 \vee v_3)$$



$$gcf(f, c) = (v_1 \rightarrow \neg v_2) \vee (\neg v_1 \rightarrow (v_2 \rightarrow v_3))$$

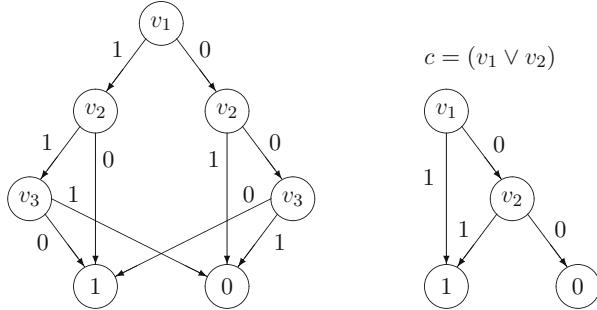


5.9.5 Strengthen

This binary operation on BDDs helps reveal inferences that are missed by **restrict** due to its sensitivity to variable ordering. Given two BDDs, b_1 and b_2 , strengthening conjoins b_1 with the *projection* of b_2 onto the variables of b_1 : that is, $b_1 \wedge \exists \vec{v} b_2$, where \vec{v} is the set of variables appearing in b_2 but not in b_1 . Strengthening each b_i against all other b_j 's sometimes reveals additional inferences or equivalences. Algorithm **strengthen** is shown in Fig. 38. Figure 39 shows an example.

Strengthening provides a way to pass important information from one BDD to another without causing a size explosion. No size explosion can occur because, before b_1 is conjoined with b_2 , all variables in b_2 that do not occur in b_1 are existentially quantified away. If an inference (of the form $v = 1$, $v = 0$, $v = w$, or $v = \neg w$) exists due to just two BDDs, then strengthening those BDDs against each other (pairwise) can *move* those inferences, even if originally spread across both BDDs, to one of the BDDs. Because **strengthen** shares information between BDDs, it can be thought of as sharing intelligence and *strengthening* the relationships between functions; the added intelligence in these strengthened functions can be exploited by a smart search heuristic. It has been found empirically that **strengthen** usually decreases the number of choicepoints when a particular search heuristic is employed, but sometimes it causes more choicepoints. It may be conjectured this is due to the delicate nature of some problems where duplicating information in the BDDs leads the heuristic astray.

$$f = (v_3 \rightarrow \neg v_2) \vee (\neg v_3 \rightarrow v_1)$$



$$gcf(f, c) = (v_1 \wedge (v_2 \rightarrow \neg v_3))$$

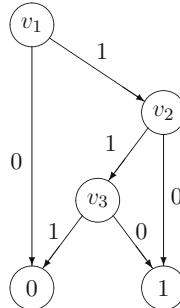


Fig. 37 Generalized cofactor operation on the same f and c of Fig. 36 and with the same variable ordering but with v_1 and v_3 swapped. In this case the result is less complicated than f and the assignment $\{v_1, v_2\}$ causes the output of gcf in this figure to have value 1, whereas the output of gcf in Fig. 36 has value 0 under the same assignment

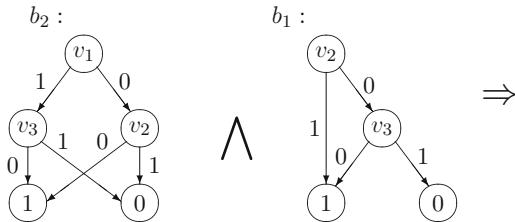
Algorithm 13.

```

Strengthen ( $b_1, b_2$ )
/* Input: BDD  $b_1$ , BDD  $b_2$                                 */
/* Output: BDD  $b_1$  strengthened by  $b_2$                   */
Set  $\vec{x} \leftarrow \{x : x \in b_2, x \notin b_1\}$ .
Repeat the following for all  $x \in \vec{x}$ :
    Set  $b_2 \leftarrow \text{exQuant}(b_2, x)$ .
Return  $b_1 \wedge b_2$ .
□

```

Fig. 38 Algorithm for strengthening a BDD by another



Strengthening example: Existentially quantify v_1 away from b_2 ...

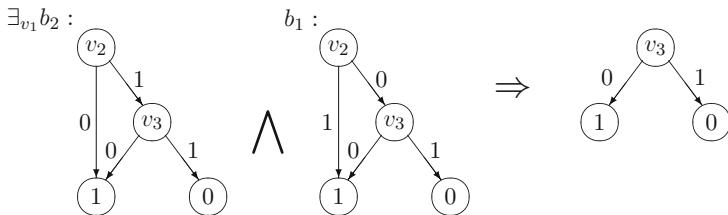


Fig. 39 ... then conjoin the two BDDs. Inference $v_3 = 0$ is revealed

Procedure **strengthen** may be applied to CNF formulas and in this case it is the same as applying Davis-Putnam resolution selectively on some of the clauses. When used on more complex functions, it is clearer how to use it effectively as the clauses being resolved are grouped with some meaning. Evidence for this comes from bounded model checking examples.

This section concludes by mentioning that for some classes of problems, resolution has polynomial complexity, while strictly BDD manipulations require exponential time, and for other classes of problems, resolution has exponential complexity, while BDD manipulations require polynomial time [67].

5.10 Decompositions

The variable elimination methods of Sects. 5.5 and 5.11 recursively *decompose* a given formula ψ into overlapping subformulas ψ_1 and ψ_2 such that the solution to ψ can be inferred from the solutions to ψ_1 and ψ_2 . The decompositions are based on occurrences of a selected variable v in clauses of ψ and each subformula has at least as many clauses as those of ψ which contain neither literal v nor literal $\neg v$. Intuitively, the speed of the methods usually depends on the magnitude of the size reduction from ψ to ψ_1 and ψ_2 . However, it is often the case that the number of occurrences of most variables in a formula or subformula is small which usually means small size reductions for most variables. For example, the average number of occurrences of a variable in a random k -SAT formula⁸ of m clauses developed

⁸Random k -SAT formulas are defined in Sect. 7, Page 440.

from n variables is km/n : so, if $k = 3$ and m/n is, say, 4, then the average number of occurrences of a randomly chosen variable is only 12. Hence, the methods often suffer computational inadequacies which can make them unusable in some cases. On the positive side, such methods can be applied to any CNF formula.

But there are other decomposition methods that sacrifice some generality for the sake of producing subformulas of relatively small size. Truemper's book [138] presents quite a few of these, all of which are capable of computationally efficient solutions to some problems that would be considered difficult for the more general variable elimination methods. This section presents one of these, called monotone decomposition, for illustration and because it is related to material that is elsewhere in this chapter.

5.10.1 Monotone Decomposition

Let CNF formula ψ of m clauses and n variables be represented as a $m \times n$ (0 ± 1)-matrix \mathcal{M}_ψ (defined in Sect. 3.1). A *monotone decomposition* of \mathcal{M}_ψ is a permutation of rows and columns of \mathcal{M}_ψ and the multiplication by -1 of some or all of its columns, referred to below as a column scaling, resulting in a partition into four submatrices as follows:

$$\left(\begin{array}{c|c} \mathcal{A}^1 & \mathcal{E} \\ \hline \mathcal{D} & \mathcal{A}^2 \end{array} \right), \quad (6)$$

where the submatrix \mathcal{A}^1 has at most one $+1$ entry per row; the submatrix \mathcal{D} contains only -1 or 0 entries; the submatrix \mathcal{A}^2 has no restrictions other than the three values of -1 , $+1$, and 0 for each entry; and the submatrix \mathcal{E} has only 0 entries.

The submatrix \mathcal{A}^1 represents a *Horn formula*. In Sect. 6.2 Horn formulas are shown to have the following two important properties: they are solved efficiently, for example, by Algorithm 20 of Fig. 46, and, by Theorem 15, there is always a unique minimum model for a satisfiable Horn formula. The second property means there is always a satisfying assignment M such that, for any other satisfying assignment M' , the variables that have value 1 according to M' are a superset of the variables set to 1 according to M (more succinctly, $M \subset M'$). This property, plus the nature of submatrices \mathcal{D} and \mathcal{E} , effectively allows a split of the problem of determining the satisfiability of ψ into two independent problems: namely, determine satisfiability for the Horn formula represented by \mathcal{A}^1 and determine satisfiability for the subformula represented by \mathcal{A}^2 . The algorithm of Fig. 40 shows this in more detail. The following theorem proves correctness of this algorithm:

Theorem 7 *Let CNF formula ψ be represented as a monotone decomposition (0 ± 1)-matrix. On input ψ , Algorithm 14 outputs “unsatisfiable” if and only if ψ is unsatisfiable, and if ψ is satisfiable, then the output set $M_1 \cup M_2$ is a model for ψ .*

Algorithm 14.

```

Monotone Decomposition Solver ( $\psi$ )
/* Input: CNF formula  $\psi$  as  $(0, \pm 1)$   $\mathcal{M}_\psi$  monotone decomposition */
/* Output: "unsatisfiable" or a model for  $\psi$  */
/* Locals: set of variables  $M_1, M_2$  */

    Let  $\mathcal{M}_\psi$  be partitioned according to (6).
    If Horn formula  $\mathcal{A}^1$  is unsatisfiable, Output "unsatisfiable."
    Let  $M_1$  be a unique minimum model for the Horn formula  $\mathcal{A}^1$ .
    Remove from  $\mathcal{A}^2$  all rows common to  $\mathcal{D}$ 's that are satisfied by  $M_1$ .
    If  $\mathcal{A}^2$  is unsatisfiable, Output "unsatisfiable."
    Let  $M_2$  be a model for  $\mathcal{A}^2$ .
    Output  $M_1 \cup M_2$ .

```

□

Fig. 40 Algorithm for determining satisfiability of a monotone decomposition

Proof Clearly, if Horn formula \mathcal{A}^1 is unsatisfiable, then so is ψ . So suppose there is a model M_1 for \mathcal{A}^1 and consider the rows of \mathcal{A}^2 remaining after rows common to those of \mathcal{D} which are satisfied by M_1 are removed. Since M_1 is a unique minimum model for \mathcal{A}^1 , no entries of \mathcal{D} are +1, and since variables of \mathcal{A}^1 are distinct from variables of \mathcal{A}^2 , no remaining row of \mathcal{A}^2 can be satisfied by any model for \mathcal{A}^1 . Therefore, if these rows of \mathcal{A}^2 are unsatisfiable, then so is ψ . On the other hand, if these rows are satisfied by model M_2 , then clearly, $M_1 \cup M_2$ is a model for ψ . □

A (0 ± 1) matrix \mathcal{M}_ψ representing CNF formula ψ may have more than one monotone decomposition. Of particular interest is the *maximum monotone decomposition* of \mathcal{M}_ψ , that is, the monotone decomposition of ψ such that \mathcal{A}^1 has the greatest number of rows and columns. A monotone decomposition is said to be maximal with respect to the dimensions of \mathcal{A}^1 . The following theorem says a unique maximal monotone decomposition is always possible.

Theorem 8 Any (0 ± 1) matrix \mathcal{M} has a unique maximal monotone decomposition.

Proof Suppose \mathcal{M} has two distinct maximal monotone decompositions, say, \mathcal{M}_1 and \mathcal{M}_2 . Let $\mathcal{A}_i^1, \mathcal{A}_i^2$, and \mathcal{D}_i , $i \in \{1, 2\}$, be the partition of \mathcal{M} , after column scaling, corresponding to \mathcal{M}_i (see the partition (6) on page 389). Construct a new partition \mathcal{M}' of \mathcal{M} into \mathcal{A}'^1 , \mathcal{A}'^2 , and \mathcal{D}' such that \mathcal{A}'^1 includes all rows and columns of \mathcal{A}_1^1 and \mathcal{A}_2^1 . For those columns of \mathcal{M}' that are also columns of \mathcal{A}_1^1 , use a column scaling that is exactly the same as the one used in \mathcal{M}_1 . For all other columns, use the same scaling as in \mathcal{M}_2 . The submatrix of \mathcal{A}'^1 that includes rows and columns of \mathcal{A}_1^1 is the same as \mathcal{A}_1^1 because the scaling of those columns is the same as for \mathcal{M}_1 . The submatrix of \mathcal{A}'^1 including rows of \mathcal{A}_1^1 and columns not in \mathcal{A}_1^1 must be a 0 submatrix by the monotone decomposition \mathcal{M}_1 . The submatrix of \mathcal{A}'^1 including columns of \mathcal{A}_1^1 and no rows of \mathcal{A}_1^1 must contain only 0 or -1 entries due to the \mathcal{M}_1 scaling and the submatrix including neither columns or rows of \mathcal{A}_1^1 must be Horn due to \mathcal{M}_2 column scaling. It follows that submatrix \mathcal{A}'^1 is Horn (at most one +1 in each row).

It is similarly easy to check that the submatrix of \mathcal{M}' consisting of rows of \mathcal{A}'^1 and columns other than those of \mathcal{A}'^1 is 0 and that the submatrix of \mathcal{M}' consisting of columns of \mathcal{A}'^1 and rows other than those of \mathcal{A}'^1 contains no +1 entries. It follows that \mathcal{M}' is a monotone decomposition. Since $\mathcal{A}_1^1 \supset \mathcal{A}'^1$ and $\mathcal{A}_2^1 \supset \mathcal{A}'^1$, neither \mathcal{M}_1 nor \mathcal{M}_2 is a maximal monotone decomposition in violation of the hypothesis. The theorem follows. \square

From [Theorem 8](#) there is always a maximum monotone decomposition for \mathcal{M}_ψ .

A maximum monotone decomposition is useful because (1) \mathcal{A}^1 , representing a Horn formula, is as large as possible so \mathcal{A}^2 is as small as possible; (2) Horn formulas may be efficiently solved by [Algorithm 20](#) and (3) a maximum monotone decomposition can be found efficiently, as will now be shown.

A maximum monotone decomposition can be found using [Algorithm 15](#) of [Fig. 41](#). The algorithm completes one or more stages where each stage produces a proper monotone decomposition of some matrix. All submatrices change dimensions during the algorithm, so primes are used as in \mathcal{E}' to refer to the current incarnation of corresponding submatrices. Initially, that matrix is \mathcal{M}_ψ . At the end of a stage, if the algorithm needs another stage to produce a bigger decomposition, \mathcal{A}^2 of the current stage becomes the entire input of the next stage and the next stage proceeds independently of previous stages. This can be done since the operation to be mentioned next does not multiply by -1 any of the rows and columns of the \mathcal{A}'^1 and \mathcal{D}' matrices of previous stages. The important operation is to move a nonpositive column that intersects \mathcal{A}'^2 to just right of the border of the current stage's \mathcal{A}'^1 matrix, move the border of \mathcal{A}'^1 and \mathcal{D}' to the right by one column, tag and move the rows containing 1 on the right boundary of the changed \mathcal{D}' up to just below the border of \mathcal{A}'^1 , and finally lower the border of \mathcal{A}'^1 and \mathcal{E}' down to include the tagged rows. Doing so keeps \mathcal{A}'^1 Horn and \mathcal{D}' nonpositive and enlarges \mathcal{A}'^1 . If no nonpositive column exists through \mathcal{A}'^2 , no column can be made nonpositive through \mathcal{A}'^2 by a -1 multiplication, and the initial moved column is not multiplied by -1 , then the initial moved column of the stage is multiplied by -1 and the stage is restarted.

Because of the following theorem, backtracking is limited to just one per stage and is used only to try to decompose with the initial moved column of the stage multiplied by -1 .

Theorem 9 Refer to [Algorithm 15](#) for specific variable names and terms:

1. If z is not a nonpositive column in \mathcal{E}' and z multiplied by -1 is not nonpositive in \mathcal{E}' , then there is no monotone decomposition at the current stage with the initial moved column v of the stage left as is.
2. If multiplying v by -1 also fails because a z cannot be made nonpositive in \mathcal{E}' , then not only does z block a monotone decomposition but multiplying any of the other columns in \mathcal{A}'^1 except v by -1 blocks a monotone decomposition as well.

Proof

1. There is no way to extend the right boundary of \mathcal{A}'^1 and stay Horn while making \mathcal{E}' 0 because column z prevents it.

Algorithm 15.

```

Find Maximum Monotone Decomposition ( $\psi$ )
/* Input: CNF formula  $\psi$  as  $(0, \pm 1)$  matrix  $\mathcal{M}_\psi$  */ 
/* Output: A maximum monotone decomposition of  $\mathcal{M}_\psi$  */ 
/* Locals: set of variables  $M_1, M_2$ , set of unusable literals  $N, L$  */ 

    Set  $N \leftarrow \emptyset$ .
    Set  $\mathcal{A}^2 \leftarrow \mathcal{M}_\psi$ .
    Repeat while there is a column  $v$  of  $\mathcal{A}^2$  such that  $v \notin N$  or  $\neg v \notin N$ :
        Remove 0 rows from  $\mathcal{A}^2$ .
        Choose any  $v$  such that either  $v \notin N$  or  $\neg v \notin N$ .
        Set  $L \leftarrow \emptyset$  and  $\alpha \leftarrow 1$ .
        If  $v \in N$ :
            Multiply all entries in column  $v$  of  $\mathcal{A}^2$  by  $-1$ .
            Set  $\alpha \leftarrow -1$ .
            Set  $N \leftarrow N \setminus \{v\} \cup \{\neg v\}$ .
        Set  $p \leftarrow v$ .
        Define  $\mathcal{A}'^1 = \mathcal{D}' = \mathcal{E}' = 0$ ,  $\mathcal{A}'^2 = \mathcal{A}^2$ , the initial partition of  $\mathcal{A}^2$ .
        Repeat the following:
            Move column  $p$  of  $\mathcal{A}'^2$  to the right border of  $\mathcal{A}'^1$ .
            Move the right border of  $\mathcal{A}'^1$  to the right by 1 column.
            Move and tag rows of  $\mathcal{D}'$  with 1 in its right column to the top.
            Move the bottom border of  $\mathcal{A}'^1$  down to include tagged rows.
            If  $\mathcal{E}' = 0$ , Set  $\mathcal{A}^2 \leftarrow \mathcal{A}'^2$  and Break.
            Choose column  $z$  through  $\mathcal{E}'$  with a non-zero entry in  $\mathcal{E}'$ .
            If  $(z \in N \text{ and } \neg z \in N) \text{ or}$ 
                 $(z \in N \text{ and column } z \text{ has } -1 \text{ entry}) \text{ or}$ 
                 $(\neg z \in N \text{ and column } z \text{ has } +1 \text{ entry}) \text{ or}$ 
                 $(\text{column } z \text{ has } +1 \text{ and } -1 \text{ entries}):$ 
                If  $\neg v \in N$  or  $\alpha = -1$ : Set  $N \leftarrow N \cup \{v, \neg v, z, \neg z\} \cup L$ .
                Break.
            Otherwise,
            If  $\neg z \notin N$  and column  $z$  has no  $-1$  entries:
                Multiply all entries in column  $z$  of  $\mathcal{A}^2$  by  $-1$ .
                If  $z \in N$ : Set  $N \leftarrow N \setminus \{z\} \cup \{\neg z\}$ .
                If  $\neg z \in L$ : Set  $L \leftarrow L \setminus \{\neg z\} \cup \{z\}$ .
                Set  $L \leftarrow L \cup \{\neg z\}$ .
                Set  $p \leftarrow z$ .
            Remove 0 rows from  $\mathcal{A}^2$ .
            Let  $\mathcal{M}$  be the permuted and scaled  $\mathcal{M}_\psi$  with lower right matrix  $\mathcal{A}^2$ .
            Output  $\mathcal{M}$ .

```

□

Fig. 41 Algorithm for finding the maximum monotone decomposition of a (0 ± 1) matrix

2. Consider columns in \mathcal{A}'^1 first. The proof is by induction on the number of columns processed in \mathcal{A}'^1 . The base case has no such column: that is, \mathcal{A}'^1 only contains the column v and is trivially satisfied. For the inductive step, change the column scaling to 1 for all columns and run the algorithm in the same column order it had been when it could not continue. Assume the hypothesis holds to k columns and consider processing at the $k + 1$ st column, call it column x . At this point \mathcal{A}'^1 has one 1 in each row, \mathcal{D}' is nonpositive, and since x is multiplied by 1, it is nonzero and nonpositive through \mathcal{E}' . If there is a monotone decomposition where x is multiplied by -1 , then x goes through \mathcal{A}^1 of that decomposition. The multiplication by -1 changes the nonzero nonpositive elements of x through \mathcal{E}' to nonzero nonnegative elements. Therefore, at least one of these elements, say, in row r , is $+1$. But \mathcal{A}^1 of the decomposition must have a $+1$ in each row, so it must be that row r has this $+1$ in, say, column c , a column of \mathcal{A}'^1 that is multiplied by -1 . But c cannot be the same as v since v multiplied by -1 blocks a monotone decomposition by hypothesis. On the other hand, if c is not v , then by the inductive hypothesis c cannot be multiplied by -1 in a monotone decomposition. Therefore, by contradiction, there is no monotone decomposition and the hypothesis holds to $k + 1$ columns.

Now consider column z . No scaling of column z can make z nonpositive in \mathcal{E}' . Then that part of z that goes through \mathcal{E}' has -1 and 1 entries. The hypothesis follows from the same induction argument as above. \square

Any column blocking a monotone decomposition need never be checked again.

The algorithm keeps track of blocking columns with set N , the long-term record of blocking columns, and set L , the temporary per stage record. If column indicator w is placed in N , it means the unmultiplied column w blocks, and if $-w$ is placed in N , it means column w multiplied by -1 blocks.

The algorithm has quadratic complexity. Complexity can be made linear by running the two possible starting points of each stage, namely, using column v as is and multiplying column v by -1 , concurrently, and breaking off computation when one of the two succeeds.

A formula ψ which has a maximum monotone decomposition where \mathcal{A}^2 is a member of an efficiently solved subclass of satisfiability obviously may be solved efficiently by [Algorithm 14](#) if \mathcal{A}^2 can efficiently be recognized as belonging to such a subclass. [Section 6](#) discusses several efficiently solved subclasses of satisfiability problems which may be suitable for testing. If \mathcal{A}^2 represents a 2-SAT formula (see [Sect. 6.1](#)), then ψ is said to be q-Horn. The class of q-Horn formulas was discovered and efficiently solved in [\[21, 22\]](#), and it was the results of that work that led to the development of maximum monotone decompositions [\[137\]](#).

5.10.2 Autarkies and Safe Assignments

Definition 1 An assignment to a set of variables is said to be *autark* or an *autarky* if all clauses that contain at least one of those variables are satisfied by the assignment. We will call a set of variables that is associated with an autarky an *autark set*.

If all clauses satisfied by an autarky are removed from a CNF formula ψ , then the resulting formula is equivalent in satisfiability to ψ . A simple example of an autarky set is a collection of one or more pure literals. A simple decomposition is to remove all clauses that contain pure literals. One can do the same for any autarky. However, discovering autarkies can be expensive. In some cases, though, an autarky can be found in polynomial time. This is treated in Sect. 6.9.

A similar decomposition exists for BDDs. Let f be a Boolean function and let $f|_v$ ($f|_{\neg v}$) denote the function obtained by setting variable v to 1 (respectively, 0).

Lemma 4 ([146]) *Given a conjunction of BDDs $f = f_1 \wedge \dots \wedge f_m$ and variable v occurring in one or more BDDs of f , let f' be the conjunction of all BDDs in f which contain v . Let $f'_v = \neg(f'|_v) \wedge (f'|_{\neg v})$. Let $f'_{\neg v} = (f'|_v) \wedge \neg(f'|_{\neg v})$.*

1. *If f'_v has value 0, then $f|_v$ is satisfiable if and only if f is satisfiable.*
2. *If $f'_{\neg v}$ has value 0, then $f|_{\neg v}$ is satisfiable if and only if f is satisfiable.*

Lemma 4 states that if any of the BDDs in f are falsified or if all of the BDDs in f are satisfied by setting v to 1 (respectively, 0), then it is safe to make that assignment because the satisfiability of f does not change by doing so. We emphasize that the safe value for v is *not necessarily inferred*. This lemma provides a way to test whether a safe value exists for a variable, that is, if $f'_v (f'_{\neg v})$ has value 0, then it is safe to set v to 1 (0) in f . However, to use this lemma directly requires conjoining all BDDs containing v , and from Sect. 5.9, this could be expensive. In pursuit of an efficient method for finding safe assignments, **Lemma 4** may be used to derive the following weaker result:

Theorem 10 ([146]) *Given a conjunction of BDDs $f = f_1 \wedge \dots \wedge f_m$ and variable v occurring in one or more BDDs of f , let f' be the conjunction of all BDDs in f which contain v . Without loss of generality, suppose $f' = f_1 \wedge \dots \wedge f_n$ where $n \leq m$. Let $f'_v = (\neg(f_1|_v) \wedge f_1|_{\neg v}) \vee \dots \vee (\neg(f_n|_v) \wedge f_n|_{\neg v})$. Let $f'_{\neg v} = (f_1|_v \wedge \neg(f_1|_{\neg v})) \vee \dots \vee (f_n|_v \wedge \neg(f_n|_{\neg v}))$.*

1. *If f'_v has value 0, then $f|_v$ is satisfiable if and only if f is satisfiable.*
2. *If $f'_{\neg v}$ has value 0, then $f|_{\neg v}$ is satisfiable if and only if f is satisfiable.*

According to **Theorem 10**, a safe assignment may be found without having to conjoin BDDs containing v . This is practical when BDDs are fairly small and is more practical than conjoining BDDs if v appears in many of them. However, it is possible that a safe assignment discovered using **Lemma 4** may be undiscoverable using **Theorem 10**.

The following is the corresponding theorem for safe assignments involving more than one variable:

Theorem 11 ([146]) *Given a conjunction of BDDs $f = f_1 \wedge \dots \wedge f_m$ and a set of variables $V = \{v_1, \dots, v_k\}$ each occurring in one or more BDDs of f , let f' be the conjunction of all BDDs in f which contain at least one of the variables in V . Let $\mathcal{M} = \{M_1, \dots, M_{2^k}\}$ be the set of all possible truth assignments to the variables*

in V . Without loss of generality, suppose $f' = f_1 \wedge \dots \wedge f_n$ where $n \leq m$. $\forall_{1 \leq i \leq 2^k}$ if $(\neg(f' |_{M_i}) \wedge (f' |_{M_1} \vee \dots \vee f' |_{M_{2^k}}))$ has value 0, then $f' |_{M_i}$ is satisfiable if and only if f' is satisfiable.

It is straightforward to turn [Theorems 10](#) and [11](#) into an algorithm, but this is not done here because numerous variants to speed up the search are possible and it is impractical to list all of them.

5.11 Branch and Bound

The DPLL algorithm of [Fig. 17](#) is sequential in nature. At any point in the search, only one node, representing a partial assignment $M_{1:}$, of the search tree is *active*: that is, open to exploration. This means exploration of a promising branch of the search space may be delayed for a potentially considerable period until search finally reaches that branch. Branch and bound aims to correct this to some extent. In branch and bound, quite a few nodes of the search space may be active at any point in the search. Each of the active nodes has a number $l(M_{1:})$ which is an aggregate estimate of how close the assignment represented at a node is to a solution or confirms that assignment cannot be extended to a *best* solution. Details concerning how $l(M_{1:})$ is computed will follow. A variable v is chosen for assignment from the subformula of the active node of lowest l value and that node is expanded. The expansion eliminates one active node and may create up to two others, one for each value to v . To help control the growth of active nodes, branch and bound maintains a monotonically decreasing number u for preventing nodes known to be unfruitful from becoming active. If the l value of any potential active node is greater than u , it is thrown out and not made active. Eventually, there are no active nodes left and the algorithm completes.

Branch and bound is intended to solve more problems than SAT, one of the most important being MAX-SAT ([Page 319](#)). It requires a function $g_\psi(M_{1:})$ which maps a given formula ψ and partial or total truth assignment $M_{1:}$ to a nonnegative number. The objective of branch and bound is to return an assignment M such that $g_\psi(M)$ is minimum over all truth assignments, partial or complete. That is,

$$M : \forall_x, g_\psi(X) \geq g_\psi(M).$$

For example, if $g_\psi(M_{1:})$ is just the number of clauses in $\psi_{M_{1:}}$, then branch and bound seeks to find M which satisfies the greatest number of clauses, maybe all.

Branch and bound discovers and discards search paths that are known to be fruitless, before they are explored, by means of a heuristic function $h(\psi_{M_{1:}})$ where $\psi_{M_{1:}}$ is obtained from ψ by removing clauses satisfied by and literals falsified by $M_{1:}$. The heuristic function returns a nonnegative number which, when added to $g_\psi(M_{1:})$, is a lower bound on $g(\psi_X)$ over all possible extensions X to $M_{1:}$. That sum is the $l(M_{1:})$ that was referred to above. That is,

Algorithm 16.

```

Branch-and-bound ( $\psi, h, g_\psi$ )
/* Input:  $n$  variable CNF formula  $\psi$ , heuristic function  $h$ ,          */
/*           objective function  $g_\psi$  mapping partial assignments to  $Z$       */
/* Output: Assignment  $M$  such that  $g_\psi(M)$  is minimized                 */
/* Locals: Integer  $u$ , priority queue  $P$                                 */

    Set  $M \leftarrow \emptyset$ ; Set  $M_{1:0} \leftarrow \emptyset$ ; Set  $u \leftarrow \infty$ .
    Insert  $P \leftarrow \langle\langle\psi, M_{1:0}\rangle, 0\rangle$ .
    Repeat the following while  $P \neq \emptyset$ :
        Pop  $\langle\psi', M_{1:i}\rangle \leftarrow P$ .
        If  $\psi' = \emptyset$  then Output  $M$ .
        Choose variable  $v$  from  $\psi'$ .
        Set  $\psi'_1 \leftarrow \{c \setminus \{v\} : c \in \psi'\text{ and } \neg v \notin c\}$ .
        Set  $\psi'_2 \leftarrow \{c \setminus \{\neg v\} : c \in \psi'\text{ and } v \notin c\}$ .
        Repeat the following for  $j = 1$  and  $j = 2$ :
            If  $j = 1$  then do the following:
                Set  $M_{1:i+1} \leftarrow M_{1:i}$ .
            Otherwise
                Set  $M_{1:i+1} \leftarrow M_{1:i} \cup \{v\}$ .
            If  $h(\psi'_j) + g_\psi(M_{1:i+1}) < u$  then do the following:
                Insert  $P \leftarrow \langle\langle\psi'_j, M_{1:i+1}\rangle, h(\psi'_j) + g_\psi(M_{1:i+1})\rangle$ .
                If  $g_\psi(M_{1:i+1}) < u$  then do the following:
                    Set  $u \leftarrow g_\psi(M_{1:i+1})$ .
                    Set  $M \leftarrow M_{1:i+1}$ .
        Output  $M$ .
    □

```

Fig. 42 Classic branch-and-bound procedure adapted to satisfiability

$$l(M_{1:}) = h(\psi_{M_{1:}}) + g_\psi(M_{1:}) \leq \min\{g_\psi(X) : X \text{ is an extension of } M_{1:}\}.$$

The algorithm maintains a number u that records the lowest g_ψ value that has been seen so far during search. If partial assignment $M_{1:i}$ is extended by one variable to $M_{1:i+1}$ and $g_\psi(M_{1:i+1}) < u$ then u , is updated to that value and $M_{1:i+1}$, the assignment that produced that value, is saved as M . In that case, $l(M_{1:i+1})$ must also be less than u because it is less than $g_\psi(M_{1:i+1})$. But if $l(M_{1:i+1}) > u$, then there is no chance, by definition of $l(M_{1:})$, that any extension to $M_{1:i+1}$ will yield the minimum g_ψ . Hence, if that test succeeds, the node that would correspond to $M_{1:i+1}$ is thrown out, eliminating exploration of that branch.

The algorithm in its general form for satisfiability is shown in Fig. 42 as **Algorithm 16**. A priority queue P is used to hold all active nodes as pairs where each pair contains a reduced formula and its corresponding partial assignment. Pairs are stored in P in increasing order of $l(M_{1:})$. It is easy to see, by definition of $l(M_{1:})$, that no optimal assignment gets thrown out. It is also not difficult to see that, given two heuristic functions h_1 and h_2 , if $h_1(\psi_{M_{1:}}) \leq h_2(\psi_{M_{1:}})$ for all M , then the search

explored using h_1 will be no larger than the search space explored using h_2 . Thus, to keep the size of the search space down, as tight a heuristic function as possible is desired. However, since overall performance is most important and since tighter heuristic functions typically mean more overhead, it is sometimes more desirable to use a weaker heuristic function which generates a larger search space in less time. [Section 5.12.2](#) shows how linear programming relaxations of integer programming representations of search nodes can be used as heuristic functions. This section concludes with an alternative to illustrate what else is possible.

Recall the problem of variable-weighted satisfiability which was defined in [Sect. 2](#): given CNF formula ψ and positive weights on variables, find a satisfying assignment for ψ , if one exists, such that the sum of weights of variables of value 1 is minimized. Let ψ_{M_1} be defined as above and let $Q_{\psi_{M_1}}$ be a subset of positive clauses of ψ_{M_1} such that no variable appears twice in $Q_{\psi_{M_1}}$. For example, $Q_{\psi_{M_1}}$ might look like this:

$$(v_1 \vee v_3 \vee v_7) \wedge (v_2 \vee v_6) \wedge (v_4 \vee v_5 \vee v_8).$$

A strictly lower bound on the minimum weight solution over all extensions to M_1 is $g_\psi(M_1) + h(\psi_{M_1})$, the sum of the weights of the minimum weight variable in each of the clauses of $Q_{\psi_{M_1}}$. Clearly, this is not a very tight bound. But it is computationally fast to acquire this bound, and the trade-off of accuracy for speed often favors this approach [61], particularly when weight calculations are made incrementally. Additionally, there are some tricks that help to find a *good* $Q_{\psi_{M_1}}$. For example, a greedy approach may be used as follows: choose a positive clause c with variables independent of clauses already in $Q_{\psi_{M_1}}$ and such that the ratio of the weight of the minimum weight variable in c to the number of variables in c is maximum [108]. The interested reader can consult [108] and [41] for additional ideas.

5.12 Algebraic Methods

A collection of Boolean constraints may be expressed as a system of algebraic equations or inequalities which has a solution if and only if the constraints are satisfiable. The attraction of these methods is that, in some cases, a single algebraic operation can simulate a large number of resolution operations. The problem is that it is not always obvious how to choose the optimal sequence of operations to take advantage of this, and often performance is disappointing due to nonoptimal choices.

5.12.1 Gröbner Bases Applied to SAT

It is interesting that at the same time resolution was being developed and understood as a search tool in the 1960s, an algebraic tool for computing a basis for highly *nonlinear* systems of equations was introduced: the basis it found was given the name Gröbner basis and the tool was called the Gröbner basis algorithm [29].

But it was not until the mid-1990s that Gröbner bases crossed paths with satisfiability when it was shown that Boolean expressions can be written as systems of multilinear equations which a simplified Gröbner basis algorithm can solve *using a number of derivations that is guaranteed to be within a polynomial of the minimum number possible* [37]. Also in that paper, it is shown that the minimum number of derivations cannot be much greater than, and may sometimes be far less than, the minimum number needed by resolution. Such powerful results led the authors to say “these results suggest the Gröbner basis algorithm might replace resolution as a basis for heuristics for NP-complete problems.”

This has not happened, perhaps partly because of the advances in the development of CNF SAT solvers in the 1990s and partly because, as is the case for resolution, it is generally difficult to find a minimum sequence of derivations leading to the desired conclusion. However, the complementary nature of algebraic and logic methods makes them an important alternative to resolution. Generally, in the algebraic world, problems that can be solved essentially using Gaussian elimination with a small or modest increase in the degree of polynomials are easy. A classic example where this is true is the systems of equations involving only the exclusive-or operator. By contrast, just expressing the exclusive-or of n variables in CNF requires 2^{n-1} clauses.

In the algebraic proof system outlined here, facts are represented as multilinear equations and new facts are derived a database of existing facts using rules described below. Let $\langle c_0, c_1, \dots, c_{2^n-1} \rangle$ be a 0-1 vector of 2^n coefficients. For $0 \leq j < n$, let $b_{i,j}$ be the j th bit in the binary representation of the number i . An input to the proof system is a set of equations of the following form:

$$\sum_{i=0}^{2^n-1} c_i v_1^{b_{i,0}} v_2^{b_{i,1}} \dots v_n^{b_{i,n-1}} = 0, \quad (7)$$

where all variables v_i can take value 0 or 1, and addition is taken modulo 2. An equation of the form (7) is said to be multilinear. A product $t_i = v_1^{b_{i,0}} v_2^{b_{i,1}} \dots v_n^{b_{i,n-1}}$, for any $0 \leq i \leq 2^n - 1$, will be referred to as a multilinear term or simply a term. The degree of t_i , denoted $\deg(t_i)$, is $\sum_{0 \leq j < n} b_{i,j}$. A term that has a coefficient of value 1 in an equation is said to be a nonzero term of that equation.

New facts may be derived from known facts using the following rules:

1. Any even sum of like nonzero terms in an equation may be replaced by 0. Thus, $v_1v_2 + v_1v_2$ reduces to 0 and $1 + 1$ reduces to 0. This reduction rule is needed to eliminate terms when adding two equations (see below).
2. A factor v^2 in a term may be replaced by v . This reduction rule is needed to ensure terms remain multilinear after multiplication (see below).
3. An equation of the form (7) may be multiplied by a term, and the resulting equation may be reduced to the form (7) by rule 2 above. Thus, $v_3v_4(v_1 + v_3 = 0)$ becomes $v_1v_3v_4 + v_3v_4 = 0$.
4. Two equations may be added to produce an equation that may be reduced by rule 1 above to the form (7). Examples will be given below.

An equation that is created by rule 3 or 4 is said to be derived. All derived equations are reduced by rules 1 and 2 before being added to the proof.

Observe that the solution spaces of two equations are complementary if they differ only in that $c_0 = 0$ for one and $c_0 = 1$ for the other. For example, the sets of solutions for the two equations

$$\begin{aligned} v_1v_2v_3 + v_1v_2 + v_2v_3 + v_1 + 1 &= 0 \text{ and} \\ v_1v_2v_3 + v_1v_2 + v_2v_3 + v_1 &= 0 \end{aligned}$$

are complementary.

It is left as evident that performing operations rules 3 and 4 with reductions rules 1 and 2 as needed results in a set of derived equations whose solution space is a superset of the original set. The set of all possible derived equations has a solution space which is identical to that of the original set of equations.

Theorem 12 *The equation $1 = 0$ is always derivable using rules 3 and 4 (and implicitly rules 1 and 2) from an inconsistent input set of multilinear equations and never derived from a consistent set.*

Proof Assume $1 = 0$ is not one of the input equations. Suppose $1 = 0$ is derived from a consistent input set. Then two equations were added to derive $1 = 0$. But the solution space of both must be complementary and therefore the solution space of the entire system must be empty. That is not possible since application of rules 3 and 4 does not reduce the solution space below that of the original set of equations, and there is at least one solution because the input set is consistent.

Suppose $1 = 0$ is not derivable from an inconsistent input set. Re-index terms, with the term of degree 0 taking the lowest index, and construct a sequence of derivations such that no two derived equations have the same nonzero term. If the derivation cannot continue to the lowest (0th) term, then the resulting system of equations is linearly independent and therefore must have a solution. But that is impossible by assumption.

Re-indexing is as follows: terms of degree i all have higher index than terms of degree j if $i > j$; among terms of the same degree, the order of index is decided lexicographically. Call the equations ψ and create set B , initially empty. Repeat the following until ψ is empty. Pick an equation e of ψ that has the highest index, nonzero term. As long as there is an equation g in B whose highest nonzero term has the same index as the highest index nonzero term of e , replace e with $e + g$. If $0 = 0$ is not produced, add e to B . This ensures B remains linearly independent. Create as many as n new equations by multiplying e by every variable and add those equations to ψ that have never been in ψ . This sets up the addition of e with all other equations in B . When ψ is empty, all original equations have been replaced by equations with the same solution space and are such that no two of them have the same highest index nonzero term. \square

Next some examples of inputs are shown and then two short derivations. The CNF clause

$$(v_1 \vee v_2 \vee v_3)$$

is represented by the equation

$$v_1(1 + v_2)(1 + v_3) + v_2(1 + v_3) + v_3 + 1 = 0,$$

which may be rewritten

$$v_1v_2v_3 + v_1v_2 + v_1v_3 + v_2v_3 + v_1 + v_2 + v_3 + 1 = 0.$$

The reader can verify this from the truth table for the clause. Negative literals in a clause are handled by replacing variable symbol v with $(1 + v)$. For example, the clause

$$(\neg v_1 \vee v_2 \vee v_3)$$

is represented by

$$(1 + v_1)(1 + v_2)(1 + v_3) + v_2(1 + v_3) + v_3 + 1 = 0,$$

which reduces to

$$v_1v_2v_3 + v_1v_2 + v_1v_3 + v_1 = 0. \quad (8)$$

As can be seen, just the expression of a clause introduces nonlinearities. However, this is not the case for some Boolean functions. For example, the exclusive-or formula

$$v_1 \oplus v_2 \oplus v_3 \oplus v_4$$

is represented by

$$v_1 + v_2 + v_3 + v_4 + 1 = 0.$$

An equation representing a BDD (Sect. 5.9) can be written directly from the BDD as a sum of algebraic expressions constructed from paths to 1 because each path represents one or more rows of a truth table and the intersection of rows represented by any two paths is empty. Each expression is constructed incrementally while tracing a path as follows: when a 1 branch is encountered for variable v , multiply by v , and when a 0 branch is encountered for variable v , multiply by $(1 + v)$. Observe that for any truth assignment, at most one of the expressions has value 1. The equation corresponding to the BDD at the upper left in Fig. 28 is

$$(1 + v_1)(1 + v_2)(1 + v_3) + (1 + v_1)v_2v_3 + v_1(1 + v_2)v_3 + v_1v_2 + 1 = 0,$$

which reduces to

$$v_1 + v_2 + v_3 + v_1 v_2 v_3 = 0.$$

Since there is a BDD for every Boolean function, this example illustrates the fact that a single equation can represent any complex function. It should be equally clear that a single equation addition may have the same effect as many resolution steps.

Addition of equations and the Gaussian-elimination nature of algebraic proofs is illustrated by showing steps that solve the following simple formula:

$$(v_1 \vee \neg v_2) \wedge (v_2 \vee \neg v_3) \wedge (v_3 \vee \neg v_1). \quad (9)$$

The equations corresponding to (9) are expressed below as (1), (2), and (3). All equations following those equations are derived as stated on the right.

$$\begin{array}{rcccl}
 v_1 v_2 & & +v_2 & = 0 & (1) \\
 & v_2 v_3 & & +v_3 & = 0 \quad (2) \\
 v_1 v_3 & & +v_1 & = 0 & (3) \\
 \hline
 v_1 v_2 v_3 & & +v_2 v_3 & = 0 & (4) \Leftarrow v_3 \cdot (1) \\
 v_1 v_2 v_3 & & & +v_3 & = 0 \quad (5) \Leftarrow (4) + (2) \\
 v_1 v_2 v_3 & & +v_1 v_3 & = 0 & (6) \Leftarrow v_1 \cdot (2) \\
 v_1 v_2 v_3 & & & +v_1 & = 0 \quad (7) \Leftarrow (6) + (3) \\
 v_1 v_2 v_3 + v_1 v_2 & & & = 0 & (8) \Leftarrow v_2 \cdot (3) \\
 v_1 v_2 v_3 & & +v_2 & = 0 & (9) \Leftarrow (8) + (1) \\
 & v_1 & +v_2 & = 0 & (10) \Leftarrow (9) + (7) \\
 v_1 & & +v_3 & = 0 & (11) \Leftarrow (5) + (7)
 \end{array}$$

The solution is given by the bottom two equations which state that $v_1 = v_2 = v_3$. If, say, the following two clauses are added to (9)

$$(\neg v_1 \vee \neg v_2) \wedge (v_3 \vee v_1),$$

the equation $v_1 + v_2 + 1 = 0$ could be derived. Adding this to (10) would give $1 = 0$ which proves that no solution exists.

Ensuring a derivation of reasonable length is difficult. One possibility is to limit derivations to equations of bounded degree where the degree of a term t , $\deg(t)$, is defined in the proof of [Theorem 12](#) and the degree of an equation is $\text{degree}(e) = \max\{\deg(t) : t \text{ is a nonzero term in } e\}$. An example is [Algorithm 17](#) of [Fig. 43](#) which is adapted from [37]. In the algorithm terms are re-indexed as in [Theorem 12](#). Then $\text{first_non-zero}(e_i)$ is used to determine the highest index of a nonzero term of e_i . The function $\text{reduce}(e)$ is an explicit statement that says reduction rules 1 and 2 are applied as needed to produce a multilinear equation.

Algorithm 17.

```

An Algebraic Solver ( $\psi, d$ )
/* Input: List of equations  $\psi = \langle e_1, \dots, e_m \rangle$ , integer  $d$  */  

/* Output: “satisfiable” or “unsatisfiable” */  

/* Locals: Set  $B$  of equations */  

Set  $B \leftarrow \emptyset$ .  

Repeat while  $\psi \neq \emptyset$ :  

    Pop  $e \leftarrow \psi$ .  

    Repeat while  $\exists e' \in B : \text{first\_non-zero}(e) = \text{first\_non-zero}(e')$ :  

        Set  $e \leftarrow \text{reduce}(e + e')$ . /* Rule 4. */  

    If  $e$  is 1 = 0: Output “unsatisfiable”  

    If  $e$  is not 0 = 0:  

        Set  $B \leftarrow B \cup \{e\}$ .  

        If  $\text{degree}(e) < d$ :  

            Repeat for all variables  $v$ :  

                If  $\text{reduce}(ve)$  has not been in  $\psi$ :  

                    Append  $\psi \leftarrow \text{reduce}(ve)$ . /* Rule 3. */  

    Output “satisfiable”.  

□

```

Fig. 43 Simple algebraic algorithm for SAT

This section concludes by comparing equations and the algebraic method with BDDs and BDD operations. Consider an example taken from [Sect. 5.9](#). Equations corresponding to f and c in [Fig. 31](#) are

$$\begin{aligned} f : v_1v_3 + v_2 + v_1v_2 &= 0 \\ c : v_2v_3 + v_3 &= 0. \end{aligned}$$

Multiply f by v_2v_3 to get $v_2v_3 = 0$ which adds with c to get $v_3 = 0$, the inference that is missed by $\text{restrict}(f, c)$ in [Fig. 31](#). The inference can be derived from BDDs by reversing the role of f and c as shown in [Fig. 32](#). Consider what multiplying f by v_2v_3 and adding to c means in the BDD world. The BDD representing v_2v_3 , call it d , consists of two internal nodes v_2 and v_3 , a path to 0 following only 1 branches, and all other paths terminating at 1. Every path that terminates at 1 in f also terminates at 1 in d . Therefore, $d \wedge c$ can safely be added as a BDD as long as f remains. But it is easy to check that $d \wedge c$ is simply $v_3 = 0$.

The process used in the above example can be applied more generally. All that is needed is some way to create a *best* factor d from f and c . This is something a generalized cofactor, which is discussed in [Sect. 5.9.4](#), can sometimes do. However, the result of finding a generalized cofactor depends on BDD variable ordering. For the ordering $v_1 < v_2 < v_3$, the generalized cofactor $g = \text{gcf}(f, c)$ turns out to be $(v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_2 \vee \neg v_3)$ which is different from d in the leading clause but

is sufficient to derive the inference when conjoined with c . By the definition of **gcf**, since $f \wedge c = g \wedge c$, g may replace f – this is not the case for d above.

Existentially quantifying v away from a BDD has a simple counterpart in algebra: just multiply two polynomials, one with restriction $v = 1$ and the other with restriction $v = 0$. For example, the BDD of Fig. 28 may be expressed as

$$v_1 v_2 v_3 + v_1 v_3 + v_1 + 1 = 0.$$

The equations under restrictions $v_2 = 1$ and $v_2 = 0$, respectively, are

$$v_1 + 1 = 0 \quad \text{and} \quad v_1 v_3 + v_1 + 1 = 0.$$

The result of existential quantification is

$$(v_1 + 1)(v_1 v_3 + v_1 + 1) = v_1 + 1 = 0,$$

which reveals the same inference. As with BDDs, this can be done only if the quantified variable is in no other equation.

The counterpart to strengthening is just as straightforward. The BDDs of Fig. 39 have equation representations

$$b_2 : v_1 v_3 + v_2 + v_1 v_2 = 0$$

$$b_1 : v_3 + v_2 v_3 = 0.$$

Existentially quantify v_1 away from b_2 to get $v_2 v_3 = 0$ and add this to b_1 to get $v_3 = 0$.

5.12.2 Integer Programming

An integer program models the problem of maximizing or minimizing a linear function subject to a system of linear constraints, where all n variables are integral:

$$\begin{aligned} & \text{maximize or minimize } \mathbf{c} \alpha \\ & \text{subject to } \mathcal{M}\alpha \leq \mathbf{b} \\ & l \leq \alpha \leq u \\ & \alpha_i \text{ integral}, 1 \leq i \leq n, \end{aligned} \tag{10}$$

where \mathcal{M} is a constraint matrix, \mathbf{c} is a linear objection function, \mathbf{b} is a constant vector, and α is a variable vector.

The integer programming problem and its relaxation to linear programming are very well studied, and a large body of techniques have been developed to assist in establishing an efficient solution to (10). They are divided into the categories of preprocessing and solving. However, an important third aspect concerns the matrix \mathcal{M} that is used to model a given instance.

Modeling is important because the effective solution of integer programs often entails the use of linear programming relaxations. A solution to such a relaxation generally provides a bound on the actual solution, and the relaxation of one formulation of the input may provide a tighter bound than another. Generally, the tighter the bound, the better.

For example, consider two formulations of the pigeonhole problem. The pigeon-hole problem is as follows: can $n + 1$ pigeons be placed in n holes so that no two pigeons are in the same hole? Define Boolean variables $v_{i,j}$, $1 \leq i \leq n$, $1 \leq j \leq n + 1$ with the interpretation that $v_{i,j}$ will take the value 1 if and only if pigeon j is in hole i and otherwise will take value 0. The following equations, one per pigeon, express the requirement that every pigeon is to be assigned to a single hole:

$$\sum_{i=1}^n v_{i,j} = 1, \quad 1 \leq j \leq n + 1, \quad (11)$$

and the following inequalities express the requirement that two pigeons cannot occupy the same hole:

$$v_{i,j} + v_{i,k} \leq 1, \quad 1 \leq i \leq n, \quad 1 \leq j < k \leq n + 1. \quad (12)$$

There is no solution to this system of equations and inequalities. Relaxing integrality constraints, there is a solution at $v_{i,j} = 1/n$ for all i, j . If running the algorithm to be shown later, practically complete enumeration is necessary before finally it is determined that no solution exists [82]. However, the requirement that at most one pigeon is in a hole may alternatively be represented by

$$\sum_{j=1}^{n+1} v_{i,j} \leq 1, \quad 1 \leq i \leq n. \quad (13)$$

which may be used instead of (12). The new constraints are much tighter than (12) and the system (11) and (13) is easily solved [82].

The purpose of preprocessing is to reformulate a given integer program and tighten its linear programming relaxation. In the process it may eliminate redundant constraints and may even be able to discover unsatisfiability.

6 Algorithms for Easy Classes of CNF Formulas

For certain classes of CNF formulas, the satisfiability problem is known to be solved efficiently by specially designed algorithms. Some classes, such as the 2-SAT and Horn classes are quite important because they show up in real applications and others are quite interesting because results on these add to our understanding of

Algorithm 18.**Unit Resolution (ψ)**

```

/* Input: set of sets CNF formula  $\psi$  */  

/* Output: pair  $\langle$  CNF formula  $\phi$ , partial assignment  $P$   $\rangle$  */  

/*  $\psi$  is satisfiable if and only if  $\phi$  is satisfiable */  

/*  $\phi$  has no unit clauses */  

/* Locals: set of variables  $P$ , set of sets CNF formula  $\phi$  */  

Set  $\phi \leftarrow \psi$ ; Set  $P \leftarrow \emptyset$ .  

Repeat the following while  $\emptyset \notin \phi$  and there is a unit clause in  $\phi$ :  

    Let  $\{l\} \in \phi$  be a unit clause.  

    If  $l$  is a positive literal, Set  $P \leftarrow P \cup \{l\}$ .  

    Set  $\phi \leftarrow \{c - \{\neg l\} : c \in \phi, l \notin c\}$ .  

Output  $\langle \phi, P \rangle$ .

```

□

Fig. 44 Unit resolution for CNF formulas

the structural properties that make formulas hard or easy. Such an understanding can help develop a search heuristic that will obtain a solution more efficiently. In particular, knowledge that a large subset of clauses of a given formula belongs to some easy class of formulas can help reduce the size of the search space needed to determine whether some partial truth assignment can be extended to a solution. Because CNF formulas are so rich in structure, much space in this section is devoted to a few special cases. Hopefully this will help to fully appreciate the possibilities. In particular, results on minimally unsatisfiable and nested formulas are greatly detailed.

The reader may have the impression that the number of polynomial-time solvable classes is quite small due to the famous dichotomy theorem of Schaefer [121]. But this is not the case. Schaefer proposed a scheme for defining classes of propositional formulas with a generalized notion of *clause*. He proved that every class definable within his scheme was either \mathcal{NP} -complete or polynomial-time solvable, and he gave criteria to determine which. But not all classes can be defined within his scheme. The class of Horn formulas can be but several others, which will be described in this section, including q-Horn, extended Horn, CC-balanced, and SLUR cannot be so defined. The reason is that Schaefer's scheme is limited to classes that can be recognized in log space.

Below, some of the more notable easy classes and algorithms for solving them are presented. A crucial component of many of these algorithms is *unit resolution*. An implementation is given in Fig. 44.

6.1 2-SAT

Every clause of a 2-SAT formula contains at most two literals. A given 2-SAT formula ψ may be solved efficiently by constructing the implication graph \vec{G}_ψ

Algorithm 19.

```

2-SAT Solver ( $\psi$ )
/* Input: set of sets 2-CNF formula  $\psi$  */  

/* Output: “unsatisfiable” or a model for  $\psi$  */  

/* Locals: variable  $s$ , set of variables  $M, M'$ , set of sets formula  $\phi$  */  

Set  $\phi \leftarrow \psi$ ; Set  $s \leftarrow 1$ ; Set  $M \leftarrow \emptyset$ ; Set  $M' \leftarrow \emptyset$ ; Set  $\phi' \leftarrow \emptyset$ .  

Repeat the following until some statement outputs a value:  

    Set  $\langle \phi, M' \rangle \leftarrow \text{Unit Resolution } (\phi)$ .  

    If  $\emptyset \in \phi$  then do the following:  

        If  $s$  has value 1 or  $\phi' = \emptyset$  then Output “unsatisfiable”.  

        Set  $s \leftarrow 1$   

        Set  $M' \leftarrow \emptyset$ ;  $\phi \leftarrow \phi'$ ;  $l \leftarrow l'$ .  

        If  $l$  is a negative literal, Set  $M' \leftarrow \neg l$ .  

        Set  $\phi \leftarrow \{c - \{l\} : c \in \phi, \neg l \notin c\}$ .  

    Otherwise, if  $\phi \neq \emptyset$  then do the following:  

        Set  $s \leftarrow 0$ .  

        Choose a literal  $l$  arbitrarily from a clause of  $\phi$ .  

        Set  $M \leftarrow M \cup M'$ .  

        Set  $M' \leftarrow \emptyset$ .  

        Set  $\phi' \leftarrow \phi$ .  

        Set  $l' \leftarrow l$ .  

        If  $l$  is a positive literal, Set  $M' \leftarrow \{l\}$ .  

        Set  $\phi \leftarrow \{c - \{\neg l\} : c \in \phi, l \notin c\}$ .  

    Otherwise, Output  $M \cup M'$ .

```

□

Fig. 45 Algorithm for determining satisfiability of 2-CNF formulas

of ψ (see Sect. 3.3) and traversing its vertices, ending either at a cycle containing complementary literals or with no additional vertices to explore. The algorithm of Fig. 45 implicitly does this.

Unit resolution drives the exploration of strongly connected components of \vec{G}_ψ . The initial application of unit resolution, if necessary, is analogous to traversing all vertices reachable from the special vertex F . Choosing a literal l arbitrarily and temporarily assigning it the value 1 after unit resolution completes is analogous to starting a traversal of some strongly connected component with another round of unit resolution. If that round completes with an empty clause, a contradiction exists so l is set to 0, ϕ and M are reset to what they were just before l was set to 1, and exploration resumes. If an empty clause is encountered before the entire component is visited (i.e., while there still exist unit clauses), then the formula is unsatisfiable. Otherwise, the value of l is made permanent and so are values that were given to other variables during traversal of the component. This process repeats until the formula is found to be unsatisfiable or all components have been explored. The variable named s keeps track of whether variable l has been given one value or two,

the variable M' holds the temporary assignments to variables during traversal of a component, and the variables ϕ' and l' save the point to return to if a contradiction is found. This algorithm is adapted from [56].

The next two theorems are stated without proof.

Theorem 13 *On input CNF formula ψ , Algorithm **2-SAT Solver** outputs “unsatisfiable” if and only if ψ is unsatisfiable, and if it outputs a set M , then M is a model for ψ .* \square

Theorem 14 *On input CNF formula ψ containing m clauses and n variables, Algorithm **2-SAT Solver** has $O(m + n)$ worst-case complexity.* \square

6.2 Horn Formulas

A CNF formula is Horn if every clause in it has at most one positive literal. This class is widely studied, in part because of its close association with logic programming. To illustrate, a Horn clause $(\neg v_1 \vee \neg v_2 \vee \dots \vee \neg v_i \vee v)$ is equivalent to the rule $v_1 \wedge v_2 \wedge \dots \wedge v_i \rightarrow v$ or the implication $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_i \rightarrow v$. However, the notion of causality is generally lost when translating from rules to Horn formulas.

The following states an important property of Horn formulas.

Theorem 15 *Every Horn formula has a unique minimum model.*

Proof Let ψ be a Horn formula. Let M be a minimal model for ψ , that is, a smallest subset of variables of value 1 that satisfies ψ . Choose any $v \in M$. Since $M \setminus \{v\}$ is not a model for ψ , there must be a clause $c \in \psi$ such that positive literal $v \in c$. Since all other literals of c are negative and M is minimal, all assignments not containing v cannot satisfy c and therefore ψ . It follows that all models other than M must have cardinality greater than $|M|$. Hence, M is a unique minimum model for ψ . \square

The satisfiability of Horn formulas can be determined in linear time using unit resolution [53, 78, 125]. One of several possible variants is shown in Fig. 46.

Theorem 16 *Given Horn formula ψ as input, Algorithm **Horn Solver** outputs “unsatisfiable” if and only if ψ is unsatisfiable, and if it outputs a set of variables M , then M is a unique minimum model for ψ .*

Proof When Algorithm **Horn Solver** completes with output set M , all remaining clauses have at least one negative literal. Since none of the remaining clauses are null and since v added to M serves to falsify negative literals, at least one of the remaining negative literals in a remaining clause has not been added to M and is therefore satisfied by M . Therefore, all remaining clauses are satisfied by M .

Algorithm 20.

```

Horn Solver ( $\psi$ )
/* Input: set of sets Horn formula  $\psi$  */  

/* Output: “unsatisfiable” or a model for  $\psi$  */  

/* Locals: set of variables  $M$  */  

Set  $M \leftarrow \emptyset$ .  

Repeat the following until no positive literal unit clauses are in  $\psi$ :  

    Choose  $v$  from a positive literal unit clause  $\{v\} \in \psi$ .  

    Set  $M \leftarrow M \cup \{v\}$ .  

    Set  $\psi \leftarrow \{c - \{\neg v\} : c \in \psi, v \notin c\}$ .  

    If  $\emptyset \in \psi$ , Output “unsatisfiable.”  

Output  $M$ .

```

□

Fig. 46 Algorithm for determining satisfiability of Horn formulas

A clause is removed when adding v to M only because it contains literal v . Therefore, all removed clauses are satisfied by M . Hence, M satisfies ψ .

Suppose M is not the unique minimum model for ψ . Then, for some variable v , the assignment $M \setminus \{v\}$ satisfies ψ . Variable v was added to M because some clause $c \in \psi$ containing positive literal v became a unit clause after all variables associated with the negative literals of c were placed in M . But then $M \setminus \{v\}$ cannot satisfy c . Therefore, M is the unique minimum model.

Now suppose the algorithm outputs *unsatisfiable* but there is a model for ψ . Let M' be the unique minimum model. Run the algorithm until reaching the point at which an empty clause is generated. Let this happen on the i th iteration of the Repeat block and let ψ' be the set of all clauses removed up to the test for an empty clause on the i -first iteration. Let v be the last variable added to M and c be the unit clause from which it was obtained. Clearly, ψ' is Horn and $M \setminus \{v\}$ is its unique minimum model. M' must contain all variables of $M \setminus \{v\}$ since it is the unique minimum model for ψ . Therefore, it cannot contain $\{v\}$. But then c is not satisfied by M' , a contradiction. □

Algorithm **Horn Solver** is only useful if the input formula is known to be Horn. It is easy to see that this can be checked in linear time.

6.3 Renamable Horn Formulas

Given CNF formula ψ and variable subset $V'_\psi \subset V_\psi$, define $switch(\psi, V'_\psi)$ to be the formula obtained from ψ by reversing the polarity of all occurrences of v and $\neg v$ in ψ for all $v \in V'_\psi$. If there exists a $V'_\psi \subset V_\psi$ such that $switch(\psi, V'_\psi)$ is Horn, then ψ is said to be *renamable Horn* or *hidden Horn*.

Renamable Horn formulas can be recognized and solved in $O(|\psi|)$ time [10, 104]. Algorithm **SLUR** on Page 410 solves renamable Horn formulas in linear time.

6.4 Linear Programming Relaxations

Let \mathcal{M}_ψ be a (0 ± 1) matrix representing a CNF formula ψ . If \mathcal{M}_ψ has a particular structure, it is possible to solve the inequalities (1) with non-integer constraints $0 \leq \alpha_i \leq 1$ to obtain a solution to ψ either directly or by rounding. Notable classes based on particular matrix structures are the extended Horn formulas and what in this chapter are called the CC-balanced formulas.

The class of *extended Horn formulas* was introduced by Chandru and Hooker [31] who were looking for conditions under which a linear programming relaxation could be used to find solutions to propositional formulas. It is based on a theorem of Chandrasekaran [30] which characterizes sets of linear inequalities for which 0-1 solutions can always be found (if one exists) by rounding a real solution obtained using an LP relaxation. Extended Horn formulas can be expressed as linear inequalities that belong to this family of 0-1 problems.

For the sake of clarity, an equivalent graph theoretic definition is presented here. A formula ψ is in the class of extended Horn formulas if one can construct a rooted directed tree T , called an extended Horn tree, indexed on the variables of ψ such that, for every clause $c \in \psi$:

1. All the positive literals of c are consecutive on a single path of T .
2. There is a partition of the negative literals of c into sets N_1, N_2, \dots, N_{n_c} , where n_c is at least 1, but no greater than the number of negative literals of c , such that for all $1 \leq i \leq n_c$, all the variables of N_i are consecutive on a single path of T .
3. For at most one i , the path in T associated with N_i begins at the vertex in T from which the path associated with positive literals begins.
4. For all remaining i , the path in T associated with N_i begins at the root of T .

Disallowing negative paths that do not originate at the root (point 3 above) gives a subclass of extended Horn called *simple extended Horn* [133]. Extended Horn formulas can be solved in polynomial time by Algorithm **SLUR** on Page 410 [122].

Chandru and Hooker showed that unit resolution alone can determine whether or not a given extended Horn formula is satisfiable. This is due to the following two properties of an extended Horn formula:

1. If ψ is extended Horn and has no unit clauses, then ψ is satisfiable.
2. If ψ is extended Horn and v is a variable in ψ , then

$$\psi_1 = \{c - \{v\} : c \in \psi, \neg v \notin c\} \text{ and } \psi_2 = \{c - \{\neg v\} : c \in \psi, v \notin c\}$$

are both extended Horn.

Chandru and Hooker proposed an algorithm that finds a model for a satisfiable extended Horn formula. First, apply unit resolution, setting values of unassigned variables to 1/2 when no unit clauses remain. Then round the result by a matrix multiplication. Their algorithm cannot, however, be reliably applied unless it is

Algorithm 21.

```

SLUR ( $\psi$ )
/* Input: a set of sets CNF formula  $\psi$  */  

/* Output: “unsatisfiable” or a model for  $\psi$  */  

/* Locals: set of variables  $M$ , flag  $d$ , formula  $\psi'$  */  

Set  $\langle \psi', M \rangle \leftarrow \text{Unit Resolution } (\psi)$ .  

If  $\emptyset \in \psi'$  then Output “unsatisfiable.”  

Set  $d \leftarrow 0$  /*  $d = 0$  iff execution is at the top level. */  

Repeat the following while  $\psi' \neq \emptyset$ :  

    Choose arbitrarily a variable  $v \in V_{\psi'}$ .  

    Set  $\langle \psi_1, M_1 \rangle \leftarrow \text{Unit Resolution } (\{c - \{v\} : c \in \psi', \neg v \notin c\})$ .  

    Set  $\langle \psi_2, M_2 \rangle \leftarrow \text{Unit Resolution } (\{c - \{\neg v\} : c \in \psi', v \notin c\})$ .  

    If  $\emptyset \in \psi_1$  and  $\emptyset \in \psi_2$  then do the following:  

        If  $d = 0$  then Output “unsatisfiable.”  

        Otherwise, Output “give up.”  

    Otherwise, do the following:  

        Arbitrarily choose  $i$  so that  $\emptyset \notin \psi_i$ .  

        Set  $\psi' \leftarrow \psi_i$ .  

        If  $i = 1$ ,  $M \leftarrow M \cup M_1$ .  

        Otherwise,  $M \leftarrow M \cup M_2 \cup \{v\}$ .  

    Set  $d \leftarrow 1$   

Output  $M$ 
□

```

Fig. 47 Algorithm for determining satisfiability of SLUR formulas

known that a given formula is extended Horn. Unfortunately, the problem of recognizing extended Horn formulas is not known to be solved in polynomial time. As will be shown later in this section, this problem has become moot since Algorithm **SLUR** solves extended Horn formulas in linear time without the need for recognition.

The class of CC-balanced formulas has been studied by several researchers (see [38] for a detailed account of balanced matrices and a description of CC-balanced formulas). The motivation for this class is the question, for SAT, when do linear programming relaxations have integer solutions? A formula ψ with (0 ± 1) matrix representation \mathcal{M}_ψ is *CC-balanced* if in every submatrix of \mathcal{M}_ψ with exactly two nonzero entries per row and per column, the sum of the entries is a multiple of 4 (this definition is taken from [138]). Recognizing that a formula is CC-balanced takes linear time. However, the recognition problem is moot because Algorithm **SLUR** solves CC-balanced formulas in linear time without the need for recognition.

As alluded to above, both extended Horn and CC-balanced formulas are subsets of a larger efficiently solved class of formulas solved by *single lookahead unit resolution* [122] (SLUR). The SLUR class is peculiar in that it is defined based on an algorithm rather than on properties of formulas. Algorithm **SLUR** of Fig. 47 selects variables sequentially and arbitrarily and considers a one-level lookahead,

under unit resolution, of both possible values that the selected variable can take. If unit resolution does not result in an empty clause in one direction, the assignment corresponding to that value choice is made permanent and variable selection continues. If all clauses are satisfied after a value is assigned to a variable (and unit resolution is applied), the algorithm returns a satisfying assignment. If unit resolution, applied to the given formula or to both subformulas created from assigning values to the selected variable on the first iteration, results in a clause that is falsified, the algorithm reports that the formula is unsatisfiable. If unit resolution results in falsified clauses as a consequence of both assignments of values to the selected variable on any iteration except the first, the algorithm reports that it has given up.

A formula is in the class SLUR if, for all possible sequences of selected variables, Algorithm **SLUR** does not give up on that formula. Observe that due to the definition of this class, the question of class recognition is avoided.

The worst-case complexity of Algorithm **SLUR**, as written, is quadratic in the length of the input formula. The complexity is dominated by the execution of $\{c - \{v\} : c \in \psi, \neg v \notin c\}$, $\{c - \{\neg v\} : c \in \psi, v \notin c\}$, and the number of unit clauses eliminated by unit resolution. The total number of times a clause is checked and a literal removed due to the first two expressions is at most the number of literals existing in the given formula if the clauses are maintained in a linked list indexed on the literals. However, the same unit clause may be removed by unit resolution on successive iterations of the Repeat block of Algorithm **SLUR** since once branch of execution is always cut. This causes quadratic worst-case complexity.

A simple modification to Algorithm **SLUR** brings the complexity down to linear time: run both calls of unit resolution simultaneously, alternating execution of their Repeat blocks. When one terminates without an empty clause in its output formula, abandon the other call.

Theorem 17 *Algorithm **SLUR** has $O(|\psi|)$ worst-case complexity if both calls to unit resolution are applied simultaneously and one call is immediately abandoned if the other finishes first without falsifying a clause.*

Proof For reasons mentioned above, only the number of steps used by unit resolution is considered. The number of times a literal from a unit clause is chosen and satisfied clauses and falsified literals removed in *non-abandoned* calls of unit resolution is $O(|\psi|)$ since no literal is chosen twice. Since the Repeat blocks of abandoned and non-abandoned calls alternate, the time used by abandoned calls is no greater than that used by non-abandoned ones. Thus, the worst-case complexity of Algorithm **SLUR**, with the interleave modification, is $O(|\psi|)$. \square

All Horn, renamable Horn, extended Horn, and CC-balanced formulas are in the class SLUR. Thus, an important outcome of the results on SLUR is the observation that no special preprocessing or testing is needed for some of the special polynomial-time solvable classes of SAT when using a reasonable variant of the DPLL algorithm.

A limitation of all the classes of this section is they do not represent many interesting unsatisfiable formulas. There are several possible extensions to the SLUR class which improve the situation. One is to add a 2-SAT solver to the unit resolution steps of Algorithm **SLUR**. This extension is at least able to handle all 2-SAT formulas which is something Algorithm **SLUR** cannot do. It can be elegantly incorporated due to the following observation: whenever Algorithm **SLUR** completes a sequence of unit resolutions and if at that time the remaining clauses are nothing but a subset of the original clauses (which they would have to be if all clauses have at most two literals), then effectively the algorithm can start all over. That is, if fixing of a variable to both values leads to an empty clause, then the formula has been proved to be unsatisfiable. Thus, one need not augment Algorithm **SLUR** by the 2-SAT algorithm, because the 2-SAT algorithm (at least one version of it) does exactly what the extended algorithm does. Another extension of Algorithm **SLUR** is to allow a polynomial number of backtracks, giving up if at least one branch of the search tree does not terminate at a leaf where a clause is falsified. This enables unsatisfiable formulas with short search trees to be solved efficiently by Algorithm **SLUR**.

6.5 q-Horn Formulas

This class of propositional formulas was developed in [22] and [23]. The class of q-Horn formulas may be characterized as a special case of maximum monotone decomposition of matrices [137, 138]. Express a CNF formula of m clauses and n variables as an $m \times n$ (0 ± 1)-matrix \mathcal{M} . In the monotone decomposition of \mathcal{M} , columns are scaled by -1 and the rows and columns are partitioned into submatrices as follows:

$$\left(\begin{array}{c|c} \mathcal{A}^1 & \mathcal{E} \\ \hline \mathcal{D} & \mathcal{A}^2 \end{array} \right),$$

where the submatrix \mathcal{A}^1 has at most one $+1$ entry per row; the submatrix \mathcal{D} contains only -1 or 0 entries; the submatrix \mathcal{A}^2 has no restrictions other than the three values of -1 , $+1$, and 0 for each entry; and the submatrix \mathcal{E} has only 0 entries. If \mathcal{A}^1 is the largest possible over columns, then the decomposition is a maximum monotone decomposition. If the maximum monotone decomposition of \mathcal{M} is such that \mathcal{A}^2 has no more than two nonzero entries per row, then the formula represented by \mathcal{M} is *q-Horn*.

Truemper [138] shows that a maximum monotone decomposition for a matrix associated with a q-Horn formula can be found in linear time (this is discussed in Sect. 5.10.1). Once a q-Horn formula is in its decomposed form, it can be solved in linear time by Algorithm **q-Horn Solver** of Fig. 48.

Theorem 18 *Given a q-Horn formula ψ , Algorithm **q-Horn Solver** outputs “unsatisfiable” if and only if ψ is unsatisfiable, and if ψ is satisfiable, then the output set $M_1 \cup M_2$ is a model for ψ .*

Algorithm 22.

```

q-Horn Solver ( $\mathcal{M}_\psi$ )
/* Input: a  $(0, \pm 1)$  matrix representation for q-Horn formula  $\psi$  */
/* Output: “unsatisfiable” or a model for  $\psi$  */
/* Locals: set of variables  $M_1, M_2$  */

    Find the maximum monotone decomposition of  $\mathcal{M}_\psi$  (Page 84).
    If Horn formula  $\mathcal{A}^1$  is unsatisfiable then Output “unsatisfiable”.
    Let  $M_1$  be a unique minimum model for the Horn formula  $\mathcal{A}^1$ .
    Remove from  $\mathcal{A}^2$  all rows whose columns in  $\mathcal{D}$  are satisfied by  $M_1$ .
    If  $\mathcal{A}^2$  is unsatisfiable then Output “unsatisfiable”.
    Let  $M_2$  be a model for the 2-SAT formula  $\mathcal{A}^2$ .
    Output  $M_1 \cup M_2$ .
```

□

Fig. 48 Algorithm for determining satisfiability of q-Horn formulas

Proof Clearly, if Horn formula \mathcal{A}^1 is unsatisfiable, then so is ψ . Suppose \mathcal{A}^2 is unsatisfiable after rows whose columns in \mathcal{D} are removed because they are satisfied by M_1 . Since M_1 is a unique minimum model for \mathcal{A}^1 and no entries of \mathcal{D} are $+1$, no remaining row of \mathcal{A}^2 can be satisfied by any model for \mathcal{A}^1 . Therefore, ψ is unsatisfiable in this case. The set $M_1 \cup M_2$ is a model for ψ if it is output since M_1 satisfies rows in \mathcal{A}^1 and M_2 satisfies rows in \mathcal{A}^2 . □

An equivalent definition of q-Horn formulas comes from the following.

Theorem 19 A CNF formula is q-Horn if and only if its satisfiability index is no greater than 1.

Proof Let ψ be a q-Horn formula with variable set V_ψ and suppose $|V_\psi| = n$. Let \mathcal{M}_ψ be a monotone decomposition for ψ . Let n_a be such that for $0 \leq i < n_a$, column i of \mathcal{M}_ψ coincides with submatrices \mathcal{A}^1 and \mathcal{D} and for $n_a \leq i < n$, column i coincides with submatrix \mathcal{A}^2 . Form the inequalities (2) from \mathcal{M}_ψ . Assign value 1 to all α_i , $0 \leq i < n_a$, and value $1/2$ to all α_i , $n_a \leq i < n$. This satisfies $0 \leq \alpha_i \leq 1$ of system (2). Since rows of \mathcal{A}^1 have nonzero entries in columns 0 to $n_a - 1$ only and at most one of those is $+1$, the maximum sum of the elements of the corresponding row of inequality (2) is 1. Since all rows of \mathcal{D} and \mathcal{A}^2 have at most two nonzero entries in columns n_a to $n - 1$ and no $+1$ entries in columns 0 to $n_a - 1$, the sum of elements of a corresponding row of inequality (2) has maximum value 1 too. Thus, inequality (2) is satisfied with $z = 1$.

Now, suppose ψ has satisfiability index I_ψ which is no greater than 1. Choose values for all α_i terms such that inequality (2) is satisfied for $z = I_\psi$. Let π be a permutation of the columns of \mathcal{M}_ψ so that $\alpha_{\pi_i} \leq \alpha_{\pi_j}$ if and only if $i < j$. Form \mathcal{M}'_ψ from \mathcal{M}_ψ by permuting columns according to π . Let n_b be such that $\alpha_{\pi_i} < 1/2$ for $0 \leq i < n_b$ and $1/2 \leq \alpha_{\pi_i}$ for $n_b \leq i < n$. Scale columns 0 through $n_b - 1$ of \mathcal{M}'_ψ by -1 . Then inequality (2), using \mathcal{M}'_ψ for \mathcal{M}_ψ , is satisfied with $z = I_\psi$ and

$1/2 \leq \alpha_i$ for all $0 \leq i < n$. It follows that each row of \mathcal{M}'_ψ can have at most two +1 entries or else the elements of the corresponding row of inequality (2) sums to greater than 1. Consider all rows of \mathcal{M}'_ψ with two +1 entries and mark all columns that contain at least one of those entries. Let ρ be a permutation of the columns of \mathcal{M}'_ψ so that all marked columns have higher index than all unmarked columns. Form \mathcal{M}''_ψ from \mathcal{M}'_ψ by permuting columns according to ρ and permuting rows so that all rows with only 0 entries in marked columns are indexed lower than rows with at least one nonzero entry in a marked column. Let n_c be such that columns n_c to $n-1$ in \mathcal{M}''_ψ are exactly the marked columns. The value of $\alpha_{\rho_{n_c}}, n_c \leq i < n$, must be exactly 1/2 or else the elements of some row of inequality (2) must sum to greater than 1. It follows that the number of nonzero entries in columns n_c to $n-1$ in any row of \mathcal{M}''_ψ must be at most two or else the elements of some row of the inequality sum to greater than 1. By construction of \mathcal{M}''_ψ , every row can have at most one +1 entry in columns 0 to $n_c - 1$. Hence, \mathcal{M}''_ψ is a monotone decomposition for ψ . \square

The following result from [23] is also interesting in light of [Theorem 19](#). It is stated without proof.

Theorem 20 *The class of all formulas with a satisfiability index greater than $1 + 1/n^\epsilon$, for any fixed $\epsilon < 1$, is NP-complete.* \square

There is an almost obvious polynomial-time solvable class larger than that of the q-Horn formulas: namely, the class of formulas which have a satisfiability index no greater than $1 + a \ln(n)/n$, where a is any positive constant. The \mathcal{M} matrix for any formula in this class can be scaled by -1 and partitioned as follows:

$$\left(\begin{array}{c|cc|c} \mathcal{A}^1 & \mathcal{E} & \mathcal{B}^1 \\ \hline \mathcal{D} & \mathcal{A}^2 & \mathcal{B}^2 \end{array} \right),$$

where submatrices \mathcal{A}^1 , \mathcal{A}^2 , \mathcal{E} , and \mathcal{D} have the properties required for q-Horn formulas and the number of columns in \mathcal{B}^1 and \mathcal{B}^2 is no greater than $O(\ln(n))$. Satisfiability for such formulas can be determined in polynomial time by solving the q-Horn system obtained after substitution of each of $2^{O(\ln(n))}$ partial truth assignments to the variables of \mathcal{B}^1 and \mathcal{B}^2 .

6.6 Matched Formulas

The matched formulas have been considered in the literature (see [136]) but not extensively studied, probably because this seems to be a rather useless and small class of formulas. Our interest in matched formulas is to provide a basis of comparison with other, well-known, well-studied classes. Let $G_\psi(V_1, V_2, E)$ be the variable-clause matching graph (see [Sect. 3.5](#)) for CNF formula ψ , where V_1 is the set of clause vertices and V_2 is the set of variable vertices. A total matching with respect

to V_1 is a subset of edges $E' \subset E$ such that no two edges in E' share an endpoint but every vertex $v \in V_1$ is an endpoint for some edge in E' . Formula ψ is a *matched formula* if its variable-clause matching graph has a total matching with respect to V_1 . A matched formula is trivially satisfied: for each edge $e \in E'$, assign the variable represented by the variable vertex in e a value that satisfies the clause represented by the clause vertex in e . The comparison with other classes is discussed in Sect. 6.12.

6.7 Generalized Matched Formulas

The class of matched formulas has been generalized by Szeider [135]. Let $G_\psi(V_1, V_2, E)$ be the variable-clause matching graph of ψ as above. Let $V'_1 \subset V_1$ and $V'_2 \subset V_2$ and suppose the subgraph $G'_\psi(V'_1, V'_2, E')$ induced by V'_1 and V'_2 is complete: that is, for every pair of vertices $v_1 \in V'_1$ and $v_2 \in V'_2$, there is an edge $e \in E'$. Call such a subgraph a *biclique*.

Theorem 21 *Let $G_\psi(V_1, V_2, E)$ be the variable-clause graph for ψ . Let $\{X_1, X_2, \dots, X_r\}$ be a collection of bicliques in $G_\psi(V_1, V_2, E)$ and suppose (1) the number of clause vertices in X_1 is less than 2 raised to the number of variable vertices in X_i , $1 \leq i \leq r$; (2) every $v_1 \in V_1$ (representing a clause) is in some X_i , $1 \leq i \leq r$; and (3) every $v_2 \in V_2$ (representing a variable) is in at most one X_i , $1 \leq i \leq r$. Then ψ is satisfiable.*

Proof Let V^{X_i} be the variables represented by vertices of X_i and let C^{X_i} be the clauses represented by the remaining vertices of X_i . Remove from all $c \in C^{X_i}$ all literals not associated with variables of V^{X_i} . Since X_i is complete, the number of literals in c is $|V^{X_i}|$ and the number of assignments to the variables of V^{X_i} which are falsified by c is 1. By condition (1), the total number of falsifying assignments for C^{X_i} is less than all possible assignments to V^{X_i} ; hence, some assignment satisfies C^{X_i} . By condition (3), $V^{X_i} \cap V^{X_j} = \emptyset$, $i \neq j$, so satisfying assignments for both C^{X_i} and C^{X_j} cannot conflict. By condition (2), every clause in ψ is in some biclique and is therefore satisfied by some assignment. \square

Clause width is typically fixed. In that case, a model for a formula satisfying the conditions of Theorem 21 can be found in time linear in the number of clauses of ψ . If the bicliques of G_ψ are all single edges, then ψ is a matched formula.

Unfortunately, the problem of recognizing a generalized matched formula is \mathcal{NP} -complete, even for 3-CNF formulas.

6.8 Nested and Extended Nested Satisfiability

The complexity of nested satisfiability, inspired by Lichtenstein's theorem of planar satisfiability [105], has been studied in [91]. Index all variables in a CNF formula consecutively from 1 to n and let positive and negative literals take the index of

their respective variables. A clause c_i is said to *straddle* another clause c_j if the index of a literal of c_j is strictly between two indices of literals of c_i . Two clauses are said to *overlap* if they straddle each other. A formula is said to be *nested* if no two clauses overlap. For example, the following formula is nested:

$$(v_6 \vee \neg v_7 \vee v_8) \wedge (v_2 \vee v_4) \wedge (\neg v_6 \vee \neg v_9) \wedge (v_1 \vee \neg v_5 \vee v_{10}).$$

The class of nested formulas is quite limited in size. A variable cannot show up in more than one clause of a nested formula where its index is strictly between the greatest and least of the clause: otherwise, two clauses overlap. Therefore, a nested formula of m clauses and n variables has at most $2m + n$ literals. Thus, no CNF formula consisting of k -literal clauses is a nested formula unless $m/n < 1/(k-2)$. This particular restriction will be understood better from the probabilistic perspective taken in [Sect. 6.13](#). Despite this, the class of nested formulas is not contained in either of the SLUR or q-Horn classes as shown in [Sect. 6.12](#). This adds credibility to the potential usefulness of the algorithm presented here. However, our main interest in nested formulas is due to an enlightening analysis and efficient dynamic programming solution which appears in [91] and is presented here.

Strong dependencies between variables of nested formulas may be exploited for fast solutions. In a nested formula, if clause c_i straddles clause c_j , then c_j does not straddle c_i , and if clause c_i straddles clause c_j and clause c_j straddles c_k , then c_i straddles c_k . Thus, the straddling relation induces a partial order on the clauses of a nested formula. It follows that the clauses can be placed in a total ordering using a standard linear time algorithm for topologically sorting a partial order: in the total ordering, a given clause does not straddle any clauses following it. In the example above, if clauses are numbered c_0 to c_3 from left to right, c_2 straddles c_0 , c_3 straddles c_1 and c_2 , and no other clause straddles any other, so these clauses are in the desired order already.

Once clauses are topologically sorted into a total order, the satisfiability question may be solved in linear time by a dynamic programming approach where clauses are processed one at a time, in order. The idea is to maintain a partition of variables as a list of intervals such that all variables in any clause seen so far are in one interval. Associated with each interval are four D values that express the satisfiability of corresponding processed clauses under all possible assignments of values to the endpoints of the interval. As more clauses are considered, intervals join and D values are assigned. By introducing two extra variables, v_0 and v_{n+1} and an extra clause $(v_0 \vee v_{n+1})$ which straddles all others and is the last one processed, there is one interval $[v_0, v_{n+1}]$ remaining at the end. A D value associated with that interval determines satisfiability for the given formula. What remains is to determine how to develop the interval list and associated D values incrementally. This task is made easy by the fact that a variable which appears in a clause c with index strictly between the highest and lowest indices of variables in c never appears in a following clause.

An efficient algorithm for determining the satisfiability of nested formulas is shown in [Fig. 49](#). The following lemma is needed to prove correctness. The actual dependence of h values on i is not shown to prevent the notation from going out of control. However, from the context, this dependence should be clear.

Algorithm 23.**Nested Solver (ψ)**

```
/* Input: set of sets CNF formula  $\psi$ , with variables indexed 1 to  $n$  */
/*           and  $m$  clauses indexed from 0 to  $m - 1$  */
/* Output: "unsatisfiable" or "satisfiable" */
/* Locals: Boolean variables  $D_{i,j}(s,t), E_{i,j}(s,t), G_{i,j}(s,t)$ ,
/*           set  $\mathcal{D}$  of Boolean variables. */
```

Apply unit resolution to eliminate all unit clauses in ψ .

Topologically sort the clauses of ψ as explained in the text.

Let c_0, \dots, c_{m-1} denote the clauses, in order.

Add the clause $c_m = \{v_0, v_{n+1}\}$.

Set $\mathcal{D} \leftarrow \{D_{j,j+1}(s,t) \leftarrow 1 : 0 \leq j \leq n, s, t \in \{0, 1\}\}$.

Repeat the following for $0 \leq i \leq m$:

Let $0 < h_1 < h_2 < \dots < h_k < n + 1$ be the intervals for $D \in \mathcal{D}$.

// That is, $\mathcal{D} = \{D_{0,h_1}(*,*), D_{h_1,h_2}(*,*), \dots, D_{h_k,n+1}(*,*)\}$.

Let $c_i = \{l_{h_p}, \dots, l_{h_q}\}$, where $l_{h_r} \in \{v_{h_r}, \neg v_{h_r}\}$, $p \leq r \leq q$.

Set $E_{h_p,h_p}(s,t) \leftarrow 0$ for all $s \in \{0, 1\}$, $t \in \{0, 1\}$.

Set $G_{h_p,h_p}(s,t) \leftarrow 0$ for all $s \in \{0, 1\}$, $t \in \{0, 1\}$.

If $l_{h_p} = v_{h_p}$ then do the following:

Set $E_{h_p,h_p}(1,1) \leftarrow G_{h_p,h_p}(0,0) \leftarrow 1$.

Set $E_{h_p,h_p}(0,0) \leftarrow G_{h_p,h_p}(1,1) \leftarrow 0$.

Otherwise, if $l_{h_p} = \neg v_{h_p}$ then do the following:

Set $E_{h_p,h_p}(1,1) \leftarrow G_{h_p,h_p}(0,0) \leftarrow 0$.

Set $E_{h_p,h_p}(0,0) \leftarrow G_{h_p,h_p}(1,1) \leftarrow 1$.

Repeat the following for $0 \leq j < q - p$, $s \in \{0, 1\}$, $t \in \{0, 1\}$:

If $t = 1$ then Set $l \leftarrow v_{h_{p+j+1}}$, Otherwise Set $l \leftarrow \neg v_{h_{p+j+1}}$.

Set $E_{h_p,h_{p+j+1}}(s,t) \leftarrow$

$(E_{h_p,h_{p+j}}(s,1) \wedge D_{h_{p+j},h_{p+j+1}}(1,t)) \vee$

$(E_{h_p,h_{p+j}}(s,0) \wedge D_{h_{p+j},h_{p+j+1}}(0,t)) \vee$

$(G_{h_p,h_{p+j}}(s,1) \wedge D_{h_{p+j},h_{p+j+1}}(1,t) \wedge l \in c_i) \vee$

$(G_{h_p,h_{p+j}}(s,0) \wedge D_{h_{p+j},h_{p+j+1}}(0,t) \wedge l \in c_i)$.

Set $G_{h_p,h_{p+j+1}}(s,t) \leftarrow$

$(G_{h_p,h_{p+j}}(s,1) \wedge D_{h_{p+j},h_{p+j+1}}(1,t) \wedge \neg(l \in c_i)) \vee$

$(G_{h_p,h_{p+j}}(s,0) \wedge D_{h_{p+j},h_{p+j+1}}(0,t) \wedge \neg(l \in c_i))$.

Repeat for $s \in \{0, 1\}$, $t \in \{0, 1\}$: Set $D_{h_p,h_q}(s,t) \leftarrow E_{h_p,h_q}(s,t)$.

Set $\mathcal{D} \leftarrow \mathcal{D} \setminus \{D_{h_p,h_{p+1}}(*,*), \dots, D_{h_{q-1},h_q}(*,*)\} \cup \{D_{h_p,h_q}(*,*)\}$.

If $D_{0,n+1}(1,1) = 1$, Output "satisfiable,"

Otherwise Output "unsatisfiable."

□

Fig. 49 Algorithm for determining satisfiability of nested formulas

Lemma 5 Assume, at the start of any iteration $0 \leq i \leq m$ of the outer Repeat loop of Algorithm **Nested Solver**, that

$$\mathcal{D} = \{D_{h_0, h_1}(*, *), D_{h_1, h_2}(*, *), \dots, D_{h_k, h_{k+1}}(*, *)\},$$

where $h_0 = 0$ and $h_{k+1} = n + 1$. Let

$$\psi_{p, p+j}^i = \{c : c \in \{c_0, \dots, c_{i-1}\}, h_p \leq \text{minIndex}(c) < \text{maxIndex}(c) \leq h_{p+j}\},$$

for any $0 \leq j \leq k - p + 1$. Then the following hold:

1. At the start of iteration i of the outer Repeat loop, each variable in c_i is the same as one of $\{v_{h_0}, v_{h_1}, \dots, v_{h_{k+1}}\}$, and for every clause $c \in \{c_0, \dots, c_{i-1}\}$, there exists an $0 \leq r \leq k$ such that all the variables of c have index between h_r and h_{r+1} . Moreover, for every h_r and h_{r+1} , there is at least one clause whose minimum indexed variable has index h_r and whose maximum indexed variable has index h_{r+1} .
2. At the start of iteration i of the outer Repeat loop, $D_{h_j, h_{j+1}}(s, t)$, $0 \leq j \leq k$, has value 1 if and only if $\psi_{j, j+1}^i$ is satisfiable with variable v_{h_j} set to value s and variable $v_{h_{j+1}}$ set to value t .
3. At the start of iteration $0 \leq j < q - p$ of the main inner Repeat loop, $E_{h_p, h_{p+j}}(s, t)$ has value 1 if and only if $\psi_{p, p+j}^i \cup \{\{l_x : l_x \in c_i, x \leq h_{p+j}\}\}$ is satisfiable with variable v_{h_p} set to value s and variable $v_{h_{p+j}}$ set to value t .
4. At the start of iteration $0 \leq j < q - p$ of the main inner Repeat loop, $G_{h_p, h_{p+j}}(s, t)$ has value 1 if and only if $\psi_{p, p+j}^i \cup \{\{l_x : l_x \in c_i, x \leq h_{p+j}\}\}$ is not satisfiable, but $\psi_{p, p+j}^i$ is satisfiable with variable h_p set to value s and variable $v_{h_{p+j}}$ set to value t .

Proof Consider point 1. At the start of iteration 0 of the outer Repeat loop, point 1 holds because all variables are in the set $\{h_0, \dots, h_{n+1}\}$. Suppose point 1 holds at the beginning of iteration i . As a result of the topological sort, c_i cannot be straddled by any clause in $\{c_0 \dots c_{i-1}\}$. If one variable of c_i is indexed strictly between h_r and h_{r+1} , by point 1, there is a clause in $\{c_0, \dots, c_{i-1}\}$ whose variable indices are as high as h_{r+1} and as low as h_r . But such a clause would straddle c_i . Hence, for every variable $v \in c_i$, $v \in \{v_{h_p}, v_{h_{p+1}}, \dots, v_{h_q}\}$. Since the last line of the outer Repeat loop replaces all $D_{h_p, h_{p+1}} \dots D_{h_{q-1}, q}$ with D_{h_p, h_q} , point 1 holds at the beginning of iteration $i + 1$ of the outer loop.

Consider point 2. At the start of iteration $i = 0$, all defined variables are $D_{j, j+1}(*, *)$, $0 \leq j \leq n$, and these have value 1. From the definition of $\psi_{p, j}^i$, $\psi_{j, j+1}^0 = \emptyset$. Thus, point 2 holds before iteration $i = 0$. Suppose point 2 holds at the start of iteration i . Since c_i contains no variables indexed less than h_p or greater than h_q and all $D_{h_r, h_{r+1}}(*, *)$ are unchanged by the algorithm for $r < p$ and $r \geq q$, then these D values are correct for iteration $i + 1$ (but the subscripts on h values change because there are fewer D variables on the next iteration). So, due to the short inner Repeat loop following the main inner Repeat loop, point 2 holds

at the start of iteration $i + 1$ if $E_{h_p, h_q}(s, t)$ has value 1 if and only if $\psi_{p,q}^i \cup \{c_i\}$ is satisfiable with variable v_{h_p} set to value s and variable v_{h_q} set to value t . This is shown below, thus taking care of point 2.

Assume, during iteration i of the outer loop and before the start of the main inner loop, that points 1 and 2 hold. Consider the iteration $j = 0$ of the main inner loop. As above, $\psi_{*,*}^0 = \emptyset$. Moreover, by point 1, $\{l_x : l_x \in c_0, x \leq h_p\} = \{l_{h_p}\}$ so $\psi_{p,p}^0 \cup \{\{l_{h_p}\}\} = \{\{l_{h_p}\}\}$. There is no satisfying assignment for this set if the value of the highest and lowest indexed variables of c_0 are opposite each other since the highest and lowest indexed variables are the same. Accordingly, $E_{p,p}(s, t)$ and $G_{p,p}(s, t)$ are set to 0 in the algorithm when s and t are of opposite value. However, if $s = t = 1$ and if $l_{h_p} = v_{h_p}$, then $\psi_{p,p}^0 \cup \{\{v_{h_p}\}\}$ is satisfiable. Accordingly, $E_{p,p}(1, 1)$ is set to 1 and $G_{p,p}(1, 1)$ is set to 0 in the algorithm. Otherwise, if $s = t = 1$ and $l_{h_p} = \neg v_{h_p}$, then $\psi_{p,p}^0 \cup \{\{\neg v_{h_p}\}\}$ is unsatisfiable. Accordingly, $E_{p,p}(1, 1)$ is set to 0 and $G_{p,p}(1, 1)$ is set to 1 in the algorithm. Similar reasoning applies to the case $s = t = 0$. Thus, points 3 and 4 hold for the case $j = 0$.

Now consider iteration i of the outer loop and iteration $j > 0$ of the main inner loop. Assume, at the start of iteration $j - 1$, that points 3 and 4 hold. Points 1 and 2 hold from before since no changes to these occur in the main inner loop. Let $c_i^{j-1} = \{l_x : l_x \in c_i, x \leq p + j - 1\}$ be the subset of literals of c_i that have index no greater than h_{p+j-1} . From Point 1, for every clause $c \in \{c_0, \dots, c_{i-1}\}$, there exists a positive integer $0 \leq r \leq k$ such that all variables of c have index between h_r and h_{r+1} . It follows that

$$\psi_{p,p+j}^i \cup \{c_i^{j-1}\} = (\psi_{p,p+j-1}^i \cup \{c_i^{j-1}\}) \cup \psi_{p+j-1,p+j}^i,$$

and $(\psi_{p,p+j-1}^i \cup \{c_i^{j-1}\}) \cap \psi_{p+j-1,p+j}^i = \emptyset$. For the sake of visualization, define $\psi_1 = \psi_{p,p+j-1}^i$ and $\psi_2 = \psi_{p+j-1,p+j}^i$. At most one variable, namely, $v_{h_{p+j-1}}$, is common to both ψ_1 and ψ_2 . Hence, when $v_{h_{p+j-1}}$ is set to some value, the following holds: both $\psi_1 \cup \{c_i^{j-1}\}$ and ψ_2 are satisfiable if and only if $\psi_{p,p+j}^i \cup \{c_i^{j-1}\}$ is satisfiable. Now consider $\psi_{p,p+j}^i \cup \{c_i^j\}$. Since every variable of c_i has an index matching one of $\{h_p, h_{p+1}, \dots, h_q\}$, $c_i^j \setminus c_i^{j-1}$, either is the empty set or $\{v_{h_{p+j}}\}$ or $\{\neg v_{h_{p+j}}\}$. Therefore, given $v_{h_{p+j-1}}$, $\psi_{p,p+j-1}^i \cup \{c_i^j\}$ is satisfiable if and only if either both $\psi_1 \cup \{c_i^{j-1}\}$ and ψ_2 are satisfiable or ψ_1 and ψ_2 are satisfiable, $\psi_1 \cup \{c_i^{j-1}\}$ is unsatisfiable, but $\psi_1 \cup \{c_i^{j-1} \cup \{l\}\}$ is satisfiable where $l \in \{v_{h_{p+j}}, \neg v_{h_{p+j}}\}$ or $\{l\} = \emptyset$. But since l does not occur in ψ_1 , $\psi_1 \cup \{c_i^{j-1}\}$ can be unsatisfiable with ψ_1 and $\psi_1 \cup \{c_i^{j-1} \cup \{l\}\}$ satisfiable only for assignments which satisfy l . Then, by hypothesis, the association of the E and G variables with $\psi_{p,p+j-1}^i$ and $\psi_{p+j-1,p+j}^i$, and the assignments to E and G in the main inner loop of Algorithm 23, the values of E and G match points 3 and 4 for the j th iteration of the main inner loop.

From point 3, upon completion of the main inner loop, $E_{h_p, h_q}(s, t)$ has value 1 if and only if $\psi_{h_p, h_q}^i \cup \{c_i\}$ is satisfiable. This matches the hypothesis of point 2, thereby completing the proof that point 2 holds. \square

Corollary 1 *Algorithm Nested Solver correctly determines satisfiability for a given nested formula.*

Proof Algorithm **Nested Solver** determines ψ is satisfiable if and only if $D_{0,n+1}(1, 1)$ has value 1. But, from Lemma 5, $D_{0,n+1}(1, 1)$ has value 1 if and only if $\psi \cup \{v_0, v_{n+1}\}$ is satisfiable and v_0 and v_{n+1} are set to value 1. The corollary follows. \square

The operation of **Nested Solver** is demonstrated by means of an example taken from [91]. Suppose the algorithm is applied to a nested formula containing the following clauses which are shown in proper order after the topological sort:

$$(v_1 \vee v_2) \wedge (v_2 \vee v_3) \wedge (\neg v_2 \vee \neg v_3) \wedge (v_3 \vee v_4) \wedge (v_3 \vee \neg v_4) \wedge (\neg v_3 \vee v_4) \wedge (\neg v_1 \vee v_2 \vee v_4).$$

The following table shows the D values that have been computed prior to the start of iteration 6 of the outer loop of the algorithm. The columns show s, t values and the rows show variable intervals.

$D_{*,*}(s, t)$	0, 0	0, 1	1, 0	1, 1
1, 2	0	1	1	1
2, 3	0	1	1	0
3, 4	0	0	0	1

Such a table could result from the following set of processed clauses at the head of the topological order (and before c): processing clause c , using the initial values and recurrence relations for E and G variables given above, produces values as shown in the following tables.

$E_{p,p+j}(s, t)$	0, 0	0, 1	1, 0	1, 1
1, 1	1	0	0	0
1, 2	0	1	0	1
1, 3	1	0	1	0
1, 4	0	0	0	1

$G_{p,p+j}(s, t)$	0, 0	0, 1	1, 0	1, 1
1, 1	0	0	0	1
1, 2	0	0	1	0
1, 3	0	0	0	1

The last line of the E table holds the new D values for the interval $[v_1, v_4]$ as shown by the following table.

$D_{*,*}(s, t)$	0, 0	0, 1	1, 0	1, 1
1, 4	0	0	0	1

Linear time is achieved by careful data structure design and from the fact that no more than $2m + n$ literals exist in a nested formula.

Theorem 22 *Algorithm Nested Solver has worst-case time complexity that is linear in the size of ψ .* \square

The question of whether the variable indices of a given formula can, in linear time, be permuted to make the formula nested appears to be open.

An extension to nested satisfiability has been proposed in [72]. The details are skipped. This extension can be recognized and solved in linear time. For details, the reader is referred to [72].

6.9 Linear Autark Formulas

This class is based on the notion of an autark assignment which was introduced in [111]. Repeating the definition from Page 393, an assignment to a set of variables is an autark assignment if all clauses that contain at least one of those variables are satisfied by the assignment. An autark assignment provides a means to partition the clauses into two groups: one that is satisfied by the autark assignment and one that is completely untouched by that assignment. Therefore, autark assignments provide a way to reduce a formula to one that is equivalent in satisfiability.

The following shows how to find an autark assignment in polynomial time. Let CNF formula ψ of m clauses and n variables be represented as a (0 ± 1) matrix \mathcal{M}_ψ . Let α be an n -dimensional real vector with components $\alpha_1, \alpha_2, \dots, \alpha_n$. Consider the following system of inequalities:

$$\begin{aligned} \mathcal{M}_\psi \alpha &\geq 0, \\ \alpha &\neq 0. \end{aligned} \tag{14}$$

Theorem 23 ([145]) *A solution to (14) implies an autark assignment for ψ .*

Proof Create the autark assignment as follows: if $\alpha_i < 0$ then assign $v_i = 0$, and if $\alpha_i > 0$ then assign $v_i = 1$, if $\alpha_i = 0$ then keep v_i unassigned. It is shown that either a clause is satisfied by this assignment or it contains only unassigned variables.

For every clause, by (1), it is necessary to satisfy the following:

$$a_1 v_1 + a_2 v_2 + \dots + a_n v_n \geq 1 - b, \tag{15}$$

where a_i factors are 0, -1 , or $+1$ and the number of negative a_i terms is b . Suppose α is a solution to (14). Write an inequality of (14) as follows:

$$a_1\alpha_1 + a_2\alpha_2 + \dots + a_n\alpha_n \geq 0. \quad (16)$$

Suppose at least one term, say, $a_i\alpha_i$, is positive. If α_i is positive, then $a_i = 1$ so, with $v_i = 1$ and since there are at most b terms of value -1 , the left side of (15) must be at least $1 - b$. On the other hand, if α_i is negative, then $a_i = -1$, $v_i = 0$, and the product $a_i v_i$ in (15) is 0. Since there are $b - 1$ negative factors left in (15), the left side must be at least $1 - b$. Therefore, in either case, inequalities of the form (16) with at least one positive term represent clauses that are satisfied by the assignment corresponding to α .

Now consider the case where no terms in (16) are positive. Since the left side of (16) is at least 0, there can be no negative terms either. Therefore, all the variables for the represented clause are unassigned. \square

A formula ψ for which the only solution to (14) is $\alpha = 0$ is said to be *linear autarky-free*.

System (14) is an instance of linear programming and can therefore be solved in polynomial time. Hence, an autark assignment, if one exists, can be found in polynomial time.

Algorithm **LinAut** of Fig. 50 is the central topic of study in this section and it will be used to define a polynomial-time solvable class of formulas called linear autarky formulas. The algorithm repeatedly applies an autark assignment as long as one can be found by solving (14). In a departure from [145], the algorithm begins with a call to **Unit Resolution** to eliminate all unit clauses. This is done to remove some awkwardness in the description of linear autarky formulas that will become clear shortly.

The following states an important property of Algorithm **LinAut**.

Lemma 6 *Let ψ be a CNF formula that is input to Algorithm **LinAut** and let ψ' be the formula that is output. If $\psi' = \emptyset$ then ψ is satisfiable and α^S transforms to a satisfying assignment for ψ .*

Proof An autark assignment t to ψ' induces a partition of clauses of ψ' into those that are satisfied by t and those that are untouched by t . Due to the independence of the latter group, a satisfying assignment for that group can be combined with the autark assignment for the former group to satisfy ψ' . If there is no satisfying assignment for either group, then ψ' cannot be satisfiable. Therefore, since each iteration finds and applies an autark assignment (via α), if $\psi' = \emptyset$ then the composition of the autark assignments of each iteration is a satisfying assignment for ψ . \square

The algorithm has polynomial-time complexity. In some cases it solves its input formula.

Algorithm 24.

```

LinAut ( $\psi$ )
/* Input: a set of sets CNF formula  $\psi$  */ 
/* Output: pair  $\langle$ real vector, subformula of  $\psi$   $\rangle$  or “unsatisfiable” */ 
Set  $\langle \psi', P \rangle \leftarrow \text{Unit Resolution}(\psi)$ .
If  $\emptyset \in \psi'$  then Output “unsatisfiable”.
Set  $\alpha^S \leftarrow 0$ .
For each  $v_i \in P$  Set  $\alpha_i^S \leftarrow 1$ .
For each  $v_i$  such that  $v_i \notin P$  and  $v_i \in V_P$  Set  $\alpha_i^S \leftarrow -1$ .
If  $\psi' = \emptyset$  then Output  $\langle \alpha^S, \emptyset \rangle$ .
Repeat the following until some statement generates output:
  Solve for  $\alpha : \mathcal{M}_{\psi'} \alpha \geq 0$  and  $\alpha_i^S = 0$  if column  $i$  in  $\mathcal{M}_{\psi'}$  is 0.
  If  $\alpha=0$  then Output  $\langle \alpha^S, \psi' \rangle$ .
  Otherwise, do the following:
    Set  $\alpha^S \leftarrow \alpha^S + \alpha$ .
    For every  $\alpha_i^S \neq 0$  do the following:
      Zero out rows of  $\mathcal{M}_{\psi'}$  with a non-zero entry in column  $i$ .
      Set  $\psi' \leftarrow \{c : c \in \psi', v_i \notin c, \neg v_i \notin c\}$ .

```

□

Fig. 50 Repeated decomposition of a formula using autark assignments. V_P is the set of variables whose values are set in partial assignment P . Output real vector α^S can be transformed to a truth assignment as described in the proof of [Theorem 23](#)

Theorem 24 Let ψ be Horn or renamable Horn. If ψ is satisfiable, then **LinAut** returns an α^S that transforms to a solution of ψ as described in the proof of [Theorem 23](#), and the formula returned by **LinAut** is \emptyset . If ψ is unsatisfiable, **LinAut** returns “unsatisfiable.”

Proof Horn or renamable Horn formula ψ is unsatisfiable only if there is at least one positive unit clause in ψ . In this case **Unit Resolution**(ψ) will output a formula containing \emptyset and **LinAut** will output *unsatisfiable*. Otherwise, **Unit Resolution**(ψ) outputs a Horn or renamable Horn formula ψ' with no unit clauses and a partial assignment P which is recorded in α^S . In the next paragraph, it will be shown that any such Horn or renamable Horn formula is not linear autarky-free, so there exists an autark assignment for it. By definition, the clauses that remain after the autark assignment is applied must be Horn or renamable Horn. Therefore, the Repeat loop of **LinAut** must continue until there are no clauses left. Then, by [Lemma 6](#), α^S transforms to a satisfying assignment for ψ .

Now it is shown that there is always an $\alpha \neq 0$ that solves (14). Observe that any Horn formula without unit clauses has at least one negative literal in every clause. Therefore, some all-negative vector α of equal components solves (14). In the case of renamable Horn, the polarity of α components in the switch set is reversed to get the same result. Therefore, there is always at least one autark assignment for a Horn or renamable Horn formula. Note that Algorithm **LinAut** may find an α that does not zero out all rows, so the Repeat loop may have more than 1 iteration, but since

the reduced formula ψ is Horn or renamable Horn, there is always an $\alpha \neq 0$ that solves (14) for the clauses that are left. \square

The following Horn formula would violate [Theorem 24](#) if the call to **Unit Resolution** had not been added to **LinAut**:

$$(v_1) \wedge (v_2) \wedge (v_3) \wedge (v_1 \vee \neg v_2 \vee \neg v_3) \wedge (\neg v_1 \vee v_2 \vee \neg v_3) \wedge (\neg v_1 \vee \neg v_2 \vee v_3).$$

This formula is satisfied with variables set to 1 but is linear autarky-free.

Theorem 25 *If ψ is a 2-SAT formula that is satisfiable, then **LinAut** outputs an α^S that represents a solution of ψ and the formula output by **LinAut** is \emptyset .*

Proof In this case **Unit Resolution**(ψ) outputs ψ and causes no change to α^S . The only nonzero autark solutions to $\mathcal{M}_\psi \alpha \geq 0$ have the following clausal interpretation: choose a literal in a clause and assign its variable a value that satisfies the clause; for all non-satisfied clauses that contain the negation of that literal, assign a value to the variable of the clause's other literal that satisfies that clause; continue the process until some clause is falsified or no variable is forced to be assigned a value to satisfy a clause. This is one iteration of Algorithm **2-SAT Solver**, so since ψ is satisfiable, the process never ends in a falsified clause. Since what is left is a satisfiable 2-SAT formula, this step is repeated until \emptyset is output. As in the proof of [Theorem 24](#), α^S transforms to a satisfying assignment for ψ . \square

If ψ is an unsatisfiable 2-SAT formula, then **LinAut**(ψ) will output an unsatisfiable 2-SAT formula.

The class of *linear autarky formulas* is the set of CNF formulas on which the application of **LinAut** either outputs *unsatisfiable* or a formula that is \emptyset or a linear autarky-free 2-SAT formula. In the middle case, the input formula is satisfiable; in the other two cases, it is unsatisfiable. Observe that in the last case, it is unnecessary to solve the remaining 2-SAT formula.

Theorem 26 *All q-Horn formulas are linear autark formulas.*

Proof Apply **LinAut** to q-Horn formula ψ . Then a linear autarky-free q-Horn formula is output. Assume that all clauses in the output formula have at least 2 literals – this can be done because all clauses of ψ' entering the Repeat loop for the first time have at least 2 literals and any subsequent autark assignment would not introduce a clause that is not already in ψ' . The rows and columns of $\mathcal{M}_{\psi'}$ representing the output q-Horn formula ψ' may be permuted, and columns may be scaled by -1 to get a form as shown on [Page 389](#) where \mathcal{A}^1 represents a Horn formula, \mathcal{D} is nonpositive, and \mathcal{A}^2 represents a 2-SAT formula. Construct vector α as follows. Assign equal negative values to all α_i where i is the index of a column through \mathcal{A}^1 (reverse the value if the column had been scaled by -1) and 0 to all other α_i . Then, since there are at least as many positive as negative entries in every

row through \mathcal{A}^1 , the product of any of those rows, unscaled, and α is greater than 0. Since there are only negative entries in rows through \mathcal{D} and all the entries through columns of \mathcal{A}^2 multiply by α_i values that are 0, the product of any of the rows through \mathcal{D} , unscaled, and α is also positive. Therefore, α shows that ψ' is not linear autarky-free, a contradiction. It follows that $\mathcal{A}^1 = \mathcal{D} = \emptyset$ and the output formula is either 2-SAT or \emptyset . If it is 2-SAT, it must be unsatisfiable as argued in [Theorem 25](#). \square

The relationship between linear autarky-free formulas and the satisfiability index provides a polynomial-time test for the satisfiability of a given CNF formula.

Theorem 27 *If the shortest clause of a CNF formula ψ has width k and if the satisfiability index of ψ is less than $k/2$, then ψ is satisfiable.*

Proof This is seen more clearly by transforming variables as follows. Define real n -dimensional vector β with components

$$\beta_i = 2\alpha_i - 1 \text{ for all } 0 \leq i < n.$$

The satisfiability index (2) for the transformed variables may be expressed, by simple substitution, as follows:

$$\mathcal{M}_\psi \beta \leq 2Z - I, \quad (17)$$

where I is an n -dimensional vector expressing the number of literals in all clauses and $Z = \langle z, z, \dots, z \rangle$. The minimum z that satisfies (17) is the satisfiability index of ψ . Apply Algorithm **LinAut** to ψ which, by hypothesis, has shortest clause of width k . The algorithm removes rows, so z can only decrease and k can only increase. Therefore, if $z < k/2$ for ψ , it also holds for the ψ' that is output by **LinAut**. Suppose $\psi' \neq \emptyset$. Formula ψ' is linear autarky-free so the only solution to $0 \leq \mathcal{M}_{\psi'} \beta$ is $\beta = 0$ which implies $0 \leq 2Z - I$. Since the shortest clause of ψ' is at least k , it follows that $k/2 \leq z$. This contradicts the hypothesis that $z < k/2$. Therefore, $\psi' = \emptyset$ and, by [Lemma 6](#), the input formula is satisfiable. \square

The effectiveness of this test on random k -CNF formulas⁹ will be discussed in [Sect. 6.13](#).

6.10 Minimally Unsatisfiable Formulas

An unsatisfiable CNF formula is *minimally unsatisfiable* if removing any clause results in a satisfiable formula. The class of minimally unsatisfiable formulas is

⁹Random k -SAT formulas are defined in [Sect. 7](#), Page 440

easily solved if the number of clauses exceeds the number of variables by a fixed positive constant k . This difference is called the formula's *deficiency*. This section begins with a discussion of the special case where deficiency is 1, then considers the case of any fixed deficiency greater than 1. The discussion includes some useful properties which lead to an efficient solver for this case and helps explain the difficulty of resolution methods for many CNF formulas.

The following result was proved in one form or another by several people (e.g., [5, 99]). A simple argument due to Truemper is presented.

Theorem 28 *If ψ is a minimally unsatisfiable CNF formula with $|V_\psi| = n$ variables, then the number of clauses in ψ must be at least $n + 1$.*

Proof Let ψ be a minimally unsatisfiable CNF formula with $n + 1$ clauses. Suppose $|V_\psi| \geq n + 1$. Let $G_\psi(V_1, V_2, E)$ be the variable-clause matching graph for ψ (see Sect. 3.5) where V_1 is the set of clause vertices and V_2 is the set of variable vertices. There is no total matching with respect to V_1 since this would imply ψ is satisfiable, a contradiction. Therefore, by Hall's Theorem [70] (stated as Theorem 36 in Sect. 6.13), there is a subset $V'_\psi \subset V_1$ with neighborhood smaller than $|V'_\psi|$. Let V'_ψ be such a subset of maximum cardinality. Define ψ_1 to be the CNF formula consisting of the clauses of ψ corresponding to the neighborhood of V'_ψ . By the minimality property of ψ , ψ_1 must be satisfiable. Delete from ψ the clauses of ψ_1 and from the remaining clauses all variables occurring in ψ_1 . Call the resulting CNF formula ψ_2 . There must be a matching of the clauses of ψ_2 into the variables of ψ_2 since otherwise V'_ψ and therefore ψ_1 were not maximal in size. Hence, ψ_2 is satisfiable. But if ψ_1 and ψ_2 are satisfiable, then so is ψ , a contraction. \square

Most of the remaining ideas of this section have been inspired by Oliver Kullmann [96].

A *saturated minimally unsatisfiable* formula ψ is a minimally unsatisfiable formula such that adding any literal l , existing in a clause of ψ , to any clause of ψ not already containing l or $\neg l$ results in a satisfiable formula. The importance of saturated minimally unsatisfiable formulas is grounded in the following lemma.

Lemma 7 *Let ψ be a saturated minimally unsatisfiable formula and let l be a literal from ψ . Then*

$$\psi' = \{c \setminus \{\neg l\} : c \in \psi, l \notin c\}$$

is minimally unsatisfiable. In other words, if satisfied clauses and falsified literals due to l taking value 1 are removed from ψ , what is left is minimally unsatisfiable.

Proof Formula ψ' is unsatisfiable if ψ is; otherwise, there is an assignment which sets l to 1 and satisfies ψ . Suppose there is a clause $c \in \psi'$ such that $\psi' \setminus \{c\}$ is unsatisfiable. Clause c must contain a literal l' that is neither l nor $\neg l$. Construct ψ'' by adding l' to all clauses of ψ containing $\neg l$. Since ψ is saturated, there is a truth assignment M satisfying ψ'' and therefore ψ' with literals l' added as

above. Assignment M must set l' to 1; otherwise, $\psi' \setminus \{c\}$ is satisfiable. Then, adjusting M so the value of l is 0 gives an assignment which also satisfies ψ' , a contradiction. \square

The following is a simple observation that is needed to prove [Theorem 29](#) below.

Lemma 8 *Every variable of a minimally unsatisfiable formula occurs positively and negatively in the formula.*

Proof Let ψ be any minimally unsatisfiable formula. Suppose there is a positive literal l such that $\exists c \in \psi : l \in c$ and $\forall c \in \psi, \neg l \notin c$. Let $\psi_1 = \{c : c \in \psi, l \notin c\}$. Let $\psi_2 = \psi \setminus \psi_1$ denote the clauses of ψ which contain literal l . Clearly, $|\psi_2| > 0$. Then, by definition of minimal unsatisfiability, ψ_1 is satisfiable by some truth assignment M . Hence, $M \cup \{l\}$ satisfies $\psi_1 \cup \psi_2 = \psi$. This contradicts the hypothesis that ψ is unsatisfiable. A similar argument applies if literal l is negative, proving the lemma. \square

Theorem 29 *Let ψ be a minimally unsatisfiable formula with $n > 0$ variables, and $n + 1$ clauses. Then there exists a variable $v \in V_\psi$ such that the literal v occurs exactly one time in ψ and the literal $\neg v$ occurs exactly one time in ψ .*

Proof If ψ is not saturated, there is some set of literals already in ψ that may be added to clauses in ψ to make it saturated. Doing so does not change the number of variables and clauses of ψ . So suppose from now on that ψ is a saturated minimally unsatisfiable formula with n variables and $n + 1$ clauses. Choose a variable v such that the sum of the number of occurrences of literal v and literal $\neg v$ in ψ is minimum. By [Lemma 8](#), the number of occurrences of literal v in ψ is at least one and the number of occurrences of literal $\neg v$ in ψ is at least one. By [Lemma 7](#),

$$\psi_1 = \{c \setminus \{\neg v\} : c \in \psi, v \notin c\} \text{ and } \psi_2 = \{c \setminus \{v\} : c \in \psi, \neg v \notin c\}$$

are minimally unsatisfiable. Clearly, $|V_{\psi_1}| = |V_{\psi_2}| = n - 1$ or else the minimality of variable v is violated. By [Theorem 28](#), the fact that ψ_1 and ψ_2 are minimally unsatisfiable and the fact that $|\psi_i| < |\psi|, i \in \{1, 2\}$, it follows that $|\psi_1| = |\psi_2| = n$. Therefore, the number of occurrences of literal v in ψ is one and the number of occurrences of literal $\neg v$ in ψ is one. \square

Lemma 9 *Let ψ be a minimally unsatisfiable CNF formula with $n > 0$ variables and $n + 1$ clauses. Let v be the variable of [Theorem 29](#) and define clauses c_v and $c_{\neg v}$ such that literal $v \in c_v$ and literal $\neg v \in c_{\neg v}$. Then there is no variable which appears as a positive literal in one of c_v or $c_{\neg v}$ and as a negative literal in the other.*

Proof Suppose there is a variable w such that literal $w \in c_v$ and literal $\neg w \in c_{\neg v}$. By the minimality of ψ and the fact that variable v appears only in c_v and $c_{\neg v}$, there exists an assignment M which excludes setting a value for variable v and

Algorithm 25.

```

Min Unsat Solver ( $\psi$ )
/* Input: a set of sets CNF formula  $\psi$  */          */
/* Output: either “unsat(1)” or “not unsat(1)” */      */
Repeat the following until all variables of  $\psi$  are eliminated:
  If the difference between clauses and variables is not 1,
    Output “not unsat(1).”
  If no variable of  $\psi$  occurs once positively and once negatively,
    Output “not unsat(1).”
  If, for some  $v$ ,  $\psi$  includes  $(v) \wedge (\neg v)$  and  $|\psi| > 2$ ,
    Output “not unsat(1).”
  Choose variable  $v$  such that  $v \in c_v$  and  $\neg v \in c_{\neg v}$ 
    and neither literals  $v$  nor  $\neg v$  are in  $\psi \setminus \{c_v, c_{\neg v}\}$ .
  Compute the resolvent  $\mathcal{R}_{c_{\neg v}}^{c_v}$  of clauses  $c_v$  and  $c_{\neg v}$ .
  Set  $\psi \leftarrow \psi \setminus \{c_v, c_{\neg v}\} \cup \{\mathcal{R}_{c_{\neg v}}^{c_v}\}$ .
Output “unsat(1).”
```

□

Fig. 51 Algorithm for determining whether a CNF formula with one more clause than variable is minimally unsatisfiable

satisfies $\psi \setminus \{c_v, c_{\neg v}\}$. But M must also satisfy either c_v , if w has value 1, or $c_{\neg v}$ if w has value 0. In the former case $M \cup \{\neg v\}$ satisfies ψ and in the latter case $M \cup \{v\}$ satisfies ψ , a contradiction. The argument can be generalized to prove the lemma. □

The next series of results shows the structure of minimally unsatisfiable formulas and how to exploit that structure for fast solutions.

Theorem 30 *Let ψ be a minimally unsatisfiable formula with $n > 1$ variables and $n + 1$ clauses. Let v be the variable of Theorem 29, let c_v be the clause of ψ containing the literal v , and let $c_{\neg v}$ be the clause of ψ containing the literal $\neg v$. Then $\psi' = \psi \setminus \{c_v, c_{\neg v}\} \cup \{\mathcal{R}_{c_{\neg v}}^{c_v}\}$ is a minimally unsatisfiable formula with $n - 1$ variables and n clauses.*

Proof By Lemma 9 the resolvent $\mathcal{R}_{c_{\neg v}}^{c_v}$ of c_v and $c_{\neg v}$ exists. Moreover, $\mathcal{R}_{c_{\neg v}}^{c_v} \neq \emptyset$ if $n > 1$ or else ψ is not minimally unsatisfiable. Therefore, ψ' is unsatisfiable, contains $n - 1$ variables, and has n nonempty clauses. Remove a clause c from $\psi \setminus \{c_v, c_{\neg v}\}$. By the minimality of ψ , there is an assignment M which satisfies $\psi \setminus \{c\}$. By Lemma 2, M also satisfies $\psi \setminus \{c, c_v, c_{\neg v}\} \cup \{\mathcal{R}_{c_{\neg v}}^{c_v}\}$. Hence, M satisfies $\psi' \setminus \{c\}$. Now remove c_v and $c_{\neg v}$ from ψ . Again, by the minimality of ψ , there is an assignment M which satisfies $\psi \setminus \{c_v, c_{\neg v}\}$ and hence $\psi' \setminus \{\mathcal{R}_{c_{\neg v}}^{c_v}\}$. Thus, removing any clause from ψ' results in a satisfiable formula and the theorem is proved. □

Theorem 30 implies the correctness of the polynomial-time algorithm of Fig. 51 for determining whether a CNF formula with one more clause than variable is

minimally unsatisfiable. If the output is “munsat(1)” the input formula is minimally unsatisfiable, with clause-variable difference of 1; otherwise, it is not.

The following results provide an interesting and useful characterization of minimally unsatisfiable formulas of the type discussed here.

Theorem 31 *Let ψ be a minimally unsatisfiable formula with $n > 1$ variables and $n + 1$ clauses. Then there exists a variable v , occurring once as a positive literal and once as a negative literal, and a partition of the clauses of ψ into two disjoint sets ψ_v and $\psi_{\neg v}$ such that literal v only occurs in clauses of ψ_v , literal $\neg v$ only occurs in clauses of $\psi_{\neg v}$, and no variable other than v that is in ψ_v is also in $\psi_{\neg v}$ and no variable other than v that is in $\psi_{\neg v}$ is also in ψ_v .*

Proof By induction on the number of variables. The hypothesis clearly holds for minimally unsatisfiable formulas of one variable, all of which have the form $(v) \wedge (\neg v)$. Suppose the hypothesis is true for all minimally unsatisfiable CNF formulas containing $k > 1$ or fewer variables and having deficiency 1. Let ψ be a minimally unsatisfiable CNF formula with $k + 1$ variables and $k + 2$ clauses. From Theorem 29 there is a variable v in ψ such that literal v occurs in exactly one clause, say, c_v , and literal $\neg v$ occurs in exactly one clause, say, $c_{\neg v}$. By Lemma 9 the resolvent $\mathcal{R}_{c_{\neg v}}^{c_v}$ of c_v and $c_{\neg v}$ exists and $\mathcal{R}_{c_{\neg v}}^{c_v} \neq \emptyset$ or else ψ is not minimally unsatisfiable. Let $\psi' = (\psi \setminus \{c_v, c_{\neg v}\}) \cup \{\mathcal{R}_{c_{\neg v}}^{c_v}\}$. That is, ψ' is obtained from ψ by resolving c_v and $c_{\neg v}$ on variable v . By Theorem 30 ψ' is minimally unsatisfiable with k variables and $k + 1$ clauses. Then, by the induction hypothesis, there is a partition $\psi'_{v'}$ and $\psi'_{\neg v'}$ of clauses of ψ' and a variable v' such that literal v' occurs only in one clause of $\psi'_{v'}$, literal $\neg v'$ occurs only in one clause of $\psi'_{\neg v'}$, and, excluding v' , there is no variable overlap between $\psi'_{v'}$ and $\psi'_{\neg v'}$. Suppose $\mathcal{R}_{c_{\neg v}}^{c_v} \in \psi'_{\neg v'}$. Then $\psi'_{v'}$ and $(\psi'_{\neg v'} \setminus \{\mathcal{R}_{c_{\neg v}}^{c_v}\}) \cup \{c_v, c_{\neg v}\}$ (denoted ψ_v and $\psi_{\neg v}$, respectively) form a nonvariable overlapping partition of clauses of ψ (excluding v' and $\neg v'$), literal v' occurs once in a clause of ψ_v , and literal $\neg v'$ occurs once in a clause of $\psi_{\neg v}$. A similar statement holds if $\mathcal{R}_{c_{\neg v}}^{c_v} \in \psi'_{v'}$. Therefore, the hypothesis holds for ψ or any other minimally unsatisfiable formula of $k + 1$ variables and $k + 2$ clauses. The theorem follows. \square

Theorem 32 *A CNF formula ψ with n variables and $n + 1$ clauses is minimally unsatisfiable if and only if there is a refutation tree for ψ in which every clause of ψ labels a leaf exactly one time, every variable of ψ labels exactly two edges (once as a positive literal and once as a negative literal), and every edge label of the tree appears in at least one clause of ψ .*

Proof (\leftarrow) By induction on the size of the refutation tree. Suppose there is such a refutation tree T_ψ for ψ . If T_ψ consists of two edges, they must be labeled v and $\neg v$ for some variable v . In addition, each clause labeling a leaf of T_ψ must consist of one literal that is opposite to that labeling the edge the leaf is the endpoint of. Hence, ψ must be $(v) \wedge (\neg v)$ which is minimally unsatisfiable.

Now suppose the theorem holds for refutation trees of $k > 2$ or fewer edges and suppose \mathcal{T}_ψ has $k + 1$ edges. Let v be the variable associated with the root of \mathcal{T}_ψ , and let ψ_v and $\psi_{\neg v}$ be the sets of clauses labeling leaves in the subtree joined to the root edges labeled v and $\neg v$, respectively. Define $\psi_1 = \{c \setminus \{\neg v\} : c \in \psi_v\}$ and $\psi_2 = \{c \setminus \{v\} : c \in \psi_{\neg v}\}$. It is straightforward to see that the relationships between each refutation subtree rooted at a child of the root of \mathcal{T}_ψ and ψ_1 and ψ_2 are as described in the statement of the theorem. Hence, by the induction hypothesis, ψ_1 and ψ_2 are minimally unsatisfiable. Let c be a clause in ψ_1 . Let M_1 be an assignment to variables of ψ_1 satisfying $\psi_1 \setminus \{c\}$. Let $M_2 = M_1 \cup \{v\}$. Clearly, M_2 satisfies ψ_v as well as all clauses of $\psi_{\neg v}$ which contain literal v . The remaining clauses of $\psi_{\neg v}$ are a proper subset of clauses of ψ_2 and are satisfied by some truth assignment M_3 to variables of ψ_2 since ψ_2 is minimally unsatisfiable. Since the variables of ψ_1 and ψ_2 do not overlap, $M_2 \cup M_3$ is an assignment satisfying $\psi_v \cup \psi_{\neg v} \setminus \{c\}$ and therefore $\psi \setminus \{c\}$. The same result is obtained if c is removed from ψ_2 . Thus, ψ is minimally unsatisfiable.

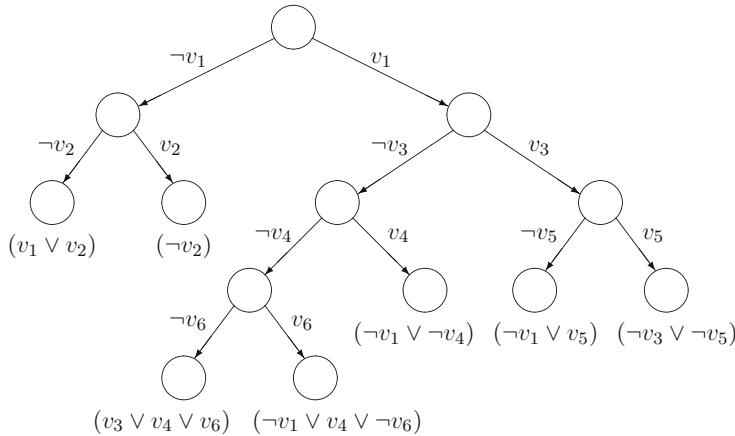
(\rightarrow) Build the refutation tree in conjunction with running Algorithm **Min Unsat Solver** as follows. Before running the algorithm, construct $n + 1$ (leaf) nodes and distinctly label each with a clause of ψ . While running the algorithm, add a new (non-leaf) node each time a resolvent is computed. Unlike the case for a normal refutation tree, label the new node with the resolvent. Construct two edges from the new node to the two nodes which are labeled by the two clauses being resolved (c_v and $c_{\neg v}$ in the algorithm). If the pivot variable is v , label the edge incident to the node labeled by the clause containing v (alternatively $\neg v$) $\neg v$ (v , respectively). Continue until a single tree containing all the original leaf nodes is formed.

The graph constructed has all the properties of the refutation tree as stated in the theorem. It must include one or more trees since each clause, original or resolvent, is used one time in computing a resolvent. Since two edges are added for each new non-leaf node and n new non-leaf nodes are added, there must be $2n + 1$ nodes and $2n$ edges in the structure. Hence, it must be a single tree. Any clause labeling a node contains all literals in clauses labeling leaves beneath that node minus all literals of pivot variables beneath and including that node. In addition, the label of the root is \emptyset . Therefore, all literals of a clause labeling a leaf are a subset of the complements of edge labels on a path from the root to that leaf. Obviously, the complement of each edge label appears at least once in leaf clauses. \square

An example of such a minimally unsatisfiable formula and corresponding refutation tree is shown in Fig. 52.

Now, attention is turned to the case of minimally unsatisfiable formulas with deficiency $\delta > 1$. Algorithms for recognizing and solving minimally unsatisfiable formulas in time $n^{O(k)}$ were first presented in [98] and [58]. The fixed-parameter tractable algorithm that is presented in [134] is discussed here.

Let ψ be a CNF formula with m clauses and n variables. The deficiency of ψ is the difference $m - n$. Each subformula of ψ has its own deficiency, namely, the difference between the number of clauses of the subformula and the number of variables it contains. The *maximum deficiency* of ψ is then the maximum of the deficiencies of all its subformulas. If, for every nonempty subset V' of variables



$$\psi = (v_1 \vee v_2) \wedge (\neg v_2) \wedge (v_3 \vee v_4 \vee v_6) \wedge (\neg v_1 \vee v_4 \vee \neg v_6) \wedge \\ (\neg v_1 \vee \neg v_4) \wedge (\neg v_1 \vee v_5) \wedge (\neg v_3 \vee \neg v_5)$$

Fig. 52 A minimally unsatisfiable CNF formula ψ of six variables and seven clauses and a corresponding refutation tree. Edge labels are variable assignments (e.g., $\neg v_1$ means v_1 has value 0). Each leaf is labeled with a clause that is falsified by the assignment indicated by the path from the root to the leaf

of ψ , there are at least $|V'| + q$ clauses $C \subset \psi$ such that some variable of V' is contained in C , then ψ is said to be q -expanding. The following four lemmas are stated without proof.

Lemma 10 *The maximum deficiency of a formula can be determined in polynomial time.* □

Lemma 11 *Let ψ be a minimally unsatisfiable CNF formula which has δ more clauses than variables. Then the maximum deficiency of ψ is δ .* □

Lemma 12 *Let ψ be a minimally unsatisfiable formula. Then for every nonempty set V' of variables of ψ , there is at least one clause $c \in \psi$ such that some variable of V' occurs in c .* □

Lemma 13 *Let ψ be a CNF formula with maximum deficiency Δ . A maximum matching of the bipartite variable-clause graph G_ψ of ψ does not cover Δ clause vertices.* □

Lemma 13 is important because it is used in the next lemma and because it shows that the maximum deficiency of a CNF formula can be computed with a $O(n^3)$ matching algorithm.

Lemma 14 Let ψ be a 1-expanding CNF formula with maximum deficiency δ . Let $\psi' \subset \psi$ be a subformula of ψ . Then the maximum deficiency of ψ' is less than δ .

Proof It is necessary to show that for any subset $\psi' \subset \psi$, if the number of variables contained in ψ' is n' , then $|\psi'| - n' < \delta$. Let G_ψ be the bipartite variable-clause graph of ψ . In what follows symbols will be used to represent clause vertices in G_ψ and clauses in ψ interchangeably. Choose a clause $c \in \psi \setminus \psi'$. Let M_{G_ψ} be a maximum matching on G_ψ that does not cover vertex c . There is always one such matching because, by hypothesis, every subset of variable vertices has a neighborhood which is larger than the subset, and this allows c 's cover to be moved to another clause vertex, if necessary. Let C be the set of all clause vertices of G_ψ that are not covered by M_{G_ψ} . By Lemma 13 $|C| = \delta$ so since $c \notin \psi'$, $|C \cap \psi'| < \delta$. Since M_{G_ψ} matches every clause vertex in $\psi' \setminus C$ to a variable that is in ψ' , the number of clauses in $\psi' \setminus C$ must be no bigger than n' . Therefore, $|\psi'| - n' \leq |\psi'| - |\psi' \setminus C|$. But $|\psi'| - |\psi' \setminus C|$ is the number of clauses in $C \cap \psi'$ which is less than δ . \square

Theorem 33 ([134]) Let ψ be a CNF formula with maximum deficiency δ . The satisfiability of ψ can be determined in time $O(2^\delta n^3)$ where n is the number of variables in ψ .

Proof Let m be the number of clauses in ψ . By hypothesis, $m \leq n + \delta$. Let $G = (V_1, V_2, E)$ be the variable-clause matching graph for ψ . Find a maximum matching M_G for G . Since $nm \leq n(n+\delta) = O(n^2)$, this can be done in $O(n^3)$ time by the well-known Hopcroft-Karp maximum cardinality matching algorithm for bipartite graphs [40]. The next step is to build a refutation tree for ψ of depth δ . \square

Theorem 34 ([134]) Let ψ be a minimally unsatisfiable CNF formula with δ more clauses than variables. Then ψ can be recognized as such in time $O(2^\delta n^4)$ where n is the number of variables in ψ .

Proof By Lemma 11 the maximum deficiency of ψ is δ which, by Lemma 13, can be checked in $O(n^3)$ time. Since, by Lemma 14, the removal of a clause from ψ reduces the maximum deficiency of ψ , the algorithm inferred by Theorem 33 may be used to check that, for each $c \in \psi$, $\psi \setminus \{c\}$ is satisfiable. The algorithm may also be used to check that ψ is unsatisfiable. Since the algorithm is applied $m + 1$ times and $m = n + \delta$, the complexity of this check is $O(2^\delta n^4)$. Therefore, the entire check takes $O(2^\delta n^4)$ time. \square

6.11 Bounded Resolvent Length Resolution

This class is considered here because it is a counterpoint to minimally unsatisfiable formulas. A CNF formula ψ is k-BRLR if Algorithm **BRLR** of Fig. 53 either generates all resolvents of ψ or returns *unsatisfiable*.

Algorithm 26.

```

BRLR( $\psi, k$ )
/* Input: a set of sets CNF formula  $\psi$  */ 
/* Output: “unsatisfiable,” “satisfiable,” or “give up” */
Repeat the following until some statement outputs a value:
  If there are clauses  $c_1, c_2 \in \psi$  that resolve to  $\mathcal{R}_{c_2}^{c_1} \notin \psi$ ,  $|\mathcal{R}_{c_2}^{c_1}| \leq k$ 
    Set  $\psi \leftarrow \psi \cup \mathcal{R}_{c_2}^{c_1}$ .
  Otherwise, if  $\emptyset \notin \psi$  do the following:
    If all resolvents have been generated then Output “satisfiable”.
    Otherwise, Output “give up”.
  If  $\emptyset \in \psi$ , Output “unsatisfiable.”
□

```

Fig. 53 Finding a k -bounded refutation

Algorithm **BRLR** implements a simplified form of *k -closure* [144]. It repeatedly applies the resolution rule to a CNF formula with the restriction that all resolvents are of size no greater than some fixed k . In other words, the algorithm finds what are sometimes called “ k -bounded” refutations.

For a given CNF formula with n variables and m clauses, the worst-case number of resolution steps required by the algorithm is

$$\sum_{i=1}^k 2^i \binom{n}{i} = 2^k \binom{n}{k} (1 + O(k/n)).$$

This essentially reflects the product of the cost of finding a resolvent and the maximum number of times a resolvent is generated. The latter is $2^k \binom{n}{k}$. The cost of finding a resolvent depends on the data structures used in implementing the algorithm.

For every clause, maintain a linked list of literals it contains, in order by index, and a linked list of possible clauses to resolve with such that the resolvent has no greater than k literals. Maintain a list \mathcal{T} of matrices of dimension $n \times 2, \binom{n}{2} \times 4, \dots, \binom{n}{k} \times 2^k$ such that each cell has value 1 if and only if a corresponding original clause of ψ or resolvent exists at any particular iteration of the algorithm. Also, maintain a list of clauses that may resolve with at least one other clause to generate a new resolvent: the list is threaded through the clauses. Assume a clause can be accessed in constant time and a clause can access its corresponding cell in \mathcal{T} in constant time by setting a link just one time as the clause is scanned the first time or created as a resolvent.

Initially, set to 1 all the cells of \mathcal{T} which correspond to a clause of ψ and set all other cells to 0. Over $\binom{m}{2}$ pairs of clauses, use $2L$ literal comparisons to determine whether the pair resolves. If so and their resolvent has k or fewer literals and the cell in \mathcal{T} corresponding to the resolvent has value 0, then set the cell to 1, add a link to the clause lists of the two clauses involved, and add to the potential resolvent

list any of the two clauses that is not already threaded through it. This accounts for complexity $O(Lm^2)$.

During an iteration, select a clause c_1 from the potential resolvent list. Scan through c_1 's resolvent list checking whether the cell of \mathcal{T} corresponding to the other clause, c_2 , has value 1. If so, delete c_2 from c_1 's list. If c_1 's list is scanned without finding a cell of value 0, delete c_1 from the potential resolvent list and try the next clause in the potential resolvent list. When some clause is paired with another having a cell of value 0 in \mathcal{T} , a new resolvent is generated. In this case, construct a new clause and create a resolvent list for the clause by checking for resolvents with all existing clauses.

6.12 Comparison of Classes

This section provides some intuition about the relative size and scope of the easy classes discussed above.

Observe that the SLUR, q-Horn, nested, and matched classes are incomparable; the class of q-Horn formulas without unit clauses is subsumed by the class of linear autark formulas and SLUR is incomparable with the linear autark formulas. All the other classes considered above are contained in one or more of the three. For example, Horn formulas are in the intersection of the q-Horn and SLUR classes and all 2-SAT formulas are q-Horn.

Any Horn formula with more clauses than distinct variables is not matched, but is both SLUR and q-Horn.

The following is a matched and q-Horn formula but is not a SLUR formula:

$$(v_1 \vee \neg v_2 \vee v_4) \wedge (v_1 \vee v_2 \vee v_5) \wedge (\neg v_1 \vee \neg v_3 \vee v_6) \wedge (\neg v_1 \vee v_3 \vee v_7).$$

In particular, in Algorithm **SLUR**, initially choosing 0 values for v_4 , v_5 , v_6 , and v_7 leaves an unsatisfiable formula with no unit clauses. To verify q-Horn membership, set $\alpha_1 = \alpha_2 = \alpha_3 = 1/2$ and the remaining α 's to 0 in (2).

The following formula is matched and SLUR but is not q-Horn:

$$(\neg v_2 \vee v_3 \vee \neg v_5) \wedge (\neg v_1 \vee \neg v_3 \vee v_4) \wedge (v_1 \vee v_2 \vee \neg v_4).$$

In particular, the satisfiability index of this formula is 4/3. To verify SLUR membership, observe that in Algorithm **SLUR** no choice sequence leads to an unsatisfiable formula without unit clauses.

The following formula is nested but not q-Horn (minimum Z is 5/4):

$$(\neg v_3 \vee \neg v_4) \wedge (v_3 \vee v_4 \vee \neg v_5) \wedge (\neg v_1 \vee v_2 \vee \neg v_3) \wedge (v_1 \vee v_3 \vee v_5).$$

The following formula is nested but not SLUR (choose v_3 first, use the branch where v_3 is 0 to enter a situation where satisfaction is impossible):

$$(v_1 \vee v_2) \wedge (v_1 \vee \neg v_2 \vee v_3) \wedge (\neg v_1 \vee v_4) \wedge (\neg v_1 \vee \neg v_4).$$

The following is a linear autark formula that is not q-Horn ($\alpha = \langle 0.5, -0.5, 0 \rangle$ and the minimum Z is 3/2):

$$(v_1 \vee v_2 \vee v_3) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3).$$

The following is SLUR but is not a linear autark formula (by symmetry, any variable choice sequence in **SLUR** leads to the same result which is a satisfying assignment and only $\alpha = \langle 0, 0, 0 \rangle$ satisfies the inequality of (14)):

$$(\neg v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee \neg v_2 \vee v_3) \wedge (v_1 \vee v_2 \vee \neg v_3) \wedge (\neg v_1 \vee \neg v_2 \vee \neg v_3).$$

The following formula is minimally unsatisfiable with one more clause than variable but is not 3-BRLR:

$$\begin{aligned} & (v_0 \vee v_1 \vee v_{48}) \wedge (v_0 \vee \neg v_1 \vee v_{56}) \wedge (\neg v_0 \vee v_2 \vee v_{60}) \wedge (\neg v_0 \vee \neg v_2 \vee v_{62}) \wedge \\ & (v_3 \vee v_4 \vee \neg v_{48}) \wedge (v_3 \vee \neg v_5 \vee v_{56}) \wedge (\neg v_3 \vee v_5 \vee v_{60}) \wedge (\neg v_3 \vee \neg v_5 \vee v_{62}) \wedge \\ & (v_6 \vee v_7 \vee v_{49}) \wedge (v_6 \vee \neg v_7 \vee \neg v_{56}) \wedge (\neg v_6 \vee v_8 \vee v_{60}) \wedge (\neg v_6 \vee \neg v_8 \vee v_{62}) \wedge \\ & (v_9 \vee v_{10} \vee \neg v_{49}) \wedge (v_9 \vee \neg v_{10} \vee \neg v_{56}) \wedge (\neg v_9 \vee v_{11} \vee v_{60}) \wedge (\neg v_9 \vee \neg v_{11} \vee v_{62}) \wedge \\ & (v_{12} \vee v_{13} \vee v_{50}) \wedge (v_{12} \vee \neg v_{13} \vee v_{57}) \wedge (\neg v_{12} \vee v_{14} \vee \neg v_{60}) \wedge (\neg v_{12} \vee \neg v_{14} \vee v_{62}) \wedge \\ & (v_{15} \vee v_{16} \vee \neg v_{50}) \wedge (v_{15} \vee \neg v_{16} \vee v_{57}) \wedge (\neg v_{15} \vee v_{17} \vee \neg v_{60}) \wedge (\neg v_{15} \vee \neg v_{17} \vee v_{62}) \wedge \\ & (v_{18} \vee v_{19} \vee v_{51}) \wedge (v_{18} \vee \neg v_{19} \vee \neg v_{57}) \wedge (\neg v_{18} \vee v_{20} \vee \neg v_{60}) \wedge (\neg v_{18} \vee \neg v_{20} \vee v_{62}) \wedge \\ & (v_{21} \vee v_{22} \vee \neg v_{51}) \wedge (v_{21} \vee \neg v_{22} \vee \neg v_{57}) \wedge (\neg v_{21} \vee v_{23} \vee \neg v_{60}) \wedge (\neg v_{21} \vee \neg v_{23} \vee v_{62}) \wedge \\ & (v_{24} \vee v_{25} \vee v_{52}) \wedge (v_{24} \vee \neg v_{25} \vee v_{58}) \wedge (\neg v_{24} \vee v_{26} \vee v_{61}) \wedge (\neg v_{24} \vee \neg v_{26} \vee \neg v_{62}) \wedge \\ & (v_{27} \vee v_{28} \vee \neg v_{52}) \wedge (v_{27} \vee \neg v_{28} \vee v_{58}) \wedge (\neg v_{27} \vee v_{29} \vee v_{61}) \wedge (\neg v_{27} \vee \neg v_{29} \vee \neg v_{62}) \wedge \\ & (v_{30} \vee v_{31} \vee v_{53}) \wedge (v_{30} \vee \neg v_{31} \vee \neg v_{58}) \wedge (\neg v_{30} \vee v_{32} \vee v_{61}) \wedge (\neg v_{30} \vee \neg v_{32} \vee \neg v_{62}) \wedge \\ & (v_{33} \vee v_{34} \vee \neg v_{53}) \wedge (v_{33} \vee \neg v_{34} \vee \neg v_{58}) \wedge (\neg v_{33} \vee v_{35} \vee v_{61}) \wedge (\neg v_{33} \vee \neg v_{35} \vee \neg v_{62}) \wedge \\ & (v_{36} \vee v_{37} \vee v_{54}) \wedge (v_{36} \vee \neg v_{37} \vee v_{59}) \wedge (\neg v_{36} \vee v_{38} \vee \neg v_{61}) \wedge (\neg v_{36} \vee \neg v_{38} \vee v_{62}) \wedge \\ & (v_{39} \vee v_{40} \vee \neg v_{54}) \wedge (v_{39} \vee \neg v_{40} \vee v_{59}) \wedge (\neg v_{39} \vee v_{41} \vee \neg v_{61}) \wedge (\neg v_{39} \vee \neg v_{41} \vee \neg v_{62}) \wedge \\ & (v_{42} \vee v_{43} \vee v_{55}) \wedge (v_{42} \vee \neg v_{43} \vee \neg v_{59}) \wedge (\neg v_{42} \vee v_{44} \vee \neg v_{61}) \wedge (\neg v_{42} \vee \neg v_{44} \vee \neg v_{62}) \wedge \\ & (v_{45} \vee v_{46} \vee \neg v_{55}) \wedge (v_{45} \vee \neg v_{46} \vee \neg v_{59}) \wedge (\neg v_{45} \vee v_{47} \vee \neg v_{61}) \wedge (\neg v_{45} \vee \neg v_{47} \vee \neg v_{62}). \end{aligned}$$

This formula was obtained from a complete binary refutation tree, variables v_0 to v_{47} labeling edges of the bottom two levels and v_{48} to v_{62} labeling edges of the top four levels. Along the path from the root to a leaf, the clause labeling that leaf contains all variables of the bottom two levels and one variable of the top four levels. Thus, resolving all clauses in a subtree rooted at variable v_{3i} , $0 \leq i \leq 15$, leaves a resolvent of four literals, all taking labels from edges in the top four levels.

To any k -BRLR formula ψ that is unsatisfiable, add another clause arbitrarily using the variables of ψ . The result is a formula that is not minimally satisfiable.

The above ideas can be extended to show that each of the classes contains a formula that is not a member of any of the others. These examples may be extended to infinite families with the same properties.

The subject of comparison of classes will be revisited in Sect. 6.13 using probabilistic measures to determine relative sizes of the classes.

6.13 Probabilistic Comparison of Incomparable Classes

Formulas that are members of certain polynomial-time solvable classes that have been considered in this section appear to be much less frequent than formulas that can usually be solved efficiently by some variant of DPLL. This statement is supported by a probabilistic analysis of these classes where a random formula is an instance of k -SAT with m clauses chosen uniformly and without replacement from all possible width k clauses taken from n variables. A few examples may help to illuminate. In what follows ψ is used to denote a random k -SAT formula.

Consider the class of Horn formulas (Sect. 6.2) first. The probability that a randomly generated clause is Horn is $(k+1)/2^k$, so the probability that ψ is Horn is $((k+1)/2^k)^m$. This tends to 0 as m tends to ∞ for any fixed k . For a hidden Horn formula (Sect. 6.3), regardless of switch set, there are only $k+1$ out of 2^k ways (k ways to place a positive literal and 1 way to place only negative literals) that a random clause can become Horn. Therefore, the expected number of successful switch sets is $2^n((k+1)/2^k)^m$. This tends to 0 for increasing m and n if $m/n > 1/(k - \log_2(k+1))$. Therefore, by Markov's inequality, ψ is not hidden Horn, with probability tending to 1, if $m/n > 1/(k - \log_2(k+1))$. Even when $k = 3$, this is $m/n > 1$. This bound can be improved considerably by finding complex structures that imply a formula cannot be hidden Horn. Such a structure is presented next.

The following result for q-Horn formulas (Sect. 6.5) is taken from [59]. For $p = \lfloor \ln(n) \rfloor \geq 4$, call a set of p clauses a *c-cycle* if all but two literals can be removed from each of $p-2$ clauses, all but three literals can be removed from two clauses, the variables can be renamed, and the clauses can be reordered in the following sequence:

$$(v_1 \vee \neg v_2) \wedge (v_2 \vee \neg v_3) \wedge \dots \wedge (v_i \vee \neg v_{i+1} \vee v_{p+1}) \wedge \dots \wedge (v_j \vee \neg v_{j+1} \vee \neg v_{p+1}) \wedge \dots \wedge (v_p \vee \neg v_1), \quad (18)$$

where $v_i \neq v_j$ if $i \neq j$. Use the term *cycle* to signify the existence of cyclic paths through clauses which share a variable: that is, by jumping from one clause to another clause only if the two clauses share a variable, one may eventually return to the starting clause. Given a c-cycle $\mathcal{C} \subset \psi$, if no two literals removed from \mathcal{C} are the same or complementary, then \mathcal{C} is called a *q-blocked c-cycle*.

If ψ has a q-blocked c-cycle, then it is not q-Horn. Let a q-blocked c-cycle in ψ be represented as above. Develop satisfiability index inequalities (2) for ψ . After rearranging terms in each, a subset of these inequalities is as follows

$$\alpha_1 \leq Z - 1 + \alpha_2 - \dots \quad (19)$$

...

$$\alpha_i \leq Z - 1 + \alpha_{i+1} - \alpha_{p+1} - \dots$$

...

$$\begin{aligned} \alpha_j &\leq Z - 1 + \alpha_{j+1} - (1 - \alpha_{p+1}) - \dots \\ &\dots \\ \alpha_p &\leq Z - 1 + \alpha_1 - \dots \end{aligned} \tag{20}$$

From inequalities (19) to (20), it can be deduced that

$$\alpha_1 \leq pZ - p + \alpha_1 - (1 - \alpha_{p+1} + \alpha_{p+1}) - \dots$$

or

$$0 \leq pZ - p - 1 + \dots$$

where all the terms in \dots are nonpositive. Thus, all solutions to (19) through (20) require $Z > (p+1)/p = 1 + 1/p = 1 + 1/\lfloor \ln^2 n \rfloor > 1 + 1/n^\beta$ for any fixed $\beta < 1$. This violates the requirement (Theorem 19) that $Z \leq 1$ in order for ψ to be q-Horn.

The expected number of q-blocked c-cycles can be found and the second moment method applied to give the following result.

Theorem 35 *A random k -SAT formula is not q-Horn, with probability tending to 1, if $m/n > 4/(k^2 - k)$.* \square

For $k = 3$ this is $m/n > 2/3$.

A similar analysis yields the same results for hidden Horn, SLUR, CC-balanced, or extended Horn classes. The critical substructure which causes ψ not to be SLUR is called a crisscross loop. An example is shown in Fig. 54 as a propositional connection graph. Just one crisscross loop in ψ prevents it from being SLUR. Comparing Fig. 54 and expression (18), it is evident that both the SLUR and q-Horn classes are *vulnerable* to certain types of *cyclic* structures. Most other polynomial-time solvable classes are similarly vulnerable to cyclic structures of various kinds. But random k -SAT formulas are constructed without consideration of such cyclic structures: at some point as m/n is increased, cycles begin to appear in ψ in abundance, and when this happens, cycles that prevent membership in one of the above named polynomial-time solvable classes also show up. Cycles appear in abundance when $m/n > 1/O(k^2)$. It follows that a random k -SAT formula is *not* a member of one of the above named polynomial-time solvable classes, with probability tending to 1, when $m/n > 1/O(k^2)$. By contrast, simple polynomial-time procedures will solve a random k -SAT formula with high probability when $m/n < 2^k/O(k)$. The disappointing conclusion that many polynomial-time solvable classes are relatively rare among random k -SAT formulas because they are vulnerable to cyclic structures is given added perspective by considering the class of matched formulas.

A k -CNF formula is a matched formula (see Page 415) if there is a total matching in its variable-clause matching bipartite graph: a property that is *not* affected by cyclic structures as above. A probabilistic analysis tells us that a random k -SAT

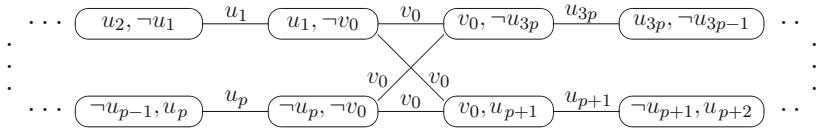


Fig. 54 A criss-cross loop of $t = 3p + 2$ clauses represented as a propositional connection graph. Only cycle literals are shown in the nodes; padding literals, required for $k \geq 3$ and different from cycle literals, are present but are not shown

generator produces *many more* matched formulas than SLUR or q-Horn formulas. Let $Q \subset \psi$ be any subset of clauses of ψ . Denote by $\mathbf{V}(Q)$ the neighborhood of Q : that is, the set of variables that occur in Q . Then $\mathbf{V}(Q)$ is the set variables corresponding to a subset of vertices in V_2 of G_ψ that are adjacent to the vertices in V_2 that correspond to clauses in Q . Denote the deficiency of Q by $\Delta(Q)$. From the definition of deficiency (Page 430), $\Delta(Q) = |Q| - |\mathbf{V}(Q)|$. A subset $Q \subset \psi$ is said to be *deficient* if $\Delta(Q) > 0$. The following theorem is well known.

Theorem 36 (Hall's Theorem [70]) *Given a bipartite graph with vertex sets V_1 and V_2 , a matching that includes every vertex of V_1 exists if and only if no subset of V_1 is deficient.* \square

Theorem 37 *Random k -SAT formulas are matched formulas with probability tending to 1 if $m/n < r(k)$ where $r(k)$ is given by the following table [59].*

k	$r(k)$
3	0.64
4	0.84
5	0.92
6	0.96
7	0.98
8	0.990
9	0.995
10	0.997

\square

Theorem 37 may be proved by finding a lower bound on the probability that a corresponding random variable-clause matching graph has a total matching. By [Theorem 36](#) it is sufficient to prove an upper bound on the probability that there exists a deficient subset of clause vertices, and then show that the bound tends to 0 for $m/n < r(k)$ as given in the theorem. This bound is obtained by the first moment method, that is, by finding the expected number of deficient subsets.

The results in this subsection up to this point are interesting for at least two reasons. First, [Theorem 37](#) says that random k -SAT formulas are matched formulas with high probability if $m/n < r(k)$ which is approximately 1. But, by [Theorem 35](#) and similar results, a random k -SAT formula is almost never a member of one of the well-studied classes mentioned earlier unless $m/n < 1/O(k^2)$. As already pointed out, this is somewhat disappointing and surprising because all the other classes were proposed for rather profound reasons, usually reflecting cases when corresponding instances of integer programming present polytopes with some special properties. Despite all the theory that helped establish these classes, the matched class, mostly ignored in the literature because it is so trivial in nature, turns out to be, in some probabilistic sense, much bigger than all the others.

Second, the results provide insight into the nature of larger polynomial-time solvable classes of formulas. Classes vulnerable to cyclic structures appear to be handicapped relative to classes that are not. In fact, the matched class has been generalized considerably to larger polynomial-time solvable classes such as linear autarkies [97, 145] which are described in [Sect. 6.9](#) and biclique satisfiable formulas of [Sect. 6.7](#).

But there is disappointing news concerning linear autarkies. Consider the test for satisfiability that is implied by [Theorem 27](#): find the satisfiability index z of the given formula and compare against the width, say, k , of the formula's shortest clause; if $z < k/2$, then the formula is satisfiable. This test is based on the fact that Algorithm **LinAut**, [Page 422](#), will output $\psi' = \emptyset$ on any formula for which $z < k/2$.

Theorem 38 *The above test for satisfiability does not succeed on a random k -SAT formula ψ with probability tending to 1 as $m, n \rightarrow \infty$ and $m/n > 1$.* □

7 The k -SAT Problem

The previous sections present SAT algorithm designs and applications. Those sections are intended for people who use SAT technology, want to find out more about what is going on under the hood, and may want to experiment with new algorithm designs to suit particular problems. Those sections may also be useful to people who find SAT interesting and want to see what others have accomplished before trying out their new ideas. This section complements those sections. The success of SAT algorithms is somewhat mysterious owing to the very simple and cumbersome representation of constraints in conjunctive normal form. Is there any particular reason that SAT solvers can often perform so well, defying normal, reasonable intuition? Why can a few tweaks to SAT algorithms result in orders of magnitude performance improvements? Theoretical results have contributed partial answers to such questions and have even influenced the design of SAT solvers. [Section 6](#) presents some of these results. This section highlights some others. In particular, the results of this section apply to the k -SAT problem. In this section a k -SAT formula is a Boolean formula expressed in conjunctive normal form where each clause has exactly k distinct literals, no two of which

are complementary. A *random k -SAT formula* (restated from Sect. 6.13) is a Boolean formula in conjunctive normal form with m clauses, each of which is chosen uniformly and without replacement from all possible width k clauses taken from n variables. The *clause density* of a random k -SAT formula is the ratio m/n .

The theoretical results on k -SAT are best understood in terms of the so-called satisfiability threshold r_k which is the limiting clause density below which random k -SAT formulas are nearly always satisfiable and above which random k -SAT formulas are nearly always unsatisfiable. In [1, 2] the second moment method was applied to a variant of k -SAT to show that $r_k = 2^k \log 2 - O(k)$. In Sect. 6.13 several algorithms that are designed to solve subclasses of CNF formulas were analyzed with respect to random k -SAT formulas. It was shown that none of these algorithms could solve a significant proportion of random k -SAT formula if density is greater than 1.

Non- or limited backtracking variants of DPLL have been analyzed with respect to random k -SAT formulas with results that are significantly better than and equally insightful to those of Sect. 6.13. In [32–34] it was shown that DPLL will nearly always solve a random k -SAT formula if $m/n < O(2^k/k)$. The analysis can be visualized as a system of clause flows between nodes that represent a collection of clauses of widths from 1 to k . Each node holds a number of clauses of a particular width and on every iteration there is some average flow out of a node as clauses are satisfied and literals are falsified. The outward flow due to falsified literals becomes an inward flow to a node holding clauses of one fewer literal. The reason for the k in the denominator of the right side becomes apparent from this analysis: the average flow of clauses into the node representing unit clauses rises above 1 if $m/n = \omega(2^k/k)$, and since only 1 clause is removed from that node in one iteration, on average, the number of unit clauses rises and eventually it becomes likely that a complementary pair of unit clauses exists in the partially assigned formula. These results may be made to apply to any *myopic algorithm* (i.e., a straight line algorithm whose distribution of expressions corresponding to a particular coalesced state under the spectral coalescence of states) can be expressed by its spectral components alone: that is, by the *number* of clauses of width i , for all $1 \leq i \leq k$, and the *number* of assigned variables. But in [3] it is shown that no myopic algorithm can succeed in solving random 3-SAT formulas with high probability if $m/n > 3.26$. However, using a non-myopic greedy algorithm, m/n may be as high as 3.52 before finding solutions becomes unlikely [69, 83, 84]. Note this is well below the presumed threshold $r_3 = 4.26$ and is also well inside the range of densities for which Walksat and Survey Propagation do a good job of solving random 3-SAT formulas.

Analyses have been performed on the unsatisfiable side of the threshold as well. In particular, the probability that the length of a shortest resolution proof (this includes DPLL algorithms) is bounded by a polynomial in n when $m/n > c'_k 2^k$, c'_k some constant, and $\lim_{m,n \rightarrow \infty} m/n = O(1)$ tends to 0 [17, 35]. But, if m is great enough, a random k -SAT formula can, with high probability, be efficiently

proven to have no solution. For example, if $m/n = \Theta(n^{k-1})$, the probability that there exist 2^k clauses containing the same variables spanning all different literal complementation patterns tends to 1, and since such a pattern can be identified in polynomial time and certifies there is no solution, random expressions can be proven not to have a solution with probability tending to 1 in this case. The best resolution can do is not much different from $m/n = \Theta(n^{k-1})$ [14]. This lack of success has led to investigation of alternatives to resolution. A notable example is to cast SAT as a HITTING SET problem and sum the number of variables that are *forced* to be set to 1 and the number that are forced to be set to 0. If this sum can be shown to be greater than n in polynomial time, a *short* proof of unsatisfiability follows. This idea has been applied to random k -SAT and yields a polynomial-time algorithm that nearly always proves unsatisfiability when $m/n = n^{k/2+o(1)-1}$ or greater, a considerable improvement over resolution results [64]. This may be regarded as an example of how probabilistic analysis has driven the search for improved, alternative algorithms.

Theoretical results on random k -SAT have provided a visual record of the nature of problems that are hard for some algorithms and not for others. If one were to produce a figure in which formulas appear as points at some distance above the figure *floor*, where distance is measured in the number of clauses that must be falsified for any assignments and where points are connected to other points which are *close* in terms of Hamming distance, one would see one or more inverted *hills*, some of which may actually touch the floor. Each hill reaches a local minimum called a *local energy minimum* (a contribution from physicists who have studied this). When clause density is low, there exists a single local energy minimum. Then as density is increased, there is a sudden change to several local minima. Finally, there is another jump where exponentially many local minima materialize. The first jump occurs at $m/n = O(2^k)$, and the second jump occurs at $m/n = O(2^k)$. Thus, it is not surprising that so many algorithms perform miserably on unsatisfiable formulas with density just above the transition point. Presumably, as density is increased further, these local minima coalesce (since the minima cannot go below 0), and verifying unsatisfiability becomes easy again. Theoretical results corroborate this [14, 64].

Theoretical results have revealed performance upper bounds on solving k -SAT. The algorithms that are inferred by these results are not considered practical, at least not on their own, and are surprisingly simple. For example, consider the probabilistic algorithm of Fig. 55. The analysis of [123] shows that this algorithm almost always terminates in time that is within a polynomial factor of 1.334^n for any 3-SAT input, of 1.5^n for any 4-SAT input, of 1.6^n for any 5-SAT input, and so on. These complexities, although not considered to represent efficient algorithms, are far below the 2^n complexity of a brute force search. Other results followed. Notably, the bound for 3-SAT was lowered to 1.324^n by combining Schöning's algorithm with the probabilistic algorithm of [115] in a nontrivial way [79], and the

best probabilistic bound is currently $O\left(2^{n\left(1-\frac{m}{\ln(n)}+\frac{1}{\ln(\ln(m))}\right)}\right)$ [48].

```

ProbSat ( $\psi$ )
/* Input:  $\psi$  is a set of sets of literals representing a CNF formula, */
/*           $\psi$  contains  $m$  clauses taken from variable set  $V$ ,  $|V| = n$  */
Randomly choose  $M \subseteq V$ . /*  $M$  is a truth assignment */
 $\psi = \{c \in \psi : \text{for all } l \in c, \text{ if } l \text{ is positive } l \notin M, \text{ otherwise } l \in M\}$ .
Repeat the following  $3n$  times:
  If  $\psi = \emptyset$ , stop and return "satisfiable"
  Randomly choose  $c \in \psi$ .
  Randomly choose  $l \in c$ .
  If  $l$  is positive,  $M = M \cup \{l\}$  else  $M = M \setminus \{\neg l\}$ .
   $\psi = \{c \in \psi : \text{for all } l \in c, \text{ if } l \text{ is positive } l \notin M, \text{ otherwise } l \in M\}$ .
  return "unknown."

```

Fig. 55 Probabilistic algorithm of Schöning

Upper bounds have also been obtained for deterministic algorithms. An early upper bound of $O(\alpha_k^n \cdot |\psi|)$ was obtained in (B. Monien and E. Speckenmeyer, 1979, 3-Satisfiability is testable in $O(1.62^r)$ steps, Unpublished manuscript) [111] for a simple algorithm that seeks and eliminates autarkies (Page 393) when possible. The factor α_k^n bounds the Fibonacci-like recursion

$$\begin{aligned} T_k(1) &= T_k(2) = \dots = T_k(k-1) = 1 \quad \text{and} \\ T_k(n) &= T_k(n-1) + T_k(n-2) + \dots + T_k(n-k+1), \quad \text{for } n \geq k, \end{aligned}$$

where $T_k(n)$ expresses the worst-case complexity for solving k -SAT in this manner. For example, $\alpha_3 \geq 1.681$, $\alpha_4 \geq 1.8393$, and $\alpha_5 \geq 1.9276$. Specifically, the algorithm looks for a shortest clause $c = \{l_1 \vee \dots \vee l_{|c|}\}$ in the current formula ψ . If c has $k-1$ or fewer literals, then ψ is split into at most $k-1$ subformulas according to the following subassignments:

$$\begin{aligned} (1) \quad l_1 &= 1 \\ (2) \quad l_1 &= 0, l_2 = 1 \\ &\dots \\ |c|) \quad l_1 &= l_2 = \dots = l_{|c|-1} = 0, l_{|c|} = 1. \end{aligned}$$

If all clauses c of ϕ have length k , then ϕ is split into k subformulas as described, and each subformula either contains a clause of length at most $k-1$ or one of the resulting subformulas only contains clauses of length k . In the latter case, the corresponding subassignment is autark (Page 393), that is, all clauses containing these variables are satisfied by this subassignment, thereby pruning a branch of the search space. Numerous results reducing the deterministic upper bounds followed. The best deterministic bound for solving k -SAT is $O((2 - \frac{2}{k+1})^n)$ [49]. For CNF formulas of unrestricted clause length, the best worst-case time bound for

a deterministic algorithm is $O(2^{n(1-\frac{1}{\log(2m)})})$ [47]. The algorithm was obtained by derandomizing a probabilistic algorithm of [124].

8 Other Topics

Several significant topics are not treated in this chapter. There are experimental algorithms that control the space of solutions with linear or quadratic cuts. The Handbook on Satisfiability [19] is a good source of information about these. Several probabilistic algorithms other than those presented here have been proposed, for example, simulated annealing and genetic algorithm variants. These were omitted in favor of more influential varieties. Message passing algorithms such as Survey Propagation have been omitted as they seem to have a special niche. More general constraint programming algorithms have been omitted as the focus here is strictly on SAT. The use of SAT in non-monotonic settings was mentioned in the applications section, but the topic is too broad to cover adequately in this chapter so stable models, well-founded semantics, and answer set programming algorithms, among others, have been omitted.

9 Conclusion

Resolution continues to be the foundation for modern SAT solvers. Resolution alone is inadequate for most inputs. However, advances known collectively as conflict-driven clause learning, plus advanced search and cache heuristics, have resulted in CNF solvers that are quite useful in a variety of roles related to problems in combinatorial optimization. One of the most useful roles of SAT to date is to support solvers of mixed logic systems, known as Satisfiability Modulo Theory solvers.

Resolution-based solvers are gaining in performance and are replacing reduced ordered binary decision diagrams in some roles. Gröbner bases applied to SAT promise to replace resolution-based algorithms in significantly many roles, but this has not happened yet because producing an optimal order of operations has remained illusive. The same can be said for other algebraic systems such as cutting planes. For awhile it seemed stochastic local search algorithms would replace resolution-based algorithms in many roles, but this has not happened due to the advances in conflict-driven clause learning noted above and because local search techniques, as developed, are not able to prove unsatisfiability.

Some classes of SAT are solvable in polynomial time. There are many such classes and many of those have been analyzed over the years. Using a probabilistic measure, the scope of such classes seems limited.

Theoretical analyses illustrate the reasons why some algorithms work under certain circumstances. Theoretical results have been used to design new algorithms such as the probabilistic algorithm presented in this chapter. Theoretical results also reveal upper bounds on algorithm performance.

Glossary

Algorithm: (328, 336, 340)

A specific set of instructions for carrying out a procedure or solving a problem, usually with the requirement that the algorithm terminate at some point.

Boolean Function: (316)

A mapping $\{0, 1\} \times \{0, 1\} \times \dots \times \{0, 1\} \mapsto \{0, 1\}$. If the dimension of the domain is n , the number of possible functions is 2^{2^n} .

Clause: (314, also see **Formula, CNF**)

In CNF formulas a clause is a disjunction of literals such as the following: $(\bar{a} \vee b \vee c \vee d)$. A clause containing only negative (positive) literals is called a *negative clause* (alternatively, a *positive clause*). In this chapter a disjunction of literals is also written as a set such as this: $\{\bar{a}, b, c, d\}$. Either way, the *width* of a clause is the number of literals it contains. In logic programming a clause is an implication such as the following: $(a \wedge b \wedge c \rightarrow g)$. In this chapter, if a formula is a conjunction or disjunction of expressions, each expression is said to be a *clause*.

Clause Width: (see **Clause**)

Edge: (see **Graph**)

Endpoint: (338, 338, 345, 415, 416, 429)

One of two vertices spanned by an edge. See **Graph**.

Formula, DNF: (344)

DNF stands for disjunctive normal form. Let a literal be a variable or a negated variable. Let a *conjunctive clause* be a single literal or a conjunction of two or more literals (see **Clause**). A DNF formula is an expression of the form $C_1 \vee C_2 \vee \dots \vee C_m$ where each C_i , $1 \leq i \leq m$, is a conjunctive clause: that is, a DNF formula is a disjunction of some number m of conjunctive clauses. A conjunctive clause evaluates to *true* under an assignment of values to variables if all its literals has value *true* under the assignment.

Formula, CNF: (327, 329, 340, 313)

CNF stands for conjunctive normal form. Let a literal be a variable or a negated variable. Let a *disjunctive clause* be a single literal or a disjunction of two or more literals (see **Clause**). A CNF formula is an expression of the form $C_1 \wedge C_2 \wedge \dots \wedge C_m$ where each C_i , $1 \leq i \leq m$, is a disjunctive clause: that is, a CNF formula is a conjunction of some number m of disjunctive clauses. In this chapter a disjunctive clause, sometimes called a clause when the context is clear, is regarded to be a set of literals and a CNF formula to be a set of clauses. Thus, the following is an example of how a CNF formula is expressed in this chapter.

$$\{\{\bar{a}, b\}, \{a, c, d\}, \{c, \bar{d}, \bar{e}\}\}.$$

A CNF formula is said to be *satisfied* by an assignment of values to its variables if the assignment causes all its clauses to evaluate to *true*. A clause evaluates to *true* under an assignment of values to variables if at least one of its literals has value *true* under the assignment.

Formula, Horn: (315, 328)

A CNF formula in which every clause has at most one positive literal. Satisfiability of Horn formulas can be determined in linear time [53, 78]. Horn formulas have the remarkable property that, if satisfiable, there exists a unique minimum satisfying assignment with respect to the value *true*. In other words, the set of all variables assigned value *true* in any satisfying assignment other than the unique minimum one includes the set of variables assigned value *true* in the minimum one.

Formula, Minimally Unsatisfiable: (317)

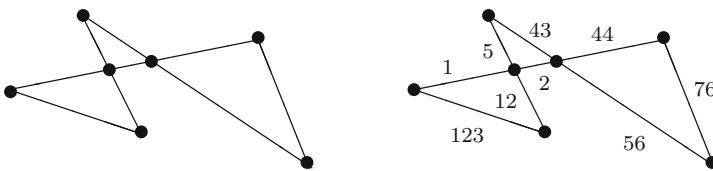
An unsatisfiable CNF formula such that removal of any clause makes it satisfiable.

Formula, Propositional or Boolean: (327, 336, 340, 342, 315)

A Boolean variable is a formula. If ψ is a formula, then (ψ) is a formula. If ψ is a formula, then $\neg\psi$ is a formula. If ψ_1 and ψ_2 are formulas and \mathcal{O}_b is a binary Boolean operator, then $\psi_1 \mathcal{O}_b \psi_2$ is a formula. In some contexts other than logic programming, \bar{a} or $\overline{\psi}$ is used instead of $\neg a$ or $\neg\psi$ to denote negation of a variable or formula. Formulas evaluate to *true* or *false* depending on the operators involved. Precedence from highest to lowest is typically from parentheses, to \neg , to binary operators. Association, when it matters as in the case of \rightarrow (implies), is typically from right to left.

Graph: (340)

A mathematical object composed of points known as vertices or nodes and lines connecting some (possibly empty) subset of them, known as edges. Each edge is said to span the vertices it connects. If weights are assigned to the edges, then the graph is a *weighted graph*. Below is an example of a graph and a weighted graph.

**Graph, Directed:** (321)

A graph in which some orientation is given to each edge.

Graph, Directed Acyclic: (324, 321, 320)

A directed graph such that, for every pair of vertices v_a and v_b , there is not both a path from v_a to v_b and a path from v_b to v_a .

Graph, Rooted Directed Acyclic: (324, 321, 320)

A connected, directed graph with no cycles such that there is exactly one vertex, known as the *root*, whose incident edges are all oriented away from it.

Literal: (313) see also **Formula, CNF**

A Boolean variable or the negation of a Boolean variable. In the context of a formula, a negated variable is called a *negative* literal and an unnegated variable is called a *positive* literal.

Logic Program, Normal: (315)

A formula consisting of implicational clauses. That is, clauses have the form $(a \wedge b \wedge c \rightarrow g)$ where atoms to the left of \rightarrow could be positive or negative literals.

Maximum Satisfiability (MAX-SAT): ()

The problem of determining the maximum number of clauses of a given CNF formula that can be satisfied by some truth assignment.

Model, Minimal Model: (316)

For the purposes of this chapter, a model is a truth assignment satisfying a given formula. A minimal model is such that a change in value of any *true* variable causes the assignment not to satisfy the formula. See also **Formula, Horn**.

\mathcal{NP} -hard: (327)

A very large class of difficult combinatorial problems. There is no known polynomial-time algorithm for solving any \mathcal{NP} -hard problem, and it is considered unlikely that any will be found. For a more formal treatment of this topic, see M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of \mathcal{NP} -completeness*, W.H. Freeman, San Francisco, 1979.

Operator, Boolean: (334, 313, 345)

A mapping from binary Boolean vectors to $\{0, 1\}$. Most frequently used binary operators are

Or	\vee :	$\{00 \mapsto 0; 10, 01, 11 \mapsto 1\}$
And	\wedge :	$\{00, 01, 10 \mapsto 0; 11 \mapsto 1\}$
Implies	\rightarrow :	$\{10 \mapsto 0; 00, 01, 11 \mapsto 1\}$
Equivalent	\leftrightarrow :	$\{01, 10 \mapsto 0; 00, 11 \mapsto 1\}$
XOR	\oplus :	$\{00, 11 \mapsto 0; 01, 10 \mapsto 1\}$

out of the 16 possible mappings. A Boolean operator \mathcal{O} shows up in a formula like this: $(v_l \mathcal{O} v_r)$ where v_l is called the left operand of \mathcal{O} and v_r is called the right operand of \mathcal{O} . Thus, the domain of \mathcal{O} is a binary vector whose first component is the value of v_l and whose second component is the value of v_r . In the text patterns of 4 bits are sometimes used to represent an operator: the first bit is the mapping from 00, the second from 01, the third from 10, and the fourth from 11. Thus, the operator 0001 applied to v_l and v_r has the same functionality as $(v_l \wedge v_r)$, the operator 0111 has the same functionality as $(v_l \vee v_r)$, and the operator 1101 has the same functionality as $(v_l \rightarrow v_r)$. The only meaningful unary operator is $\neg:\{1 \mapsto 0; 0 \mapsto 1\}$. Sometimes \bar{a} or ψ is written for $\neg a$ or $\neg \psi$ where a is a variable and ψ is a formula.

Operator, Temporal: (334, 334)

An operator used in temporal logics. Basic operators include *henceforth* ($\square \psi_1$), *eventually* ($\diamond \psi_1$), *next* ($\circ \psi_1$), and *until* ($\psi_1 \mathcal{U} \psi_2$).

Satisfied Clause: see **Formula, CNF**

Satisfiability (SAT): (327, 330, 340)

The problem of deciding whether there is an assignment of values to the variables of a given Boolean formula that makes the formula *true*. In 1971, Cook showed that the general problem is NP-complete even if restricted to CNF formulas containing clauses of width 3 or greater or if the Boolean operators are restricted

to any truth-functionally complete subset. However, many efficiently solved subclasses are known.

State: (334, 334, 336)

A particular assignment of values to the parameters of a system.

Subgraph: (340, 340, 342)

A graph whose vertices and edges form subsets of the vertices and edges of a given graph where an edge is contained in the subset only if both its endpoints are.

Unit Clause: (342)

A clause consisting of one literal. See also **Clause** in the glossary.

Variable, Propositional or Boolean: (330, 340, 342))

An object taking one of two values {0, 1}.

Vertex: see **Graph****Vertex Weighted Satisfiability:** (328)

The problem of determining an assignment of values to the variables of a given CNF formula, with weights on the variables, that satisfies the formula and maximizes the sum of the weights of *true* variables. The problem of finding a minimal model for a given satisfiable formula is a special case.)

Weighted Maximum Satisfiability: (340, 342)

The problem of determining an assignment of values to the variables of a given CNF formula, with weights on the clauses, which maximizes the sum of the weights of satisfied clauses.

Acknowledgements We thank John S. Schlipf for his help in writing some of the text, particularly the sections on the diagnosis of circuit faults and Binary Decision Diagrams. We especially appreciate John's attention to detail which was quite valuable to us.

Cross-References

- ▶ [Hardness and Approximation of Network Vulnerability](#)
 - ▶ [Probabilistic Verification and Non-approximability](#)
 - ▶ [Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work](#)
-

Recommended Reading

1. D. Achlioptas, C. Moore, The asymptotic order of the random k -SAT thresholds, in *43rd Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, 2002)
2. D. Achlioptas, Y. Peres, The threshold for random k -SAT is $2^k \log 2 - O(k)$. *J. Am. Math. Soc.* **17**, 947–973 (2004)
3. D. Achlioptas, G. Sorkin, Optimal myopic algorithms for random 3-SAT, in *41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, California (IEEE Computer Society, Los Alamitos, 2000), pp. 590–600

4. A. Agrawal, P. Klein, R. Ravi, When trees collide: an approximation algorithm for the generalized Steiner problem on networks. *SIAM J. Comput.* **24**, 440–456 (1995)
5. R. Aharoni, N. Linial, Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. *J. Comb. Theory A* **43**, 196–204 (1986)
6. S.B. Akers, Binary decision diagrams. *IEEE Trans. Comput.* **C-27**, 509–516 (1978)
7. M. Alekhnovich, J. Johannsen, T. Pitassi, A. Urquhart, An exponential separation between regular and general resolution. *Theory Comput.* **3**, 81–102 (2007). <http://theoryofcomputing.org>
8. R. Amara, A.J. Lipinski, *Business Planning for an Uncertain Future: Scenarios & Strategies* (Elsevier, Amsterdam, 1983)
9. H. Andersen, An introduction to binary decision diagrams. Technical report, Department of Information Technology, Technical University of Denmark, 1998
10. B. Aspvall, Recognizing disguised NR(1) instances of the satisfiability problem. *J. Algorithms* **1**, 97–103 (1980)
11. G. Audemard, L. Simon, Predicting learnt clauses quality in modern SAT solvers, in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, Pasadena, CA, 2009, pp. 399–404
12. C. Barrett, A. Stump, C. Tinelli, The SMT-LIB Standard: Version 1.2. Linked from <http://combination.cs.uiowa.edu/smtlib/>
13. C. Barrett, R. Sebastiani, S.A. Sanjit, C. Tinelli, Satisfiability Modulo Theories, in *Handbook of Satisfiability*, ed. by A. Biere, M. Heule, H. van Maaren, T. Walsh (IOS Press, Amsterdam, 2009), pp. 825–886
14. P. Beame, R.M. Karp, T. Pitassi, M. Saks, On the complexity of unsatisfiability proofs for random k -CNF formulas, in *30th Annual Symposium on the Theory of Computing* (Association for Computing Machinery, New York, 1998), pp. 561–571
15. P. Beame, H. Kautz, A. Sabharwal, Towards understanding and harnessing the potential of clause learning. *J. Artif. Intell. Res.* **22**, 319–351 (2004)
16. P. Beame, H. Kautz, A. Sabharwal, Understanding the power of clause learning, in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007, pp. 1194–1201
17. E. Ben-Sasson, A. Wigderson, Short proofs are narrow – resolution made simple. *J. Assoc. Comput. Mach.* **48**, 149–169 (2001)
18. A. Biere, A. Cimatti, E. Clarke, Y. Zhu, Symbolic model checking without BDDs. *Lect. Notes Comput. Sci.* **1579**, 193–207 (1999)
19. A. Biere, M. Heule, H. van Maaren, T. Walsh (eds.) *Handbook of Satisfiability* (IOS Press, Amsterdam, 2009)
20. A. Blake, Canonical expressions in Boolean algebra. Ph.D. Dissertation, Department of Mathematics, University of Chicago, 1937
21. E. Boros, Y. Crama, P.L. Hammer, Polynomial-time inference of all valid implications for Horn and related formulae. *Ann. Math. Artif. Intell.* **1**, 21–32 (1990)
22. E. Boros, P.L. Hammer, X. Sun, Recognition of q-Horn formulae in linear time. *Discret. Appl. Math.* **55**, 1–13 (1994)
23. E. Boros, Y. Crama, P.L. Hammer, M. Saks, A complexity index for satisfiability problems. *SIAM J. Comput.* **23**, 45–49 (1994)
24. E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, I. Muchnik, An implementation of logical analysis of data. RUTCOR Research Report RRR 04-97, RUTCOR, Rutgers University, 1996
25. E. Boros, P.L. Hammer, T. Ibaraki, A. Kogan, Logical analysis of numerical data. *Math. Program.* **79**, 163–190 (1997)
26. D. Brand, Verification of large synthesized designs, in *Proceeding of the International Conference on Computer Aided Design* (IEEE Computer Society Press, Los Alamitos, 1993), pp. 534–537

27. R.E. Bryant, Graph-based algorithms for Boolean function manipulation. *IEEE Trans. Comput.* **C-35**, 677–691 (1986)
28. R.E. Bryant, Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Comput. Surv.* **24**, 293–318 (1992)
29. B. Buchberger, An algorithm for finding a basis for the residue of a class ring of a zero-dimensional polynomial ideal. Ph.D. Thesis, Universität Innsbruck, Institut für Mathematik, 1965
30. R. Chandrasekaran, Integer programming problems for which a simple rounding type of algorithm works, in *Progress in Combinatorial Optimization*, ed. by W. Pulleyblank (Academic, Toronto, 1984), pp. 101–106
31. V. Chandru, J.N. Hooker, Extended Horn sets in propositional logic. *J. Assoc. Comput. Mach.* **38**, 205–221 (1991)
32. M.-T. Chao, J. Franco, Probabilistic analysis of two heuristics for the 3-Satisfiability problem. *SIAM J. Comput.* **15**, 1106–1118 (1986)
33. M.-T. Chao, J. Franco, Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k -satisfiability problem. *Inf. Sci.* **51**, 289–314 (1990)
34. V. Chvátal, B. Reed, Mick gets some (the odds are on his side), in *33rd Annual Symposium on Foundations of Computer Science*, Pittsburgh, Pennsylvania (IEEE Computer Society, Los Alamitos, 1992), pp. 620–627
35. V. Chvátal, E. Szemerédi, Many hard examples for resolution. *J. Assoc. Comput. Mach.* **35**, 759–768 (1988)
36. E. Clarke, A. Biere, R. Raimi, Y. Zhu, Bounded model checking using satisfiability solving. *Form. Methods Syst. Des.* **19**(1), 7–34 (2001)
37. M. Clegg, J. Edmonds, R. Impagliazzo, Using the Groebner basis algorithm to find proofs of unsatisfiability, in *28th ACM Symposium on the Theory of Computing* (ACM, Philadelphia, 1996), pp. 174–183
38. M. Conforti, G. Cornuéjols, A. Kapoor, K. Vušković, M.R. Rao, Balanced matrices, in *Mathematical Programming: State of the Art*, ed. by J.R. Birge, K.G. Murty, Braunschweig, United States. Produced in association with the 15th International Symposium on Mathematical Programming (University of Michigan, Ann Arbor, 1994)
39. S.A. Cook, R.A. Reckhow, Corrections for “On the Lengths of Proofs in the Propositional Calculus.” *SIGACT News* (ACM Special Interest Group on Automata and Computability Theory) **6** (1974)
40. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edn. (MIT/McGraw-Hill, 2001)
41. O. Coudert, On solving covering problems, in *Proceedings of the 33rd Design Automation Conference* (IEEE Computer Society, Los Alimitos, 1996), pp. 197–202
42. O. Coudert, J.C. Madre, A unified framework for the formal verification of sequential circuits, in *Proceedings of the 1990 International Conference on Computer-Aided Design (ICCAD '90)* (IEEE Computer Society, Los Alamitos, 1990), pp. 126–129
43. O. Coudert, C. Berthet, J.C. Madre, Verification of synchronous sequential machines based on symbolic execution. *Lect. Notes Comput. Sci.* **407**, 365–373 (1990). Springer
44. Y. Crama, P.L. Hammer, T. Ibaraki, Cause-effect relationships and partially defined Boolean functions. *Ann. Oper. Res.* **16**, 299–326 (1998)
45. W. Craig, Linear reasoning: a new form of the Herbrand-Gentzen theorem. *J. Symb. Logic* **22**(3), 250–268 (1957)
46. J.M. Crawford, A.B. Baker, Experimental results on the application of satisfiability algorithms to scheduling problems, in *Proceedings of the National Conference on Artificial Intelligence (AAAI/MIT, Menlo Park, 1994)*, pp. 1092–1097
47. E. Dantsin, A. Wolpert, Derandomization of Schuler’s algorithm for SAT, in *6th International Conference on the Theory and Applications of Satisfiability Testing. Lecture Notes in Computer Science*, vol. 2919 (Springer, New York, 2004), pp. 69–75

48. E. Dantsin, A. Wolpert, An improved upper bound for SAT, in *8th International Conference on the Theory and Applications of Satisfiability Testing*. Lecture Notes in Computer Science, vol. 3569 (Springer, New York, 2005), pp. 400–407
49. E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, U. Schöning, A deterministic $(2 - 2/(k+1))^n$ algorithm for k -SAT based on local search. *Theor. Comput. Sci.* **289**, 69–83 (2002)
50. A. Darwiche, K. Pipatsrisawat, Complete algorithms, in *Handbook of Satisfiability*, ed. by A. Biere, M. Heule, H. van Maaren, T. Walsh (IOS Press, Amsterdam, 2009), pp. 99–130
51. M. Davis, H. Putnam, A computing procedure for quantification theory. *J. ACM* **7**, 201–215 (1960)
52. M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving. *Commun. ACM* **5**, 394–397 (1962)
53. W.F. Dowling, J.H. Gallier, Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *J. Logic Program.* **1**, 267–284 (1984)
54. C. Ducot, G.J. Lubben, A typology for scenarios. *Future* **12**, 51–57 (1980)
55. B. Dutertre, L. de Moura, *A Fast Linear-Arithmetical Solver for DPLL(T)**. Lecture Notes in Computer Science, vol. 4144 (Springer, Berlin/Heidelberg, 2006), pp. 81–94
56. S. Even, A. Itai, A. Shamir, On the complexity of timetable and multi-commodity flow problems. *SIAM J. Comput.* **5**, 691–703 (1976)
57. R. Feldmann, N. Sensen, Efficient algorithms for the consistency analysis in scenario projects. Technical Report, Universität Paderborn, Germany, 1997
58. H. Fleischner, O. Kullmann, S. Szeider, Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable difference. *Theor. Comput. Sci.* **289**(1), 503–516 (2002)
59. J. Franco, A. Van Gelder, A perspective on certain polynomial time solvable classes of satisfiability. *Discret. Appl. Math.* **125**(2–3), 177–214 (2003)
60. J.W. Freeman, Improvements to propositional satisfiability search algorithms. Ph.D. Thesis, University of Pennsylvania, Computer and Information Science, 1995
61. Z. Fu, S. Malik, Solving the minimum-cost Satisfiability problem using SAT based branch-and-bound search, in *Proceedings of the International Conference on Computer Aided Design* (Association for Computing Machinery, New York, 2006), pp. 852–859
62. J. Gausemeier, A. Fink, O. Schlake, *Szenario-Management* (Carl Hanser Verlag, München-Wien, 1995)
63. M. Godet, *Scenarios and Strategic Management* (Butterworths Publishing (Pty) Ltd., South Africa, 1987)
64. A. Goerdt, M. Krivelevich, *Efficient Recognition of Random Unsatisfiable k -SAT Instances by Spectral Methods*. Lecture Notes in Computer Science, vol. 2010 (Springer, Berlin, 2001), pp. 294–304
65. E. Goldberg, Y. Novikov, *How Good Can a Resolution Based SAT-Solver Be?* Lecture Notes in Computer Science, vol. 2919 (Springer, Berlin, 2003), pp. 35–52
66. E. Grégoire, B. Mazure, L. Saïs, Logically-complete local search for propositional non-monotonic knowledge bases, in *Proceedings of the 7th International Workshop on Non-monotonic Reasoning*, Trente, 1998, pp. 37–45
67. J.F. Groote, H. Zantema, Resolution and Binary Decision Diagrams cannot simulate each other polynomially, in *Perspectives of System Informatics: 4th International Andrei Ershov Memorial Conference*, ed. by D. Björner, M. Broy, A.V. Zamulin. Lecture Notes in Computer Science, vol. 2244 (Springer, New York, 2001), pp. 33–38
68. S. Haim, T. Walsh, Restart strategy selection using machine learning techniques, in *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, Swansea, UK, 2009, pp. 312–325
69. M.T. Hajiaghayi, G.B. Sorkin, The satisfiability threshold of random 3-SAT is at least 3.52. Available from <http://arxiv.org/pdf/math.CO/0310193>
70. P. Hall, On representatives of subsets. *J. Lond. Math. Soc.* **10**, 26–30 (1935)
71. P.L. Hammer, Partially defined Boolean functions and cause-effect relationships, in *Proceedings of the International Conference on Multi-Attribute Decision Making Via OR-Based Expert Systems*, University of Passau, Passau, Germany, 1986

72. P. Hansen, B. Jaumard, G. Plateau, An extension of nested satisfiability. *Les Cahiers du GERAD*, G-93-27, 1993
73. H.H. Hoos, On the run-time behaviour of stochastic local search algorithms for SAT, in *Proceedings of the 16th National Conference on Artificial Intelligence* (AAAI/MIT, Menlo Park, 1999), pp. 661–666
74. H.H. Hoos, An adaptive noise mechanism for WalkSAT, in *Proceedings of the National Conference on Artificial Intelligence* (AAAI/MIT, Menlo Park, 2002), pp. 655–660
75. H.H. Hoos, D.A.D. Tompkins, Adaptive Novelty+. Technical Report, Computer Science Department, University of British Columbia, 2007. Available from: <http://www.satcompetition.org/2007/adaptnovelty.pdf>
76. J. Huang, The effect of restarts on the efficiency of clause learning, in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, 2003, pp. 2318–2323
77. W. Hunt, S. Swords, Centaur technology media unit verification, in *Proceedings of the International Conference on Computer Aided Design* (Association for Computing Machinery, Grenoble, 2009), pp. 353–367
78. A. Itai, J. Makowsky, On the complexity of Herbrand’s theorem. Working paper 243, Department of Computer Science, Israel Institute of Technology, 1982
79. K. Iwama, S. Tamaki, Improved upper bounds for 3-SAT, in *15th ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, 2003), pp. 328–328
80. R.E. Jeroslow, J. Wang, Solving propositional satisfiability problems. *Ann. Math. AI* **1**, 167–187 (1990)
81. D.S. Johnson, Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.* **9**, 256–278 (1974)
82. E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, Progress in linear programming-based algorithms for integer programming: an exposition. *INFORMS J. Comput.* **12**(1), 2–23 (2000)
83. A.C. Kaporis, L.M. Kirousis, E.G. Lalas, The probabilistic analysis of a greedy satisfiability algorithm, in *10th Annual European Symposium on Algorithms*, Rome, Italy, 2002
84. A.C. Kaporis, L.M. Kirousis, E.G. Lalas, Selecting complementary pairs of literals. *Electron. Notes Discret. Math.* **16**(1), 1–24 (2004)
85. R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, ed. by R.E. Miller, J.W. Thatcher (Plenum, New York, 1972), pp. 85–103
86. H. Kautz, B. Selman, MAXWalkSat. Available from: <http://www.cs.rochester.edu/~kautz/walksat/>
87. H. Kautz, B. Selman, Y. Jiang, *Stochastic Search for Solving Weighted MAX-SAT Encodings of Constraint Satisfaction Problems* (AT&T Laboratories, Murray Hill, 1998, Preprint)
88. H. Kautz, E. Horvitz, Y. Ruan, C. Gomes, B. Selman, Dynamic restart policies, in *Proceedings of the National Conference on Artificial Intelligence* (AAAI/MIT, Menlo Park, 2002), pp. 674–681
89. H. Kautz, A. Sabharwal, B. Selman, Incomplete algorithms, in *Handbook of Satisfiability*, ed. by A. Biere, M. Heule, H. van Maaren, T. Walsh (IOS Press, Amsterdam, 2009), pp. 185–204
90. F. Klaedtke, Decision Procedures for Logical Theories, Lecture 12: Combining Decision Procedures: Nelson-Oppen, Department of Computer Science, ETH Zurich, 2006
91. D. Knuth, Nested satisfiability. *Acta Inform.* **28**, 1–6 (1990)
92. F. Krohm, A. Kuehlmann, Equivalence checking using cuts and heaps, in *Proceedings of the 34th Design Automation Conference* (IEEE Computer Society, Los Alamitos, 1997), pp. 263–268
93. F. Krohm, A. Kuehlmann, A. Mets, The use of random simulation in formal verification, in *Proceedings of the IEEE International Conference on Computer Design* (IEEE Computer Society, Los Alamitos, 1997), pp. 371–376
94. A. Kuehlmann, Dynamic transition relation simplification for bounded property checking, in *Proceeding of the IEEE International Conference on Computer Aided Design* (IEEE Computer Society, Los Alamitos, 2004), pp. 50–57

95. A. Kuehlmann, V. Paruthi, F. Krohm, M.K. Ganai, Robust Boolean reasoning for equivalence checking and functional property verification. *IEEE Trans. Comput. Aided Des.* **21**(12), 1377–1394 (2002)
96. O. Kullmann, A first summary on minimal unsatisfiable clause-sets. Technical report, University of Toronto, June, 1998
97. O. Kullmann, Investigations on autark assignments. *Discret. Appl. Math.* **107**, 99–137 (2000)
98. O. Kullmann, An application of Matroid theory to the SAT problem, in *Proceedings of the 15th Annual IEEE Conference on Computational Complexity* (IEEE, 2000), pp. 116–124
99. H. Kleine Büning, On the minimal unsatisfiability problem for some subclasses of CNF, in *Abstracts of 16th Int'l Symposium on Mathematical Programming*, Lausanne, 1997
100. M. Kouril, J. Franco, Resolution tunnels for improved SAT solver performance, in *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing*, St. Andrews, UK, 2005, pp. 136–141
101. M. Kouril, J. Paul, The van der Waerden Number W(2, 6) Is 1132. *Exp. Math.* **17**(1), 53–61 (2008)
102. R.A. Kowalski, A proof procedure using connection graphs. *J. ACM* **22**, 572–595 (1974)
103. C.Y. Lee, Representation of switching circuits by binary decision programs. *Bell Syst. Tech. J.* **38**, 985–999 (1959)
104. H.R. Lewis, Renaming a set of clauses as a Horn set. *J. Assoc. Comput. Mach.* **25**, 134–135 (1978)
105. D. Lichtenstein, Planar formulae and their uses. *SIAM J. Comput.* **11**, 329–343 (1982)
106. I. Lynce, J.P.M. Silva, Efficient data structures for backtrack search SAT solvers. *Ann. Math. Artif. Intell.* **43**(1), 137–152 (2005)
107. I. Lynce, J.P.M. Silva, SAT in Bioinformatics: making the Case with Haplotype Inference, in *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*, Seattle, WA, 2006, pp. 136–141
108. V.M. Manquinho, J.P.M. Silva, Search pruning techniques in SAT-based branch-and-bound algorithms for the binate covering problem. *IEEE Trans. CAD Integr. Circuits Syst.* **21**(5), 505–516 (2002)
109. D. McAllester, B. Selman, H. Kautz, Evidence for invariants in local search, in *Proceedings of the 14th National Conference on Artificial Intelligence* (AAAI/MIT, Menlo Park, 1997), pp. 321–326
110. I. Mironov, L. Zhang, Applications of SAT solvers to cryptanalysis of hash functions, in *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*, Seattle, WA, 2006, pp. 102–115
111. B. Monien, E. Speckenmeyer, Solving satisfiability in less than 2^n steps. *Discret. Appl. Math.* **10**, 287–295 (1985)
112. M.W. Moskewicz, C.F. Madigan, Y. Zhao, L. Zhang, S. Malik, Chaff: engineering an efficient SAT solver, in *Proceedings of the 38th Design Automation Conference*, Las Vegas, NV, USA, 2001, pp. 530–535
113. G. Nelson, D.C. Oppen, Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.* **1**(2), 245–257 (1979)
114. D.C. Oppen, Complexity, convexity and combinations of theories. *Theor. Comput. Sci.* **12**, 291–302, 1980
115. R. Paturi, P. Pudlák, M.E. Saks, F. Zane, An improved exponential-time algorithm for k-SAT, in *39th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, 1998)
116. W.V.O. Quine, A way to simplify truth functions. *Am. Math. Mon.* **62**, 627–631 (1955)
117. R. Reiter, A theory of diagnosis from first principles. *Artif. Intell.* **32**, 57–95 (1987)
118. J.A. Robinson, A machine-oriented logic based on the resolution principle. *J. ACM* **12**, 23–41 (1965)
119. L. Ryan, Efficient algorithms for clause-learning SAT solvers. Masters Thesis, Simon Fraser University, 2004

120. E.W. Samson, B.E. Mills, Circuit minimization: algebra and algorithms for new Boolean canonical expressions. Air Force Cambridge Research Center Technical Report 54–21, 1954
121. T.S. Schaefer, The complexity of satisfiability problems, in *Proceedings of 10th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1978), pp. 216–226
122. J.S. Schlipf, F. Annexstein, J. Franco, R. Swaminathan, On finding solutions for extended Horn formulas. Inf. Process. Lett. **54**, 133–137 (1995)
123. U. Schöning, A probabilistic algorithm for k -SAT and constraint satisfaction problems, in *40th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, 1999), pp. 410–414
124. R. Schuler, An algorithm for the satisfiability problem of formulas in conjunctive normal form. J. Algorithms **54**(1), 40–44 (2005)
125. M.G. Scutella, A note on Dowling and Gallier’s top-down algorithm for propositional Horn satisfiability. J. Logic Program. **8**, 265–273 (1990)
126. R. Sebastiani, Lazy Satisfiability Modulo Theories. J. Satisfiability Boolean Model. Comput. **3**(3–4), 141–224 (2007)
127. B. Selman, H. Kautz, B. Cohen, Local search strategies for satisfiability testing, in *Cliques, Coloring, and Satisfiability: The Second DIMACS Implementation Challenge*, ed. by D.S. Johnson, M.A. Trick (American Mathematical Society, Providence, 1996), pp. 521–532
128. Y. Shang, B.W. Wah, A discrete Lagrangian-based global search method for solving satisfiability problems. J. Glob. Optim. **12**(1), 61–100 (1998)
129. C.E. Shannon, A symbolic analysis of relay and switching circuits. Masters Thesis, Massachusetts Institute of Technology, 1940. Available from <http://hdl.handle.net/1721.1/11173>
130. J.P.M. Silva, K.A. Sakallah, GRASP – a new search algorithm for satisfiability, in *Proceedings of the 1996 International Conference on Computer-Aided Design* (ICCAD ’96) (IEEE Computer Society, Los Alamitos, 1996), pp. 220–227
131. J.P.M. Silva, I. Lynce, S. Malik, Conflict-driven clause learning SAT solvers, in *Handbook of Satisfiability*, ed. by A. Biere, M. Heule, H. van Maaren, T. Walsh (IOS Press, Amsterdam, 2009), pp. 131–153
132. E.W. Smith, D.L. Dill, Automatic formal verification of block Cipher implementations, in *Proceedings of the Formal Methods in Computer-Aided Design Conference*, Portland, Oregon, 2008, pp. 1–7
133. R.P. Swaminathan, D.K. Wagner, The arborescence–realization problem. Discret. Appl. Math. Comb. Oper. Res. Comput. Sci. **59**(3), 267–283 (1995)
134. S. Szeider, Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable. J. Comput. Syst. Sci. **69**, 656–674 (2004)
135. S. Szeider, Generalizations of matched CNF formulas. Ann. Math. Artif. Intell. **43**(1), 223–238 (2005)
136. C.A. Tovey, A simplified NP-complete satisfiability problem. Discret. Appl. Math. **8**, 85–89 (1984)
137. K. Truemper, Monotone decomposition of matrices. Technical Report UTDCS-1-94, University of Texas at Dallas, 1994
138. K. Truemper, *Effective Logic Computation* (Wiley, New York, 1998)
139. G.S. Tseitin, On the complexity of derivations in propositional calculus, in *Structures in Constructive Mathematics and Mathematical Logic, Part II*, ed. by A.O. Slisenko (Nauka, Moscow, 1968), pp. 115–125 (translated from Russian)
140. L.G. Valiant, A theory of the learnable. Commun. ACM **27**, 1134–1142 (1984)
141. A. Van Gelder, Generalizations of watched literals for backtracking search, in *Seventh International Symposium on Artificial Intelligence and Mathematics*, 2002
142. A. Van Gelder, Verifying propositional unsatisfiability: Pitfalls to avoid, in *Proceedings of the 10th International Conference on Theory and Applications of Satisfiability Testing*, Lisbon, Portugal, 2007, pp. 328–333
143. A. Van Gelder, Improved conflict-clause minimization leads to improved propositional proof traces, in *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, Swansea, UK, 2009, pp. 141–146

144. A. Van Gelder, Y.K. Tsuji, Satisfiability testing with more reasoning and less guessing, in *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, ed. by D.S. Johnson, M. Trick (American Mathematical Society, Providence, 1996)
145. H. van Maaren, A short note on some tractable classes of the Satisfiability problem. Inf. Comput. **158**(2), 125–130 (2000)
146. S. Weaver, J. Franco, J. Schlipf, Extending existential quantification in conjunctions of BDDs. J. Satisfiability Boolean Model. Comput. **1**, 89–110 (2006)
147. Z. Wu, B.W. Wah, Trap escaping strategies in discrete Lagrangian methods for solving hard satisfiability and maximum satisfiability problems, in *Proceedings of the 16th National Conference on Artificial Intelligence* (AAAI/MIT, Menlo Park, 1999), pp. 673–678
148. Z. Wu, B.W. Wah, An efficient global-search strategy in discrete Lagrangian methods for solving hard satisfiability problems, in *Proceedings of the 17th National Conference on Artificial Intelligence* (AAAI/MIT, Menlo Park, 2000), pp. 310–315
149. M. Yoeli, *Formal Verification of Hardware Design* (IEEE Computer Society, Los Alamitos, 1988)
150. L. Zhang, S. Malik, Validating SAT solvers using an independent resolution-based checker: practical implementations and other applications, in *Proceedings of the Conference on Design, Automation and Test in Europe*, Washington, DC, USA, 2003, pp. 880–885
151. H. Zhang, D. Li, H. Shen, A SAT based scheduler for tournament schedules, in *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing*, Vancouver, BC, 2004

Bin Packing Approximation Algorithms: Survey and Classification

Edward G. Coffman, János Csirik, Gábor Galambos, Silvano Martello
and Daniele Vigo

Contents

1	Introduction	456
2	Definitions and Classification	458
2.1	Basic Definitions for Classical Bin Packing	459
2.2	General Definitions	461
2.3	Classification Scheme	463
3	On-Line Algorithms	466
3.1	Classical Algorithms	467
3.2	Weighting Functions	468
3.3	Any-Fit and Almost Any-Fit Algorithms	470
3.4	Bounded-Space Algorithms	470
3.5	Variations and the Best-in-Class	473
3.6	APR Lower Bounds	476
3.7	Semi-on-line Algorithms	478
4	Off-line Algorithms	480
4.1	Algorithms with Presorting	480
4.2	Linear Time, Randomization, and Other Approaches	483

E.G. Coffman

Department of Computer Science, Columbia University, New York, NY, USA

e-mail: coffman@cs.columbia.edu; egc@ee.columbia.edu

J. Csirik

Department of Computer Algorithms and Artificial Intelligence, University of Szeged, Szeged,
Hungary

e-mail: csirik@inf.u-szeged.hu

G. Galambos

Faculty of Education, Department of Computer Science, University of Szeged, Szeged, Hungary
e-mail: galambos@jgypk.u-szeged.hu

S. Martello (✉) • D. Vigo

DEI “Guglielmo Marconi”, University of Bologna, Bologna, Italy

e-mail: silvano.martello@unibo.it; daniele.vigo@unibo.it

4.3	Asymptotic Approximation Schemes.....	484
4.4	Anomalies.....	487
5	Variations on Bin Sizes.....	488
5.1	Variable-Sized Bins.....	488
5.2	Resource Augmentation.....	491
6	Dual Versions.....	495
6.1	Minimizing the Bin Capacity.....	495
6.2	Maximizing the Number of Items Packed.....	497
6.3	Maximizing the Number of Full Bins.....	499
7	Variations on Item Packing.....	500
7.1	Dynamic Bin Packing.....	500
7.2	Selfish Bin Packing.....	502
7.3	Bin Packing with Rejection.....	504
7.4	Item Fragmentation.....	505
7.5	Fragile Objects.....	507
7.6	Packing Sets and Graphs.....	507
8	Additional Conditions.....	508
8.1	Item-Size Restrictions.....	508
8.2	Cardinality Constrained Problems.....	510
8.3	Class Constrained Bin Packing.....	513
8.4	LIB Constraints.....	514
8.5	Bin Packing with Conflicts.....	515
8.6	Partial Orders.....	517
8.7	Clustered Items.....	518
8.8	Item Types.....	518
9	Examples of Classification.....	518
	Cross-References.....	520
	Recommended Reading.....	521

Abstract

The survey presents an overview of approximation algorithms for the classical bin packing problem and reviews the more important results on performance guarantees. Both on-line and off-line algorithms are analyzed. The investigation is extended to variants of the problem through an extensive review of dual versions, variations on bin sizes and item packing, as well as those produced by additional constraints. The bin packing papers are classified according to a novel scheme that allows one to create a compact synthesis of the topic, the main results, and the corresponding algorithms.

1 Introduction

In the classical version of the bin packing problem, one is given an infinite supply of bins with capacity C and a list L of n items with sizes no larger than C : the problem is to pack the items into a minimum number of bins so that the sum of the sizes in each bin is no greater than C . In simpler terms, a set of numbers is to be partitioned into a minimum number of blocks subject to a sum constraint common to each block. Bin packing rather than partitioning terminology will be used, as it eases considerably the problem of describing and analyzing algorithms.

The mathematical foundations of bin packing were first studied at Bell Laboratories by M.R. Garey and R.L. Graham in the early 1970s. They were soon joined by J.D. Ullman; then at Princeton, these three published the first [103] of many papers to appear in the conferences of the computer science theory community over the next 40 years. The second such paper appeared within months; in that paper, D.S. Johnson [125] extended the results in [103] and studied general classes of approximation algorithms. In collaboration with A. Demers at Princeton, researchers Johnson, Garey, Graham, and Ullman published the first definitive analysis of bin packing approximation algorithms [129]. In parallel with the research producing this landmark paper, Johnson completed his 1973 Ph.D. thesis at MIT which gave a more comprehensive treatment and more detailed versions of the abbreviated proofs in [129].

The pioneering work in [129] opened an extremely rich research area; it soon turned out that this simple model could be used for a wide variety of different practical problems, ranging from a large number of cutting stock applications to packing trucks with a given weight limit, assigning commercials to station breaks in television programming, or allocating memory in computers. The problem is well-known to be \mathcal{NP} -hard (see, e.g., Garey and Johnson [100]); hence, it is unlikely that efficient (i.e., polynomial-time) optimization algorithms can be found for its solution. Researchers have thus turned to the study of approximation algorithms, which do not guarantee an optimal solution for every instance, but attempt to find a near-optimal solution within polynomial time. Together with closely related partitioning problems, bin packing has played an important role in applications of complexity theory and in both the combinatorial and average-case analysis of approximation algorithms (see, e.g., the volume edited by Hochbaum [115]).

Starting with the seminal papers mentioned above, most of the early research focused on combinatorial analysis of algorithms leading to bounds on worst-case behavior, also known as performance guarantees. In particular, letting $A(L)$ be the number of bins used by an algorithm A and letting $OPT(L)$ be the minimum number of bins needed to pack the items of L , one tries to find a least upper bound on $A(L)/OPT(L)$ over all lists L (for a more formal definition, see Sect. 2). Much of the research can be divided along the boundary between *on-line* and *off-line* algorithms. In the case of on-line algorithms, items are packed in the order they are encountered in a scan of L ; the bin in which an item is packed is chosen without knowledge of items not yet encountered in L . These algorithms are the only ones that can be used in certain situations. For example, the items to be packed may arrive in a sequence according to some physical process and have to be assigned to a bin as soon as they arrive. Off-line algorithms have complete information about the entire list throughout the packing process.

Since the early 1980s, progressively more attention has been devoted to the probabilistic analysis of packing algorithms. A book by Coffman and Lueker [46] covers the methodology in some detail (see also the book by Hofri [119, Chap. 10]). Nevertheless, combinatorial analysis remains a central research area, and from time to time, numerous new results need to be collected into survey papers. The first comprehensive surveys of bin packing algorithms were by Garey and Johnson [101]

in 1981 and by Coffman et al. [50] in 1984. The next such survey was written some 10 years later by Galambos and Woeginger [96] who gave an overview restricted to on-line algorithms. New surveys appeared at the end of the 1990s. Csirik and Woeginger [62] concentrated on on-line algorithms, while Coffman et al. [52] extended the coverage to include off-line algorithms, and Coffman et al. [53] considered both classes of algorithms in an earlier version of the present survey. Worst-case and average-case analysis are surveyed in [62] and [52]. More recently, classical bin packing and its variants were covered in Coffman and Csirik [43] and in Coffman et al. [54, 55].

This survey gives a broad summary of results in the one-dimensional bin packing arena, concentrating on combinatorial analysis, but in contrast to other surveys, greater space is devoted to variants of the classical problem. Important new variants continue to arise in many different settings and help account for the thriving interest in bin packing research. The survey does not attempt to give a self-contained work, for example, it does not present all algorithms in detail nor does it give formal proofs, but it refers to certain proof techniques which have been used frequently to establish the most important results. Also, in the coverage of variants, the restriction to partitioning problems in one dimension is maintained. Packing in higher dimensions, for example, strip packing and two-dimensional bin packing, is itself a big subject, one deserving its own survey. The interested reader is referred to the recent survey by Epstein and van Stee [79].

Another important feature of this survey is that it adopts the novel classification scheme for bin packing papers recently introduced by Coffman and Csirik [44]. This scheme allows one to create a compact synthesis of the topic, the main results, and the corresponding algorithms.

Section 2 covers the main definitions and as well as a full description of the classification scheme. For the classical bin packing problem, on-line (and semi-on-line) algorithms are discussed in Sect. 3 and off-line algorithms in Sect. 4. (Section 4.4 is a slight departure from this organization: it deals with anomalous behavior both for on-line and off-line algorithms.)

The second part of this survey concentrates on special cases and variants of the classical problem. Variations on bin size are considered in Sect. 5, dual versions in Sect. 6, variations on item packing in Sect. 7, and additional conditions in Sect. 8. Finally, Sect. 9 gives a series of examples to familiarize the reader with the classification scheme. The classification of the papers considered in the survey is also given, when appropriate, in the Bibliography.

2 Definitions and Classification

This section introduces more formally all of the relevant notation required to define, analyze, and classify bin packing problems. Starting with the classical problem, the notation is then extended to more general versions, and finally, the new classification scheme is described. Additional definitions needed by specific variants are introduced in the appropriate sections.

2.1 Basic Definitions for Classical Bin Packing

The classical bin packing problem is defined by an infinite supply of bins with capacity C and a list $L = (a_1, \dots, a_n)$ of items (or elements). A value $s_i \equiv s(a_i)$ gives the size of item a_i and satisfies $0 < s_i \leq C$, $1 \leq i \leq n$. The problem is to pack the items into a minimum number of bins under the constraint that the sum of the sizes of the items in each bin is no greater than C .

In the classical problem, the bin capacity C is just a scale factor, so without loss of generality, one can adopt the normalization $C = 1$. Unless stated otherwise, this convention is in force throughout. One of the exceptions will be in the sections treating variants of the classical problem in which bins B_j have varying sizes. In those sections, bin sizes will be denoted by $s(B_j)$.

In the algorithmic context, nonempty bins will be classified as either *open* or *closed*. Open bins are available for packing additional items. Closed bins are unavailable and can never be reopened. It is convenient to regard a completely full bin as open under any given algorithm until it is specifically closed, even though such bins can receive no further items. Denote the bins by B_1, B_2, \dots . When no confusion arises, B_j will also denote the set of items packed in the j -th bin, and $|B_j|$ will denote the number of such items.

The items are always organized into a list L (or a list L_j in some indexed set of lists). The notation for list *concatenation* is as usual; $L = L_1 L_2 \dots L_k$ means that the items of list L_i are followed by the items of list L_{i+1} for each $i = 1, 2, \dots, k - 1$. When the number of items in a list is needed as part of the notation, an index in parentheses is used: $L_{(n)}$ denotes a list of n items.

If B_j is nonempty, its current *content* or *level* is defined as

$$c(B_j) = \sum_{a_i \in B_j} s_i.$$

A bin (necessarily empty) is *opened* when it receives its first item. Of course, it may be closed immediately thereafter, depending on the algorithm and the item size. It is assumed, without loss of generality, that all algorithms open bins in order of increasing index. Within any collection of nonempty bins, the *earliest opened*, the *leftmost*, and the *lowest indexed* all refer to the same bin.

In general, in the considered lists, the item sizes are taken from the interval $(0, \alpha]$, with $\alpha \in (0, 1]$. It is usually assumed that $\alpha = \frac{1}{r}$, for some integer $r \geq 1$, and many results apply only to $r = \alpha = 1$.

There are two principal measures of the worst-case behavior of an algorithm. Recall that $A(L)$ denotes the number of bins used by algorithm A to pack the elements of L ; OPT denotes an optimal algorithm, one that uses a minimum number of bins. Define the set V_α of all lists L for which the maximum size of the items is bounded from above by α . For every $k \geq 1$, let

$$R_A(k, \alpha) = \sup_{L \in V_\alpha} \left\{ \frac{A(L)}{k} : OPT(L) = k \right\}.$$

Then the *asymptotic worst-case ratio* (or *asymptotic performance ratio*, APR) as a function of α is given by

$$R_A^\infty(\alpha) = \overline{\lim}_{k \rightarrow \infty} R_A(k, \alpha).$$

Clearly, $R_A^\infty(\alpha) \geq 1$, and this number measures the quality of the packings produced by algorithm A compared to optimal packings in the worst case. In an equivalent definition, $R_A^\infty(\alpha)$ is a smallest number such that there exists a constant $K \geq 0$ for which

$$A(L) \leq R_A^\infty(\alpha) OPT(L) + K.$$

for every list $L \in V_\alpha$. Hereafter, if α is left unspecified, the APR of algorithm A refers to $R_A^\infty \equiv R_A^\infty(1)$.

The second way to measure the worst-case behavior of an algorithm A is the *absolute worst-case ratio* (AR)

$$R_A(\alpha) = \sup_{L \in V_\alpha} \left\{ \frac{A(L)}{OPT(L)} \right\}.$$

The comparison of algorithms by asymptotic bounds can be strikingly different from that by absolute bounds. Generally speaking, the number of items n must be sufficiently large (how large will depend on the algorithm) for the asymptotic bounds to be the better measure for purposes of comparison. Note that the ratios are bounded below by 1; the better algorithms have the smaller ratios. When algorithm A is an on-line algorithm, the asymptotic ratio is also called the *competitive ratio*.

There are some further proposals to measure the quality of a packing, like *differential approximation measure* (see Demange et al. [66], [67]), *random-order ratio* (see Kenyon [136]), *relative worst-order ratio* (see Boyar and Favrholdt [27]), and *accommodation function* (see Boyar et al. [28]).

It is finally worth mentioning a special sequence t_i of integers which was investigated by Sylvester [170] in connection with a number theoretic problem and later generalized by Golomb [107]. (In describing the performance of various algorithms in later sections, such sequence will occasionally be considered.) For an integer $r \geq 1$, define

$$t_1(r) = r + 1$$

$$t_2(r) = r + 2$$

$$t_{i+1}(r) = t_i(r)(t_i(r) - 1) + 1, \quad \text{for } i \geq 2.$$

(The Sylvester sequence was defined for $r = 1$). It was conjectured by Golomb that for $r = 1$, this sequence gives the closest approximation to 1 from below among

the approximations by sums of reciprocals of k integers, the basis of the conjecture being

$$\sum_{i=1}^k \frac{1}{t_i(1)} + \frac{1}{t_{k+1}(1) - 1} = 1.$$

Furthermore, it is also easily proved that the following expression is valid for the Golomb sequences:

$$\frac{(r-1)}{t_1(r)} + \sum_{i=2}^k \frac{1}{t_i(r)} + \frac{1}{t_{k+1}(r) - 1} = 1 \quad \text{for any } r \geq 1.$$

On the other hand, the following value appears in several results:

$$h_\infty(r) = 1 + \sum_{i=2}^{\infty} \frac{1}{t_i(r) - 1}.$$

The first few values of $h_\infty(r)$ are the following: $h_\infty(1) \approx 1.69103$, $h_\infty(2) \approx 1.42312$, $h_\infty(3) \approx 1.30238$.

2.2 General Definitions

This survey covers results on several variants of the one-dimensional bin packing problem. In such variants, the meaning of a number of notations can be different, so more general definitions are necessary. These will be introduced in the present section, together with the basic ideas behind the adopted classification scheme.

The notion of packing items into a sequence of initially empty bins helps visualize algorithms for constructing partitions. It is also helpful in classifying algorithms according to the various constraints under which they must operate in practice. The items are normally given in the form of a sequence or *list* $L = (a_1, \dots, a_n)$, although the ordering in many cases will not have any significance. To economize on notation, a harmless abuse is adopted whereby s_i denotes the name as well as the size of the i -th item. The generic symbol for packing is \mathbf{P} , the number of items in \mathbf{P} is denoted by $|\mathbf{P}|$, the sum of the sizes of the items in \mathbf{P} is denoted by $c(\mathbf{P})$, and the number of bins in \mathbf{P} is denoted by $\#\mathbf{P}$. In the classical bin packing optimization problem, the objective is to find a packing \mathbf{P} of all the items in L such that $\#\mathbf{P}$ is minimized over all partitions of L satisfying the sum constraints. Recall that C is only a scale factor in this problem; it is usually convenient to replace the s_i by s_i/C and take $C = 1$.

Let $\mathbf{P}_A(L)$ denote the packing of L produced by algorithm A . In the literature, one finds the notation $A(L)$ representing metrics such as $\#\mathbf{P}$, but since $A(L)$ may denote different metrics for different problems (the same algorithm A may apply to problems with different objective functions), the alternative notation will be

necessary on occasion. The minimum of $\#\mathbf{P}$ over all partitions \mathbf{P} of L satisfying the sum constraints will have the notation $OPT(L) := \min_{\pi(L)} \#\mathbf{P}(L)$, where $\pi(L)$ denotes the set of all such partitions. The notation $OPT(L)$ suffers from the same ambiguity as before, that is, the objective function to which it applies is determined by context. Moreover, in contrast to other algorithm notation, OPT does not denote a unique algorithm.

There are two variants of classical bin packing that arise immediately from the definition. In one, a general bin capacity C and a number m of bins are part of the problem instance, and the problem is to find a maximum cardinality subset of $\{s_i\}$ that can be packed into m bins of capacity C . In the other, all n items must be packed into m bins, and the problem is to find a smallest bin capacity C such that there exists a packing of all the items in m bins of capacity C . The first problem suggests yet another in which the total size of the items in a subset S is the quantity of interest rather than the number $|S|$ of items. Problems fixing the number of bins fall within scheduling theory whose origins in fact predate those of bin packing theory. In scheduling theory, which is very large in its own right, makespan scheduling is more likely to be described as scheduling a list of tasks or jobs (items) on m identical processors (bins) so as to minimize the schedule length or makespan (bin capacity). This incursion into scheduling problems will be limited to the most elementary variants and applications of bin packing problems, such as those above.

Bin covering problems are also included in bin packing theory for classification purposes, and these change the sum constraints to $c(B_j) \geq 1$, where again, bin capacity is normalized to 1. The classical covering problem asks for a partition, \mathbf{C} , called a *cover*, which *maximizes* the number $\#\mathbf{C}$ of bins satisfying the new constraint. Both the packing and covering combinatorial optimization problems are \mathcal{NP} -hard. With problems defined on restricted item sizes or number of items per bin being the major exceptions, this will be the case for nearly all problem variants in the classification scheme. Note that there are immediate variants to bin covering, just as there were for bin packing. For example, one can take the number m of bins to be fixed and ask for a *minimum* cardinality subset of $\{a_i\}$ from which a cover of m bins can be obtained. Or one can consider the problem of finding a *largest* capacity C such that L can be made to cover m bins of capacity C .

Order-of-magnitude estimates of time complexities of fundamental algorithms and their extensions are usually easy to derive. The analysis of parallel algorithms for computing packings is an example where deriving time complexities is not always so easy. However, the research in this area, in which results take the form of complexity measures, has been very limited.

Several results quantify the trade-off between the running time of algorithms and the quality of the packings. They produce polynomial-time (or fully polynomial-time) approximation schemes [100], denoted by PTAS (or FPTAS). In simplified terms, a typical form of such results is illustrated by “Algorithm A produces packings with $O(\varepsilon)$ wasted space and has a running time that is polynomial in $1/\varepsilon$.”

The most common approach to the analysis of approximation algorithms has been *worst-case* analysis by which the worst possible performance of an algorithm is compared with the performance of an optimization algorithm. (Detailed definitions will be provided shortly.) The term *performance guarantee* puts a more positive slant on results of this type. So also does the term *competitive analysis*, which usually refers to a worst-case analysis comparing an on-line approximation algorithm with an optimal off-line algorithm. Probability models also enjoy wide use and have grown in popularity, as they bring out typical, *average-case* behavior rather than the normally rare worst-case behavior. In probabilistic analysis, algorithms have random inputs; the items are usually assumed to be independent, identically distributed random variables. For a given algorithm A , $A(L_n)$ is a random variable whose distribution becomes the goal of the analysis. Because of the major differences in probabilistic results and in the analysis that produces them, they are not covered in this survey, which focuses exclusively on combinatorial analysis. For a general treatment of average-case analysis of packing and related partitioning problems, see the text by Coffman and Lueker [46].

2.3 Classification Scheme

The scheme for classifying problems and solutions is aimed at giving the reader a good idea of the results in bin packing theory to be found in any given paper on the subject. Although in many, if not most cases, it is impractical to describe every result contained in a paper, an indication of the main results should be useful to the reader.

The classification uses four fields, presented in the form

problem | algorithm class | results | parameters

For a brief preview, observe that, normally, the *problem* is to minimize or maximize a metric under sum constraints and refers, for example, to the number of bins of fixed capacity and the capacity of a fixed number of bins. The *algorithm class* refers to paradigms such as on-line, off-line, or bounded space. The *results* field specifies performance in terms of absolute or asymptotic worst-case ratios, problem complexity, etc. Lastly, the *parameters* field describes restrictions on problem parameters, such as a limit on item sizes or on the number of items per bin, and a restriction of all data to finite or countable sets. In many cases, there are no additional parameters to specify, in which case the *parameters* field will be omitted. The three remaining fields will normally be nonempty.

In the following, a detailed description of the classification scheme is introduced. Section 9 gives a number of annotated examples to familiarize the reader with the proposed classification. The special terms or abbreviations adopted for entries will be given in bold face. In the large majority of cases, the entry will have a mnemonic quality that makes the precise meaning of an entry clear in spite of its compact form.

2.3.1 Problem

The combinatorial optimization problems of interest can easily be expressed in the notation given so far, but for readability, abbreviations of the names by which the problems are commonly known will be used. Most, but not all, of the problems below were discussed earlier. Moreover, some of them are not treated in this survey and are listed for the sake of completeness.

1. **pack** refers to the problem of minimizing $\#\mathbf{P}(L)$. The default sum constraints are $c(B_j) \leq 1$. But they are $c(B_j) \leq s(B_j)$, if variable bin capacities $s(B_j)$ are considered, and more simply $c(B_j) \leq C$ if there is only a common bin capacity C .
2. **maxpack** problems invert the problem above, that is, $\#\mathbf{P}$ is to be maximized. Clearly, such problems trivialize unless there is some constraint placed on opening new bins. The tacit assumption will be that packings under approximation algorithms must obey a conservative, any-fit constraint: a new bin cannot be opened for an item s_i unless s_i cannot fit into any currently open bin. Optimal packings by definition must be such that, for some permutation of the bins, no item in B_i fits into B_j for any $j < i$.
3. **mincap** refers to the problem of minimizing the common bin capacity needed to pack L into a given number m of bins. *Bin-stretching* analysis applies if the problem instances are restricted to those for which an optimization rule can pack L into m unit-capacity bins. Under this assumption, in the analysis of algorithm A , one asks how large must the bin capacity be (how much must it be stretched) for algorithm A to pack L into the same number of bins.
4. **maxcard(subset)** has the number m of bins and their common capacity C as part of the problem instance. The problem is to find a largest cardinality subset of L that can be packed into m bins of capacity C .
5. **maxsize(subset)** has the number m of bins and their common capacity C as part of the problem instance. The problem is to find a subset of L with maximum total item size that can be packed into m bins of capacity C .
6. **cover** refers to the problem of finding a partition of L into bins that maximizes $\#\mathbf{P}(L)$ subject to the constraints $c(B_j) \geq 1$. The partition is called a *cover*.
7. **capcover** refers to the problem of finding, for a given number m of bins, the *maximum* capacity C such that a covering of m bins of capacity C can be obtained from L .
8. **cardcover(subset)** is the related problem of minimizing the cardinality of the subset of L needed to cover a given number m of bins with a given capacity C .

2.3.2 Algorithm Class

1. **on-line** algorithms sequentially assign items to bins, in the order encountered in L , without knowledge of items not yet packed. Thus, the bin to which a_i is assigned is a function only of the sizes s_1, \dots, s_i .
2. **off-line** algorithms have no constraints beyond the intrinsic sum constraints; an off-line algorithm simply maps the entire list L into a packing $\mathbf{P}(L)$. All items are known in advance, so the ordering of L plays no role.

3. **bounded-space** algorithms decide where an item is to be packed based only on the current contents of at most a finite number k of bins, where k is a parameter of the algorithm. A more precise definition and further discussion of these algorithms appear later.
4. **linear-time** algorithms have $O(n)$ running time. More precisely, all such algorithms take constant time to pack each item.

The three characterizations above are orthogonal. But the literature suggests that the following convention allows to use one term in classifying algorithms most of the time: *bounded space implies linear time and linear time implies online*. Exceptions will be noted explicitly: it will be seen below (under **repack**) how off-line algorithms can be linear time.

5. **open-end** refers to certain cover approximation algorithms. An open bin B_j , that is, one for which $c(B_j) < s(B_j)$ and further packing is allowed, must be closed just as soon as $c(B_j) \geq s(B_j)$. (The item causing the “overflow” is left in the bin.)
6. **conservative** algorithms are those required to pack the current item into an open bin with sufficient space, whenever such a bin exists; in particular, it cannot choose to open a new bin. Scheduling algorithms satisfying a similar constraint are sometimes called *work conserving*.
7. **repack** refers to packing problems which allow the repacking (possibly limited in some way) of items, that is, moving an item, say s_i , from one bin to another.
8. **dynamic** packing introduces the time dimension; an instance L of this problem consists of a sequence of triples (s_i, b_i, d_i) with b_i and d_i denoting arrival and departure times, respectively. Under a packing algorithm A , $A(L, t)$ denotes the number of bins occupied at time t , that is, the number of bins occupied by those items a_i for which $b_i \leq t < d_i$.

2.3.3 Results

Almost all results fall into the broad classes mentioned in Sect. 2.1.

1. R_A^∞ is the asymptotic worst-case ratio, with algorithm A specified when appropriate. When only a bound is proved, the word **bound** is appended.
2. R_A is the absolute worst-case ratio. When only a bound is proved, the word **bound** is appended.
3. Where possible, complexity of the problem will be given in the standard notation of problem complexity. Approximation schemes are classified as complexity results and have entries like **PTAS** and **FPTAS** as noted earlier.
4. Complexity of the algorithm may also be a result; it refers to running-time complexity and will be signaled by the entry **running time**.

A paper classified as a worst-case analysis may also have complexity results (but not conversely, unless both types of results figure prominently in the paper, in which case both classifications will be given).

2.3.4 Parameters

These typically correspond to generalizations whereby limitations can be placed on the problem instance, or further properties of the algorithm classification. In some

cases, problems may be simplified by certain limitations, such as a specific upper limit of two to the number of items per bin.

1. **{B_i}** means that there can be more than one bin size and an unlimited supply of each size.
2. **stretch** refers to certain asymptotic bounds which compare the number of bins of capacity C needed to pack L by algorithm A to the number of unit-capacity bins needed by an optimal packing.
3. **mutex** stands for mutual exclusion and introduces constraints, where needed, in the form of a sequence of pairs (s_i, s_j) , $i \neq j$, meaning that s_i and s_j cannot be put in the same bin.
4. **card(B) ≤ k** gives a bound on the number of items that can be packed in a bin.
5. $s_i \leq \alpha$ or $s_i \geq \alpha$ denotes bounds on item sizes, the former being far more common in the literature. In some cases, α is specialized to a discrete parameter $1/k$, k an integer. The problems with item-size restrictions are called *parametric* cases in the literature.
6. **restricted s_i** refers to simplified problems where the number of different item sizes is finite.
7. **discrete** calls for discrete sets of item sizes, in particular, item sizes that, for a given integer r , are all multiples of $1/r$ with $C = 1$. Equivalently, the bin size could be taken as r and item sizes restricted to the set $\{1, \dots, j\}$ for some $j \in \{1, \dots, r\}$. While this may not be a significant practical constraint, it will affect the difficulty of the analysis and may create significant changes in the results.
8. **controllable** means that one has the possibility not to pack an item as is, but to first take some decision about it, such as rejecting it or splitting it into parts.

3 On-Line Algorithms

Recall that a bin packing algorithm is called *on-line* if it packs each element as soon as it is inspected, without any knowledge of the elements not yet encountered (either the number of them or their sizes). This review starts with results for some classical algorithms and then generalizes to the *Any-Fit* and the *Almost Any-Fit* classes of on-line algorithms. The subclass of *bounded-space* on-line algorithms will also be considered: an algorithm is bounded space if the number of open bins at any time in the packing process is bounded by a constant. (The practical significance of this condition is clear; e.g., one may have to load trucks at a depot where only a limited number of trucks can be at the loading dock). This section will be concluded by a detailed discussion of lower bounds on the APRs of certain classes of on-line algorithms, but before doing so, relevant variations of old algorithms and the current best-in-class algorithms will be presented. Anomalous behavior occurring in on-line algorithms is discussed in Sect. 4.4.

3.1 Classical Algorithms

In describing an on-line algorithm, it will occasionally be convenient, just before a decision point, to refer to the next item to be packed as the *current* item; right after A packs a_i , $i < n$, a_{i+1} becomes the current item. A simple approach is to pack the bins one at a time according to

Next-Fit (NF): After packing the first item, NF packs each successive item in the bin containing the last item to be packed, if it fits in that bin; if it does not fit, NF closes that bin and packs the current item in an empty bin.

The time complexity of NF is clearly $O(n)$. Note that only one bin is ever open under NF, so it is bounded space. This advantage is compensated by a relatively poor APR, however.

Theorem 1 (Johnson et al. [129]) *One has*

$$R_{\text{NF}}^{\infty}(\alpha) = \begin{cases} 2 & \text{if } \frac{1}{2} \leq \alpha \leq 1 \\ (1 - \alpha)^{-1} & \text{if } 0 < \alpha < \frac{1}{2} \end{cases}.$$

Fisher [86] discovered an interesting property that NF does not share with other classical approximation algorithms. He proved that NF packs any list and its reverse into the same number of bins. The conspicuous disadvantage of NF is that it closes bins that could be used for packing later items. An immediate improvement would seem to be never to close bins. But then the next question is: if an item can be put into more than one open bin, which bin should be selected? One possible rule drawn from scheduling theory (where it is known as the *greedy* or *largest processing time* rule) is the following

Worst-Fit (WF): If there is no open bin in which the current item fits, then WF packs the item in an empty bin. Otherwise, WF packs the current item into an open bin of smallest content in which it fits; if there is more than one such bin, WF chooses the lowest indexed one.

Although one might expect WF to behave better than NF, it does not.

Theorem 2 (Johnson [127]) *For all $\alpha \in (0, 1]$*

$$R_{\text{WF}}^{\infty}(\alpha) = R_{\text{NF}}^{\infty}(\alpha).$$

To achieve smaller APRs, there are many better rules for choosing from among the open bins. One that quickly comes to mind is

First-Fit (FF): FF packs the current item in the lowest indexed nonempty bin in which it fits, assuming there is such a bin. If no such bin exists, FF packs the current item in an empty bin. FF is neither linear time nor bounded space.

A natural complement to WF packs each item into a bin that *minimizes* the space leftover.

Best-Fit (BF): If there is no open bin in which the current item fits, then BF packs the item in an empty bin. Otherwise, BF packs the current item into an open bin of largest content in which it fits; if there is more than one, such bin BF chooses the lowest indexed one.

By adopting appropriate data structures for representing packings, it is easy to verify that the time complexity of these algorithms is $O(n \log n)$. The analysis of the worst-case behavior of the packings they produce is far more complicated. The basic idea of the upper bound proofs is the *weighting function* technique, which has played a fundamental role in bin packing theory. So, before proceeding with other algorithms, it is convenient to describe this technique, which was introduced in [103, 129] and subsequently applied in many other papers (see, e.g., [13, 92, 127, 143]).

3.2 Weighting Functions

To bound the asymptotic worst-case behavior of an algorithm A , one can try to find a function $W_A : (0, 1] \rightarrow \mathbb{R}$ with the properties: (i) There exists a constant $K \geq 0$ such that for any list L

$$\sum_{a \in L} W_A(a) \geq A(L) - K \quad (1)$$

and (ii) there exists a constant K^* such that for any set B of items summing to no more than 1

$$\sum_{a \in B} W_A(a) \leq K^*. \quad (2)$$

The value $W_A(a)$ is the *weight* of item a under A 's *weighting function* W_A . Note that (1) requires that, for large packings (large $A(L)$), the average total weight of the items in a bin must satisfy a lower bound close to 1 for all lists L . On the other hand, (2) says that the total weight in any bin of any packing is at most K^* , so for an optimal packing

$$\sum_{a \in L} W_A(a) = \sum_{j=1}^{OPT(L)} \sum_{a \in B_j} W_A(a) \leq K^* OPT(L). \quad (3)$$

Together, (1) and (3) obviously imply the bound $A(L) \leq K^* OPT(L) + K$, and hence $R_A^\infty \leq K^*$. Note that the technique above is only representative; it is easy to find weaker conditions on the weighting function which will produce the same upper bound for the APR.

The proof of the NF upper bound is not appreciably simplified by a weighting function argument, but it does offer a simple example of such arguments. Consider the case $\alpha = 1$ and define $W_{\text{NF}}(a) = 2s(a)$ for all a , where $s(a)$ denotes the size of item a . An observation is needed about NF when $\text{NF}(L) > 1$: Since the first item of B_{j+1} did not fit in B_j , $1 \leq j < \text{NF}(L)$, the sum of the item sizes in $B_j \cup B_{j+1}$

exceeds 1, and hence the sum of the weights of the items in $B_j \cup B_{j+1}$ exceeds 2. Then

$$\begin{aligned} \sum_{j=1}^{\text{NF}(L)} \sum_{a \in B_j} W_{\text{NF}}(a) &\geq \sum_{j=1}^{\lfloor \text{NF}(L)/2 \rfloor} \sum_{a \in B_{2j-1} \cup B_{2j}} W_{\text{NF}}(a) \\ &\geq 2 \lfloor \text{NF}(L)/2 \rfloor \geq \text{NF}(L) - 1, \end{aligned}$$

so (1) holds with $K = 1$. The inequality (2) with $K^* = 2$ is immediate from the definition of W_{NF} , so $R_{\text{NF}}^\infty \leq 2$, as desired.

There is no systematic way to find appropriate weighting functions, and the approach can be difficult to work out. For example, consider the proof of the following result.

Theorem 3 (Johnson et al. [129])

$$R_{\text{FF}}^\infty = R_{\text{BF}}^\infty = \begin{cases} \frac{17}{10} & \text{if } \frac{1}{2} < \alpha \leq 1 \\ 1 + \lfloor \frac{1}{\alpha} \rfloor^{-1} & \text{if } 0 < \alpha \leq \frac{1}{2} \end{cases}.$$

The proof of the $\frac{17}{10}$ upper bound consists of verifying that the following weighting function suffices

$$W(x) = \begin{cases} \frac{6}{5}x & \text{if } 0 \leq x \leq \frac{1}{6} \\ \frac{9}{5}x - \frac{1}{10} & \text{if } \frac{1}{6} < x \leq \frac{1}{3} \\ \frac{6}{5}x + \frac{1}{10} & \text{if } \frac{1}{3} < x \leq \frac{1}{2} \\ \frac{6}{5}x + \frac{2}{5} & \text{if } \frac{1}{2} < x \leq 1 \end{cases}$$

and it requires a substantial effort. (The argument encompasses several pages of case analysis.) Yet, despite a few hints that emerge in this effort, a clear understanding of how this function was obtained in the first place requires still more effort.

Theorem 3 for $\alpha \leq \frac{1}{2}$ can be proved without weighting functions (see [129]), but the reader may find it instructive to prove the $1 + \lfloor \frac{1}{\alpha} \rfloor^{-1}$ upper bound with the weighting function

$$W_{\text{FF}}(a) = \frac{r+1}{r} s(a), \quad \alpha = \frac{1}{r}, \quad r \geq 2.$$

Sequences of specific examples establish lower bounds for R_A^∞ . In particular, one seeks a sequence of lists $L_{(n_1)}, L_{(n_2)}, \dots$ satisfying $L_{(n_k)} \in V_\alpha$ ($k = 1, 2, \dots$), $\lim_{k \rightarrow \infty} OPT(L_{(n_k)}) = \infty$, and for some constant K_* ,

$$\lim_{k \rightarrow \infty} \frac{A(L_{(n_k)})}{OPT(L_{(n_k)})} = K_*.$$

Then $R_A^\infty \geq K_*$, and if K_* is equal to K^* of the upper bound analysis, one has the APR for the algorithm considered. Finding worst-case examples can be anywhere from quite easy to quite hard. For example, the reader would have little trouble with NF but would probably find FF to be quite challenging.

3.3 Any-Fit and Almost Any-Fit Algorithms

The algorithms described so far belong to a much larger class of on-line heuristics satisfying similar worst-case properties. It is clear that FF, WF, and BF satisfy the following condition:

Any-Fit constraint: *If B_1, \dots, B_j are the current nonempty bins, then the current item will not be packed into B_{j+1} unless it does not fit in any of the bins B_1, \dots, B_j .*

The class of on-line heuristics satisfying the Any-Fit constraint will be denoted by \mathcal{AF} . The following result shows that FF and WF are best and worst algorithms in \mathcal{AF} , in the APR sense.

Theorem 4 (Johnson [127]) *For every algorithm $A \in \mathcal{AF}$ and for every $\alpha \in (0, 1]$*

$$R_{\text{FF}}^\infty(\alpha) \leq R_A^\infty(\alpha) \leq R_{\text{WF}}^\infty(\alpha).$$

By a slight tightening of the Any-Fit constraint, one can eliminate the high-APR algorithms like WF and define a class of heuristics all having the same APR.

Almost Any-Fit constraint: *If B_1, \dots, B_j are the current nonempty bins, and B_k ($k \leq j$) is the unique bin with the smallest content, then the current item will not be packed into B_k unless it does not fit in any of the bins to the left of B_k .*

Clearly, WF does not satisfy this condition, but it is easy to verify that both FF and BF do. The class of on-line algorithms satisfying both constraints above will be denoted by \mathcal{AAF} .

Theorem 5 (Johnson [127]) *$R_A^\infty(\alpha) = R_{\text{FF}}^\infty(\alpha)$ for every A in \mathcal{AAF} .*

Almost Worst-Fit (AWF) is a modification of WF whereby the current item is always placed in a bin having the second lowest content, if such a bin exists and the current item fits in it; the current item is packed in a bin with smallest content only if it fits nowhere else. Interestingly, though it seems to differ little from WF, AWF has a substantially better APR, since it is in \mathcal{AAF} and hence $R_{\text{AWF}}^\infty(\alpha) = R_{\text{FF}}^\infty(\alpha) = \frac{17}{10}$.

3.4 Bounded-Space Algorithms

An on-line bin packing algorithm uses *k-bounded space* if, for each item, the choice of where to pack it is restricted to a set of at most k open bins. One obtains bounded-space counterparts of the algorithms of the previous section by specifying a suitable policy for closing bins.

As previously observed, NF uses only 1-bounded space; the only algorithm to use less is the trivial algorithm that puts each item in a separate bin. To improve on the APR of NF, yet stay with bounded-space algorithms, Johnson [126] proposed an algorithm that packs items according to the First-Fit rule, but considers as candidates only the k most recently opened bins; when a new bin has to be opened and there are already k open bins, then the lowest indexed open bin is closed. It can be expected that the APR of the resulting algorithm, which is known as *Next- k -Fit* (NF_k), tends to $\frac{17}{10}$ as k increases. Finding the exact bound was not an easy task, although Johnson did give a narrow range for the APR. Later, Csirik and Imreh [58] constructed the worst-case sequences, and then Mao was able to prove the exact bound:

Theorem 6 (Mao [153]) *For any $k \geq 2$,* $R_{\text{NF}_k}^{\infty} = \frac{17}{10} + \frac{3}{10(k-1)}$.

In general, a bounded-space algorithm is defined by specifying the packing and closing rules. An interesting class of such rules is based on FF and BF as follows:

- **Packing rules:** The elements are packed following either the First-Fit rule or the Best-Fit rule.
- **Closing rules:** The next bin to close is either the lowest indexed one or one of largest content.

The algorithm that uses packing rule X, closing rule Y, and k -bounded space is denoted by AXY_k , where $X = F$ or B for FF or BF and $Y = F$ or B for the lowest indexed (First) open bin or the largest-content (Best) open bin. With this terminology, NF_k can also be classified as AFF_k . Note that, independently of the chosen rules, if $k = 1$, then one always gets NF.

Algorithm ABF_k was first analyzed by Mao, who called it *Best- k -Fit*. He proved that for any fixed k , this algorithm is slightly better than NF_k .

Theorem 7 (Mao [152]) *For any $k \geq 2$,* $R_{\text{ABF}_k}^{\infty} = \frac{17}{10} + \frac{3}{10k}$.

The tight asymptotic bound for AFB_k was found by Zhang [184] (see also the paper version [187]).

Theorem 8 (Zhang [187]) *For any $k \geq 2$,* $R_{\text{AFB}_k}^{\infty} = R_{\text{NF}_k}^{\infty}$.

Finally, consider the algorithm ABB_k whose asymptotic behavior is, rather surprisingly, independent of $k \geq 2$ and equal to that of FF and BF.

Theorem 9 (Csirik and Johnson [60]) *If $k \geq 2$ then $R_{\text{ABB}_k}^{\infty} = \frac{17}{10}$.*

Since all of the above algorithms fulfill the Any-Fit constraint with respect to the open bins, the overall bound $R_A^{\infty} \geq \frac{17}{10}$ is to be expected from [Theorem 4](#). A better on-line algorithm can only be obtained without the Any-Fit constraint. In the remaining part of this section, it is shown how the fruitful idea of *reservation techniques* (introduced by Yao [180] for unbounded-space algorithms and discussed

in the next section) led to on-line algorithms which are neither in class \mathcal{AF} nor in \mathcal{AAF} .

Yao's idea appeared in the work of Lee and Lee [143] who developed the *Harmonic-Fit* algorithm, which will be denoted by HF_k since, for the case $\alpha = 1$, it uses at most k open bins. The algorithm divides the interval $(0, 1]$ into subintervals $I_j = (\frac{1}{j+1}, \frac{1}{j}]$ ($1 \leq j \leq k-1$) and $I_k = (0, \frac{1}{k}]$. An element is called an I_j -element if its size belongs to interval I_j . Similarly, there are k different bin types: an I_j -bin is reserved for I_j -elements only. An I_j -element is always packed into an I_j -bin following the Next-Fit rule, and so at most, k bins are open at the same time. Galambos [92] extended the idea to general α . Observe that, by selecting $\alpha \in (\frac{1}{r+1}, \frac{1}{r}]$, the number, say M , of bin types exceeds the space bound k by $r-1$; this is because I_j -bins for $j < r$ are never opened. Instead of notation HF_k with k the space bound, the literature often uses HF_M with $M = k + r - 1$ being the number of bin types.

The general APR can be formulated as follows. (See the end of Sect. 2 for the definitions of the quantities $t_s(r)$ and $h_\infty(r)$.)

Theorem 10 (Lee and Lee [143], Galambos [92]) *Suppose that $L \in V_\alpha$ with $\alpha \in (\frac{1}{r+1}, \frac{1}{r}]$ for some positive integer r and choose any sequence k_s , $s \geq 1$, such that $t_s(r) < k_s + 1 \leq t_{s+1}(r)$. Then*

$$\lim_{s \rightarrow \infty} R_{\text{HF}_{k_s}}^\infty(\alpha) = \lim_{s \rightarrow \infty} \left(1 + \sum_{j=2}^s \frac{1}{t_j(r)-1} + \frac{k_s + r - 1}{(t_{s+1}(r)-1)(M-1)} \right) = h_\infty(r).$$

The results in [92, 143] gave tight bounds only for the cases $k = t_j(r) - r + 1$ and $k = t_{j+1}(r) - r$ for integers $j \geq 1$. Also, considering only the $\alpha = 1$ case, one can see that, to obtain an APR better than $\frac{17}{10}$, at least seven open bins are needed. These observations raised two further questions:

- For the case $\alpha = 1$, is there an on-line, bounded-space algorithm that uses fewer than 7 bins and has an APR better than $\frac{17}{10}$?
- What are tight bounds on HF_k for specific k ?

An affirmative answer to the first question was given by Woeginger [175]. Using a more sophisticated interval structure, one based on the Golomb sequences, the performance of his *Simplified Harmonic* (SH_k) algorithm improved on the $\frac{17}{10}$ bound with six open bins; precisely, $R_{\text{SH}_6}^\infty \approx 1.69444$. Moreover, Woeginger proved the following deeper, more general result.

Theorem 11 (Woeginger [175]) *To achieve the worst-case performance ratio of heuristic HF_k with k open bins and $\alpha = 1$, heuristic SH_k only needs $O(\log \log k)$ open bins.*

The second question was investigated by Csirik and Johnson [60] (see also [59]) and van Vliet [172, 173]. They gave tight bounds for the case $\alpha = 1$ with $k = 4$ and 5. Tight bounds for further k remain open problems.

Table 1 APR values for bounded-space algorithms, rounded to five decimals. The values in column $\text{HF}_k \leq$ are upper bounds and are tight if starred. Note that $42 = t_4(1) - 1$

k	NF_k	ABF_k	ABB_k	$\text{HF}_k \leq$	SH_k	best
2	2.00000	1.85000	1.70000	2.00000*	2.00000	1.70000
3	1.85000	1.80000	1.70000	1.75000*	1.75000	1.70000
4	1.80000	1.77500	1.70000	1.71429*	1.72222	1.70000
5	1.77500	1.76000	1.70000	1.70000*	1.70000	1.70000
6	1.76000	1.75000	1.70000	1.70000*	1.69444	1.69444
7	1.75000	1.74286	1.70000	1.69444*	1.69388	1.69388
8	1.74286	1.73750	1.70000	1.69388	1.69106	1.69106
9	1.73750	1.73333	1.70000	1.69345	1.69104	1.69104
10	1.73333	1.73000	1.70000	1.69312	1.69104	1.69104
42	1.70732	1.70714	1.70000	1.69106*	1.69103	1.69103
$+\infty$	1.70000	1.70000	1.70000	1.69103	1.69103	1.69103

Similarly, the general case has not been discussed exhaustively, and some questions raised by Woeginger [175] are still open:

- What is the smallest k such that there exists an on-line heuristic using k -bounded space and having an APR strictly less than $\frac{17}{10}$?
- What is the best possible APR for any on-line heuristic using 2-bounded space? (ABB_2 achieves a worst-case ratio of $\frac{17}{10}$.)
- By considering only algorithms that pack the items by the Next-Fit rule according to some fixed partition of $(0, 1]$ into k subintervals, which partition gives the best APR ? (It is known that for $k \leq 2$, the best possible APR is 2 (see Csirik and Imreh [58]), but for $k \geq 3$, no tight bound is known.)

Tables 1 and 2 show the best results known for bounded-space algorithms. Note that the worst-case ratios of all algorithms in Table 1 are never smaller than $h_\infty(1) \approx 1.69103$. As pointed out by Lee and Lee [143] for the $\alpha = 1$ case, bounded-space algorithms cannot do better. The result holds for general α too, as shown by Galambos, that is,

Theorem 12 (Lee and Lee [143], Galambos [92]) *Every bounded-space on-line bin packing algorithm A satisfies $R_A^\infty(\alpha) \geq h_\infty(r)$ for all α , $\frac{1}{r+1} < \alpha \leq \frac{1}{r}$.*

None of the known bounded-space algorithms achieves the lower bound using a finite number of open bins. It will be later shown (see Sect. 3.7) that the bound can be achieved with three open bins if repacking among the open bins is allowed, but without such a relaxation, the question remains open.

3.5 Variations and the Best-in-Class

Chronologically, Yao [180] was the first to break through the $h_\infty(1)$ barrier with his *Refined First-Fit* (RFF) algorithm. RFF classifies items into types 1, 2, 3, or 4

Table 2 APR values for $\text{HF}_k(\frac{1}{r})$. Starred values are tight

k	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 6$
2	2.00000*	1.50000*	1.33333*	1.25000*	1.20000*	1.18888*
3	1.75000*	1.44444*	1.31250*	1.24000*	1.19444*	1.16326*
4	1.71429*	1.43750	1.31000	1.23888	1.19387	1.16294
5	1.70000*	1.43333	1.30833	1.23809	1.19345	1.16269
6	1.70000*	1.43055	1.30714	1.23750	1.19312	1.16250
7	1.69444*	1.42857	1.30625	1.23703	1.19285	1.16233
8	1.69388	1.42708	1.30555	1.23666	1.19264	1.16220
9	1.69345	1.42592	1.30500	1.23636	1.19246	1.16208
10	1.69312	1.42499	1.30454	1.23611	1.19230	1.16198
$+\infty$	1.69103	1.42307	1.30238	1.23441	1.19102	1.16102

accordingly as their sizes are in the respective intervals $(0, \frac{1}{3}]$, $(\frac{1}{3}, \frac{2}{5}]$, $(\frac{2}{5}, \frac{1}{2}]$, and $(\frac{1}{2}, 1]$. RFF packs four sequences of bins, one for each type. With one exception, RFF packs type- i items into the sequence of type- i bins using First-Fit. The exception is that every sixth type-2 item (with a size in $(\frac{1}{3}, \frac{2}{5}]$) is thrown in with the type-4 items, that is, packed by First-Fit into the sequence of type-4 bins. It is easily verified that RFF uses unbounded space and has a time complexity $O(n \log n)$. Yao proved that $R_{\text{RFF}}^{\infty} = \frac{5}{3} = 1.666\dots$. Yao did not use the usual weighting function technique but based his proof on enumeration of the elements in each class. Also, the APR remains unchanged if the special treatment given every sixth type-2 item is instead given every m -th type-2 item, where m is taken to be one of 7, 8, or 9.

It was immediately clear that this reservation technique was a promising approach, a fact supported by the Harmonic-Fit algorithm discussed in the previous section. The main disadvantage of the latter algorithm is that each I_1 -element, even with a size slightly over $\frac{1}{2}$, is packed alone into a bin. An immediate improvement is to try to add other items to these bins. In their algorithm *Refined Harmonic-Fit* (RHF), Lee and Lee [142, 143] modified HF_{20} by subdividing intervals I_1 and I_2 into two subintervals:

$$\begin{aligned} I_1 &= I_{1,s} \cup I_{1,b}, \\ I_2 &= I_{2,s} \cup I_{2,b} \end{aligned}$$

with $I_{2,s} = (\frac{1}{3}, \frac{37}{96}]$, $I_{2,b} = (\frac{37}{96}, \frac{1}{2}]$, $I_{1,s} = (\frac{1}{2}, \frac{59}{96}]$, and $I_{1,b} = (\frac{59}{96}, 1]$. This brought the number of bin types to $M = 22$. The packing strategy for I_j -elements ($3 \leq j \leq M - 2 = 20$), $I_{1,b}$ -elements, and $I_{2,b}$ -elements is the same as in Harmonic-Fit, but the $I_{1,s}$ -elements and the $I_{2,s}$ -elements are allowed to share the same bins in certain situations. The details are omitted. Note however that the time complexity of RHF is $O(n)$, but the algorithm is no longer bounded space. Its APR is given by

Theorem 13 (Lee and Lee [143]) $R_{\text{RHF}}^{\infty} \leq \frac{373}{228} \approx 1.63596$.

It is not known whether this bound is tight.

In 1989, several improved algorithms were presented by Ramanan et al. [159]. The first one, called *Modified Harmonic-Fit* (MHF), applies Yao's idea in a more sophisticated way. Instead of choosing $\frac{59}{96}$ as the point dividing $(\frac{1}{2}, 1]$, the problem is handled in a more general fashion. Let the number of bin types satisfy $M \geq 5$, and consider the subdivision of $(0, 1]$:

$$(0, 1] = \bigcup_{j=1}^{M-2} I_j$$

with $I_1 = I_{1,s} \cup I_{1,b}$ and $I_2 = I_{2,s} \cup I_{2,b}$ as earlier, but now

$$\begin{aligned} I_{1,s} &= (\frac{1}{2}, 1 - y] & I_{2,s} &= (\frac{1}{3}, y] \\ I_{1,b} &= (1 - y, 1] & I_{2,b} &= (y, \frac{1}{2}] \end{aligned}$$

for some y satisfying $\frac{1}{3} < y < \frac{1}{2}$. Initially, the set of empty bins is divided into M infinite classes, each associated with a subinterval. All $I_{1,b}$ -bins, $I_{2,s}$ -bins, $I_{2,b}$ -bins, and I_j -bins ($3 \leq j \leq M - 2$) are only used to pack elements from the associated interval (as in Harmonic-Fit). $I_{1,s}$ -bins on the other hand can contain $I_{1,s}$ -elements and some of the $I_{2,s}$ -elements and I_j -elements ($j \in \{3, 6, 7, \dots, M - 2\}$). The algorithm also includes more complicated rules for packing and for deciding when items with sizes in different intervals can share the same bin. For this algorithm with $M = 40$, the following bounds hold.

Theorem 14 (Ramanan et al. [159])

$$1.6156146 < \frac{3}{2} + \frac{1}{9} + \frac{1}{222} - \frac{1}{987012} \leq R_{\text{MHF}}^{\infty} < \frac{3}{2} + \frac{1}{9} + \frac{1}{222} = 1.615615\dots$$

The above algorithm was further generalized by Ramanan et al. [159] who introduced a sequence of classes of on-line linear-time algorithms, called $C^{(h)}$; an algorithm A belongs to $C^{(h)}$, for a given $h \geq 1$, if it divides $(0, 1]$ into disjoint subintervals including

$$\begin{aligned} I_{1,b} &= (1 - y_1, 1], & I_{2,b} &= (y_h, \frac{1}{2}], \\ I_{1,j} &= (1 - y_{j+1}, 1 - y_j], & I_{2,j} &= (y_{h-j}, y_{h-j+1}], \quad 1 \leq j \leq h, \end{aligned}$$

and

$$I_{\lambda,1} = (0, \lambda], \quad I_{\lambda,2} = (\lambda, \frac{1}{3}], \quad 0 < \lambda \leq \frac{1}{3}$$

where $\frac{1}{3} = y_0 < y_1 < \dots < y_h < y_{h+1} = \frac{1}{2}$, and $I_{\lambda,2} = \emptyset$ if $\lambda = \frac{1}{3}$. Elements are classified, as usual, according to the intervals in which their sizes fall.

Note that algorithm MHF belongs to $C^{(1)}$. Ramanan et al. [159] developed an algorithm (*Modified Harmonic-2*, MH2), which is in $C^{(2)}$ and proved that $R_{\text{MH2}}^{\infty} \leq 1.612\dots$. The algorithm is quite elaborate and beyond the scope of this survey.

The authors discussed further improvements aimed at reducing the APR to 1.59. They also proved the lower bound result.

Theorem 15 (Ramanan et al. [159]) *There is no on-line algorithm A in $C^{(h)}$ such that $R_A^\infty < \frac{3}{2} + \frac{1}{12} = 1.58333\dots$*

The HARMONIC+1 algorithm of Richey [160] subdivides intervals $(\frac{1}{2}, 1]$ and $(\frac{1}{3}, \frac{1}{2}]$ very finely (using 76 classes of subintervals!) in order to allow a precise item pairing in these two intervals. It also allows items of size $(\frac{1}{4}, \frac{1}{3}]$ to be mixed with larger items of various sizes and not just with those of size at least $\frac{1}{2}$. Seiden [163] showed that such result was flawed and gave a new algorithm, HARMONIC++, whose asymptotic performance ratio is at most 1.58889, which is the current best APR for on-line bin packing.

3.6 APR Lower Bounds

In previous sections, some results concerning lower bounds on the APR were mentioned, but the fundamental problem was not considered yet: what is the best an on-line algorithm can do in the asymptotic worst case? In this section, lower bound results are discussed in chronological order.

Most of the existing lower bounds come from the same idea. To obtain a good packing, it seems advisable to first pack the large elements so that either a bin is “full enough” or its empty space can be reduced by subsequent small elements. Therefore, in order to force bad behavior on a heuristic algorithm A , one should challenge it with a list in which small items come first. If A adopts a policy that packs these small items tightly, then it will not be able to find a good packing for the large items which may come later. If, instead, A leaves space for large items while packing the small ones, then the expected large items might not appear in the list. In both cases, the resulting packing will be poor.

To give a more precise description, consider a simple example involving two lists L_1 and L_2 , each containing n identical items. The size of each element in L_1 is $\frac{1}{2} - \varepsilon$, and the size of each element in L_2 is $\frac{1}{2} + \varepsilon$. The asymptotic behavior of an arbitrary approximation algorithm A will be investigated on two lists: L_1 alone and the concatenated list $L_1 L_2$. It is easy to see that $OPT(L_1) = \frac{n}{2}$ and $OPT(L_1 L_2) = n$. Consider the behavior of this algorithm on L_1 : it will pack some elements alone into bins (say x of them), and it will match up the remaining $n - x$. Hence $A(L_1) = \frac{n+x}{2}$. For the concatenated list $L_1 L_2$, when processing the elements of L_2 , the best that A can do is to add one element of L_2 to each of x bins containing a single element of L_1 and to pack alone the remaining $n - x$ elements. Therefore, $A(L_1 L_2) = \frac{3n-x}{2}$. Since n may be arbitrarily large,

$$R_A^\infty \geq \max \left\{ \frac{A(L_1)}{OPT(L_1)}, \frac{A(L_1 L_2)}{OPT(L_1 L_2)} \right\} = \max \left\{ \frac{n+x}{n}, \frac{3n-x}{2n} \right\}$$

Table 3 Lower bounds and R_A^∞ for various algorithms

r	Lower bound	R_{HF}^∞	R_{FF}^∞	R_{NF}^∞	Best known
1	1.54037	1.69103	1.70000	2.00000	1.58889
2	1.38966	1.42307	1.50000	2.00000	1.42307
3	1.29144	1.30238	1.33333	1.50000	1.30238
4	1.22986	1.23441	1.25000	1.33333	1.23441
5	1.18881	1.19102	1.20000	1.25000	1.19102
6	1.15982	1.16102	1.16667	1.20000	1.16102

for which the minimum is attained when $x = \frac{n}{3}$, implying a lower bound of $\frac{4}{3}$ for the APR of *any* on-line algorithm A .

The above idea can be easily generalized by taking a carefully chosen series of lists L_1, \dots, L_k and evaluating the performance of a heuristic on the concatenated lists $L_1 \dots L_j$, ($1 \leq j \leq k$). The first step along these lines was made by Yao [180]. He proved a lower bound of $\frac{3}{2}$ based on three lists of equal-size elements, the sizes being $\frac{1}{2} + \varepsilon$, $\frac{1}{3} + \varepsilon$, and $\frac{1}{6} - 2\varepsilon$. Using Sylvester's sequences, Brown [32] and Liang [147] independently gave a further improvement to 1.53634577 (The largest sizes in their sequences were as follows: $\frac{1}{2} + \varepsilon$, $\frac{1}{3} + \varepsilon$, $\frac{1}{7} + \varepsilon$, $\frac{1}{43} + \varepsilon$, and $\frac{1}{1807} + \varepsilon$.) Galambos [92] used the Golomb sequences to extend the idea to general α . The proof in [92] was considerably simplified by Galambos and Frenk [93]. van Vliet gave an exhaustive analysis of the lower bound constructions with a linear programming technique applied to all α and gave the following lower bound:

Theorem 16 (van Vliet [171]) *For any on-line algorithm A , $R_A^\infty \geq 1.54015 \dots$*

The current best-in-class, $R_A^\infty \geq 1.54037$, was given by Balogh et al. [15]. Table 3 gives a comparison for several values of $\alpha = \frac{1}{r}$ between the best lower bounds and the corresponding upper bounds for various algorithms. It is interesting to note that the gap between the lower and upper bounds becomes rather small for $r \geq 2$.

Faigle, Kern, and Turán [83] proved that if there are only two item sizes, then no on-line algorithm can be better than 4/3.

Chandra [38] has examined the effect on lower bounds when randomization is allowed in the construction of on-line algorithms, that is, when coin flips are allowed in determining where to pack items. The performance ratio for randomized algorithm A is now $E[A(L)]/OPT(L)$, where $E[A(L)]$ is the expected number of bins needed by A to pack the items in L . Chandra has shown that there are lists such that this ratio exceeds 1.536 for all randomized on-line algorithms, and so from this limited standpoint, results suggest that randomization is not a valuable tool in the design of on-line algorithms.

3.7 Semi-on-line Algorithms

The APR of on-line algorithms cannot break through the $1.540\dots$ barrier (see Sect. 3.6). For almost 20 years, only the pure on-line and off-line algorithms were analyzed, and no attention was paid to algorithms lying between these two classes. In a more general setting, one can consider giving the algorithm more information about the list and/or more freedom with respect to the pure on-line case. This section deals with *semi-on-line* (SOL) algorithms that can

- Repack elements
- Look ahead to later elements before assigning the current one
- Assume some preordering of the elements

First consider the case where repacking is allowed. SOL algorithms allowing only a restricted number of elements to be repacked at each step are called *c-repacking SOL* algorithms. It was seen in Sect. 3.4 that no known on-line bounded-space algorithm reaches the bound $h_\infty(1)$ using finitely many open bins. It will be shown that this is possible with SOL algorithms.

In 1985, Galambos [91] made a first step in this direction. His *Buffered Next-Fit* (BNF) algorithm uses two open bins, say B_1 and B_2 . The arriving elements are initially packed into B_1 , until the first element arrives for which B_1 does not have enough space. This element and those currently packed in B_1 are then reordered by decreasing size and repacked in B_1 and B_2 following the Next-Fit rule. B_1 is now closed, B_2 is renamed B_1 , and a new bin B_2 is opened. Using a weighting function approach, it was proved that $h_\infty(1) \leq R_{\text{BNF}}^\infty \leq \frac{18}{10}$.

Galambos and Woeginger [95] generalized the above idea, adopting a better weighting function. Let $w(B)$ be the sum of the weights associated with the items currently packed in bin B . Their *Repacking* algorithm (REP_3) uses three open bins. When a new item a_i arrives, the following steps are performed: (i) a_i is packed into an empty bin; (ii) all the elements in the three open bins, sorted by nonincreasing size, are repacked by the FF strategy, with the result that either one bin becomes empty or at least one bin B has $w(B) \geq 1$; and (iii) all bins B with $w(B) \geq 1$ are closed and replaced by new empty bins. In [95], it was proved that this repacking in fact helps, since $R_{\text{REP}_3}^\infty = h_\infty(1)$. It is not known whether the same result can be obtained with two bins.

Gambosi et al. [99] (see also [97] and [98]) were the first to beat the 1.540 on-line bound via repacking. They gave two algorithms. Both of them use an unusual step to repack the elements: the small elements are grouped into a “bundle” of $O(n)$ elements, which can be repacked in a single step.

In the first algorithm, A_1 , the interval $(0, 1]$ is divided into four subintervals, and the elements are packed into bins in a Harmonic-like way. As each new item is packed, groups of small elements can be repacked – in a bundle – so as to fill up gaps in bins that are not full enough. By using appropriate data structures, this repacking is performed in constant time. The algorithm has linear time and space complexity, and it has an APR, $R_{A_1}^\infty \leq \frac{3}{2}$. In the second algorithm, A_2 , the unit interval is divided into six subintervals, and, as

each new item is encountered, the elements are repacked more carefully, in $O(\log n)$ time, by means of a pairing technique analogous to that introduced by Martel (see Sect. 4.2). The time complexity of A_2 is $O(n \log n)$, and its APR is $R_{A_2}^\infty \leq \frac{4}{3}$.

Ivkovic and Lloyd [122] gave a further improvement on SOL algorithms achieving a $\frac{5}{4}$ worst-case ratio. Their algorithm is much more complicated than the previous ones, as it was designed for handling the dynamic packing case. The dynamic bin packing problem will be considered in Sect. 7.1 in detail, but here it is enough to know that in case of dynamic packing, deletions of some elements are allowed in each step and $A(L)$ is considered as the maximum number of occupied bins during the packing. Ivkovic and Lloyd proved a $\frac{4}{3}$ lower bound for the c-repacking SOL algorithms. This result was improved by Balogh et al. [14].

Theorem 17 (Balogh et al. [14]) *For any c-repacking SOL algorithm A, the APR satisfies $R_A^\infty \geq 1.3871 \dots$*

Ivkovic and Lloyd [121] also presented approximation schemes for their dynamic, SOL model, applying the techniques of Fernandez de la Vega and Lueker and those of Karmarkar and Karp cited in Sect. 4.3.

Consider now the case in which the algorithm is allowed to look ahead, in the sense that, when an element arrives, it is not necessary to pack it immediately; one is allowed to collect further elements whose sizes can effect the packing decision. In order to avoid relaxation to off-line algorithms, consider the case of *bounded* lookahead. Grove [111] proposed a k -bounded algorithm which had, in addition, a capacity (or *warehouse*) constraint W . The algorithm can delay the packing of item a_i until it has collected all subsequent elements a_{i+1}, \dots, a_j such that $\sum_{r=i}^j a_r \leq W$. For any fixed k and W , the $h_\infty(1)$ lower bound (see Theorem 12) remains valid, but Grove's *Revised Warehouse* (RW) algorithm reaches the bound if W is sufficiently large. In his proof, Grove uses a weighting function argument.

Another subclass of the lookahead SOL problems arises if the input list is divided into a number of batches. If an algorithm works on a batched list, then it has to pack each batch as an off-line list (i.e., lookahead is only possible within the current batch). However, while packing elements of the current batch, the items of earlier batches cannot be moved. If the number of batches is bounded by m , the problem is called *m-batched bin packing* (BBP) problem. Gutin et al. [112] were the first to study the BBP problem: they investigated in depth the 2-BBP problem and proved a lower bound of $1.3871 \dots$ for this case.

The rarely considered class of algorithms in which it is assumed that the input list is *presorted* is finally considered. Because of the lower bound constructions, it is easy to see that, if the list is presorted by increasing item size, the on-line lower bounds remain valid. Moreover, the Johnson result holds: if the list is presorted by decreasing item size, then $\frac{11}{9} \leq R_A^\infty \leq \frac{5}{4}$ for any algorithm $A \in \mathcal{AF}$ (see Theorem 21). This begged the broader question of lower bounds: how good can an arbitrary algorithm be? Based on two different lists, a partial answer was given in

the early 1980s by Csirik et al. [64]. They proved that if the list is presorted by decreasing item size, then $R_A^\infty \geq \frac{8}{7}$ for all algorithms A. Although this approach seemed to be very simple (the authors used only two different list types), no progress was made until very recently Balogh et al. [15] gave an improved lower bound:

Theorem 18 (Balogh et al. [15]) *If the list is presorted by decreasing item size, then $R_A^\infty \geq \frac{54}{47}$ for all algorithms A.*

4 Off-line Algorithms

An *off-line* algorithm has all items available for preprocessing, reordering, grouping, etc. before packing. It has been seen that most of the classical on-line algorithms achieve their worst-case ratio when the items are packed in increasing order of size (see, e.g., FF and BF), or if small and large items are merged (see, e.g., NF). Thus, one is led to expect improved behavior by a sorting of the items in decreasing order of size. Note that the $O(n \log n)$ sorting step makes the algorithm no longer linear time.

The review starts with results for approaches that sort the items before executing one of the on-line algorithms. Linear-time heuristics are then considered. The section is concluded by approximation schemes and by a discussion of the anomalous behavior that is exhibited by many bin packing algorithms, including both on-line and off-line algorithms.

4.1 Algorithms with Presorting

When the sorted list is packed according to the Next-Fit rule, one obtains the *Next-Fit Decreasing* (NFD) algorithm. This heuristic was investigated by Baker and Coffman, who proved by a weighting function argument that its APR is slightly better than that of FF and BF:

Theorem 19 (Baker and Coffman [13]) *If $\alpha \in (\frac{1}{r+1}, \frac{1}{r}]$ ($r \geq 1$), then $R_{\text{NFD}}^\infty(\alpha) = h_\infty(r)$.*

Packing the sorted list according to First-Fit or Best-Fit gives the algorithms *First-Fit Decreasing* (FFD) and *Best-Fit Decreasing* (BFD), with much better asymptotic worst-case performance.

Theorem 20 (Johnson et al. [129]) $R_{\text{FFD}}^\infty = R_{\text{BFD}}^\infty = \frac{11}{9}$.

The original proof was based on the weighting function technique, but subsequent proofs introduced dramatic changes; the giant case analysis made in 1973 by Johnson [126] was considerably shortened by Baker [12] in 1985, and in 1991, Yue

[183] presented the shortest proof known so far. In parallel, the additive constant was also improved; Johnson had proved that $\text{FFD}(L) \leq \frac{11}{9} \text{OPT}(L) + 4$, but Baker reduced the constant to 3, and Yue reduced it to 1. Then, in 1997, Li and Yue [146] further reduced the constant to $\frac{7}{9}$ and conjectured the tight value to be $\frac{5}{9}$. However, in 2007, Dósa [68] closed the issue by proving that the tight value is $\frac{6}{9}$.

The behavior of BFD for general α is not known, but that of FFD has been intensively investigated. Johnson et al. [129] analyzed several cases, showing that

$$R_{\text{FFD}}^{\infty}(\alpha) = \begin{cases} \frac{11}{9} & \text{if } \frac{1}{2} < \alpha \leq 1 \\ \frac{71}{60} & \text{if } \frac{8}{29} < \alpha \leq \frac{1}{2} \\ \frac{7}{6} & \text{if } \frac{1}{4} < \alpha \leq \frac{8}{29} \\ \frac{23}{20} & \text{if } \frac{1}{5} < \alpha \leq \frac{1}{4} \end{cases}$$

and conjecturing that, for any integer $m \geq 4$,

$$R_{\text{FFD}}^{\infty}\left(\frac{1}{m}\right) = F_m := 1 + \frac{1}{m+2} - \frac{2}{m(m+1)(m+2)}.$$

Twenty years later, Csirik [57] proved that the above conjecture is valid only for m even, and that, for m odd,

$$R_{\text{FFD}}^{\infty}\left(\frac{1}{m}\right) = G_m := 1 + \frac{1}{m+2} - \frac{1}{m(m+1)(m+2)}.$$

In the same year, the complete analysis of FFD for arbitrary values of $\alpha \leq \frac{1}{4}$ was published by Xu [178] (see also [179]), who showed that if m is even, then F_m is the correct APR for any α in $(\frac{1}{m+1}, \frac{1}{m}]$, while for m odd, the interval has to be divided into two parts, with

$$R_{\text{FFD}}^{\infty}(\alpha) = \begin{cases} F_m, & \text{if } \frac{1}{m+1} < \alpha \leq d_m \\ G_m, & \text{if } d_m < \alpha \leq \frac{1}{m} \end{cases}$$

where $d_m := (m+1)^2/(m^3 + 3m^2 + m + 1)$.

Recall that, after a presorting stage, all of the above algorithms belong to the Any-Fit class (see Sect. 3.3). Johnson showed that, after a presort in increasing order, Any-Fit algorithms do not perform well in the worst case; for example, their APRs must be at least $h_{\infty}(1)$ when $\alpha = 1$. But presorting in decreasing order is much more useful.

Theorem 21 (Johnson [126, 127]) *Any algorithm $A \in \mathcal{AF}$ operating on a list presorted in decreasing-size order must have*

$$\frac{11}{9} \leq R_A^{\infty}(1) \leq \frac{5}{4}$$

$$\frac{1}{m+2} - \frac{2}{m(m+1)(m+2)} \leq R_A^{\infty}(\alpha) \leq \frac{1}{m+2}, \text{ where } m = \lfloor \frac{1}{\alpha} \rfloor \text{ and } \alpha < 1.$$

For a long time, the FFD bound was the smallest proved APR. Johnson [126] made an interesting attempt to obtain a better APR. His *Most-k-Fit* (MF_k) algorithm takes elements from both ends of the sorted list, packing bins one at a time. At any step, after trying to place the largest unpacked element into the current bin, the algorithm attempts to fill up the remaining space in the bin using the smallest k (or fewer) as yet unpacked items. As soon as the available space becomes smaller than the smallest unpacked element, the algorithm starts packing a new bin. The algorithm has time complexity $O(n^k \log n)$, so it is practical only for small k . Johnson conjectured that $\lim_{k \rightarrow \infty} R_{\text{MF}_k}^\infty = \frac{10}{9}$, but almost 20 years later, the conjecture was contradicted by Friesen and Langston [90], who gave examples for which $R_{\text{MF}_k}^\infty \geq \frac{5}{4}$, $k \geq 2$.

Yao [180] devised the first improvement to FFD. He presented a complicated $O(n^{10} \log n)$ algorithm, called *Refined First-Fit Decreasing* (RFFD), with worst-case ratio $R_{\text{RFFD}}^\infty \leq \frac{11}{9} - 10^{-7}$. Following this result, further efforts were made to develop better off-line algorithms. Garey and Johnson [102] proposed the *Modified First-Fit Decreasing* (MFFD) algorithm. The main idea is to supplement FFD with an attempt to improve that part of the packing containing bins with items of sizes larger than $\frac{1}{2}$ by trying to pack in these bins pairs of items (to be called S items) with sizes in $(\frac{1}{6}, \frac{1}{3}]$. The non-FFD decisions of MFFD occur only during the packing of S items. At the time these items come up for packing, the bins currently containing a single item larger than $\frac{1}{2}$ are packed first, where possible, in decreasing-gap order as follows. In packing the next such bin, MFFD first checks whether there are two still unpacked S items that can fit into the bin; if not, MFFD finishes out the remaining packing just like FFD. Otherwise, the smallest available S item is packed first in the bin; the largest remaining available S item that fits with it is packed second. The running time of MFFD is not appreciably larger than that for FFD, but Garey and Johnson proved that

Theorem 22 (Garey and Johnson [102]) $R_{\text{MFFD}}^\infty = \frac{71}{60} = 1.18333 \dots$

Another modification of FFD was presented by Friesen and Langston [90]. Their *Best Two-Fit* (B2F) algorithm starts by filling one bin at a time, greedily; when no further element fits into the current bin, and the bin contains more than one element, an attempt is made to replace the smallest one by two unpacked elements with sizes at least $\frac{1}{6}$. When all the unpacked elements have sizes smaller than $\frac{1}{6}$, the standard FFD algorithm is applied. Friesen and Langston proved that $R_{\text{B2F}}^\infty = \frac{5}{4}$, which is worse than $\frac{11}{9}$. However, they further showed that a combined algorithm (CFB), which runs both B2F and FFD and takes the better packing, has an improved APR.

Theorem 23 (Friesen and Langston [90]) $1.16410 \dots = \frac{227}{195} \leq R_{\text{CFB}}^\infty \leq \frac{6}{5} = 1.2$.

Concerning the absolute worst-case ratio, Johnson et al. [129] had already conjectured that, if the number of bins in the optimal solution is more than 20, then the absolute worst-case ratio of FF is no more than 1.7. The first results were given by

Johnson et al. [129] and Simchi-Levi [169]. They showed that FF and BF have an absolute performance bound not greater than 1.75 and that FFD and BFD have an absolute performance ratio of 1.5. The latter is the best possible for the classical bin packing problem, unless $\mathcal{P} = \mathcal{NP}$. Xie and Liu [177] improved the bound for FF to 1.737. Zhang et al. [188] provided a linear-time bounded-space off-line approximation algorithm with an absolute worst-case ratio of 1.5 and a linear-time bounded-space on-line algorithm with an absolute worst-case ratio of 1.75. Berghammer and Reuter [24] gave a different linear-time algorithm with an absolute worst-case ratio of 1.5.

4.2 Linear Time, Randomization, and Other Approaches

The off-line algorithms analyzed so far have time complexity at least $O(n \log n)$. It is also interesting to see what can be accomplished in linear time – in particular, without sorting. The first such heuristic was constructed by Johnson [127]. His *Group-X-Fit Grouped* (GXFG) algorithm depends on the choice of a set of breakpoints X , defined by a sequence of real numbers $0 = x_0 < x_1 < \dots < x_p = 1$. For a given X , the algorithm partitions the items, according to their size, into at most p classes, and renames them in such a way that items of the same class are consecutive and classes are ordered by decreasing maximum size. The bins are also collected into p groups according to their *actual gap*, defined as the maximum x_j such that the current empty space in the bin is at least x_j . The items are packed using the Best-Fit rule with respect to the actual gaps. The algorithm can be implemented so as to require linear time and has the following APR.

Theorem 24 (Johnson [127]) *For all $m \geq 1$, if X contains $\frac{1}{m+2}, \frac{1}{m+1}$, and $\frac{1}{m}$, then $R_{\text{GXFG}}^{\infty}(\alpha) = \frac{m+2}{m+1}$ for all $\alpha \leq 1$ such that $m = \lfloor \frac{1}{\alpha} \rfloor$.*

For $\alpha = 1$, the above theorem gives $R_{\text{GXFG}}^{\infty} = \frac{3}{2}$, a bound subsequently improved by Martel. His algorithm, H_4 , uses a set $X = \{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}\}$, but it does not reorder the items. It instead inserts them into heaps and uses a linear search for the median-size item. The packing strategy makes use of an elaborate pairing technique.

Theorem 25 (Martel [154]) $R_{H_4}^{\infty} = \frac{4}{3}$.

Later, Békési et al. [23] applied the Martel idea to a set $X = \{\frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{3}{8}, \frac{1}{2}, \frac{2}{3}, \frac{4}{5}\}$ and improved the bound to $\frac{5}{4}$. Making experimental comparisons, they further showed that this linear-time algorithm is faster than the less complicated FFD rule even for small problem instances. They also mentioned that, by using more breakpoints, one can further improve the above result, but the resulting algorithms would be linear time with a constant term so large that they would not be useful in practice.

Table 4 Worst-case ratios of off-line algorithms. NA means “not analyzed”

Algorithm	Time	$R_A^\infty(1)$	$R_A^\infty(2)$	$R_A^\infty(3)$	$R_A^\infty(4)$
NFD	$O(n)$	1.691...	1.424...	1.302...	1.234
FFD	$O(n \log n)$	1.222...	1.183...	1.183...	1.15
BFD	$O(n \log n)$	1.222...	1.183...	1.183...	1.15
GXFD	$O(n)$	1.5	1.333...	1.25	1.2
MFFD	$O(n \log n)$	1.183...	1.183...	1.183...	1.15
H ₄	$O(n)$	1.333...	NA	NA	NA
H ₇	$O(n)$	1.25	NA	NA	NA
B2F	$O(n \log n)$	1.25	NA	NA	NA
CFB	$O(n \log n)$	1.2	$\leq 1.183\dots$	$\leq 1.183\dots$	≤ 1.15

Table 4 summarizes the tightest worst-case ratios of off-line algorithms (H_7 denotes the algorithm of Békési and Galambos [22]).

As a final note on fast off-line algorithms, the reader is referred to the work of Anderson et al. [4] for the implementation of approximation algorithms on parallel architectures. They show that, with $n / \log n$ processors in the EREW PRAM model, a packing can be obtained in parallel in $O(\log n)$ time which has the same asymptotic $\frac{11}{9}$ bound as FFD.

Caprara and Pferschy [34] proposed a simple (although non-polynomial) algorithm, discussed in Sect. 7.2, for which they proved an upper bound on the worst-case ratio of $4/3 + \ln 4/3 = 1.62102\dots$.

4.3 Asymptotic Approximation Schemes

In 1980, Yao [180] raised an interesting question: does there exist an $\varepsilon > 0$ such that every $O(n)$ -time algorithm A must satisfy the lower bound $R_A^\infty \geq 1 + \varepsilon$? Fernandez de la Vega and Lueker [84] answered the question in the negative by constructing an asymptotic linear approximation scheme for bin packing. In their important paper, LP (linear programming) relaxations were first introduced as a technique for devising bin packing approximation algorithms. (See Sect. 8.1 for an integer programming formulation of the bin packing problem.) In this section, their result is first discussed, and then some improvements proposed by Johnson and by Karmarkar and Karp are introduced. Further discussion can be found in [52, 116].

The main idea in [84] is the following. Given an ε , $0 < \varepsilon < 1$, define ε_1 so that $\frac{\varepsilon}{2} < \varepsilon_1 \leq \frac{\varepsilon}{\varepsilon+1}$. Instead of packing the given list L , the algorithm packs a concatenation of three lists $L_1 L_2 L_3$ determined as follows:

- L_1 contains all the elements of L with sizes smaller than ε_1 .
- L_2 is a list of $(m - 1)h$ dummy elements with “rounded” sizes, corresponding to the $(m - 1)h$ smallest elements of $L \setminus L_1$, where $m = \lceil \frac{4}{\varepsilon^2} \rceil$ and $h = \left\lfloor \frac{|L \setminus L_1|}{m} \right\rfloor$.

The elements of L_2 have only $m - 1$ different sizes, and, for each size s , the list has h elements; the sizes of the corresponding h elements in $L \setminus L_1$ are no greater than s .

- L_3 contains the remaining elements of $L \setminus L_1$, that is, those not having a corresponding rounded element in L_2 , hence $|L_3| \leq 2h - 1$.

By construction, for each group of h elements of L_2 having the same size s , there exists a distinct group of h elements of $L \setminus L_1$ having sizes at least s . It follows that $OPT(L_2) \leq OPT(L \setminus L_1)$.

It is first proved that for a given ε , one can construct, in linear time via an LP relaxation, a packing for the elements of L_2 that requires no more than $OPT(L \setminus L_1) + \frac{4}{\varepsilon}$ bins. The next step is to pack each element of L_3 into a separate bin, so the number of open bins at this point is bounded by

$$OPT(L \setminus L_1) + \frac{4}{\varepsilon} + 2h - 1 < OPT(L \setminus L_1) + \frac{4}{\varepsilon} + |L \setminus L_1| \frac{\varepsilon^2}{2} \leq OPT(L \setminus L_1)(1 + \varepsilon) + \frac{4}{\varepsilon}$$

(using the fact that $OPT(L \setminus L_1) \geq \frac{\varepsilon}{2}|L \setminus L_1|$).

Finally, the items of L_1 are added to the current packing, one bin at a time, using the Next-Fit rule. If no new bin is opened in this phase, the resulting packing has the desired performance. If at least one new bin is opened, this implies that all the bins, except possibly the last one, are filled to at least $(1 - \varepsilon_1)$, and hence the total number of bins is bounded by

$$\frac{OPT(L)}{1 - \varepsilon_1} + 1 \leq OPT(L)(1 + \varepsilon) + 1 \leq OPT(L)(1 + \varepsilon) + \frac{4}{\varepsilon}.$$

Combining this with an analysis of the time to pack L_2 , Fernandez de la Vega and Lueker proved the following result.

Theorem 26 (Fernandez de la Vega and Lueker [84]) *For any $\varepsilon > 0$, there exists a bin packing algorithm A with asymptotic worst-case ratio $R_A^\infty \leq 1 + \varepsilon$, requiring time $C_\varepsilon + Cn \log \frac{1}{\varepsilon}$, where C_ε depends only on ε and C is an absolute constant.*

Although the time complexity of the above approximation scheme is polynomial in the number of elements, it is exponential in $\frac{1}{\varepsilon}$.

The first improvement was given by Johnson [128], who observed that, if ε is allowed to grow suitably slowly with $OPT(L)$, one can use the above approach to construct a polynomial-time algorithm A such that $A(L) \leq OPT(L) + o(OPT(L))$; incorporating a scheme suggested by Karmarkar and Karp, he achieved $A(L) \leq OPT(L) + O(OPT(L)^{1-\delta})$, where δ is a positive constant. Thus, as a corollary to [Theorem 26](#), there exists a polynomial-time approximation algorithm for bin packing with an APR equal to 1.

Karmarkar and Karp [132] made further improvements and produced an asymptotic fully polynomial-time approximation scheme. They brought several new

techniques into play. The ellipsoid method applied to an LP relaxation drove the approach; a feasible packing was determined by an extension of the approach of Fernandez de la Vega and Lueker. A function T , estimated below, describes the time complexity of the LP problem in terms of properties of the problem instance. Let $c(L)$ denote the total size of the items in L and recall that k denotes the number, perhaps infinite, of possible item sizes. The main results are several different forms of asymptotic optimality and their associated running-time bounds. Recalling that the absolute error for an algorithm A is simply $A(L) - OPT(L)$, one has the following.

Theorem 27 (Karmarkar and Karp [132]) *For each row in the table below, there is an approximation algorithm with the given running-time and absolute-error bounds; the approximation parameters in the last two rows satisfy $\alpha \in [0, 1]$ and $\varepsilon > 0$:*

Running-time bound	Absolute-error bound
$O(T_n(c(L)))$	$O(\log^2 OPT(L))$
$O(T_n(k))$	$O(\log^2 k)$
$O(T_n(c(L)^{1-\alpha}))$	$O(OPT(L)^\alpha)$
$O(T_n(\varepsilon^{-2}))$	$\varepsilon OPT(L) + O(\varepsilon^{-2})$

where

$$T_n(v) = O(v^8 \log v \log^2 n + v^4 n \log v \log n).$$

The bound on $T_n(v)$ is not especially attractive, so this approach as it stands is not likely to be very practical, although the authors mention that a mixture of their technique with a column generation method [105, 106] may be very efficient in practice.

Taking a complementary approach, Hochbaum and Shmoys [118] (see also [117]) define an ε -dual approximation algorithm for bin packing, denoted in the following by M_ε . For a given $\varepsilon > 0$, M_ε finds in polynomial time a packing of L in $OPT(L)$ bins, each of capacity $C = 1 + \varepsilon$. A primary objective of M_ε is the approximation scheme for the capacity minimization problem discussed in Sect. 6.1, but such algorithms also find use in bin packing settings where precise bin capacities vary or are unknown.

The above dual approach can be likened to a search for near-feasible, optimal solutions rather than feasible, near-optimal solutions. The construction of M_ε exploits a reduction of the problem to bin packing with a finite number of item sizes and hence the integer program formulation in Sect. 5.2. A general discussion of the details has been given by Hochbaum [116], who describes a version of M_ε that requires only linear running time (with constants depending on ε).

4.4 Anomalies

For a given algorithm A , one usually expects that if a list is made shorter, or its items made smaller, then the number of bins needed by A to pack the list could not increase, and if the reductions were large enough, then the number of bins required would actually decrease. This is certainly true for optimal algorithms, but as this section shows, it is not the case for many of the better on-line and off-line approximation algorithms. This anomalous behavior can explain the difficulty in analyzing algorithms; with such behavior, inductive arguments cannot normally be expected to work. The following instances illustrate bin packing anomalies. Define

$$\begin{aligned} L &= (0.7, 0.68, 0.5399, 0.3201, 0.15, 0.14, 0.08, 0.08, 0.08, 0.08, 0.08, 0.07) \\ L' &= (0.7, 0.68, 0.5399, 0.32, 0.15, 0.14, 0.08, 0.08, 0.08, 0.08, 0.08, 0.07) \end{aligned}$$

and note that the two instances differ only in the fourth element, which is slightly smaller in L' . BF and BFD produce the same packing of L :

$$\begin{aligned} c(B_1) &= 0.7 + 0.15 + 0.08 + 0.07 \\ c(B_2) &= 0.68 + 0.08 + 0.08 + 0.08 \\ c(B_3) &= 0.5399 + 0.3201 + 0.14 \end{aligned}$$

BF and BFD also produce the same packing of L' , which is larger by one bin:

$$\begin{aligned} c(B_1) &= 0.7 + 0.15 + 0.14 \\ c(B_2) &= 0.68 + 0.32 \\ c(B_3) &= 0.5399 + 0.08 + 0.08 + 0.08 + 0.08 + 0.08 \\ c(B_4) &= 0.07 \end{aligned}$$

A list $L_{(n)} = (a_1, a_2, \dots, a_n)$ *dominates* a list $L'_{(m)} = (a'_1, a'_2, \dots, a'_m)$ if $n \geq m$ and the size of each element a_i is no less than the size of the corresponding element a'_i . An algorithm A is *monotonic* if $A(L) \geq A(L')$ whenever L dominates L' . An *anomalous* algorithm is one that is not monotonic.

Graham [110] and Johnson [126] observed that algorithms FF and FFD are anomalous. However, the current insights into the anomalous behavior of bin packing algorithms are due mainly to Murgolo. In the notation of Sect. 3.4 (with W meaning Worst-Fit), he proved

Theorem 28 (Murgolo [156]) *Algorithms NF and NF₂ are monotonic, but each of BF, BFD, WF, WFD, ABF₂, AWF₂, and AFF_k with k ≥ 3 is anomalous.*

Murgolo also obtained upper and lower bounds on the behavior of anomalous algorithms. An algorithm is called *conservative* if it never opens a new bin unless

the current element cannot fit into an open bin. For conservative algorithms, the following bound applies.

Theorem 29 (Murgolo [156]) *If L' dominates L and A is conservative, then $A(L') \leq 2A(L)$. In addition, if $A \in \{FFD, BFD\}$, then $A(L') \leq \frac{11}{9}A(L) + 4$.*

He also proved the following complementary results:

Theorem 30 (Murgolo [156]) *There exist arbitrarily long lists L_i and L'_i , with L'_i dominating L_i , ($i = 1, 2, 3$), such that*

$$\begin{aligned} \text{if } A \in \{\text{BF}, \text{BFD}\} & \text{ then } A(L_1) \geq \frac{43}{42}A(L'_1), \\ \text{if } A \in \{\text{FF}\} & \text{ then } A(L_2) \geq \frac{76}{75}A(L'_2), \\ \text{if } A \in \{\text{WF}, \text{WFD}\} & \text{ then } A(L_3) \geq \frac{16}{15}A(L'_3). \end{aligned}$$

The following sections discuss problems in which special types of lists, alternative objective functions, or any of various constraints on items or packings are considered. It will be seen that in some cases the problem becomes easier, in the sense of approximability, while in others it becomes harder. Similarly, adaptations of classical heuristics give more or less attractive results depending on the new problem.

5 Variations on Bin Sizes

5.1 Variable-Sized Bins

Consider the case where one is given different bin types, B_1, B_2, \dots, B_k with sizes $1 = s(B_1) > s(B_2) > \dots > s(B_k)$. For each type, an unlimited number of bins are available, and the aim is to pack a list L of elements a_i , with $s_i \in (0, 1]$, into a subset of bins having the smallest total size. Let $s(A, L)$ denote the total size of the bins used by algorithm A to pack the items of L . Then

$$R_A^\infty = \lim_{k \rightarrow +\infty} \sup \left\{ \frac{s(A, L)}{s(OPT, L)} : s(OPT, L) \geq k \right\}.$$

The problem has practical importance in many of the classical bin packing applications, for example, cutting stock problems, the assignment of commercials to variable size breaks in television or radio broadcasting, and memory allocation for computer operating systems.

At first glance, one might expect variable-sized bin packing to be harder than the classical problem. However, this need not be true in general. For example, if there are bin types for all sizes between $\frac{1}{2}$ and 1, then packing large elements with sizes at

least $\frac{1}{2}$ can be done without wasted space. Similarly, smaller elements can be packed together without any waste, so long as their sum is at least $\frac{1}{2}$.

For general sets of bin sizes, an on-line algorithm must select the bin size for packing the current element whenever a new bin is opened. Friesen and Langston [89] proposed an on-line algorithm based on the Next-Fit rule, which always selects the largest bin size (*Next-Fit, using Largest possible bins*, NFL), and proved that its APR is 2. Kinnersley and Langston [138] showed that the same APR is obtained by adopting the First-Fit rule, both for an algorithm that uses the largest possible bins and for an algorithm that uses the smallest bins. Burkard and Zhang [33] pointed out that this APR characterizes a large class of algorithms that use one of the above bin-opening rules.

Kinnersley and Langston [138] analyzed other fast on-line algorithms. They proposed a scheme based on a user-specified *fill factor*, $f \geq \frac{1}{2}$, and proved that this strategy guarantees an APR smaller than $\frac{3}{2} + \frac{f}{2} \leq 1.75$. Zhang [185] proved that with $f = \frac{1}{2}$, the APR is $\frac{17}{10}$.

Csirik [56] investigated an algorithm based on the Harmonic-Fit strategy (*Variable Harmonic-Fit*, VH) and showed that its APR is at most $h_\infty(1) \approx 1.69103$. For special collections of bin types, the algorithm may perform better; for example, Csirik proved that if there are only two types of bins, with $s(B_1) = 1$ and $s(B_2) = \frac{7}{10}$, then $R_{\text{VH}}^\infty = \frac{7}{5}$.

This result is very interesting since the bound is smaller than the 1.54 on-line lower bound of Balogh et al. [15] for the classical problem (see Sect. 3.6), and implies that, for certain sets of two or more bin sizes, on-line algorithms can behave better than those restricted to a single bin size.

Csirik's VH algorithm suffers from the same “illness” as the classical HF_k algorithm (see Sect. 3.4); it reaches the $h_\infty(1)$ bound only for a very large number of open bins. More precisely, let $M > 1$ be a positive integer, and suppose that the algorithm can use l different bin types. Let $M_j = \lceil M s(B_j) \rceil$ ($j = 1, \dots, l$), and denote by VH_M the resulting VH algorithm, which is a k -bounded-space algorithm with $k = \sum_{j=1}^l M_j - l + 1$. If $M_l \leq 5$ then $R_{\text{VH}_M}^\infty \geq \frac{17}{10}$.

This raised further questions. If there are only two different bin types, which combination of sizes produces the smallest APR? What lower bounds depending on bin sizes can be proved? What can be said about the problem with at least three (or an arbitrary number of) bin sizes?

Some of the above questions were answered by Seiden [162], who revisited the VH algorithm. For the case where there are two bin sizes, he developed an algorithm that provides a lower bound for any fixed α . Specifically, he proved the following result:

$$1.37530 < \frac{78,392,621}{57,000,000} \leq \inf_{\alpha \in (0,1]} R_{OPT}^\infty(\alpha) \leq \frac{395,101,163}{287,280,000} < 1.37532.$$

He additionally proved that the upper bound $h_\infty(1) \approx 1.69103$, given in [56], is tight and that algorithm VH is optimal among bounded-space algorithms.

To obtain this result, Seiden used the weighting function technique and developed “twin” mathematical programs, which have the same sets of conditions but different objective functions. He used the mathematical programs to get lower and upper bounds for the asymptotic performance ratio of VH. He proved the asymptotic optimality of VH, although the precise value could not be computed.

The third question remained open even in the case of three different sizes.

Burkard and Zhang [33] investigated other bounded-space algorithms for variable-sized bin packing. They distinguished three different components: the opening rule, packing rule, and closing rule. The opening rule was fixed as follows. If the current item has size $s_i > \frac{1}{2}$ and there exist bin sizes smaller than 1 that can accommodate a_i , then the rule opens the smallest such bin; otherwise, it opens a bin of size 1. The packing rules analyzed were First-Fit (the F rule) and Best-Fit (the B rule). The closing rule closes a bin of size less than 1, if one exists; otherwise, it closes either the lowest indexed bin (the first-bin or F rule) or the most nearly full bin (the best-bin or B rule). With this terminology, VXY_k denotes the k -bounded algorithm that incorporates packing rule X and closing rule Y . Burkard and Zhang showed that, among the four resulting algorithms (VFF_k , VFB_k , VBF_k , and VBB_k), VBB_k is the best and that the following holds.

Theorem 31 (Burkard and Zhang [33]) $R_{VBB_k}^\infty = \frac{17}{10}$. Moreover, $R_{VH_M}^\infty \geq R_{VBB_k}^\infty$ if and only if $M_l < 7$, with $k \geq 6l + 1$.

The authors proved the theorem by a weighting function technique. They also mentioned that, with a small modification in the weighting function, they could prove that $R_{VFB_k}^\infty = \frac{17}{10} + \frac{3}{10(k-1)}$ for any $k \geq 2$. So, one of the more interesting questions remains open: does there exist an on-line algorithm with an APR strictly less than $h_\infty(1)$ for all collections of bin sizes?

For the off-line case, few results have been proved. Friesen and Langston [89] presented two algorithms based on the First-Fit Decreasing strategy. The first one, *FFD using Largest bins and Repack* (FFDLR), begins by packing the presorted elements into the largest bins, then repacks the contents of each open bin into the smallest possible empty bin. The second algorithm, *FFD using Largest bins and Shift* (FFDLS), improves on the first phase of FFDLR: whenever the bin (say of type B_j) where the current item has been packed contains an item of size at least $\frac{1}{3}$, the contents of the bin are shifted, if possible, to the smallest empty bin (say of type B_h) such that $s(B_h) \geq \tilde{c} \geq \frac{3}{4}s(B_h)$, where \tilde{c} denotes the total item size packed into the current bin. Friesen and Langston proved that $R_{FFDLR}^\infty = \frac{3}{2}$ and $R_{FFDLS}^\infty = \frac{4}{3}$.

Seiden et al. [164] made great strides by improving the upper bound and giving the first – and to this very day the only – general lower bound for the case with two bin sizes. First, they introduced a parameter $\mu \in (\frac{1}{3}, \frac{1}{2})$. They gave an algorithm $VH1(\mu)$ for all $\alpha \in (0, 1)$, and an algorithm $VH2(\mu)$, only defined for $\alpha > \max\{\frac{1}{2(1-\mu)}, \frac{1}{3\mu}\}$. Both algorithms are combinations of the Harmonic and Refined Harmonic algorithms. The upper bound was achieved by optimizing μ over each choice of α , and the authors chose the best value among $VH1(\mu)$, $VH2(\mu)$, and

VH. The resulting upper bound is at most $\frac{373}{228} < 1.63597$ for all α . Note that this is the asymptotic performance ratio of the RHF algorithm in the classic bin packing context. The lower bound result is summarized by the following theorem.

Theorem 32 (Seiden et al. [164]) *Any on-line algorithm for the variable-sized bin packing problem with two bin sizes has asymptotic performance ratio at least $\frac{495,176,908,800}{370,749,511,199} > 1.33561$.*

The result is quite good since the largest gap between the two bounds is 0.18193 for $\alpha = 0.9071$, and the smallest gap is 0.003371 for $\alpha = 0.6667$.

Murgolo [157] proposed an efficient approximation scheme. He first gave an algorithm with a running time linear in the number of items but exponential in $\frac{1}{\varepsilon}$ and the number of bin sizes. Then, following the ideas in Karmarkar and Karp [132], he solved a linear programming formulation of the problem by the ellipsoid method, thus obtaining a fully polynomial-time approximation scheme.

Zhang [186] considered a variant of the on-line version of the problem, in which item-size information is known in advance, but no information on bin sizes is available, except that each bin size is known to be no less than the size of the largest item. Bins arrive one at a time, and one has to decide which items to pack into the current bin. Zhang proved that the analogues of the classical NF, FF, NFD, and FFD algorithms all have an APR equal to 2 and left as an open question whether one can devise an algorithm with an APR better than 2.

Kang and Park [130] investigated a problem where there are m different bin sizes, and the total cost of a unit of each bin size does not increase with the bin size, that is,

$$\frac{c_{i_1}}{s(B_{i_1})} \leq \frac{c_{i_2}}{s(B_{i_2})}, \quad \text{if } 1 \leq i_1 < i_2 \leq m.$$

They analyzed two algorithms, *Iterative First-Fit Decreasing* (IFFD) and *Iterative Best-Fit Decreasing* (IBFD), for three cases. If both the bin and item sizes are divisible, then both algorithms are optimal. If only the bin sizes are divisible, then $R_{\text{IFFD}} = R_{\text{IBFD}} = \frac{11}{9}$, whereas for the general (non-divisible) case, $R_{\text{IFFD}} = R_{\text{IBFD}} = \frac{3}{2}$. In addition, they proved that an algorithm by Coffman et al. [51] for the variable-sized bin case does not provide the optimal solution.

The case of weak divisibility has not been treated.

5.2 Resource Augmentation

The idea of *resource augmentation* was introduced by Csirik and Woeginger [63]. The motivation is that the worst-case examples are “pathological,” that is, if for a unit-capacity bin one considers items with size $\frac{1}{2} + \varepsilon$ with a sufficiently small ε , then only one item per bin can be packed. If the size of the bins is slightly increased, then two items can be packed in a bin, thus sparing a lot of capacity. (This is also called *bin stretching*.)

In this problem, an algorithm A has to pack a list L of elements in $(0, 1]$ into a minimal number of bins of capacity $C \geq 1$. Denote by $A_C(L)$ the number of bins used by A for packing L into bins of capacity C . Then the APR is defined as follows:

$$R_C^\infty = \lim_{OPT_1(L) \rightarrow \infty} \sup_L A_C(L),$$

where $OPT_1(L)$ is the optimal (off-line) packing of L into unit-capacity bins. Csirik and Woeginger considered on-line bounded-space algorithms and gave a complete analysis. For each C , a sequence $T(C) = \langle t_1, t_2, \dots, \rangle$ of positive integers is defined:

$$t_1 = \lfloor 1 + C \rfloor \quad \text{and} \quad r_1 = \frac{1}{C} - \frac{1}{t_1}$$

and, for $i = 2, 3, \dots$,

$$t_i = \left\lfloor 1 + \frac{1}{r_{i-1}} \right\rfloor \quad \text{and} \quad r_i = r_{i-1} - \frac{1}{t_i}.$$

It is clear that this is a simple transformation of the Sylvester sequence, so

$$\frac{1}{C} = \sum_{i=1}^{\infty} \frac{1}{t_i}.$$

They defined a new function

$$\rho(C) = \sum_{i=1}^{\infty} \frac{1}{t_i - 1}$$

for which it is easy to prove that $\rho(C)$ is strictly decreasing on $[1, \infty)$, and $\rho(1) = h_\infty(1) \approx 1.69103$, $\rho(2) \approx 0.69103$. Furthermore, the following inequalities hold:

$$\frac{1}{m} \leq \rho(m) \leq \frac{1}{m-1} \text{ for integer } m \geq 2.$$

For a given $l \geq 3$, they defined t_l intervals: $\mathcal{I}_j = (\frac{C}{j+1}, \frac{C}{j}]$ for $j = 1, \dots, t_l - 1$, and $\mathcal{I}_{t_l} = (0, \frac{C}{t_l}]$. For these intervals, they constructed a Harmonic-Fit type algorithm. There is one active bin for every interval \mathcal{I}_j , and all items from the interval $\mathcal{I}_j \cap (0, 1]$ are packed into it using a Next-Fit rule. Using a weighting function (which is an adaptation of the one defined for the classical Harmonic-Fit algorithm), they proved the following theorem.

Theorem 33 (Csirik and Woeginger [63]) *For every bin size $C \geq 1$, there exist on-line, bounded-space bin packing algorithms with APRs arbitrarily close to $\rho(C)$.*

For every bin size $C \geq 1$, the bound $\rho(C)$ cannot be beaten by an on-line bounded-space bin packing algorithm.

Azar and Regev [7] presented two on-line algorithms, each guaranteeing a worst-case performance of $5/3$ for any number of bins. They also combined the algorithms to obtain an algorithm with a tight worst-case performance of $13/8 = 1.625$. This turns out to be an interesting result, as the best lower bound for any algorithm is equal to $4/3$ for $m \geq 2$.

On-line bounded-space bin packing with limited repacking was considered by Epstein and Kleiman [75], who extended to this case the ideas of Galambos and Woeginger [95]. They defined a semi-on-line algorithm, *Resource Augmented REP₃(C)* (*RAR₃(C)*), which is allowed to repack the elements within three active bins, and proved the following theorem.

Theorem 34 (Epstein and Kleiman [75]) *For every bin size $C \geq 1$, and for any on-line bounded-space bin packing algorithm A that allows repacking within k active bins, $R_A^\infty(C) \geq \rho(C)$. Algorithm RAR₃(C) has the best possible APR.*

Epstein and van Stee [80] extended the study to on-line algorithms, investigating both lower and upper bounds. For the upper bound, they started with an analysis of the harmonic type algorithms. Recall that by introducing two new interval endpoints, $\Delta > 1/2$ and $1 - \Delta$, one can combine one small element with an item in the interval $(1/2, \Delta]$. Also recall that in this case, only a fraction of the bins belonging to interval i can be used to reserve space for a (possibly later arriving) element from the interval $(1/2, \Delta]$. This fraction is denoted as α^i . (The values of α^i 's obviously depend on the number of intervals, and they are the crucial point of any such algorithm.) On this basis, they developed four algorithms, which are applied to the intervals $[1, 6/5]$, $[6/5, 4/3]$, $[4/3, 12/7]$, and $[12/7, 2]$, respectively.

An analytical solution was given only for the *Tiny Modified-Fit* algorithm, which was applied to interval $[12/7, 2]$. To examine the worst-case behavior of the other algorithms, they defined two weighting functions and constructed a linear program $P(f)$. By applying it to the cost function, they could use the method introduced by Seiden [163], thus obtaining upper bounds for the asymptotic behavior of the algorithms. The cost functions were not given explicitly, but program P was solved for many C values, and a diagram was given to show the upper bounds.

Concerning lower bounds, Epstein and van Stee [80] followed the argument laid down by van Vliet [171]. In this case too, an LP was generated for many values of C , and appropriate sublists were obtained through specialized greedy algorithms.

Chan et al. [37] further extended the study to dynamic bin packing. They considered the case where the items may depart at any time and where moving items among the open bins is disallowed. They first investigated Any-Fit algorithms, proving the following theorem.

Theorem 35 (Chan et al. [37]) Consider the resource augmentation problem with bin capacity C , and let A be an Any-Fit algorithm. Then A can achieve 1-competitiveness if and only if $C = 2$.

They also gave a new lower bound, as a function of C , for on-line algorithms. Let m be the largest integer such that $C - \frac{1}{m} < 1$, and define, for any positive integer $1 \leq i \leq m$,

$$\alpha(1) = m!(m-1)! \quad \beta(i) = \sum_{j=1}^i \alpha(j) \quad \alpha(i) = \frac{\beta(i-1)}{(m-i+2)(m-i+1)}.$$

Theorem 36 (Chan et al. [37]) Let A be an on-line algorithm. Then

$$R_A^\infty(C) \geq \max \left\{ \frac{\beta(m)}{m!(m-1)!}, \frac{2}{C} \right\}.$$

They also analyzed certain classical algorithms:

Theorem 37 (Chan et al. [37])

$$R_{\text{FF}}^\infty(C) \leq \min \left\{ \frac{2C+1}{2C-1}, \frac{5-C}{2C-1}, \frac{C^2+3}{C(2C-1)} \right\},$$

$$R_{\text{BF}}^\infty(C) = \frac{1}{C-1}$$

and

$$\min \left\{ \frac{2C+2}{2C}, \frac{C^2-8C+20}{C(4C)} \right\} \leq R_{\text{WF}}^\infty(C) \leq \frac{4}{C^2}. \quad (4)$$

Very recently, Boyar et al. [31] gave a complete APR analysis of algorithms WF and NF for the resource augmentation case. They showed that WF is strictly better than NF in the interval $1 < C < 2$ but has the same APR for all $C \geq 2$:

Theorem 38 (Boyar et al. [31]) Let $t_C = \lfloor \frac{1}{C-1} \rfloor$. Then

$$R_{\text{WF}}^\infty(C) = \begin{cases} \frac{2C}{3C-2}, & \text{if } C \in [1, 2] \\ \frac{1}{C-1}, & \text{if } C \in [2, \infty) \end{cases}$$

and

$$R_{\text{NF}}^\infty(C) = \frac{2t_C^2 C - 2t_C^2 - 4t_C + 2 + 2Ct_C}{t_C^2 C + 2Ct_C - t_C^2 - 3t_C - 2 + 2C}.$$

6 Dual Versions

6.1 Minimizing the Bin Capacity

Suppose that the number of bins is fixed, and let the objective be a smallest C such that a given list L of items can be packed in m bins of capacity C . This dual of bin packing actually predates bin packing and is known as multiprocessor or parallel-machine scheduling. It is a central problem of scheduling theory and is surveyed in depth elsewhere (see, e.g., Lawler et al. [141] and Hall [114]), so only relevant results and research directions will be highlighted. The problem is referred to as $P||C_{\max}$ in scheduling notation.

Optimal and heuristic capacities normally differ by at most a maximum item size, so absolute worst-case ratios rather than asymptotic ones are considered. The first analysis of approximation algorithms was presented by Graham [108, 109], who studied the worst-case performance of various algorithms. For the on-line case, he investigated the Worst-Fit algorithm (better known as the *List Scheduling*, LS algorithm), which initially opens m empty bins and then iteratively packs the current item into the bin having the minimum current content (breaking ties in favor of the lowest bin index). Graham proved that $R_{\text{LS}} = 2 - \frac{1}{m}$, a bound that for many years was thought to be best among on-line algorithms. Galambos and Woeginger were first to prove that the bound could be improved. Let $A(L, m)$ denote the bin capacity needed under A for m bins, and define

$$R_A(m) = \sup_L \left\{ \frac{A(L, m)}{\text{OPT}(L, m)} \right\} \quad \text{and} \quad R(m) = \inf_A R_A(m).$$

Theorem 39 (Galambos and Woeginger [94]) *There exists a sequence of nonnegative numbers $\varepsilon_1, \varepsilon_2, \dots$, with $\varepsilon_m > 0$, $m \geq 4$ and $\varepsilon_m \rightarrow 0$ as $m \rightarrow \infty$, such that $R(m) \leq 2 - \frac{1}{m} - \varepsilon_m$.*

This result encouraged further research; important milestones on the way to the current position on the problem are as follows. Bartal et al. [19] presented an algorithm with a worst-case ratio $2 - \frac{1}{70} = 1.98571\dots$ for all $m \geq 70$. Earlier, Faigle et al. [83] proved lower bounds for different values of m . They pointed out that for $m \leq 3$, the LS algorithm is optimal among on-line algorithms, and they proved a $1 + \frac{1}{\sqrt{2}} = 1.70710\dots$ lower bound for $m \geq 4$. Then these bounds were improved, as shown below.

Theorem 40 (Chen et al. [41]) *For all $m = 80 + 8k$ with k a nonnegative integer, $R(m) \geq 1.83193$.*

Theorem 41 (Bartal et al. [20]) *For all $m \geq 3,454$, $R(m) \geq 1.8370$.*

For some time, the following result gave the best upper bound for $R(m)$ and was a generalization of the methods in [20].

Theorem 42 (Karger et al. [131]) $R(m) \leq 1.945$ for all m .

A tighter result was found by Albers [2], who proved the next two bounds.

Theorem 43 (Albers [2]) $R(m) \leq 1.923$ for all m .

Just as previous authors, Albers recognized that, during the packing process, a good algorithm must try to avoid packings in which the content of each of the bins is about the same, for in such cases many small items followed by a large item can create a poor worst-case ratio. Albers' new algorithm attempts to maintain throughout the packing process $\lfloor \frac{m}{2} \rfloor$ bins with small total content and $\lceil \frac{m}{2} \rceil$ bins with large total content. The precise aim is always to have a total content in the small-content bins that is at most γ times the total content in the large-content bins. With an optimal choice of γ , one obtains a worst-case ratio of at most 1.923.

For her new lower bound, Albers proved

Theorem 44 (Albers [2]) $R(m) \geq 1.852$ for all $m \geq 80$.

Attempts to further reduce the general gap of about .07 continue. For the special case $m = 4$, it is proved in Chen et al. [41] that $1.73101 \leq R(4) \leq \frac{52}{30} = 1.7333\dots$, but the exact value of $R(4)$ remains an open problem. Another enticing open problem is: does $R(m) < R(m + 1)$ hold for any m ?

The off-line case is now considered. By preordering the elements according to nonincreasing size and applying the LS algorithm, one obtains the so-called *Largest Processing Time* (LPT) algorithm for $P \parallel C_{\max}$. Graham [109] proved that $R_{\text{LPT}} = \frac{4}{3} - \frac{1}{3m}$. The *Multifit* algorithm by Coffman et al. [47] adopts a different strategy, leading to better worst-case behavior. The algorithm determines, by binary search over an interval with crude but easily established lower and upper bounds, the smallest capacity C such that the FFD algorithm can pack all the elements into m bins. Coffman, Garey, and Johnson proved that if k binary search iterations are performed, then the algorithm, denoted by MF_k , requires $O(n \log n + kn \log m)$ time and has an absolute worst-case ratio satisfying

$$R_{\text{MF}_k} \leq 1.22 + 2^{-k}.$$

Friesen [87] improved the bound uniform in k to 1.2, but Yue [182] later settled the question by proving that this bound is $\frac{13}{11}$. Friesen and Langston [89] developed a different version of the Multifit algorithm, proving that its worst-case ratio is bounded by $\frac{72}{61} + 2^{-k}$.

A different off-line approach was proposed by Graham [109]. His algorithm Z_k optimally packs the $\min\{k, n\}$ largest elements and completes the solution by packing the remaining elements, if any, according to the LS algorithm. Graham

proved that

$$R_{Z_k} \leq 1 + \frac{1 - \frac{1}{m}}{1 + \left\lfloor \frac{k}{m} \right\rfloor}$$

and that the bound is tight when m divides k . Algorithm Z_k implicitly defines an approximation scheme. By selecting $k = k_\varepsilon = \lceil \frac{m(1-\varepsilon)-1}{\varepsilon} \rceil$, one obtains $R_{Z_{k_\varepsilon}} \leq 1 + \varepsilon$. The running time is $O(n \log n + m^{m(1-\varepsilon)/\varepsilon})$, and so the method is unlikely to be practical. The result was improved by Sahni [161], in the sense that the functional dependence on m and ε was reduced substantially. His algorithm A_ε has running time $O(n(\frac{n^2}{\varepsilon})^{m-1})$ and satisfies $R_{A_\varepsilon} \leq 1 + \varepsilon$; hence it is a fully polynomial-time approximation scheme for any fixed m .

For m even moderately large, the approach of Hochbaum and Shmoys [118] offers major improvements. They developed an ε -approximate algorithm that removed the running-time dependence on m and reduced the dependence on n . The dependence on $\frac{1}{\varepsilon}$ is exponential, which is to be expected, since a polynomial dependence would imply $\mathcal{P} = \mathcal{NP}$. Their algorithm is based on the binary search of an interval (like MF_k), but it uses the ε -dual approximation algorithm of Sect. 4.3 at each iteration. Hochbaum and Shmoys show that, after k steps of the binary search, the bin capacity is at most $(1 + \varepsilon)(1 + 2^{-k})$ times optimal. From this fact and the properties of M_ε given in Sect. 4.3, one obtains a linear-time approximation scheme for the capacity minimization problem. (See also Hochbaum's [116] for a more extensive discussion of this interesting technique.) The Hochbaum and Shmoys result was generalized by Alon et al. [3], who gave conditions under which a number of scheduling problems on parallel machines admit a polynomial-time approximation scheme.

6.2 Maximizing the Number of Items Packed

This section considers another variant in which the number of bins is fixed; the objective is now to pack a maximum cardinality subset of a given list $L_{(n)}$. The problem arises in computing applications, when one wants to maximize the number of records stored in one or more levels of a memory hierarchy or to maximize the number of tasks performed on multiple processors within a given time interval.

The classical bin packing notation is extended in an obvious way by defining

$$\hat{R}_A(m, n) = \min \left\{ \frac{A(L_{(n)}, m)}{OPT(L_{(n)}, m)} \right\}$$

so that the APR is defined by

$$\hat{R}_A^\infty(m) = \liminf_{n \rightarrow \infty} \hat{R}_A(m, n)$$

and has values less than or equal to 1.

First consider off-line algorithms. The problem was first studied by Coffman et al. [48], who adapted the *First-Fit Increasing* (FFI) heuristic. The items are preordered according to nondecreasing size, the m bins are opened, and the current item is packed into the lowest indexed bin into which it fits, stopping the process as soon as an item is encountered that does not fit into any bin. It is proved in [48] that $\hat{R}_{\text{FFI}}^\infty(m) = \frac{3}{4}$ for all m .

The *Iterated First-Fit Decreasing* (IFFD) algorithm was investigated by Coffman and Leung [45]. The algorithm sorts the items by nonincreasing size and iteratively tries to pack all the items into the m bins following the First-Fit rule. Whenever the current item cannot be packed, the process is stopped, the largest item is removed from the list, and a new FFD iteration is performed on the shortened list. The search terminates as soon as FFD is able to pack all the items of the current list. Coffman and Leung showed that IFFD performs at least as well as FFI on every list and that $\frac{6}{7} \leq \hat{R}_{\text{IFFD}}^\infty(m) \leq \frac{7}{8}$. The time complexity of IFFD is $O(n \log n + mn \log m)$, but a tight APR is not known.

Lower bounds on the APR are now considered. If an algorithm A packs items a_1, \dots, a_t from a list $L = (a_1, \dots, a_n)$ so that $s_i > 1 - s(B_j)$ for any j and for any $i > t$ (i.e., no unpacked item fits into any bin), then it is said to be a *prefix algorithm*. Note that both FFI and IFFD are prefix algorithms.

Theorem 45 (Coffman et al. [48]) *Let A be a prefix algorithm and let $k = \min_{1 \leq j \leq m} \{|B_j|\}$ be the least number of items packed into any bin. Then*

$$\hat{R}_A^\infty(m) \geq \frac{mk}{mk + m - 1}.$$

Moreover, the bound is achievable for all $m \geq 1$ and $k \geq 1$.

The case of divisible item sizes was investigated by Coffman et al. [51]. They proved that both FFI and IFFD produce optimal packings if (L, C) is strongly divisible. IFFD remains optimal even if (L, C) is weakly divisible, but FFI does not.

The case of variable-sized bins was first analyzed by Langston [140], who proved that if the bins are rearranged by nonincreasing size, then $\hat{R}_{\text{FFI}}^\infty(m) = \frac{1}{2}$ and $\frac{2}{3} \leq \hat{R}_{\text{IFFD}}^\infty(m) \leq \frac{8}{11}$ for all m . Friesen and Kuhl [88] gave a new efficient algorithm, which is a hybrid of two algorithms defined earlier: *First-Fit Decreasing* (FFD) and *Best-Two-Fit* (B2F). The hybrid is iterative: it attempts to pack smaller and smaller suffixes of its list of items until it succeeds in packing the entire suffix. During each attempt, the hybrid partitions the current list of items and packs one part by B2F and then the other part by FFD. They proved an APR of $\frac{3}{4}$ for this hybrid.

Coming to the on-line case, Azar et al. [9] (see also [8]) studied the behavior of so-called fair and unfair algorithms. When there is a fixed number of bins, an on-line algorithm is *fair* if it rejects an item only if it does not fit in any bin, while an *unfair* algorithm is allowed to discard an item even if it fits. They show that an unfair algorithm may get better performance compared with the performance achieved by the fair version. In particular, they show that an unfair variant of

First-Fit can pack approximately $2/3$ of the items for sequences for which an optimal off-line algorithm can pack all the items, while the standard (fair) First-Fit algorithm has an asymptotically tight performance arbitrarily close to $5/8$. Later, Epstein and Favrholdt [74] proved that the APR of any fair, deterministic algorithm is $\frac{1}{2} \leq R_A \leq \frac{2}{3}$ and that a class of algorithms including BF has an APR of $\frac{n}{2n-1}$.

6.3 Maximizing the Number of Full Bins

In this variant of the problem, the objective is to pack a list of items into a maximum number of bins subject to the constraint that the content of each bin be no less than a given threshold T . Each such bin is said to be full. Potential practical applications are (i) the packing of canned goods so that each can contains at least its advertised net weight and (ii) the stimulation of economic activity during a recession by allocating tasks to a maximum number of factories, all working at or beyond the minimal feasible level.

This problem is yet another dual of bin packing. A comprehensive treatment can be found in Assmann's Ph.D. thesis [5]; Assmann's collaboration with Johnson, Kleitman and Leung (see [6]) established the main results. They first analyzed an on-line algorithm, a dual version of NF (*Dual Next-Fit*, DNF): pack the elements into the current bin B_j until $s(B_j) \geq T$, then open a new empty bin B_{j+1} as the current bin. If the current bin is not full when all items have been packed, then merge its contents with those of other full bins. All on-line algorithms are allowed this last repacking step to ensure that all bins are full. It is proved in [6] that $\tilde{R}_{\text{DNF}}^\infty = \frac{1}{2}$, where

$$\tilde{R}_A^\infty = \liminf_{m \rightarrow \infty} \left(\min \left\{ \frac{A(L)}{\text{OPT}(L)} : \text{OPT}(L) = m \right\} \right).$$

Assmann et al. [6] also studied a parametrized dual of FF, called *Dual First-Fit* (DFF[r]). Given a parameter r ($1 < r < 2$), the current item a_i is packed into the first bin B_j for which $c(B_j) + s_i \leq rT$. If there are at least two nonempty and unfilled bins (i.e., bins B_j with $0 < c(B_j) < T$) when all items have been packed, an item is removed from the rightmost such bin and added to the leftmost, thus filling it. Finally, if a single nonempty and unfilled bin remains, then its contents are merged with those of previously packed bins. It was shown that $\tilde{R}_{\text{DFF}[r]}^\infty = \frac{1}{2}$, although a better bound might have been expected. But some years later, Csirik and Totik proved that DNF is optimal among the on-line algorithms.

Theorem 46 (Assmann et al. [6] and Csirik and Totik [61]) $\tilde{R}_{\text{DFF}[r]}^\infty = \frac{1}{2}$ and there is no on-line dual bin packing algorithm A for which $\tilde{R}_A^\infty > \frac{1}{2}$.

Turning now to off-line algorithms, first consider the observation of Assmann et al. [6] that presorting does not help the adaptations of algorithms Next-Fit Decreasing and Next-Fit Increasing, for which $\tilde{R}_{\text{NFD}}^\infty = \tilde{R}_{\text{NFI}}^\infty = \frac{1}{2}$. On the other hand, the

off-line version DFFD[r] of DFF[r], obtained by presorting the items according to nonincreasing size, has better performance.

Theorem 47 (Assmann et al. [6]) $\tilde{R}_{\text{DFFD}[r]}^{\infty} = \frac{2}{3}$ if $\frac{4}{3} < r < \frac{3}{2}$ and $\lim_{r \rightarrow 1} \tilde{R}_{\text{DFFD}[r]}^{\infty} = \lim_{r \rightarrow 2} \tilde{R}_{\text{DFFD}[r]}^{\infty} = \frac{1}{2}$.

The *Iterated Lowest-Fit Decreasing* (ILFD) algorithm was also investigated in [6]. It is analogous to algorithm IFFD described in Sect. 6.2. The items are preordered by nonincreasing size. At each iteration, a prefixed number m of bins is considered, and a Lowest-Fit packing is obtained by iteratively assigning the current item to the bin with minimum contents. Binary search on m determines the maximum value for which all m bins are filled. Since n is an obvious upper bound on the optimal solution value, it is easily seen that the algorithm has $O(n \log^2 n)$ time complexity. Moreover,

Theorem 48 (Assmann et al. [6]) $\tilde{R}_{\text{ILFD}}^{\infty} = \frac{3}{4}$.

It is not difficult to see that DNF does not produce optimal packings even when (L, C) is strongly divisible. However, the following optimality results hold.

Theorem 49 (Coffman et al. [51]) *If (L, C) is strongly divisible, then the dual version of NFD (DNFD) produces an optimal packing. For weakly divisible lists, DNFD is no longer optimal, but ILFD is.*

Concerning approximation schemes for the dual bin packing problem, first observe that the approach used by Fernandez de la Vega and Lueker [84] and Karmarkar and Karp [132] (see Sect. 4.3), which eliminates the effect of small items on worst-case behavior, does not appear applicable, as in the dual problem, small items can play an important role in filling small gaps. Csirik et al. [65] have given a PTAS for this problem. Later, an FPTAS was presented by Jansen and Solis-Oba [124].

7 Variations on Item Packing

7.1 Dynamic Bin Packing

The idea of *Dynamic Bin Packing* (DBP) was introduced by Coffman et al. [49]. In the case of dynamic packing, deletion of some elements is allowed at each step, and $A(L)$ is defined as the maximum number of bins used during the packing.

Consider the generalization in which each item a_i is characterized by a triple (s_i, b_i, d_i) , where b_i is the start (arrival) time, d_i (with $d_i > b_i$) is the departure time, and, as usual, s_i is the size. Item a_i remains in the packing during the time interval $[b_i, d_i]$. Assume that $b_i \leq b_j$ if $i < j$. The problem calls for the minimization

of the maximum number, $OPT_D(L)$, of bins ever required in dynamically packing list L when repacking of the current set of items is allowed each time a new item arrives. More formally, if $A(L, t)$ denotes the number of bins used by algorithm A to pack the current set of items at time t , the objective is to minimize

$$A(L) = \max_{0 \leq t \leq b_n} A(L, t).$$

Note that, in this case, an open bin can later become empty, so the classical open/close terminology is no longer valid. The definition of the APR is straightforward:

$$R_A^\infty = \lim_{n \rightarrow \infty} \sup \left\{ \frac{A(L)}{OPT_D(L)} : |L| = n \right\}.$$

The problem models an important aspect of multiprogramming operating systems: the dynamic memory allocation for paged or other virtual memory systems (see, e.g., Coffman [42]), or data storage problems where the bins correspond to storage units (e.g., disk cylinders or tracks), and the items correspond to records which must be stored for certain specified periods of time (see Coffman et al. [49]).

Research on dynamic packing is at the boundary between bin packing and dynamic storage allocation. The most significant difference between the two areas lies in the repacking assumption of dynamic bin packing; repacking is disallowed in dynamic storage allocation, so fragmentation of the occupied space can occur. Coffman et al. [49] studied two versions of the classical FF algorithm. The first is a direct generalization of the classical algorithm. The second is a variant (*Modified First-Fit*, MFF) in which the elements with $s_i \geq \frac{1}{2}$ are handled separately, in an attempt to pair each of them with smaller elements.

Theorem 50 (Coffman et al. [49]) *If $\frac{1}{k+1} < \max\{s_i\} \leq \frac{1}{k}$, then*

$$\frac{11}{4} \leq R_{\text{FF}}^\infty(k) \leq \frac{5}{2} + \frac{3}{2} \ln \frac{\sqrt{13}-1}{2} = 2.89674 \dots \text{ if } k = 1;$$

$$\frac{k+1}{k} + \frac{1}{k^2} \leq R_{\text{FF}}^\infty(k) \leq \frac{k+1}{k} + \frac{1}{k-1} \ln \frac{k^2}{k^2-k+1} \text{ if } k \geq 2;$$

$$2.77 \leq R_{\text{MFF}}^\infty(1) \leq \frac{5}{2} + \ln \frac{4}{3} = 2.78768 \dots$$

For $k = 2$, one gets $\frac{7}{4} \leq R_{\text{FF}}^\infty(2) \leq 1.78768 \dots$. Although these results are worse than their classical counterparts, relative performance is much better than one might think. This can be seen in the following lower bounds.

Theorem 51 (Coffman et al. [49]) *For any on-line dynamic bin packing algorithm A , $R_A^\infty \geq \frac{5}{2}$ and $R_A(k) \geq 1 + \frac{k+2}{k(k+1)}$ if $k \geq 2$, which gives $R_A(2) \geq 1.666 \dots$*

By restricting attention to strongly divisible instances (see Sect. 8.1), then matters improve, as expected. The gap between the general lower bound and that for FF disappears. Indeed,

Theorem 52 (Coffman et al. [51]) *If (L, C) is strongly divisible and L is such that $s_1 > s_2 > \dots > s_k$ ($k \geq 2$), then the APR of the dynamic version of FF is*

$$R_{\text{FF}}^{\infty} = \prod_{i=1}^{k-1} \left(1 + \frac{s_i}{C} - \frac{s_{i+1}}{C} \right).$$

Moreover, for any on-line algorithm A that operates only on strongly divisible instances, $R_A^{\infty} \geq R_{\text{FF}}^{\infty}$.

By a straightforward inductive argument, it can be shown that the continued product has the upper bound

$$\lim_{k \rightarrow +\infty} \left(1 + \frac{1}{2^{k-2}} \right) \prod_{i=1}^{k-2} \left(1 + \frac{1}{2^i} \right) = 2.384 \dots$$

Other algorithms were developed, which, in addition to deletion, allow at least one of the following capabilities: repacking, look ahead, and preordering. For those algorithms which may use all of the above operations, Ivkovic and Lloyd [122] (see also [120]) introduced the *fully dynamic bin packing* (FDBP) term and presented an algorithm, MMP, which is $\frac{5}{4}$ -competitive.

7.2 Selfish Bin Packing

Consider the case in which the items are controlled by selfish agents, and the cost of a bin is split among the agents in proportion to the fraction of the occupied bin space their items require. Namely, if an agent packs its item a of size $s(a)$ into bin B , then its cost is $\frac{s(a)}{c(B)}$, where $c(B)$ is the content of the bin. In other words, the selfish agents would like their items to be packed in a bin that is as full as possible. Hence, given a feasible solution, the agents are interested in bins B' such that $s(a) + c(B') \leq 1$ and $\frac{s(a)}{s(a)+c(B')} < \frac{s(a)}{c(B)}$. If such a bin exists, then a unilaterally migrates from B to B' , as in this new feasible packing its cost decreases. When no such bin exists for any agent, a stable state has been reached. The *social optimum* is to minimize the number of bins used, which is the aim of the standard BP problem.

This problem corresponds to a noncooperative game, and the stable state is a pure *Nash Equilibrium* (NE). There is a *Strong Nash Equilibrium* (SNE) if there is no subset of agents, which can profit by jointly moving their items to different bins so that all agents in the subset obtain a strictly smaller cost.

More precisely, a noncooperative strategic game is a tuple $G = \langle N, (S_i)_{i \in N}, (c_i)_{i \in N} \rangle$ where N is a finite set of players. Each player has a finite set S_i of strategies and a cost function c_i , and each player has complete information about the strategy of all players. Players choose their strategies independently

of each other. A combination of strategies chosen by the players is denoted by $s = (x_j)_{j \in N} \in \times_{j \in N} S_j$ and is called a *strategy profile*. The *social cost* of a game is an objective function $SC(s) : X \rightarrow \mathbb{R}$, that gives the cost of an outcome of the game for a strategy profile $s \in X$, where X denotes the set of all strategy profiles. The *social optimum* of the game is then $OPT(G) = \min_{s \in X} SC(s)$.

The quality of an NE is measured by the *Price of Anarchy* (PoA) and the *Price of Stability* (PoS), which are defined as follows:

$$PoA = \sup_{s \in NE(G)} \frac{SC(s)}{OPT(G)} \quad PoS = \inf_{s \in NE(G)} \frac{SC(s)}{OPT(G)}.$$

By using the SNE, one gets the *Strong Price of Anarchy* (SPoA) and the *Strong Price of Stability* (SPoS):

$$SPoA = \sup_{s \in SNE(G)} \frac{SC(s)}{OPT(G)} \quad SPoS = \inf_{s \in SNE(G)} \frac{SC(s)}{OPT(G)}.$$

For the bin packing problem asymptotic measures are used:

$$PoA(BP) = \limsup_{OPT(G) \rightarrow \infty} \sup_{G \in BP} PoA(G) \quad PoS(BP) = \limsup_{OPT(G) \rightarrow \infty} \inf_{G \in BP} PoS(G)$$

and

$$SPoA(BP) = \limsup_{OPT(G) \rightarrow \infty} \sup_{G \in BP} SPoA(G) \quad SPoS(BP) = \limsup_{OPT(G) \rightarrow \infty} \inf_{G \in BP} SPoS(G).$$

To stress the connection with game theory the problem is called the *bin packing game*. The first publication which discussed this problem from a game theoretical perspective was presented by Biló (see [25]). He showed that starting from an arbitrary feasible packing, one can always reach an NE using an exponential number of migration steps. This result implies that, for every instance of the bin packing game, there exists an optimal packing which provides a NE, that is, $PoS(BP) = 1$. He also proved that computing the best NE is $\mathcal{NP-hard}$. Yu and Zhang [181] constructed an $O(n^4)$ -time Recursive First-Fit Decreasing (RFFD) algorithm for finding an NE. They also investigated the $PoA(BP)$ and proved the following theorem:

Theorem 53 (Yu and Zhang [181]) *For the bin packing game,*

$$1.6416 \approx \sum_{l=1}^{\infty} \frac{1}{2^{\frac{l(l-1)}{2}}} \leq PoA(BP) \leq \frac{41 + \sqrt{145}}{32} \approx 1.6575.$$

Epstein and Kleiman [76] improved the upper bound to 1.64286 and conjectured that the best bound is equal to the lower bound.

Bin packing algorithms look for a good global solution; hence, in general, they do not produce a Nash equilibrium. An exception is the *Subset Sum* (SS) algorithm by Caprara and Pferschy [34], which produces an NE by repeatedly solving a knapsack problem to pack a bin as full as possible. Unfortunately, the knapsack problem is not polynomial unless $\mathcal{P} = \mathcal{NP}$. Indeed, Epstein and Kleiman [76] on the one hand proved equality among the $SPoS(BP)$ and the $SPoA(BP)$ and the asymptotic behavior of the Subset Sum algorithm and on the other hand showed that finding an SNE is \mathcal{NP} -hard.

Caprara and Pferschy proved that $1.6062 \leq R_{SS}^{\infty} \leq 1.6210$. So, by comparing $PoA(BP)$ and $SPoA(BP)$, it appears that they have similar values. Therefore, the efficiency in the bin packing game is only due to selfishness, and coalitions do not help. In addition, $PoS(BP)$ is significantly better than $SPoS(BP)$, which in turn equals $SPoA(BP)$, implying that in the bin packing game the best equilibrium is less efficient if coalition among agents is allowed.

7.3 Bin Packing with Rejection

This variation is motivated by the following problem, arising in file management. Suppose that files are used by a local system. Either a file is downloaded in advance or the system downloads it only when it is actually requested. The first option needs space on a local server, and it has a local transmission cost, while a program uses the file. The second option has a communication cost that one incurs, while the system downloads the file from an external server. An algorithm which manages the local file system has the choice of either packing a file (item) on the local storage device or (according to the costs) downloading it when needed. The problem can be modeled as the following bin packing problem. A pair (s_i, r_i) is associated with each item i ($i = 1, \dots, n$), where s_i is the size and r_i is the *rejection cost* of the item. An algorithm has the choice of either packing an element into the bins or rejecting it. The objective is to minimize the number of bins used, plus the sum of all rejection costs. (If all rejection costs are larger than 1 then every item will be packed, and a standard problem arises.) This model was introduced by Dósa and He [69], who considered both off-line and on-line algorithms, and investigated their absolute and asymptotic behavior.

For off-line algorithms, Dósa and He gave an algorithm with absolute worst-case ratio 2. They also observed that the lower bound for the standard problem remains valid, that is, there is no algorithm with absolute worst-case ratio better than $\frac{3}{2}$, unless $\mathcal{P} = \mathcal{NP}$. The lower bound was reached by Epstein [73] (see also [70]). Based on the ratio r_i/s_i , Dósa and He classified the elements into subclasses and used a sophisticated greedy-type algorithm, RFF4, for which they proved that $R_{RFF4}^{\infty} \leq \frac{3}{2}$.

For the on-line case, they investigated an algorithm, called RFF1, and proved that the absolute worst-case ratio of the best on-line algorithm lies in the interval (2.343, 2.618). For the asymptotic competitive ratio, they introduced another algorithm, RFF2, and proved the following:

Theorem 54 (Dósa and He [69]) *For any positive integer $m \geq 2$, $R_{RFF_2}^\infty(m) \leq \frac{7m-3}{4m-2}$. Hence there exists an on-line algorithm with an APR arbitrarily close to $7/4 = 1.75$.*

Not surprisingly, this result was beaten by a variant of the HF algorithm proposed by Epstein [73]. Her RejH_k algorithm classifies the items as HF. Then thresholds are defined for the rejection choice as follows. Suppose that the next item is $a_i \in I_j$, where $I_j = (\frac{1}{j+1}, \frac{1}{j}]$ for some $1 \leq j \leq k$. Item a_i will be rejected if either $j = k$ and $r_i \leq \frac{k}{k-1}s_i$ or $j < k$ and $r_i \leq \frac{1}{j}$. Using the weighting function technique, Epstein proved the following theorem.

Theorem 55 (Epstein [73]) *The APR of RejH_k tends to $h_\infty(1) = 1.6901$ if $k \rightarrow \infty$. No algorithm with a constant number of open bins can have a smaller APR.*

For the unbounded-space case, the latter result was improved by a *Rejective Modified Harmonic* (MHR) algorithm, which is an adaptation of the Modified Harmonic analyzed by Ramanan et al. [159]:

Theorem 56 (Epstein [73]) *The APR of MHR is at most $\frac{538}{333}$.*

Starting from the classical results of Fernandez de la Vega and Lueker [84] and Hochbaum and Shmoys [118], Epstein [73] also proposed an APTAS for bin packing with rejection, having time complexity $O(n^{O((\varepsilon^{-4})^{\varepsilon}-1)})$. Using some results on the multiple knapsack problem (see Chekuri and Khanna [40] and Kellerer [134]) a faster, $O(n^{O(\varepsilon^{-2})})$ time, approximation scheme was constructed by Bein et al. [21].

7.4 Item Fragmentation

Menakerman and Rom [155] investigated a variant of the bin packing problem in which items may be subdivided into pieces of smaller size, called *fragments*. Fragmenting an item is associated with a cost, which makes the problem \mathcal{NP} -hard. They studied two possible cost functions. In the first variant, called *Bin Packing with Size-Increasing Fragmentation* (BP-SIF), whenever an item is fragmented, overhead units are added to the size of every fragment. In the second variant, called *Bin Packing with Size-Preserving Fragmentation* (BP-SPF), each item has a size and a cost, and, whenever it is fragmented, one overhead unit is added to its cost without changing its total size. (Actually, problem BP-SIF was first introduced by Mandal et al. [150] who showed that it is \mathcal{NP} -hard.)

It is supposed that the item sizes are positive integers and the bins are all of size C . It is obvious that a good algorithm should try to perform the minimum number of fragmentations. Menakerman and Rom developed algorithms that do not fragment items unnecessarily. More precisely, an algorithm is said to prevent unnecessarily fragmentation if it follows the next two rules:

1. No unnecessary fragmentation: An item (or a fragment of an item) is fragmented only if it must be packed into a bin that cannot contain it. In case of fragmentation, the item (or fragment) is divided into two fragments. The first fragment must fill one of the bins, while the second is packed according to the rules of the algorithm.
2. No unnecessary bins: An item is packed into a new bin only if it cannot fit in any of the open bins.

For the BP-SIF problem, it is proved that for any algorithm A that prevents unnecessary fragmentation, $R_A^\infty \leq \frac{C}{C-2}$ for every $C > 2$. For the obvious generalization of the Next-Fit algorithm, called *Next-Fit with item fragmentation* (NF_f), a $\frac{C}{C-2}$ bound for all $C \geq 6$ was proved. For a more sophisticated algorithm, which is actually an iterated version of First-Fit Decreasing, they proved that the APR is not greater than $\frac{C}{C-1}$ when $C \leq 15$, while it is between $\frac{C}{C-1}$ and $\frac{C}{C-2}$ when $C \geq 16$.

As to the BP-SPF problem, the performance of an algorithm for a given list L , $OH_A(L)$ is measured by its overhead, that is, by the difference between the cost of the solution produced by the algorithm and the cost of an optimal solution. The worst-case performance of an algorithm is

$$OH_A^m = \inf\{h : OH_A(L) \leq h \text{ for all } L \text{ with } OPT(L) = m\}.$$

Menakerman and Rom [155] proved that for any algorithm A that prevents unnecessary fragmentation, $OH_A^m \leq m - 1$ and that an appropriate modification of Next-Fit reaches this bound. First-Fit Decreasing performs better when the bin size is small.

Naaman and Rom [158] considered two generalizations of the BP-SPF. The first one is quite simple: r overhead units are added to the size of every fragment. This changes the performance of Next-Fit to $\frac{C}{C-2r}$. The other generalization is more challenging: a set B of m bins is given, whose (integer) bin sizes may be different. Denote by $\bar{C} = \frac{1}{m} \sum_{j=1}^m s(B_j)$ the average bin size. In this case, the asymptotic worst-case bound for Next-Fit is $\frac{\bar{C}}{\bar{C}-2r}$ for every $\bar{C} > 4r$.

Shachnai et al. [168] studied slightly modified versions of BP-SIF and BP-SPF. For the BP-SIF, a header of fixed size Δ is attached to each (whole or fragmented) item, that is, the size required for packing a_i is $s_i + \Delta$. When an item is fragmented, each fragment gets the header. For the BP-SPF, the number of possible splits is bounded by a given value. They presented a dual PTAS and an APTAS for each of the problems. The dual PTASs pack all the items in $OPT(L)$ bins of size $(1 + \varepsilon)$, while the APTASs use at most $(1 + \varepsilon)OPT(L) + 1$ bins. All of these schemes have running times that are polynomial in n and exponential in $1/\varepsilon$. They also showed that both problems admit a dual AFPTAS, which packs the items in $OPT(L) + O(1/\varepsilon^2)$ bins of size $1 + \varepsilon$. Shachnai and Yehezkely [167] developed fast AFPTASs for both problems. Their schemes pack the items in $(1 + \varepsilon)OPT(L) + O(\varepsilon^{-1} \log \varepsilon^{-1})$ bins in a time that is linear in n and polynomial in $1/\varepsilon$.

Epstein and van Stee [81] investigated a version where the items may be split without extra cost, but each bin may contain at most k (parts of) items, for a given constant k . They provided a polynomial-time approximation scheme and a dual approximation scheme for this problem.

7.5 Fragile Objects

Bansal et al. [17] (see also [16]) investigated two interrelated optimization problems: bin packing with fragile objects and frequency allocation in cellular networks (which is a special case of the former problem). In this problem, each item has two attributes: size and fragility. The goal is to pack the items so that the sum of the item sizes in each bin is not greater than the fragility of any item in the bin. They provided a 2-approximation algorithm for the problem of minimizing the number of bins and proved a lower bound of $\frac{3}{2}$ on the asymptotical approximation ratio. They also considered the approximation with respect to fragility and gave a 2-approximation algorithm.

Chan et al. [35] considered the on-line version of the problem and proved that the asymptotic competitive ratio of any on-line algorithm is at least 2. They also considered the case where the ratio between maximum and minimum fragility is bounded by a value k , showing that the APR of an Any-Fit algorithm (including First-Fit and Best-Fit) is at least k . For the case where k is bounded by a constant, they developed a class of on-line algorithms which achieves an APR of $\frac{1}{4} + 3\frac{r}{2}$ for any $r > 1$.

7.6 Packing Sets and Graphs

Let the bin capacity C be a positive integer and suppose that the items are sets of elements drawn from some universal set. A collection of items can fit into a bin if and only if their union has at most C elements. As usual, the object is to pack the items in the smallest number of bins. Good approximation algorithms have yet to be analyzed for this problem; adaptations of the classical bin packing approximation algorithms do not have finite APRs. One observes immediately that while classical algorithms prioritize items based on cardinality and position in sequence, a good algorithm for set packing will have to consider the intersections among a given set of items, an orthogonal basis for prioritizing the items. It is thought that approximation algorithms with a finite APR for this problem are unlikely to exist, but a rigorous treatment of this question has yet to appear.

Dror (Dror, Private communication) points out that scheduling setups in a manufacturing process is an application of set packing. As an example, he describes a machine that produces many types of circuit boards, each requiring a given set of components. The machine has a magazine (bin) that holds at most C components. At any time, only the circuit boards with their component sets (items) in the magazine

can be in production. The problem is to plan the overall production process so as to minimize the number of setups.

Khanna (Khanna, Private communication) mentions that, in connection with studies of multimedia communications, graph packing is an interesting special case. Items are edges (pairs of vertices) in a given graph G . An edge is packed in a bin if both of the vertices to which it is incident are in the bin, and so the problem is to pack the edges of G into as few bins as possible subject to the constraint that there can be at most C vertices in any bin. The approximability of this problem has yet to be studied.

Katona [133] investigated a special case of graph packing. A graph is called *p-polyp* if it consists of p simple paths of the same length and sharing a common end vertex. *Polyp Packing* is the following generalization of bin packing: pack a set of paths of different lengths into a minimum number of edge disjoint polyps. He proved that the problem is \mathcal{NP} -hard and gave bounds on the performance of a modification of First-Fit.

8 Additional Conditions

8.1 Item-Size Restrictions

Perhaps the simplest (and most practical) restriction is simply to have a finite number k of item sizes, say s_1, \dots, s_k , and thus a finite number N of feasible item configurations in a bin. This class of problems arose in studies on approximation schemes (see [84, 132]) and was considered in its own right by Blazewicz and Ecker [26]. Let the i -th configuration be denoted by $p_i = (p_{i1}, \dots, p_{ik})$ where p_{ij} is the number of items of size s_j in p_i . By the definition of feasibility, $\sum_{j=1}^k p_{ij} s_j \leq 1$ for all i . Let y_i be the number of bins in a packing of a given list L that contain configuration p_i , and let n_j be the number of items of size s_j in L . Then a solution to the bin packing problem for L is a solution to the integer programming formulation:

$$\begin{aligned} \min & \sum_{i=1}^N y_i \\ \text{subject to } & \sum_{i=1}^N p_{ij} y_i \geq n_j \quad (j = 1, \dots, k) \\ & y_i \geq 0 \text{ and integer } (i = 1, \dots, N). \end{aligned}$$

Note that, if $\frac{1}{r}$ lower bounds the item sizes, then $N = O(k^{r-1})$, so the constraint matrix above has k rows and $O(k^{r-1})$ columns. This problem can be solved in time polynomial in the number of constraints (see Lenstra [144]), which is a constant independent of the number of items. The optimal overall packing can then be constructed in linear time.

If the exact solution is not needed, then one can use the classical approach of Gilmore and Gomory [105, 106]) and compute LP relaxations (see Sect. 4.3); this method gives solutions that are at most k off the optimal in significantly less time.

Gutin et al. [113] investigated an on-line problem in which one only has two different element sizes. They gave a $\frac{4}{3}$ lower bound for this problem and provided an asymptotically optimal algorithm for it. Epstein and Levin [77] focused on the parametric case, where both item sizes are bounded from above by $\frac{1}{k}$ for some natural number $k \geq 1$.

Next consider the classical problem restricted to lists L of items whose sizes form a *divisible sequence*, that is, the distinct sizes $s_1 > s_2 > \dots$ taken on by the items are such that s_{i+1} divides s_i for all $i \geq 1$. The number of items of each size is arbitrary. Given the bin capacity C , the pair (L, C) is *weakly divisible* if L has divisible item sizes and *strongly divisible* if in addition the largest item size s_1 in L divides C .

This variant has practical applications in computer memory allocation, where device capacities and memory block sizes are commonly restricted to powers of 2. It is important to recognize such applications when they are encountered, because approximation algorithms for many types of \mathcal{NP} -hard bin packing problems generate significantly better packings when items satisfy divisibility constraints. In some cases, the restriction leads to algorithms that are asymptotically optimal. The problem was studied by Coffman, Garey, and Johnson, who proved the following result for classical off-line algorithms.

Theorem 57 (Coffman et al. [51]) *If (L, C) is strongly divisible, then NFD and FFD packings are optimal.*

Indeed, the residual capacity in a bin is always either zero or at least as large as the last (smallest) item packed in the bin. The last packed item is at least as large as any remaining (unpacked) item, so either the bin is totally full or it has room for the next item. Therefore, the FFD and NFD algorithms have the same behavior: they initialize a new bin only when the previous one is totally full, so the packings they produce are identical and perfect.

The optimality of FFD holds in the less restrictive case of the following theorem as well.

Theorem 58 (Coffman et al. [51]) *If (L, C) is weakly divisible, then an FFD packing is always optimal.*

For strongly divisible instances, optimal performance can also be obtained without sorting the items.

Theorem 59 (Coffman et al. [51]) *If (L, C) is strongly divisible, then an FF packing is always optimal.*

In the *Unit-Fractions Bin Packing Problem*, n items are given, and the size of each item i is $\frac{1}{w_i}$, with w_i a positive integer ($1 \leq i \leq n$). This problem was initially studied by Bar-Noy et al. [18]. Let $H = \lceil \sum_{i=1}^n \frac{1}{w_i} \rceil$ be a lower bound on the required number of bins. They presented an off-line algorithm which uses at most $H + 1$ bins. For the on-line version, they analyzed an algorithm which uses $H + O(\sqrt{H})$ bins. It is also proved that any on-line algorithm for the unit-fractions bin packing problem uses at least $H + \Omega(\ln H)$ bins.

Dynamic bin packing problems with unit-fractions items were considered by Chan et al. [36]. They investigated the family of Any-Fit algorithms and proved the following results:

$$R_{\text{BF}}^\infty = R_{\text{WF}}^\infty = 3$$

$$2.45 \leq R_{\text{FF}}^\infty \leq 2.4985.$$

In addition they proved the following lower bound result:

Theorem 60 (Chan et al. [36]) *For a dynamic bin packing problem with unit-fractions items, there is no on-line algorithm A with $R_A^\infty < 2.428$.*

It would be interesting to examine how approximation algorithms behave with other special size sequences (e.g., Fibonacci numbers), as mentioned by Chandra et al. [39] in the context of memory allocation.

8.2 Cardinality Constrained Problems

The present section deals with the generalization in which the maximum number of items packed into a bin is bounded by a positive integer p . The problem has practical applications in multiprocessor scheduling with a single resource constraint, when the number p of processors is fixed, or in multitasking operating systems where the number of tasks is bounded. If the maximum number of items in a bin is bounded by p , then the problem is called the *p-Cardinality Constrained Bin Packing Problem*. Observe which is the difficulty when one tries to construct a good algorithm for this problem. Without the cardinality constraint, a huge number of small elements slightly affect the number of occupied bins: either they can be packed together into few bins, or they can be packed into bins which are “almost full.” In the considered case, having packed many small elements results in almost empty bins, because of the cardinality constraint. Therefore, when a large item arrives, it has to be packed into a new bin even if it would fit into an already open bin.

In the context of the above applications, Krause et al. [139] modified classical algorithms so as to deal with the restricted number of items per bin. Their on-line algorithm, pFF, is FF with an additional check on the number of items in open bins. They proved that if $p \geq 3$, then

$$\frac{27}{10} - \frac{37}{10p} \leq R_{\text{pFF}}^{\infty} \leq \frac{27}{10} - \frac{24}{10p}.$$

As p tends to ∞ , the bound is much worse than the corresponding APR of the unrestricted problem (2.7 versus 1.7). Whether this upper bound can be improved remains an open question.

On-line algorithms were also studied by Babel et al. [11] (see also [10]), who investigated four algorithms, denoted as A_1, \dots, A_4 .

- Algorithm A_1 operates with the BF strategy while packing the subsequent element. Different classes of blocked bins are defined according to the number of items in a bin. The algorithm tries to pack the element first into all blocked bins that satisfy a threshold condition, then into all formerly blocked and currently unblocked bins, and finally into the remaining bins. It is proved that for every $p \geq 3$,

$$R(p) = 2 + \frac{p + k_p(k_p - 3)}{(p - k_p + 1)k_p}, \quad \text{where} \quad k_p = \left[\frac{-p + \sqrt{p^3 - 2p}}{p - 2} \right],$$

so $\lim_{p \rightarrow \infty} A_1 = 2$.

- In algorithm A_2 , a bin is closed if $c(B) \geq \frac{1}{2}$ and it contains at least $\frac{p}{2}$ items. Furthermore, a pair of bins, B_1 and B_2 , is also closed if $c(B_1) + c(B_2) \geq 1$ and $|B_1| + |B_2| \geq k$. The open bins are subdivided into three classes, and the algorithm tries to pack the current item, in turn, into bins of such classes. If the item does not fit in any of these bins, a new bin is open. It is proved that for every $p \geq 3$, algorithm A_2 has an APR of 2.
- Algorithm A_3 deals with the case $p = 2$. The basic idea of the algorithm is that, when a small item is packed, a balance should be kept between the number of bins with only one small item and the number of bins which have two such items. It is proved that $R_{A_3} = 1 + \frac{1}{\sqrt{5}}$.
- For the 3-cardinality problem, a Harmonic-type algorithm is defined, with four bin classes. The score intervals are as follows: $I_1 = (0; \frac{1}{3}]$, $I_2 = (\frac{1}{3}, \frac{1}{2}]$, $I_3 = (\frac{1}{2}, \frac{2}{3}]$, and $I_4 = (\frac{2}{3}, 1]$. The resulting Algorithm A_4 is a much more complicated version of the classical HF algorithm and has an asymptotic competitive ratio of 1.8.

Two lower bounds are also given:

Theorem 61 (Babel et al. [11]) *There is no 2-cardinality on-line bin packing algorithm A with APR $R_A < \sqrt{2}$.*

Theorem 62 (Babel et al. [11]) *There is no 3-cardinality on-line bin packing algorithm A with APR $R_A < \frac{3}{2}$.*

For unbounded-space algorithms, some further cases were investigated by Epstein [71]. She considered a Harmonic-type algorithm that defines five

subintervals and packs the items into bins with a more sophisticated procedure. Her algorithm for the case $p = 3$ improves the APR of [11] to $\frac{7}{4} = 1.75$. For the cases $p = 4$, $p = 5$ and $p = 6$, she refined the basic algorithm, designing three algorithms with APRs $\frac{71}{38} = 1.86842$, $\frac{771}{398} = 1.93719$, and $\frac{284}{144} = 1.99306$. She proved that an algorithm with an APR strictly better than 2 cannot be bounded space (unless $p \leq 3$).

For the bounded-space case, Epstein defined the *Cardinality Constrained Harmonic- p* (CCHp) algorithm. She considered the Sylvester sequence, defined the sum

$$S_p = \sum_{i=1}^p \left\{ \frac{1}{t_i - 1}, \frac{1}{p} \right\},$$

and proved the following theorems:

Theorem 63 (Epstein [71]) *The value of S_p is a strictly increasing function of $p \geq 2$ such that $\frac{3}{2} \leq S_p < h_\infty(1) + 1$ and $\lim_{p \rightarrow \infty} S_p = h_\infty(1) + 1 \approx 2.69103$.*

Theorem 64 (Epstein [71]) *For every $p \geq 2$, the APR of CCHp is S_p , and no on-line algorithm which uses bounded space can have a better competitive ratio.*

In addition, Epstein extended her results to the variable-sized and the resource-augmentation cases (see Sects. 5.1 and 5.2).

An off-line algorithm (*Largest Memory First*, LMF), based on an adaptation of FFD, was studied by Krause et al. [139]. They proved that $R_{\text{LMF}}^\infty = 2 - \frac{2}{p}$ if $p \geq 2$. Here the gap between the unrestricted and the restricted case is large (2 versus $\frac{11}{9}$). The authors' search for an algorithm with better worst-case behavior resulted in the *Iterated Worst-Fit Decreasing* (IWFD) algorithm. IWFD starts by sorting the items according to nonincreasing size, and by opening q empty bins, where $q := \lceil \max(\frac{n}{p}, \sum_{i=1}^n s_i) \rceil$ is an obvious lower bound on $OPT(L)$. Then (i) IWFD tries to place the items into these q bins using the Worst-Fit rule (an item is placed into the open bin whose current number of items is less than p and whose current content is minimum, breaking ties by highest bin index); (ii) whenever the current item a_i does not fit in any of the q bins, q is increased by 1, all items are removed from the bins and the processing is restarted from (i). If one implements this approach using a binary search on q , the time complexity of IWFD is $O(n \log^2 n)$. Krause, Shen, and Schwetman proved that the algorithm behaves very well in certain special cases:

- If $p = 2$, then $IWFD(L) = OPT(L)$ for any list L .
- If in the final schedule no capacity constraint is operative, that is, no open bin has residual capacity smaller than the size of the smallest item, then $IWFD(L) = OPT(L)$.
- If in the final schedule no cardinality constraint is operative, that is, no open bin containing p items has residual capacity at least equal to the size of the smallest item, then $R_{\text{IWFD}}^\infty < \frac{4}{3}$.

For the general case, where the final packing contains both bins for which the capacity constraint is operative and bins for which it is not, the constants are

significantly worse but still small: $\frac{4}{3} \leq R_{\text{IWFD}}^{\infty} \leq 2$. The authors conjecture that $R_{\text{IWFD}}^{\infty} = \frac{4}{3}$.

For a long time, it was an open question whether there exists a heuristic algorithm with an APR better than 2. Kellerer and Pferschy [135] proposed a new algorithm based on a method that proved to be fruitful in scheduling theory, namely, one that performs a binary search on the possible $OPT(L)$ values. At each search step, they run a procedure that tries to pack all items into a number of bins equal to $\frac{3}{2}$ times the current search value. The procedure basically tries to solve a multiprocessor scheduling problem using the classical LPT rule with a cardinality constraint. The time complexity of this *Binary Search* (BS) algorithm is $O(n \log^2 n)$, and the improved bound is $\frac{4}{3} \leq R_{\text{BS}}^{\infty} \leq \frac{3}{2}$. Although the gap has been reduced, the exact bound is still not known.

8.3 Class Constrained Bin Packing

In the *Class Constrained Bin Packing Problem* (CLCP), each item a_i has two parameters: the size s_i and the color c_i , where $s_i \in (0, 1]$ and a color is represented by a positive integer, that is, $c_i \in [1, q]$. A further parameter $Q \in \mathbb{N}$ is associated with the problem as a bound for the number of the different colors within a bin. Different items may have the same color, and the items with the same color are classified into *color classes*. The objective is to pack the items into the minimum number of bins $\{B_1, B_2, \dots, B_m\}$, such that $\sum_{a_j \in B_i} s_j \leq 1$ for $i = 1, \dots, m$ and each such bin has items from at most Q color classes. Clearly, if $q = n$, then one gets the p -cardinality constrained problem (see Sect. 8.2), and if $q \leq Q$, then a classical bin packing problem arises. Shachnai and Tamir [165] proved that the problem is \mathcal{NP} -hard even in the case of identical item sizes.

For the case of on-line bounded-space algorithms, Xavier and Miyazawa [176] presented inapproximability results. More precisely, they proved that if $s_i \in [K_1, K_2]$, with $0 < K_1 < K_2 \leq 1$, for all $i = 1, \dots, n$, then any bounded-space on-line algorithm has an APR of $\Omega(\frac{1}{QK_1})$.

Shachnai and Tamir [166] studied unbounded-space algorithms for the case of identically sized items. Suppose that $s_i \in (\frac{1}{r+1}, \frac{1}{r}]$ for each a_i : they denoted by $S_{Q,r}(k, h)$ the set of those lists which have $kQ < q \leq (k+1)Q$ and $hr < n \leq (h+1)n$, where n is the number of elements in the list. They proved that for any deterministic algorithm A ,

$$R_A \geq 1 + \frac{k+1 - \lceil \frac{kQ+1}{r} \rceil}{h+1},$$

and that the lower bound can be reached by the FF algorithm.

Sachnai and Tamir also introduced the notion of *Color-Set algorithm* (CS). This algorithm partitions the colors into $\lceil q/Q \rceil$ color sets and packs the items of each color set in a greedy way. Note that only one bin is open for each color-set at a time.

They proved that $R_{CS} < 2$. The set of Any-Fit algorithms was also investigated, and it was shown that $R_A \leq \min(r/Q, Q+1)$ for any such algorithm. They further proved that the absolute worst-case ratio of the Next-Fit algorithm is $R_{NF} = r/Q$.

The idea of CS was used to develop new algorithms. Color classes are divided *on-line* into sets of Q colors each, in order of appearance. Each open bin is assigned one color set. As a new item arrives, the algorithm tries to pack it into a bin having the color of the item. If the FF rule is used, then one gets the *Color-Set First-Fit* (CSFF) algorithm, while the NF rule gives the *Color-Set Next-Fit* (CSNF) algorithm. For the case of identical item sizes considered in [165], the authors proved that the competitive ratio of both CSFF and CSNF is at most 2.

The case $Q = 2$ was exhaustively studied by Epstein et al. [82], who proved that the upper bound for identical items is tight for algorithms FF and CSFF. Furthermore, they showed that, for arbitrary item sizes, the competitive ratio is at least $9/4$. A 1.5652 lower bound was also given for any on-line algorithm with arbitrary item sizes, thus showing that, even if $Q = 2$, the on-line algorithms for CLCP behave worse than those for the classical bin packing. For what concerns upper bounds, they proved that for arbitrary values of Q , $R_{CSFF} \infty \leq 3 - \frac{1}{Q}$.

Xavier and Miyazawa [176] investigated the FF algorithm and proved that $R_{FF} \in [2.7, 3]$. They defined algorithm \mathcal{A}_Q , which subdivides the items, in a harmonic way, into three classes, and packs the next item, according to its size, into the class it belongs to. The algorithm also considers the color constraint Q . It was proved that $R_{\mathcal{A}_Q} \in (2.666, 2.75]$. Epstein et al. [82] developed an algorithm which has a competitive ratio of 2.65492, valid for every Q .

For fixed values of q , there are approximability results for CLCP. Note that, if q is fixed, then Q is also constant since $Q \leq q$. For equal item sizes, Shachnai and Tamir [165] constructed a dual PTAS whose time complexity is polynomial in n . The scheme uses the optimal number of bins, but the bin capacity is $1 + \varepsilon$. Xavier and Miyazawa [176] developed an APTAS. They also gave an AFPTAS with time complexity $O(\frac{n}{\varepsilon^2})$. Epstein et al. [82] proved that if q is considered as a parameter, then there is no APTAS for CLCP for any value of Q . They also gave an AFPTAS for the case of constant q , with time complexity $O(\frac{n}{\varepsilon^2})$.

8.4 LIB Constraints

In certain applications, the positions of the items within a bin are restricted: a larger item cannot be put on top of a smaller one. This leads to the variant with the LIB (Larger Item on the Bottom) constraint. The problem was first discussed by Finlay and Manyem [85] and Manyem et al. [151]. They studied the Next-Fit algorithm and proved that it cannot achieve a finite APR. For the First-Fit algorithm, they showed that $2 \leq R_{FF} \leq 3$. They also analyzed a variant of Harmonic-Fit (HARM) which partitions the items into classes following the classical rule and packs them by applying to each class of bins the FF rule (instead of NF, which performs

very badly). They showed that $2 \leq R_{\text{HARM}}^\infty$. The parametric case of HARM was considered in Epstein [72], who proved the following result:

Theorem 65 (Epstein [72]) *Let $r > 0$ be a positive integer, and suppose that $\frac{1}{r+1} < \max\{s_i\} \leq \frac{1}{r}$. Let k denote the number of bin classes. Then $R_{\text{HARM}}(r) = \Theta(k - r + 1)$ for any fixed value of $k < \infty$.*

This result has an interesting consequence: contrary to the standard problem, for which the performance of the Harmonic-Fit algorithm improves when the number of bin classes is increased, in this case, the competitive ratio worsens, and algorithm HARM has an unbounded ratio as $k \rightarrow \infty$. Unfortunately, neither BF nor WF nor AWF behave better than HARM, that is, their APRs are also unbounded. In the same paper, Epstein analyzed the FF algorithm, proving that $R_{\text{FF}}(1) = 2.5$ and $R_{\text{FF}}(r) = 2 + \frac{1}{r}$ if $r \geq 2$. Very recently, her result was improved by Dósa, Tuza, and Ye (Dósa et al., 2010, Private communication), who proved that $R_{\text{FF}}(1) = 2 + \frac{1}{6}$ and $R_{\text{FF}}(r) = 2 + \frac{1}{r(r+2)}$ if $r \geq 2$.

As to lower bounds, very little is known. Clearly, the FFD algorithm automatically fulfills the LIB constraint, so $\frac{11}{9}$ is a lower bound for any semi-on-line algorithm which packs preordered elements. The only nontrivial lower bound was obtained by Epstein [72]:

Theorem 66 (Epstein [72]) *The competitive ratio of any on-line algorithm for bin packing with LIB constraints is at least 2. The result is valid for the parametric case as well, for any value of $r \geq 1$.*

8.5 Bin Packing with Conflicts

In this problem, a set of items $L = a_1, a_2, \dots, a_n$ with sizes s_1, s_2, \dots, s_n and a conflict graph $G(L, E)$ are given. The objective is to find the minimum number of independent sets such that the sum of the sizes within each set is at most one. The problem is a generalization of both the classical bin packing problem (when $E = \emptyset$) and the graph coloring problem (when $s_i = 0$ for all i). In the on-line version of the problem, when a new item arrives, one gets its size and the edges of the conflict graph which connect it to previous items. Instead of examining the APR, most investigations focused on the absolute worst-case ratio AR, since the conflict graph that proves the worst case is valid both for small and large instances. Therefore, the APR may not be better than the AR.

It is known that graph coloring is hard to approximate for general graphs (see, e.g., Lovász et al. [149]); the problem at hand has been studied for specific graphs. The early studies on the *bin packing with conflicts* (BPC) dealt with *perfect graphs*, since the coloring problem is polynomially solvable for them. Jansen and Öhring [123] gave an algorithm with an APR of 2.7. Epstein and Levin [78] improved this result with an algorithm having the performance guarantee $h_\infty(1) + 1 = 2.690103$.

Jansen and Öhring also proposed an algorithm that uses the so-called *Precoloring Extension Problem* (PEP) as a preprocessing stage to get a partition of the conflict graph. In the PEP, one is given an undirected graph $G = (V, E)$ and k distinct vertices v_1, v_2, \dots, v_k . The aim is to find a minimum coloring f of G such that $f(v_i) = i$ for $i = 1, 2, \dots, k$. The minimum number of colors is denoted by $\chi_I(G)$. The PEP is polynomially solvable for a number of graph classes (interval graphs, forests, K-trees, etc.), whereas it is \mathcal{NP} -hard for bipartite graphs. Let \mathcal{C} be the class of those graphs for which the PEP is polynomially solvable for any induced subgraph of G .

Let $G \in \mathcal{C}$ be a conflict graph and let $BIG = \{a_j \in L : s_j > \frac{1}{2}\}$. The algorithm by Jansen and Öhring uses BIG as the set of pre-colored vertices and then determines a feasible coloring of G using $\chi_I(G)$ colors: in this coloring, each pair of items in BIG has different colors. Finally, it uses a bin packing heuristic for each color class. Denote this algorithm by $\mathcal{A}(H)$, where H is the bin packing heuristic. They proved the following theorem:

Theorem 67 (Jansen and Öhring [123]) *For the BPC problem, the $\mathcal{A}(FFD)$ has a performance ratio $2.4231 = h_\infty(2) + 1 \leq R_{\mathcal{A}(FFD)} \leq 2.5$.*

Later, Epstein and Levin [78] revisited the algorithm, proving that the lower bound of $R_{\mathcal{A}(FFD)}$ is tight. Instead of using the PEP for preprocessing, they used a matching-based preprocessing (MP) algorithm, for which they proved that $R_{MP} = 2.5$, and that the result remains valid for all classes of graphs for which there exists a polynomial-time algorithm to find a coloring with a minimum number of colors. They also suggested a greedy algorithm which tries to pack two or three independent items into a single bin and then applies FFD to each color class. They proved that the approximation ratio of this algorithm is $\frac{7}{3}$.

As already mentioned, the PEP is \mathcal{NP} -hard for bipartite graphs. However, Jansen and Öhring [123] proved that by defining a simple algorithm which finds a coloring of the conflict graph with two colors, and packs each color class with NF, one gets an AR of 2. They also pointed out that the AR does not change if NF is replaced by FFD. Using a more complicated pairing technique, Epstein and Levin [78] presented an algorithm with approximation ratio exactly equal to $\frac{7}{4}$.

For the on-line version, the BPC is hard to approximate for many classes of graphs. Success depends on the on-line coloring phase. For *interval graphs*, Kierstead and Trotter [137] presented an on-line coloring algorithm, which uses $3\omega - 2$ colors in the worst case, where ω is the maximum clique size of the graph. Epstein and Levin [78] pointed out that, using NF for each color class, one obtains a 5-competitive algorithm. They also proved a combination of FFD with the algorithm in [137] produces an algorithm with competitive ratio 4.7. For interval graphs, there exists a lower bound for any on-line algorithm:

Theorem 68 (Epstein and Levin [78]) *The competitive ratio of any on-line algorithm for BPC on interval graphs is at least $\frac{155}{36} \approx 4.30556$.*

The theorem is a direct consequence of two interesting lemmas:

Lemma 1 (Epstein and Levin [78]) *Let c be a lower bound on the APR of any on-line algorithm for classical bin packing, which knows the value OPT in advance. Then the AR of any on-line algorithm for BPC on interval graphs is at least $3 + c$.*

Lemma 2 (Epstein and Levin [78]) *Any on-line algorithm for classical bin packing, which knows the value OPT in advance, has an AR of at least $\frac{47}{36} \approx 1.30556$.*

The on-line problem for other graph classes is still open.

8.6 Partial Orders

In this generalization, precedence constraints among the elements are given, where precedence refers to the relative ordering of bins. Let \prec denote the partial order giving the precedence constraints. Then $a_i \prec a_j$ means that, if a_i and a_j are packed in B_r and B_s , respectively, then $r \leq s$. Call the model *strict* if $r < s$ replaces $r \leq s$ in this definition.

Two practical applications have been considered in the literature. The first one is the assembly line balancing problem, in which the assembly line consists of identical workstations (the bins) where the products stop for a period of time equal to the bin capacity. The item sizes are the durations of tasks to be performed, and a partial order is imposed: $a_i \prec a_j$ means that the workstation to which a_i is assigned cannot be downstream of the one to which a_j is assigned. The second application arises in multiprocessing scheduling; here each item corresponds to a unit-duration process having a memory (or other resource) requirement equal to the item size. The bin capacity measures the total memory availability. In the given partial order, $a_i \prec a_j$ imposes the requirement that a_i must be executed before a_j finishes. The objective is then to find a feasible schedule that finishes the set of processes in minimum time (number of bins).

The first problem was studied by Wee and Magazine [174] and the second one by Garey et al. [104]. In both cases, the *Ordered First-Fit Decreasing* (OFFD) algorithm was applied. An item is called *available* if all its immediate predecessors have already been packed. At each stage, the set of currently available items is sorted according to nonincreasing size, and each item is packed into the lowest indexed bin where it fits and no precedence constraint is violated. Note that, if no partial order is given, this algorithm produces the same packing as FFD. In general, however, its worst-case behavior is considerably worse. The APR is $R_{\text{OFFD}}^\infty = 2$, except in the strict model, where $R_{\text{OFFD}}^\infty = \frac{27}{10}$.

Liu and Sidney [148] considered the case in which there is an *ordinal assumption*: initially, the actual sizes of the items are unknown, but their ordering is known, that is, $s_1 \geq s_2 \geq \dots \geq s_n \geq 0$. Now assume that perfect knowledge of some of the sizes can be “purchased” by specifying the ordinal position of the desired sizes. It is shown that a worst-case performance ratio of ρ (where $\rho \geq 2$ is an

integer) can be achieved if knowledge of $\lfloor \ln(n(\rho - 1) + 1)/\ln \rho \rfloor$ weights can be purchased.

8.7 Clustered Items

In this generalization, a function $f(a_i, a_j)$ is given, measuring the “distance” between items a_i and a_j . A distance constraint D is also given; two items a_i and a_j may be packed into the same bin only if $f(a_i, a_j) \leq D$. The problem has several obvious practical applications in contexts where geographical location constraints are present. Chandra et al. [39] studied different cases, their main result being for the case where the items in a bin must all reside within the same unit square. They proposed a geometric algorithm A and proved that $3.75 \leq R_A^\infty \leq 3.8$.

8.8 Item Types

In studying the packing of advertisements on Web pages, Adler et al. [1] encountered a variant of bin packing in which items are classified into *types*; the set of types is given, and there are no constraints on the number of items of any type. The problem is to pack the items into as few bins as possible subject to the constraint that no bin can have two items of the same type. They design optimal algorithms for restricted cases and then bound the performance of these algorithms viewed as approximations to the general problem.

9 Examples of Classification

In this section, the classification scheme proposed in Sect. 2.3 is resumed. The following examples should help familiarize the reader with it:

1. The classification of [157] shows a bin-size extension of **pack**.

pack|off-line|FPTAS|{B_i}

Results: An approximation algorithm for variable-sized bin packing is presented which for any given positive ε produces a scheme with approximation ratio $1 + \varepsilon$. This algorithm has time complexity polynomial in the number of items, the number of bins, and $1/\varepsilon$.

2. The classification of [145] gives another such example:

pack|on-line, off-line, open-end|R_A[∞]bound; FPTAS

Results: For this hybrid of *pack* and *cover*, it is shown that any open-end version of on-line algorithms must have an asymptotic worst-case ratio

of at least 2. Next-Fit achieves this ratio. There is a fully polynomial-time approximation scheme for this problem.

3. The classification of [7] illustrates a capacity minimization problem with bin-stretching analysis:

$$\min_{\text{cap}} \text{on-line} |R_A| \text{bound} | \text{stretching}$$

Results: A combined algorithm achieves a worst-case bound of 1.625. The best lower bound for any on-line algorithm is 4/3.

4. Reference [48] studies a subset-cardinality objective function:

$$\max_{\text{card}} (\text{subset}) | \text{off-line} |R_A|$$

Results: $R_{\text{WFI}} = \frac{1}{2}$, $R_{\text{FFI}} = \frac{3}{4}$.

5. Reference [61] is classified as a covering problem.

$$\text{cover} | \text{on-line} |R_A^\infty| \text{bound}$$

Results: Asymptotic bound: $R_A^\infty \leq 1/2$ for any on-line algorithm A . There exists an asymptotically optimal on-line algorithm.

6. The classification of [90] is

$$\text{pack} | \text{off-line} |R_A^\infty| \text{bound}$$

Algorithm: Combined Best-Fit (CBF) which takes the better of the First-Fit Decreasing solution and Best Two-Fit (B2F) solution, where the latter algorithm is a grouping version of Best-Fit limiting the number of items per bin.

Results: $R_{\text{B2F}}^\infty = 5/4$, $\frac{227}{195} \leq R_{\text{CBF}}^\infty \leq \frac{6}{5}$

Note that the word “variant” may be simplistic in that it occasionally hides details of relatively complicated algorithms.

7. Reference [95] shows an example for the combination of two algorithm classes:

$$\text{pack} | \text{bounded-space, repack} |R_A^\infty| \text{bound}$$

Algorithm: REP₃, an adaptation of FFD using at most three open bins.

Result: $R_{\text{REP}_3}^\infty \approx 1.69 \dots$

8. The classification of [139] illustrates a case where one is allowed to constrain the number of items per bin.

$$\text{pack} | \text{on-line, off-line} |R_A^\infty| \text{bound} | \text{card}(B) \leq k$$

Result:

$$\left(\frac{27}{10} - \left\lceil \frac{37}{10k} \right\rceil \right) \leq R_{\text{FF}_k}^\infty \leq \left(\frac{27}{10} - \frac{24}{10k} \right),$$

where FF_k is the obvious adaptation of FF.

9. For [175], the classification mentions yet another constraint:

$$\text{pack}|\text{bounded-space, on-line}|R_A^\infty$$

Result: $R_{SH_k}^\infty(k) = R_{H_k}^\infty$, where SH_k is a simplified version of H_k that uses only $O(\log k)$ open bins at any time.

10. Classification of [6] aggregates several results:

$$\text{cover}|\text{on-line, off-line, open-end}|R_A^\infty$$

Algorithms: Open-end variant of Next-Fit called DNF, of First-Fit Decreasing with a parameter r (FFD_r), and of an iterated version of Worst-Fit (IWFD).

Results:

$$R_{\text{DNF}}^\infty = \frac{1}{2}, \text{FFD}_r^\infty = \frac{2}{3} \text{ for all } r, \frac{4}{3} \leq r \leq \frac{3}{2}, \text{ and } R_{\text{IWFD}}^\infty = \frac{3}{4}.$$

11. A more recent such publication is [29, 30]:

$$\text{maxpack}|\text{on-line, off-line}|R_A^\infty|s_i \leq 1/k$$

Results:

- a. No off-line approximation algorithm can have an asymptotic bound larger than $17/10$, a bound achieved by FFD.
 - b. For the off-line case, $R_{\text{FFI}}^\infty = 6/5$, $R_{\text{FFI}}^\infty(1/k) = 1 + 1/k$, $k \geq 2$
 - c. For the on-line case, $R_{\text{FF}}^\infty(1/k) = R_{\text{BF}}^\infty(1/k) = 1 + 1/(k-1)$, $k \geq 3$.
 - d. No deterministic algorithm A has a finite asymptotic bound R_A^∞ for either the `card(subset)_min` or `orc(subset)_min` problems, except for the latter problem in the parametric case with $k > 1$, in which case the bound is again $1 + 1/(k-1)$.
12. An analysis with similarities to bin stretching is used in [31] to compare WF and NF.

$$\text{pack}|\text{on-line}|R_A^\infty|\text{stretching}$$

Results: A derivation of the asymptotic ratios for WF and NF under the assumption that they use bins of capacity C while OPT uses unit-capacity bins shows that WF and NF have the same asymptotic bound for all $C \geq 1$, except when $1 < C < 2$, in which case WF has the smaller bound.

Acknowledgements The second author was supported by Project “TÁMOP-4.2.1/B-09/1/KONV-2010-0005 - Creating the Center of Excellence at the University of Szeged,” supported by the European Union and cofinanced by the European Regional Development Fund.

Cross-References

- Advances in Scheduling Problems
- Complexity Issues on PTAS
- Online and Semi-online Scheduling

Recommended Reading

1. M. Adler, P.B. Gibbons, Y. Matias, Scheduling space sharing for internet advertising. *J. Sched.* **5**, 103–119 (2002)
 - $\text{pack}|\text{off-line}| \text{running-time} | \text{mutex}$.
2. S. Albers, Better bounds for on-line scheduling, in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, TX, 1997, pp. 130–139
 - $\text{mincap}|\text{on-line}| R_A \text{bound}$.
3. N. Alon, Y. Azar, G.J. Woeginger, T. Yadid, Approximation schemes for scheduling, in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA (SIAM, 1997), pp. 493–500
 - $\text{mincap}|\text{off-line}| PTAS$.
4. R.J. Anderson, E.W. Mayr, M.K. Warmuth, Parallel approximation algorithms for bin packing. *Inf. Comput.* **82**, 262–277 (1989)
 - $\text{pack}|\text{off-line}| \text{running-time}$.
5. S.F. Assmann, *Problems in Discrete Applied Mathematics*. PhD thesis, Mathematics Department MIT, Cambridge, MA, 1983
6. S.F. Assmann, D.S. Johnson, D.J. Kleitman, J.Y.-T. Leung, On a dual version of the one-dimensional bin packing problem. *J. Algorithms* **5**, 502–525 (1984)
 - $\text{cover}|\text{on-line}, \text{off-line}, \text{open-end}| R_A^\infty$.
7. Y. Azar, O. Regev, On-line bin-stretching. *Theor. Comput. Sci.* **268**, 17–41 (2001)
 - $\text{mincap}|\text{on-line}| R_A \text{bound} | \text{stretching}$.
8. Y. Azar, J. Boyar, L.M. Favrholdt, K.S. Larsen, M.N. Nielsen, Fair versus unrestricted bin packing, in *SWAT '00, 7th Scandinavian Workshop on Algorithm Theory*, Bergen, Norway. Lecture Notes in Computer Science, vol. 1851 (Springer, 2000), pp. 200–213. This is the preliminary version of [9]
9. Y. Azar, J. Boyar, L. Epstein, L.M. Favrholdt, K.S. Larsen, M.N. Nielsen, Fair versus unrestricted bin packing. *Algorithmica* **34**, 181–196 (2002)
 - $\text{maxcard}(\text{subset})|\text{on-line}, \text{conservative}| R_A \text{bound}$.
10. L. Babel, B. Chen, H. Kellerer, V. Kotov, On-line algorithms for cardinality constrained bin packing problems, in *ISAAC 2001*, Christchurch, New Zealand. Lecture Notes in Computer Science, vol. 2223 (Springer, 2001), pp. 695–706. This is the preliminary version of [11]
11. L. Babel, B. Chen, H. Kellerer, V. Kotov, On-line algorithms for cardinality constrained bin packing problems. *Discret. Appl. Math.* **143**, 238–251 (2004)
 - $\text{pack}|\text{on-line}| R_A^\infty | s_i \leq 1/k$.
12. B.S. Baker, A new proof for the first-fit decreasing bin-packing algorithm. *J. Algorithms* **6**, 49–70 (1985)
 - $\text{pack}|\text{off-line}| R_A^\infty$.
13. B.S. Baker, E.G. Coffman Jr., A tight asymptotic bound for next-fit-decreasing bin-packing. *SIAM J. Algebra. Discret. Methods* **2**, 147–152 (1981)
 - $\text{pack}|\text{off-line}| R_A^\infty | s_i \leq 1/k$.
14. J. Balogh, J. Békési, G. Galambos, M.C. Markót, Improved lower bounds for semi-online bin packing problems. *Computing* **84**, 139–148 (2009)
 - $\text{pack}|\text{on-line}, \text{repack}| R_A^\infty \text{bounds}$.
15. J. Balogh, J. Békési, G. Galambos, New lower bounds for certain bin packing algorithms, in *WAOA 2010*, Liverpool, UK. Lecture Notes in Computer Science, vol. 6534 (Springer, 2011), pp. 25–36
 - $\text{pack}|\text{on-line}, \text{off-line}| R_A^\infty \text{bound}$.
16. N. Bansal, Z. Liu, A. Sankar, Bin-packing with fragile objects, in *2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)*, vol. 223 (Montréal, Québec, Canada, 2001), pp. 38–46
17. N. Bansal, Z. Liu, A. Sankar, Bin-packing with fragile objects and frequency allocation in cellular networks. *Wirel. Netw.* **15**, 821–830 (2009)

- $\text{pack}|\text{off-line}|R_A^\infty|\text{controllable}.$
18. A. Bar-Noy, R.E. Ladner, T. Tamir, Windows scheduling as a restricted version of bin packing. *ACM Trans. Algorithms* **3**, 1–22 (2007)
- $\text{pack}|\text{off-line}|R_A|\text{discrete}.$
19. Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an ancient scheduling problem, in *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, Victoria, Canada, 1992, pp. 51–58
- $\text{mincap}|\text{on-line}|R_A\text{bound}.$
20. Y. Bartal, H. Karloff, Y. Rabani, A better lower bound for on-line scheduling. *Inf. Process. Lett.* **50**, 113–116 (1994)
- $\text{mincap}|\text{on-line}|R_A\text{bound}.$
21. W. Bein, J.R. Correa, X. Han, A fast asymptotic approximation scheme for bin packing with rejection. *Theor. Comput. Sci.* **393**, 14–22 (2008)
- $\text{pack}|\text{off-line}|PTAS|\{B_i\}, \text{controllable}.$
22. J. Békési, G. Galambos, A 5/4 linear time bin packing algorithm. Technical report OR-97-2, Teachers Trainer College, Szeged, Hungary, 1997. This is the preliminary version of [23]
23. J. Békési, G. Galambos, H. Kellerer, A 5/4 linear time bin packing algorithm. *J. Comput. Syst. Sci.* **60**, 145–160 (2000)
- $\text{pack}|\text{off-line, linear-time}|R_A^\infty.$
24. R. Berghammer, F. Reuter, A linear approximation algorithm for bin packing with absolute approximation factor 3/2. *Sci. Comput. Program.* **48**, 67–80 (2003)
- $\text{pack}|\text{linear-time}|R_A.$
25. V. Bilo, On the packing of selfish items, in *Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece (IEEE, 2006), pp. 25–29
- $\text{pack}|\text{repack}|R_A^\infty.$
26. J. Blazewicz, K. Ecker, A linear time algorithm for restricted bin packing and scheduling problems. *Oper. Res. Lett.* **2**, 80–83 (1983)
- $\text{mincap}, \text{pack}|\text{off-line, linear-time}|running-time|\text{restricted}.$
27. J. Boyar, L.M. Favrholdt, The relative worst order ratio for online algorithms. *ACM Trans. Algorithms* **3**, 1–24 (2007)
- $\text{pack}, \text{maxcard}(\text{subset})|\text{on-line}.$
28. J. Boyar, K.S. Larsen, M.N. Nielsen, The accomodation function: a generalization of the competitive ratio. *SIAM J. Comput.* **31**, 233–258 (2001)
- $\text{maxcard}(\text{subset})|\text{on-line, conservative}|R_A\text{bound}|\text{restricted}.$
29. J. Boyar, L. Epstein, L.M. Favrholdt, J.S. Kohrt, K.S. Larsen, M.M. Pedersen, S. Wøhlk, The maximum resource bin packing problem, in *Fundamentals of Computation Theory*. Lecture Notes in Computer Science, vol. 3623 (Springer, Berlin/New York, 2005), pp. 387–408. This is the preliminary version of [30]
30. J. Boyar, L. Epstein, L.M. Favrholdt, J.S. Kohrt, K.S. Larsen, M.M. Pedersen, S. Wøhlk, The maximum resource bin packing problem. *Theor. Comput. Sci.* **362**, 127–139 (2006)
- $\text{maxpack}|\text{on-line, off-line}|R_A^\infty|s_i \leq 1/k.$
31. J. Boyar, L. Epstein, A. Levin, Tight results for Next Fit and Worst Fit with resource augmentation. *Theor. Comput. Sci.* **411**, 2572–2580 (2010)
- $\text{pack}|\text{on-line}|R_A^\infty|\text{stretching}.$
32. D.J. Brown, A lower bound for on-line one-dimensional bin-packing algorithms. Technical report R-864, University of Illinois, Coordinated Science Laboratory, Urbana, IL, 1979
- $\text{pack}|\text{on-line}|R_A^\infty\text{bound}.$
33. R.E. Burkard, G. Zhang, Bounded space on-line variable-sized bin packing. *Acta Cybern.* **13**, 63–76 (1997)
- $\text{pack}|\text{bounded-space}|R_A^\infty|\{B_i\}.$
34. A. Caprara, U. Pferschy, Worst-case analysis of the subset sum algorithm for bin packing. *Oper. Res. Lett.* **32**, 159–166 (2004)
- $\text{pack}|\text{off-line}|R_A^\infty\text{bound}|s_i \leq 1/k.$

35. W.-T. Chan, F.Y.L. Chin, D. Ye, G. Zhang, Y. Zhang, Online bin packing of fragile objects with application in cellular networks. *JOCO* **14**(4), 427–435 (2007)
 • $\text{pack}|\text{on-line}|R_A^\infty|\text{controllable}$.
36. J.W. Chan, T. Lam, P.W.H. Wong, Dynamic bin packing of unit fractions items. *Theor. Comput. Sci.* **409**, 521–529 (2008)
 • $\text{pack}|\text{on-line}, \text{dynamic}|R_A^\infty\text{bounds}|\text{discrete}$.
37. J.W. Chan, P.W.H. Wong, F.C.C. Yung, On dynamic bin packing: an improved lower bound and resource augmentation analysis. *Algorithmica* **53**, 172–206 (2009)
 • $\text{pack}|\text{on-line}, \text{dynamic}|R_A^\infty\text{bound}|\text{discrete, stretching}$.
38. B. Chandra, Does randomization help in on-line bin packing? *Inf. Process. Lett.* **43**, 15–19 (1992)
 • $\text{pack}|\text{on-line}|R_A^\infty\text{bound}$.
39. A.K. Chandra, D.S. Hirschler, C.K. Wong, Bin packing with geometric constraints in computer network design. *Oper. Res.* **26**, 760–772 (1978)
 • $\text{pack}|\text{off-line}|R_A^\infty$.
40. C. Chekuri, S. Khanna, A polynomial time approximation scheme for the multiple knapsack problem. *SIAM J. Comput.* **35**, 713–728 (2005)
41. B. Chen, A. van Vliet, G.J. Woeginger, New lower and upper bounds for on-line scheduling. *Oper. Res. Lett.* **16**, 221–230 (1994)
 • $\text{mincap}|\text{on-line}|R_A\text{bounds}$.
42. E.G. Coffman Jr., An introduction to combinatorial models of dynamic storage allocation. *SIAM Rev.* **25**, 311–325 (1983)
43. E.G. Coffman Jr., J. Csirik, Performance guarantees for one-dimensional bin packing, in *Handbook of Approximation Algorithms and Metaheuristics*, chapter 32, ed. by T. Gonzales (Taylor and Francis Books/CRC, Boca Raton, 2006), pp. 32–1–32–18
44. E.G. Coffman Jr., J. Csirik, A classification scheme for bin packing theory. *Acta Cybern.* **18**, 47–60 (2007)
 • $\text{pack}|\text{on-line}, \text{off-line}|R_A^\infty, R_A$.
45. E.G. Coffman Jr., J.Y.T. Leung, Combinatorial analysis of an efficient algorithm for processor and storage allocation. *SIAM J. Comput.* **8**, 202–217 (1979)
 • $\text{maxcard}(\text{subset})|\text{off-line}|R_A\text{bound}$.
46. E.G. Coffman Jr., G.S. Lueker, *Probabilistic Analysis of Packing and Partitioning Algorithms* (Wiley, New York, 1991)
47. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, An application of bin-packing to multiprocessor scheduling. *SIAM J. Comput.* **7**, 1–17 (1978)
 • $\text{mincap}|\text{off-line}|R_A\text{bound}$.
48. E.G. Coffman Jr., J.Y.T. Leung, D.W. Ting, Bin packing: maximizing the number of pieces packed. *Acta Inform.* **9**, 263–271 (1978)
 • $\text{maxcard}(\text{subset})|\text{off-line}|R_A$.
49. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Dynamic bin packing. *SIAM J. Comput.* **12**, 227–258 (1983)
 • $\text{pack}|\text{dynamic, repack}|R_A^\infty\text{bound}|s_i \leq 1/k$.
50. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin-packing: An updated survey, in *Algorithm Design for Computer System Design*, ed. by G. Ausiello, M. Lucertini, P. Serafini (Springer, Wien, 1984), pp. 49–106
51. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Bin packing with divisible item sizes. *J. Complex.* **3**, 405–428 (1987)
 • $\text{pack, cover, maxcard}(\text{subset})|\text{on-line}, \text{off-line}, \text{dynamic}|\{\mathcal{B}_i\}, \text{restricted}$.
52. E.G. Coffman Jr., M.R. Garey, D.S. Johnson, Approximation algorithms for bin packing: a survey, in *Approximation Algorithms for NP-Hard Problems*, ed. by D.S. Hochbaum (PWS Publishing Company, Boston, 1997), pp. 46–93
53. E.G. Coffman Jr., G. Galambos, S. Martello, D. Vigo, Bin packing approximation algorithms: Combinatorial analysis, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos (Kluwer, Boston, 1999)

54. E.G. Coffman Jr., J. Csirik, J.Y.-T. Leung, Variable-sized bin packing and bin covering, in *Handbook of Approximation Algorithms and Metaheuristics*, chapter 34, ed. by T. Gonzales (Taylor and Francis Books/CRC, Boca Raton, 2006), pp. 34–1–34–11
55. E.G. Coffman Jr., J. Csirik, J.Y.-T. Leung, Variants of classical one-dimensional bin packing, in *Handbook of Approximation Algorithms and Metaheuristics*, chapter 33, ed. by T. Gonzales (Taylor and Francis Books/CRC, Boca Raton, 2006), pp. 33–1–33–13
56. J. Csirik, An on-line algorithm for variable-sized bin packing. *Acta Inform.* **26**, 697–709 (1989)
 - $\text{pack}|\text{on-line}|R_A^\infty|\{B_i\}$.
57. J. Csirik, The parametric behaviour of the first fit decreasing bin-packing algorithm. *J. Algorithms* **15**, 1–28 (1993)
 - $\text{pack}|\text{off-line}|R_A^\infty|s_i \leq 1/k$.
58. J. Csirik, B. Imreh, On the worst-case performance of the Next-k-Fit bin-packing heuristic. *Acta Cybern.* **9**, 89–105 (1989)
 - $\text{pack}|\text{bounded-space}|R_A^\infty\text{bounds}$.
59. J. Csirik, D.S. Johnson, Bounded space on-line bin-packing: best is better than first, in *Proceedings of the 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, 1991, pp. 309–319. This is the preliminary version of [60]
60. J. Csirik, D.S. Johnson, Bounded space on-line bin-packing: best is better than first. *Algorithmica* **31**, 115–138 (2001)
 - $\text{pack}|\text{bounded-space}|R_A^\infty$.
61. J. Csirik, V. Totik, On-line algorithms for a dual version of bin packing. *Discret. Appl. Math.* **21**, 163–167 (1988)
 - $\text{cover}|\text{on-line}|R_A^\infty\text{bound}$.
62. J. Csirik, G.J. Woeginger, Online packing and covering problems, in *Online Algorithms: The State of the Art*, ed. by A. Fiat, G.J. Woeginger. Lecture Notes in Computer Science, vol. 1442 (Springer, Berlin, 1998), pp. 154–177
63. J. Csirik, G.J. Woeginger, Resource augmentation for online bounded space bin packing. *J. Algorithms* **44**, 308–320 (2002)
 - $\text{pack}|\text{bounded-space}|R_A^\infty\text{bound}|\text{stretching}$.
64. J. Csirik, G. Galambos, G. Turan, Some results on bin-packing, in *Proceedings of the EURO VI Conference*, Vienna, Austria, 1983, pp. 52
 - $\text{pack}|\text{on-line}, \text{off-line}|R_A^\infty$.
65. J. Csirik, D.S. Johnson, C. Kenyon, Better approximation algorithms for bin covering, in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, Washington, DC, 2001, pp. 557–566
 - $\text{cover}|\text{off-line}|PTAS$.
66. M. Demange, P. Grisoni, V.T. Paschos, Differential approximation algorithms for some combinatorial optimization problems. *Theor. Comput. Sci.* **209**, 107–122 (1998)
 - $\text{pack}|\text{off-line}$.
67. M. Demange, J. Monnot, V.T. Paschos, Bridging gap between standard and differential polynomial approximation: the case of bin-packing. *Appl. Math. Lett.* **12**, 127–133 (1999)
 - $\text{pack}|\text{off-line}|R_A^\infty$.
68. G. Dósa, The tight bound of first fit decreasing bin-packing algorithm is $FFD(I) \leq (11/9) OPT(I) + 6/9$, in *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*. ed. by B. Chen, M. Paterson, G. Zhang. Lecture Notes in Computer Science, vol. 4614 (Springer, Berlin, 2007), pp. 1–11
 - $\text{pack}|\text{off-line}|R_A^\infty$.
69. G. Dósa, Y. He, Bin packing problems with rejection penalties and their dual problems. *Inf. Comput.* **204**, 795–815 (2006)
 - $\text{pack}, \text{cover}|\text{on-line}, \text{off-line}|R_A^\infty, R_A|\text{controllable}$.
70. L. Epstein, Bin packing with rejection revisited, in *WAOA*, Zurich, Switzerland. Lecture Notes in Computer Science, vol. 4368 (Springer, 2006), pp. 146–159. This is the preliminary version of [73]

71. L. Epstein, Online bin packing with cardinality constraints. *SIAM J. Discret. Math.* **20**, 1015–1030 (2007)
 • $\text{pack}|\text{bounded-space}|R_A^\infty|\{B_i\}$, stretching, $\text{card}(B) \leq k$.
72. L. Epstein, On online bin packing with LIB constraints. *Nav. Res. Logist. (NRL)* **56**, 780–786 (2009)
 • $\text{pack}|\text{on-line}|R_A^\infty|s_i \leq 1/k$, controllable.
73. L. Epstein, Bin packing with rejection revisited. *Algorithmica* **56**, 505–528 (2010)
 • $\text{pack}|\text{off-line}, \text{bounded-space}|R_A^\infty \text{bound}$, PTAS|controllable.
74. L. Epstein, L.M. Favrholt, On-line maximizing the number of items packed in variable-sized bins. *Acta Cybern.* **16**, 57–66 (2003)
 • $\text{maxcard}(\text{subset})|\text{on-line}, \text{conservative}|R_A|\{B_i\}$.
75. L. Epstein, E. Kleiman, Resource augmented semi-online bounded space bin packing. *Discret. Appl. Math.* **157**, 2785–2798 (2009)
 • $\text{pack}|\text{bounded-space}, \text{repack}|R_A^\infty|\text{stretching}$.
76. L. Epstein, E. Kleiman, Selfish bin packing. *Algorithmica* (2011). To appear (online first: doi:10.1007/s00453-009-9348-6)
 • $\text{pack}|\text{repack}|R_A^\infty \text{bounds}$.
77. L. Epstein, A. Levin, More on online bin packing with two item sizes. *Discret. Optim.* **5**(4), 705–713 (2008)
 • $\text{pack}|\text{on-line}|R_A^\infty|\text{restricted}, s_i \leq 1/k$.
78. L. Epstein, A. Levin, On bin packing with conflicts. *SIAM J. Optim.* **19**, 1270–1298 (2008)
 • $\text{pack}|\text{on-line}, \text{off-line}|R_A|\text{mutex}$.
79. L. Epstein, R. van Stee, Multidimensional packing problems, in *Handbook of Approximation Algorithms and Metaheuristics*, chapter 35, ed. by T. Gonzales (Taylor and Francis Books/CRC, Boca Raton, 2006), pp. 35–1–35–15
80. L. Epstein, R. van Stee, Online bin packing with resource augmentation. *Discret. Optim.* **4**, 322–333 (2007)
 • $\text{pack}|\text{on-line}|R_A^\infty \text{bound}|\text{stretching}$.
81. L. Epstein, R. van Stee, Approximation schemes for packing splittable items with cardinality constraints, in *WAOA*, Eilat, Israel. Lecture Notes in Computer Science, vol. 4927 (Springer, 2008), pp. 232–245
 • $\text{pack}|\text{off-line}|PTAS|\text{controllable}$.
82. L. Epstein, C. Imreh, A. Levin, Class constrained bin packing revisited. *Theor. Comput. Sci.* **411**, 3073–3089 (2010)
 • $\text{pack}|\text{on-line}, \text{off-line}|R_A^\infty, \text{FPTAS}|\text{restricted}, \text{card}(B) \leq k \text{ colors}$.
83. U. Faigle, W. Kern, Gy. Turán, On the performance of on-line algorithms for partition problems. *Acta Cybern.* **9**, 107–119 (1989)
 • $\text{pack}, \text{mincap}|\text{on-line}|R_A^\infty \text{bound}|\text{restricted}$.
84. W. Fernandez de la Vega, G.S. Lueker, Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica* **1**, 349–355 (1981)
 • $\text{pack}|\text{off-line}|PTAS$.
85. L. Finlay, P. Manyem, Online LIB problems: heuristics for bin covering and lower bounds for bin packing. *RAIRO Rech. Oper.* **39**, 163–183 (2005)
 • $\text{pack}, \text{cover}|\text{on-line}|R_A^\infty|\text{controllable}$.
86. D.C. Fisher, Next-fit packs a list and its reverse into the same number of bins. *Oper. Res. Lett.* **7**, 291–293 (1988)
 • $\text{pack}|\text{bounded-space}$.
87. D.K. Friesen, Tighter bounds for the multifit processor scheduling algorithm. *SIAM J. Comput.* **13**, 170–181 (1984)
 • $\text{mincap}|\text{off-line}|R_A^\infty \text{bound}$.
88. D.K. Friesen, F.S. Kuhl, Analysis of a hybrid algorithm for packing unequal bins. *SIAM J. Comput.* **17**, 23–40 (1988)
 • $\text{maxcard}(\text{subset})|\text{off-line}|R_A|\{B_i\}$.

89. D.K. Friesen, M.A. Langston, Variable sized bin packing. *SIAM J. Comput.* **15**, 222–230 (1986)
 • $\text{pack}|\text{on-line}|R_A^\infty|\{B_i\}$.
90. D.K. Friesen, M.A. Langston, Analysis of a compound bin-packing algorithm. *SIAM J. Discret. Math.* **4**, 61–79 (1991)
 • $\text{pack}|\text{off-line}|R_A^\infty \text{bound}$.
91. G. Galambos, A new heuristic for the classical bin-packing problem. Technical report 82, Institute fuer Mathematik, Augsburg, 1985
 • $\text{pack}|\text{repack}|R_A^\infty \text{bound}|s_i \leq 1/k$.
92. G. Galambos, Parametric lower bound for on-line bin-packing. *SIAM J. Algebra. Discret. Meth.* **7**, 362–367 (1986)
 • $\text{pack}|\text{on-line}|R_A^\infty \text{bound}|s_i \leq 1/k$.
93. G. Galambos, J.B.G. Frenk, A simple proof of Liang's lower bound for on-line bin packing and the extension to the parametric case. *Discret. Appl. Math.* **41**, 173–178 (1993)
 • $\text{pack}|\text{on-line}|R_A^\infty \text{bound}|s_i \leq 1/k$.
94. G. Galambos, G.J. Woeginger, An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM J. Comput.* **22**, 345–355 (1993)
 • $\text{mincap}|\text{on-line}|R_A$.
95. G. Galambos, G.J. Woeginger, Repacking helps in bounded space on-line bin-packing. *Computing* **49**, 329–338 (1993)
 • $\text{pack}|\text{bounded-space}, \text{repack}|R_A^\infty \text{bound}$.
96. G. Galambos, G.J. Woeginger, On-line bin packing – a restricted survey. *Z. Oper. Res.* **42**, 25–45 (1995)
97. G. Gambosi, A. Postiglione, M. Talamo, New algorithms for on-line bin packing, in *Algorithms and Complexity*, ed. by R. Petreschi, G. Ausiello, D.P. Bovet (World Scientific, Singapore, 1990), pp. 44–59. This is the preliminary version of [99]
98. G. Gambosi, A. Postiglione, M. Talamo, On-line maintenance of an approximate bin-packing solution. *Nord. J. Comput.* **4**, 151–166 (1997)
 • $\text{pack}|\text{repack}|R_A^\infty$.
99. G. Gambosi, A. Postiglione, M. Talamo, Algorithms for the relaxed online bin-packing model. *SIAM J. Comput.* **30**, 1532–1551 (2000)
 • $\text{pack}|\text{on-line}, \text{repack}|R_A^\infty$.
100. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman, New York, 1979)
101. M.R. Garey, D.S. Johnson, Approximation algorithm for bin-packing problems: a survey, in *Analysis and Design of Algorithm in Combinatorial Optimization*, ed. by G. Ausiello, M. Lucertini (Springer, New York, 1981), pp. 147–172
102. M.R. Garey, D.S. Johnson, A 71/60 theorem for bin packing. *J. Complex.* **1**, 65–106 (1985)
 • $\text{pack}|\text{off-line}|R_A^\infty$.
103. M.R. Garey, R.L. Graham, J.D. Ullmann, Worst-case analysis of memory allocation algorithms, in *Proceedings of the 4th Annual ACM Symposium Theory of Computing*, Denver, CO (ACM, New York, 1972), pp. 143–150
104. M.R. Garey, R.L. Graham, D.S. Johnson, A.C.-C. Yao, Resource constrained scheduling as generalized bin packing. *J. Comb. Theory Ser. A* **21**, 257–298 (1976)
 • $\text{pack}|\text{on-line}|R_A^\infty \text{bound}|s_i \leq 1/k, \text{controllable}$.
105. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting-stock problem. *Oper. Res.* **9**, 849–859 (1961)
106. P.C. Gilmore, R.E. Gomory, A linear programming approach to the cutting stock problem – (Part II). *Oper. Res.* **11**, 863–888 (1963)
107. S.W. Golomb, On certain nonlinear recurring sequences. *Am. Math. Mon.* **70**, 403–405 (1963)
108. R.L. Graham, Bounds for certain multiprocessing anomalies. *Bell Syst. Tech. J.* **45**, 1563–1581 (1966)

109. R.L. Graham, Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* **17**, 263–269 (1969)
 • $\min cap|on-line, off-line|R_A$.
110. R.L. Graham, Bounds on multiprocessing anomalies and related packing algorithms, in *Proceedings of 1972 Spring Joint Computer Conference* (AFIPS Press, Montvale, 1972), pp. 205–217
 • $pack, \min cap|on-line, off-line|R_A bound$.
111. E.F. Grove, Online bin packing with lookahead, in *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA (SIAM, 1995), pp. 430–436
 • $pack|on-line, repack|R_A^\infty bound$.
112. G. Gutin, T.R. Jensen, A. Yeo, Batched bin packing. *Discret. Optim.* **2**, 71–82 (2005)
 • $pack|on-line, repack|R_A^\infty$.
113. G. Gutin, T.R. Jensen, A. Yeo, On-line bin packing with two item sizes. *Algorithm. Oper. Res.* **1**, 72–78 (2006)
 • $pack|on-line|R_A|restricted$.
114. L.A. Hall, Approximation algorithms for scheduling, in *Approximation Algorithms for NP-Hard Problems*, ed. by D.S. Hochbaum (PWS Publishing Company, Boston, 1997), pp. 1–45
115. D.S. Hochbaum (ed.), *Approximation Algorithms for NP-Hard Problems* (PWS Publishing Company, Boston, 1997)
116. D.S. Hochbaum, Various notions of approximation: good, better, best, and more, in *Approximation Algorithms for NP-Hard Problems*, ed. by D.S. Hochbaum (PWS Publishing Company, Boston, 1997), pp. 389–391
117. D.S. Hochbaum, D.B. Shmoys, A packing problem you can almost solve by sitting on your suitcase. *SIAM J. Algebra. Discret. Methods* **7**, 247–257 (1986)
 • $pack|off-line|complexity|restricted$.
118. D.S. Hochbaum, D.B. Shmoys, Using dual approximation algorithms for scheduling problems: theoretical and practical results. *J. ACM* **34**, 144–162 (1987)
 • $\min cap|off-line|R_A$.
119. M. Hofri, *Analysis of Algorithms* (Oxford University Press, New York, 1995)
120. Z. Ivković, E. Lloyd, Fully dynamic algorithms for bin packing: being myopic helps, in *Proceedings of the 1st European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 726 (Springer, New York, 1993), pp. 224–235. This is the preliminary version of [122]
121. Z. Ivković, E. Lloyd, Partially dynamic bin packing can be solved within $1 + \epsilon$ in (amortized) polylogarithmic time. *Inf. Process. Lett.* **63**, 45–50 (1997)
 • $pack|dynamic|PTAS$.
122. Z. Ivković, E. Lloyd, Fully dynamic algorithms for bin packing: being (mostly) myopic helps. *SIAM J. Comput.* **28**, 574–611 (1998)
 • $pack|dynamic, repack|R_A^\infty$.
123. K. Jansen, S. Öhring, Approximation algorithms for time constrained scheduling. *Inf. Comput.* **132**, 85–108 (1997)
 • $pack|off-line|R_A bound|mutex$.
124. K. Jansen, R. Solis-Oba, An asymptotic fully polynomial time approximation scheme for bin covering. *Theor. Comput. Sci.* **306**, 543–551 (2003)
 • $cover|off-line|FPTAS$.
125. D.S. Johnson, Fast allocation algorithms, in *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, New York, 1972, pp. 144–154
126. D.S. Johnson, *Near-Optimal Bin Packing Algorithms*. PhD thesis, MIT, Cambridge, MA, 1973
127. D.S. Johnson, Fast algorithms for bin packing. *J. Comput. Syst. Sci.* **8**, 272–314 (1974)
 • $pack|on-line, off-line|R_A^\infty|s_i \leq 1/k$.
128. D.S. Johnson, The NP-completeness column: an ongoing guide. *J. Algorithms* **3**, 89–99 (1982)

129. D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.* **3**, 256–278 (1974)
 • $\text{pack}|\text{on-line}, \text{off-line}|R_A^\infty, R_A|s_i \leq 1/k$.
130. J. Kang, S. Park, Algorithms for the variable sized bin packing problem. *Eur. J. Oper. Res.* **147**, 365–372 (2003)
 • $\text{pack}|\text{off-line}|R_A^\infty|\{B_i\}$.
131. D.R. Karger, S.J. Phillips, E. Torng, A better algorithm for an ancient scheduling problem. *J. Algorithms* **20**, 400–430 (1996)
 • $\text{mincap}|\text{on-line}|R_A$.
132. N. Karmarkar, R.M. Karp, An efficient approximation scheme for the one-dimensional bin-packing problem, in *Proceedings of the 23rd Annual IEEE Symposium on Foundations Computer Science*, Chicago, IL, 1982, pp. 312–320
 • $\text{pack}|\text{off-line}|FPTAS$.
133. G.Y. Katona, Edge disjoint polyp packing. *Discret. Appl. Math.* **78**, 133–152 (1997)
 • $\text{pack}|\text{on-line}|R_A^\infty \text{bounds}$.
134. H. Kellerer, A polynomial time approximation scheme for the multiple knapsack problem, in *RANDOM-APPROX*, Berkeley, CA. Lecture Notes in Computer Science, vol. 1671 (Springer, 1999), pp. 51–62
135. H. Kellerer, U. Pferschy, Cardinality constrained bin-packing problems. *Ann. Oper. Res.* **92**, 335–348 (1999)
 • $\text{pack}|\text{off-line}|R_A^\infty|\text{card}(B) \leq k$.
136. C. Kenyon, Best-fit bin-packing with random order, in *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms* Atlanta, GA (ACM/SIAM, Philadelphia, 1996), pp. 359–364
 • $\text{pack}|\text{on-line}$.
137. K.A. Kierstead, W.T. Trotter, An extremal problem in recursive combinatorics. *Congr. Numer.* **33**, 143–153 (1981)
138. N.G. Kinnersley, M.A. Langston, Online variable-sized bin packing. *Discret. Appl. Math.* **22**, 143–148 (1988–1989)
 • $\text{pack}|\text{on-line}|R_A^\infty|\{B_i\}$.
139. K.L. Krause, Y.Y. Shen, H.D. Schwetman, Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *J. ACM* **22**, 522–550 (1975)
 • $\text{pack}|\text{on-line}, \text{off-line}|R_A^\infty \text{bound}|\text{card}(B) \leq k$.
140. M.A. Langston, Improved 0/1 interchanged scheduling. *BIT* **22**, 282–290 (1982)
 • $\text{maxcard}(\text{subset})|\text{off-line}|R_A^\infty|\{B_i\}$.
141. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Sequencing and scheduling: algorithms and complexity, in *Logistics of Production and Inventory*, ed. by S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipkin. Handbooks in Operations Research and Management Science, vol. 4 (North-Holland, Amsterdam, 1993), pp. 445–522
142. C.C. Lee, D.T. Lee, A new algorithm for on-line bin-packing. Technical report 83-03-FC-02, Department of Electrical Engineering and computer Science Northwestern University, Evanston, IL, 1983
143. C.C. Lee, D.T. Lee, A simple on-line bin-packing algorithm. *J. ACM* **32**, 562–572 (1985)
 • $\text{pack}|\text{on-line}|R_A^\infty \text{bound}$.
144. H.W. Lenstra Jr., Integer programming with a fixed number of variables. *Math. Oper. Res.* **8**, 538–548 (1983)
145. J.Y.-T. Leung, M. Dror, G.H. Young, A note on an open-end bin packing problem. *J. Sched.* **4**, 201–207 (2001)
 • $\text{pack}|\text{on-line}, \text{off-line}, \text{open-end}|R_A^\infty \text{bound}; FPTAS$.
146. R. Li, M. Yue, The proof of $FFD(L) \leq 11/9 OPT(L) + 7/9$. *Chin. Sci. Bull.* **42**, 1262–1265 (1997)
 • $\text{pack}|\text{off-line}|R_A^\infty$.

147. F.M. Liang, A lower bound for on-line bin packing. *Inf. Process. Lett.* **10**, 76–79 (1980)
 • $\text{pack}|\text{on-line}|R_A^\infty\text{bound}$.
148. W.-P. Liu, J.B. Sidney, Bin packing using semi-ordinal data. *Oper. Res. Lett.* **19**, 101–104 (1996)
 • $\text{pack}|\text{off-line}|R_A^\infty$.
149. L. Lovász, M. Saks, W.T. Trotter, An on-line graph-coloring algorithm with sublinear performance ratio. *Discret. Math.* **75**, 319–325 (1989)
150. C.A. Mandal, P.P. Chakrabarti, S. Ghose, Complexity of fragmentable object bin packing and an application. *Comput. Math. Appl.* **35**, 91–97 (1998)
 • $\text{pack}|\text{off-line}| \text{running-time}|\text{controllable}$.
151. R.L. Manyem, P. Salt, M.S. Visser, Approximation lower bounds in online lib bin packing and covering. *J. Autom. Lang. Comb.* **8**, 663–674 (2003)
 • $\text{pack}|\text{on-line}|R_A^\infty\text{bound}|\text{controllable}$.
152. W. Mao, Best-k-fit bin packing. *Computing* **50**, 265–270 (1993)
 • $\text{pack}|\text{bounded-space}|R_A^\infty$.
153. W. Mao, Tight worst-case performance bounds for next-k-fit bin packing. *SIAM J. Comput.* **22**, 46–56 (1993)
 • $\text{pack}|\text{bounded-space}|R_A^\infty$.
154. C.U. Martel, A linear time bin-packing algorithm. *Oper. Res. Lett.* **4**, 189–192 (1985)
 • $\text{pack}|\text{off-line}, \text{linear-time}|R_A^\infty$.
155. N. Menakerman, R. Rom, Bin packing with item fragmentation, in *WADS*, Providence, RI. Lecture Notes in Computer Science, vol. 2125 (Springer, 2001), pp. 313–324
 • $\text{pack}|\text{on-line}, \text{off-line}|R_A|\text{controllable}$.
156. F.D. Murgolo, Anomalous behaviour in bin packing algorithms. *Discrete Appl. Math.* **21**, 229–243 (1988)
 • $\text{pack}|\text{on-line}, \text{off-line}, \text{conservative}$.
157. F.D. Murgolo, An efficient approximation scheme for variable-sized bin packing. *SIAM J. Comput.* **16**, 149–161 (1988)
 • $\text{pack}|\text{off-line}|FPTAS|\{B_i\}$.
158. N. Naaman, R. Rom, Packet scheduling with fragmentation, in *INFOCOM 2002*, New York, NY (IEEE, 2002)
 • $\text{pack}|\text{on-line}, \text{off-line}|R_A|\text{controllable}$.
159. P. Ramanan, D.J. Brown, C.C. Lee, D.T. Lee, On-line bin packing in linear time. *J. Algorithms* **10**, 305–326 (1989)
 • $\text{pack}|\text{on-line}, \text{linear-time}|R_A^\infty\text{bound}$.
160. M.B. Richey, Improved bounds for harmonic-based bin packing algorithms. *Discret. Appl. Math.* **34**, 203–227 (1991)
 • $\text{pack}|\text{on-line}, \text{linear-time}|R_A^\infty\text{bound}$.
161. S. Sahni, Algorithms for scheduling independent tasks. *J. ACM* **23**, 116–127 (1976)
162. S.S. Seiden, An optimal online algorithm for bounded space variable-sized bin packing. *SIAM J. Discret. Math.* **14**, 458–470 (2001)
 • $\text{pack}|\text{on-line}, \text{bounded-space}|R_A^\infty\text{bound}|\{B_i\}$.
163. S.S. Seiden, On the online bin packing problem. *J. ACM* **49**, 640–671 (2002)
 • $\text{pack}|\text{on-line}|R_A^\infty\text{bound}$.
164. S.S. Seiden, R. van Stee, L. Epstein, New bounds for variable-sized online bin packing. *SIAM J. Comput.* **33**, 455–469 (2003)
 • $\text{pack}|\text{on-line}|R_A^\infty\text{bound}|\{B_i\}$.
165. H. Shachnai, T. Tamir, Polynomial time approximation schemes for class-constrained packing problems. *J. Sched.* **4**, 313–338 (2001)
 • $\text{pack}|\text{off-line}|PTAS|\text{card}(B) \leq k\text{colors}$.
166. H. Shachnai, T. Tamir, Tight bounds for online class-constrained packing. *Theor. Comput. Sci.* **321**, 103–123 (2004)
 • $\text{pack}|\text{on-line}|R_A|\text{card}(B) \leq k\text{colors}$.

167. H. Shachnai, O. Yehezkeyl, Fast asymptotic FPTAS for packing fragmentable items with costs, in FCT, Budapest, Hungary. Lecture Notes in Computer Science, vol. 4639 (Springer, 2007), pp. 482–493
 • $\text{pack}|\text{off-line}| \text{FPTAS}|\text{controllable}$.
168. H. Shachnai, T. Tamir, O. Yehezkeyl, Approximation schemes for packing with item fragmentation. Theory Comput. Syst. **43**, 81–98 (2008)
 • $\text{pack}|\text{off-line}| \text{PTAS}|\text{controllable}$.
169. D. Simchi-Levi, New worst-case results for the bin packing problem. Nav. Res. Logist. Q. **41**, 579–585 (1994)
 • $\text{pack}|\text{on-line}, \text{off-line}| R_A^{\infty} \text{bound}$.
170. J. Sylvester, On a point in the theory of vulgar fractions. Am. J. Math. **3**, 332–335 (1880)
171. A. van Vliet, An improved lower bound for on-line bin packing algorithms. Inf. Process. Lett. **43**, 277–284 (1992)
 • $\text{pack}|\text{on-line}| R_A^{\infty} \text{bound} | s_i \leq 1/k$.
172. A. van Vliet, *Lower and Upper Bounds for On-Line Bin Packing and Scheduling Heuristic*. PhD thesis, Erasmus University, Rotterdam, 1995
173. A. van Vliet, On the asymptotic worst case behavoir of harmonic fit. J. Algorithms **20**, 113–136 (1996)
 • $\text{pack}|\text{on-line}, \text{bounded-space}| R_A^{\infty} \text{bound} | s_i \leq 1/k$.
174. T.S. Wee, M.J. Magazine, Assembly line balancing as generalized bin packing. Oper. Res. Lett. **1**, 56–58 (1982)
 • $\text{pack}|\text{off-line}| R_A^{\infty}$.
175. G.J. Woeginger, Improved space for bounded-space, on-line bin-packing. SIAM J. Discret. Math. **6**, 575–581 (1993)
 • $\text{pack}|\text{on-line}, \text{bounded-space}| R_A^{\infty}$.
176. E.C. Xavier, F.K. Miyazawa, The class constrained bin packing problem with applications to video-on-demand. Theor. Comput. Sci. **393**, 240–259 (2008)
 • $\text{pack}|\text{on-line}, \text{off-line}| R_A^{\infty}, \text{PTAS}|\text{card}(B) \leq k$.
177. J. Xie, Z. Liu, New worst-case bound of first-fit heuristic for bin packing problem, Unpublished manuscript.
 • $\text{pack}|\text{on-line}| R_A^{\infty} \text{bound}$.
178. K. Xu, *A Bin-Packing Problem with Item Sizes in the Interval $(0, \alpha]$ for $\alpha \leq \frac{1}{2}$* . PhD thesis, Chinese Academy of Sciences, Institute of Applied Mathematics, Beijing, China, 1993
179. K. Xu, The asymptotic worst-case behavior of the FFD heuristics for small items. J. Algorithms **37**, 237–246 (2000)
 • $\text{pack}|\text{off-line}| R_A^{\infty} | s_i \leq 1/k$.
180. A.C.-C. Yao, New algorithms for bin packing. J. ACM **27**, 207–227 (1980)
 • $\text{pack}|\text{on-line}, \text{off-line}| R_A^{\infty}$.
181. G. Yu, G. Zhang, Bin packing of selfish items, in WINE 2008, Shanghai, China. Lecture Notes in Computer Science, vol. 5385 (Springer, 2008)
 • $\text{pack}|\text{repack}| R_A^{\infty} \text{bounds}$.
182. M. Yue, On the exact upper bound for the multifit processor scheduling algorithm. Ann. Oper. Res. **24**, 233–259 (1991)
 • $\text{mincap}|\text{off-line}| R_A$.
183. M. Yue, A simple proof of the inequality $\text{FFD}(L) \leq \frac{11}{9}\text{OPT}(L) + 1 \forall L$ for the FFD bin packing algorithm. Acta Math. Appl. Sin. **7**, 321–331 (1991)
 • $\text{pack}|\text{off-line}| R_A^{\infty}$.
184. G. Zhang, Tight worst-case performance bound for AFB_k bin packing. Technical report 15, Institute of Applied Mathematics. Academia Sinica, Beijng, China, 1994. This is the preliminary version of [187]
185. G. Zhang, Worst-case analysis of the FFH algorithm for on-line variable-sized bin paking. Computing **56**, 165–172 (1996)
 • $\text{pack}|\text{on-line}| R_A^{\infty} | \{B_i\}$.

186. G. Zhang, A new version of on-line variable-sized bin packing. *Discret. Appl. Math.* **72**, 193–197 (1997)
 - *pack|on-line, off-line|R_A[∞]|\{B_i\}.*
187. G. Zhang, M. Yue, Tight performance bound for *AFB_k* bin packing. *Acta Math. Appl. Sin. Engl. Ser.* **13**, 443–446 (1997)
 - *pack|bounded-space|R_A[∞].*
188. G. Zhang, X. Cai, C.K. Wong, Linear time-approximation algorithms for bin packing. *Oper. Res. Lett.* **26**, 217–222 (2000)
 - *pack|on-line, off-line|R_A.*

Binary Unconstrained Quadratic Optimization Problem*

Gary A. Kochenberger, Fred Glover and Haibo Wang

Contents

1	Introduction	534
1.1	Rewriting into the Unified Framework.....	534
1.2	Accommodating General Linear Constraints.....	535
2	Solving UQP.....	540
3	Applications.....	542
4	Illustrative Computational Experience.....	542
4.1	The Task Allocation Problem.....	542
4.2	The Max 2-Sat Problem.....	543
4.3	The Group Technology Problem.....	544
4.4	The Set Packing Problem.....	545
4.5	The Set Partitioning Problem.....	546
4.6	The Linear Ordering Problem.....	547
4.7	The Max-Cut Problem.....	548
4.8	Comments on Computational Experience.....	549
4.9	Important Alternative Model for Assignment Problems.....	549
5	Promising New Application Areas.....	550
6	Conclusion.....	554
	Cross-References.....	555
	Recommended Reading.....	555

*This is an updated version of the previous paper given in [38].

G.A. Kochenberger (✉)

School of Business, University of Colorado, Denver, CO, USA

F. Glover

OptTek Systems, Inc, Boulder, CO, USA

H. Wang

College of Business Administration, Texas A&M International University, Laredo, TX, USA

Abstract

In recent years the unconstrained quadratic binary program (UQP) has emerged as a unified framework for modeling and solving a wide variety of combinatorial optimization problems. The unexpected versatility of the UQP model is opening doors to the solution of a diverse array of important and challenging applications. Developments in this evolving area are illustrated by describing its methodology with examples and by reporting substantial computational experience demonstrating the viability and robustness of latest methods for solving the UQP model, showing that they obtain solutions to wide-ranging instances of the model that rival or surpass the best solutions obtained by today's best special-purpose algorithms.

1 Introduction

The unconstrained quadratic binary program (UQP) has a lengthy history as an interesting and challenging combinatorial problem. Simple in its appearance, the model is given by

$$\text{UQP : } \text{Opt } x'Qx$$

where x is an n -vector of binary variables and Q is an n -by- n symmetric matrix of constants. Published accounts of this model go back at least as far as the 1960s in the work of Hammer and Rudeanu [31] and have applications in such diverse areas as spin glasses [18, 30], machine scheduling [1], the prediction of epileptic seizures [35], solving satisfiability problems [12, 13, 31, 33], and determining maximum cliques [13, 55, 56]. The application potential of UQP is much greater than might be imagined, due to the reformulation possibilities afforded by the use of quadratic infeasibility penalties as an alternative to imposing constraints in an explicit manner. In fact, any linear or quadratic discrete (deterministic) problem with linear constraints in bounded integer variables can *in principle* be recast in the form of UQP via the use of such penalties.

As will be shown, this outcome has more than theoretical significance. A broad range of challenging problems in combinatorial optimization can not only be reexpressed as UQP problems but can be solved in a highly effective manner when expressed this way. The process of reformulating a given combinatorial problem into an instance of UQP is easy to carry out. The common modeling framework that results, coupled with recently reported advances in solution methods for UQP, serve to make the model a viable alternative to more traditional combinatorial optimization models as illustrated in the sections that follow.

1.1 Recasting into the Unified Framework

For certain types of constraints, equivalent quadratic penalty representations are known in advance making it easy to embody the constraints within the UQP

objective function. For instance, let x_i and x_j be binary variables and consider the constraint

$$x_i + x_j \leq 1 \quad (1)$$

which precludes setting both variables to one simultaneously. A quadratic infeasibility penalty that imposes the same condition on x_i and x_j is

$$Px_i x_j \quad (2)$$

where P is a large positive scalar. This penalty function is positive when both variables are set to one (i.e., when (1) is violated), and otherwise the function is equal to zero. For a minimization problem then, adding the penalty function to the objective function is an alternative equivalent to imposing the constraint of (1) in the traditional manner.

In the context of transformations involving UQP, a penalty function is said to be a *valid infeasible penalty (VIP)* if it is zero for feasible solutions and otherwise positive. Including quadratic VIPs in the objective function for each constraint in the original model yields a transformed model in the form of UQP. VIPs for several commonly encountered constraints are given below (where x and y are binary variables and P is a large positive scalar):

Note that the penalty term in each case is zero if the associated constraint is satisfied, and otherwise the penalty is positive. These penalties, then, can be directly employed as an alternative to explicitly introducing the original constraints. For other more general constraints, however, VIPs are not known in advance and need to be “discovered.” A simple procedure for finding an appropriate VIP for any linear constraint is given in the next section.

1.2 Accommodating General Linear Constraints

To recast a constrained problem in the form of UQP when the VIPs are not known in advance, consider as a starting point the general constrained problem

$$\begin{aligned} & \min x_0 = xQx \\ & \text{subject to} \\ & \quad Ax = b, \quad x \text{ binary} \end{aligned} \quad (3)$$

This model accommodates both quadratic and linear objective functions since the linear case results when Q is a diagonal matrix (observing that $x_j^2 = x_j$ when x_j is a 0-1 variable). Under the assumption that A and b have integer components, problems with inequality constraints can also be put in this form by representing their bounded slack variables by a binary expansion. These constrained quadratic optimization models are converted into equivalent UQP models by adding a quadratic infeasibility penalty function to the objective function in place of explicitly imposing the constraints $Ax = b$.

Specifically, for a positive scalar P ,

$$\begin{aligned} x_0 &= xQx + P(Ax - b)^t(Ax - b) \\ &= xQx + xDx + c \\ &= x\hat{Q}x + c \end{aligned} \tag{4}$$

where the matrix D and the additive constant c result directly from the matrix multiplication indicated. Dropping the additive constant, the equivalent unconstrained version of the constrained problem becomes

$$UQP : \min x\hat{Q}x, \quad x \text{ binary} \tag{5}$$

From a theoretical standpoint, a suitable choice of the penalty scalar P can always be chosen so that the optimal solution to UQP is the optimal solution to the original constrained problem. Remarkably, as demonstrated later, it is often easy to find such a suitable value in practice as well.

The preceding general transformation that transforms (3) and (4) into (5) will be called *Transformation 1*. A fuller discussion of this transformation along with related material can be found in [13, 32, 34]. Transformation 1 provides the general procedure alluded to earlier that can in principle be employed to transform any problem in the form of (3) into an equivalent instance of UQP .

For problems where VIPs are known in advance, as by the penalty transformations given in Table 1, it is usually preferable to use the known VIP directly rather than applying Transformation 1. One special constraint in particular

$$x_j + x_k \leq 1$$

where the VIP takes the simple form Px_jx_k appears in many important applications. Due to the broad applicability of this constraint, it is convenient to refer to this special case as *Transformation 2*.

This process of transforming a given problem into the unified framework of xQx is illustrated by the following three examples.

Example 1 Set Packing

Set packing problems have the form $\text{Max } cx : Ax \leq e$ and x binary where A is a matrix of 0s and 1s and c and e are both vectors of 1s. These problems are important in the field of combinatorial optimization due to their application potential and their computational challenge. Consider the following example:

$$\begin{aligned} \max x_0 &= x_1 + x_2 + x_3 + x_4 \\ \text{st} \\ x_1 + x_3 + x_4 &\leq 1 \\ x_1 + x_2 &\leq 1 \end{aligned}$$

Table 1 Known penalties

Classical constraint	Equivalent penalty (VIP)
$x + y \leq 1$	$P(xy)$
$x + y \geq 1$	$P(1 - x - y + xy)$
$x + y = 1$	$P(1 - x - y + 2xy)$
$x \leq y$	$P(x - xy)$
$x_1 + x_2 + x_3 \leq 1$	$P(x_1x_2 + x_1x_3 + x_2x_3)$

The VIPs in [Table 1](#) make it possible to immediately recast this classical model into the UQP unified framework. Representing the positive scalar penalty P by $2M$, the equivalent unconstrained problem is

$$\max x_0 = x_1 + x_2 + x_3 + x_4 - 2Mx_1x_3 - 2Mx_1x_4 - 2Mx_3x_4 - 2Mx_1x_2$$

which can be rewritten as

$$\max (x_1 \ x_2 \ x_3 \ x_4) \begin{bmatrix} 1 & -M & -M & -M \\ -M & 1 & 0 & 0 \\ -M & 0 & 1 & -M \\ -M & 0 & -M & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$$

This model has the form $\max x'Qx$ where Q , as shown above, is a square, symmetric matrix. The procedure illustrated here can be used with any set packing problem and has proven to be an effective approach for solving problems with thousands of variables and constraints (see [\[4\]](#)).

Example 2 Set Partitioning

The classical set partitioning problem has the form $\text{Min } dx: Ax = e$, x binary where again A is a matrix of 0s and 1s, e is a vector of 1s, and (in contrast to the vector c of set packing problems) d is a vector of integers, typically nonnegative. The set partitioning problem is found in applications that range from vehicle routing to crew scheduling [\[36, 51\]](#). As an illustration, consider the following small example:

$$\min x_0 = 3x_1 + 2x_2 + x_3 + x_4 + 3x_5 + 2x_6$$

subject to

$$\begin{aligned} x_1 + x_3 + x_6 &= 1 \\ x_2 + x_3 + x_5 + x_6 &= 1 \\ x_3 + x_4 + x_5 &= 1 \\ x_1 + x_2 + x_4 + x_6 &= 1 \end{aligned}$$

and x binary. Applying Transformation 1 with $P = 10$ gives the equivalent UQP model:

$$\min x \hat{Q}x, \quad x \text{ binary}$$

where the additive constant, c , is 40 and the symmetric \hat{Q} matrix is

$$\hat{Q} = \begin{bmatrix} -17 & 10 & 10 & 10 & 0 & 20 \\ 10 & -18 & 10 & 10 & 10 & 20 \\ 10 & 10 & -29 & 10 & 20 & 20 \\ 10 & 10 & 10 & -19 & 10 & 10 \\ 0 & 10 & 20 & 10 & -17 & 10 \\ 20 & 20 & 20 & 10 & 10 & -28 \end{bmatrix}$$

Solving this UQP formulation provides an optimal solution $x_1 = x_5 = 1$ (with all other variables equal to 0) to yield $x_0 = 6$. In the straightforward application of Transformation 1 to this example, the replacement of the original problem formulation by the UQP model does not require any new variables to be introduced. Set partitioning problems with thousands of variables have been successfully solved by reformulating them in this manner, as reported in [47].

In many applications, Transformations 1 and 2 can be used in concert to produce an equivalent UQP model, as demonstrated next.

Example 3 The K-Coloring Problem

Vertex coloring problems seek to assign colors to nodes of a graph in such a way that adjacent nodes receive different colors. The K-coloring problem attempts to find such a coloring using exactly K colors. A wide range of applications, ranging from frequency assignment problems to printed circuit board design problems, can be represented by the K-coloring model.

These problems can be modeled as satisfiability problems using the assignment variables as follows:

Let x_{ij} be 1 if node i is assigned color j , and 0 otherwise.

Since each node must be colored,

$$\sum_{j=1}^K x_{ij} = 1 \quad i = 1, \dots, n \quad (6)$$

where n is the number of nodes in the graph. A feasible coloring, in which adjacent nodes are assigned different colors, is assured by imposing the constraints

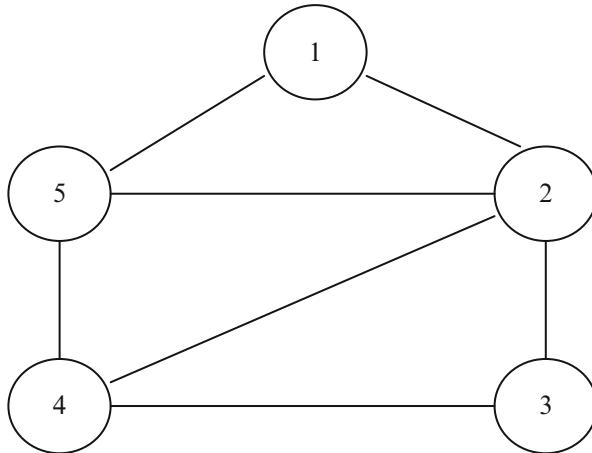
$$x_{ip} + x_{jp} \leq 1 \quad p = 1, \dots, K \quad (7)$$

for all adjacent nodes (i, j) in the graph.

This problem can be recast in the form of UQP by using Transformation 1 on the assignment constraints of (6) and Transformation 2 on the adjacency constraints of (7). No new variables are required. Since the resulting model has no explicit objective function, any positive value for the penalty P will do. The following example gives a concrete illustration of the reformulation process.

Find a feasible coloring of the following graph using three colors.

Thus, the goal is to find a solution to the system:



$$x_{i1} + x_{i2} + x_{i3} = 1 \quad i = 1, 5 \quad (8)$$

$$x_{ip} + x_{jp} \leq 1 \quad p = 1, 3 \quad (9)$$

(for all adjacent nodes i and j)

In this traditional form, the model has 15 variables and 26 constraints. To recast this problem in the form of UQP, it suffices to use Transformation 1 on the equations of (8) and Transformation 2 on the inequalities of (9). Arbitrarily choosing the penalty P to be 4, the equivalent problem in unified form is given by

$$\text{UQP}(Pen) : \min x \hat{Q} x$$

where the \hat{Q} matrix is

$$\hat{Q} = \begin{bmatrix} -4 & 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 4 & -4 & 4 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 4 & 4 & -4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 \\ 4 & 0 & 0 & -4 & 4 & 4 & 4 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 4 & -4 & 4 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 4 & 4 & 4 & -4 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 4 & 0 & 0 & -4 & 4 & 4 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 4 & -4 & 4 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 4 & 4 & -4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & -4 & 4 & 4 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 0 & 4 & -4 & 4 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 4 & 4 & 4 & -4 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 4 & 4 & 4 & 4 & -4 & 0 & 0 & 4 \\ 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & -4 & 4 & 4 & 4 \\ 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 4 & 0 & 4 & -4 & 4 \\ 0 & 0 & 4 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 4 & 4 & 4 & 4 & 4 & -4 \end{bmatrix}$$

Solving this unconstrained model, $x\hat{Q}x$, yields the feasible coloring:

$$x_{11}, x_{22}, x_{33}, x_{41}, x_{53}, = 1 \text{ all other } x_{ij} = 0$$

This approach to coloring problems has proven to be very effective for a wide variety of coloring instances from the literature as disclosed in [41].

2 Solving UQP

Employing the UQP unified framework to solve combinatorial problems requires the availability of a solution method for xQx . The recent literature reports major advances in such methods involving modern metaheuristic methodologies. The reader is referred to references [6–9, 14, 16, 24–26, 37, 45, 49, 50, 53, 54] for a description of some of the methods that have provided valuable contributions to the area. The pursuit of further advances in solution methods for xQx remains an active research arena.

The computational work reported later in this chapter derives from previous studies of three methods: a basic tabu search method due to Glover, Kochenberger, and Alidaee [23–25], a tabu search method due to Lewis [47], and the more recent tabu search method of Glover, Jin-Kao, and Lu [29]. For convenience these methods will be referred to as methods 1, 2, and 3 respectively. A brief overview of each approach is given below. Complete details are provided in the aforementioned references.

Method 1 Overview: This tabu search metaheuristic for UQP is centered around the use of strategic oscillation, which constitutes one of the primary strategies of tabu search. The method alternates between constructive phases that progressively

set variables to 1 (whose steps are called “add moves”) and destructive phases that progressively set variables to 0 (whose steps are called “drop moves”). To control the underlying search process, the method uses a memory structure that is updated at *critical events*, identified by conditions that generate a subclass of locally optimal solutions. Solutions corresponding to critical events are called *critical solutions*.

A parameter SPAN is used to indicate the amplitude of oscillation about a critical event. To begin *span* is set equal to 1 and then is gradually increased until reaching some limiting value. For each value of *span*, a series of alternating constructive and destructive phases is executed before progressing to the next value. At the limiting point, *span* is gradually decreased, allowing again for a series of alternating constructive and destructive phases. When *span* reaches a value of 1, a *complete span cycle* has been completed and the next cycle is launched. The search process is typically allowed to run for a preset number of SPAN cycles.

Information stored at critical events is used to influence the search process by penalizing potentially attractive add moves (during a constructive phase) and inducing drop moves (during a destructive phase) associated with assignments of values to variables in recent critical solutions. Cumulative critical event information is used to introduce a subtle long-term bias into the search process by means of additional penalties and inducements similar to those discussed above. Other standard elements of tabu search such as short- and long-term memory structures are also included.

Method 2 Overview: This method is a modification of the previous method that implements a basic multi-start, tabu search procedure with path-relinking. A local search is performed using a 1-opt mechanism. When no improvements to the current solution can be found using 1-opt local search, including tabu aspiration possibilities, a path-relinking procedure is initiated between the current solution and an elite set of diverse solutions saved during the search process. After relinking, a random portion of the current solution is perturbed and testing continues. If after a number of iterations (specified by the restart limit) no improvements are found, then a larger perturbation is invoked based on long-term memory.

Method 3 Overview: The DDTs method repeatedly alternates between a simple version of tabu search (TS) and a diversification phase founded on a memory-based perturbation operator. Starting from an initial random solution, DDTs uses the TS procedure to reach a local optimum. Then, the perturbation operator is applied to displace the solution to a new region, whereupon a new round of TS is launched. To facilitate achieving effective diversification, the perturbation operator is guided by information from a special memory structure.

This tabu search procedure uses a neighborhood defined by the single 1-flip moves, which consists of changing (flipping) the value of a single variable x_j to its complement value $1 - x_j$. The implementation of this neighborhood uses a fast incremental evaluation technique to calculate the cost of candidate moves. The diversification strategy utilizes a memory-based perturbation operator composed of three parts: a flip frequency memory, an elite solution memory, and an elite value frequency memory. These memory structures are used jointly by the perturbation operator to enhance the diversification of the search process. This method has proven

to be highly effective for solving large instances of UQP. Complete details of this method are given in Glover, Lu, and Hao [29].

3 Applications

To date several important classes of combinatorial problems have been successfully modeled and solved by employing the unified framework. Results with the unified framework applied to these problems have been uniformly attractive in terms of both solution quality and computation times. While the three solution methods described above are designed for the completely general form of UQP, without any specialization to take advantage of particular types of problems reformulated in this general representation, the outcomes typically prove competitive with or even superior to those of specialized methods designed for the specific problem structure at hand. The broad base of experience with UQP as a modeling and solution framework obtained by applying the three methods above includes a substantial range of problem classes including quadratic assignment problems, capital budgeting problems, multiple knapsack problems, task allocation problems (distributed computer systems), maximum diversity problems, p-median problems, asymmetric assignment problems, symmetric assignment problems, side constrained assignment problems, quadratic knapsack problems, constraint satisfaction problems (CSPs), set partitioning problems, fixed charge warehouse location problems, maximum clique problems, maximum independent set problems, maximum cut problems, graph coloring problems, graph partitioning problems, number partitioning problems, and-linear ordering problems.

Additional test problems representing a variety of other applications have also been reformulated and solved via UQP. The section below reports specific computational experience with some large-scale applications. [Section 5](#) then suggests promising new applications areas for UQP.

4 Illustrative Computational Experience

The following summarizes results obtained on large-scale test problems of a variety of well-known problem classes from combinatorial optimization. Because methods 1, 2, and 3 were developed at different points in time, and each has been applied to classes of problems different from those treated by the other two, outcomes for these classes of problems are reported by describing the findings for the three methods in the sequence in which they have been applied.

4.1 The Task Allocation Problem

The task allocation problem, which consists of assigning tasks to processors, can be described as follows: Let $P = \{P_1, P_2, \dots, P_m\}$ be a set of distributed processors

and $T = \{T_1, T_2, \dots, T_n\}$ be a set of tasks to be run on the processors. Let c_{ij} be the communications cost between tasks T_i and T_j and q_{tp} be the execution cost of task t on processor p . Then stipulating that x_{tp} equals 1 if task T_t is assigned to processor P_p and is otherwise equal to 0, the model becomes

$$\begin{aligned} & \min \sum_{t=1}^n \sum_{p=1}^m q_{tp} x_{tp} + \sum_{i < j} \sum_{p=1}^m c_{ij} x_{ip} (1 - x_{jp}) \\ & \text{st} \\ & \quad \sum_{p=1}^m x_{tp} = 1 \quad \text{for } t = 1, \dots, n \end{aligned}$$

which is of the form

$$\min x'Qx$$

st

$$Ax = b$$

Applying Transformation 1 of Sect. 1.2 yields a model in the form of UQP. In a study [46] comparing CPLEX 8.1 and method 1 on 16 large test problems ranging in size from 1,000 to 3,000 variables, CPLEX was unable to prove optimality on any of the problems (within a 48-h limit). Running method 1 for a total of 300 SPAN cycles took less than 12 min on the largest of the problems. Across all 16 problems, the best solutions produced by CPLEX had objective function values that were, on average, 52 % inferior to those produced by method 1, in spite of being allowed to run 240 times longer.

4.2 The Max 2-Sat Problem

It is well known (see for instance [13] and [33]) that the Max 2-Sat problem can be formulated as an unconstrained binary quadratic program of the form $\min x'Qx$. Yet little computational experience treating the Max 2-Sat problem as an instance of UQP has appeared in the literature prior to the computational study of [42] which reports on the successful application of the tabu search Method 1 to a variety of old and new test problems from this class. In all cases, method 1 was run for 50 SPAN cycles. Applied to a public data set with 16 problems ranging from 50 variables and 100 clauses to 150 variables and 600 clauses, method 1 found best known solutions in less than 3 s while more than half of these problem instances could not be solved within a 12-h limit by the well-known Maxsat [10] procedure.

On another publicly available set of 20 problems with 200 variables and 1,000–2,000 clauses, method 1 again found best known solutions in less than 3 s. On a third data set of 17 problem instances ranging in size from 100 to 1,000 variables and 626 to 22,883 clauses, method 1 was tested against CPLEX 8.0 by applying the latter to the classical MIP formulation of Max 2-Sat. CPLEX was only able to solve and terminate naturally on the smallest of these problems, requiring 147 s in

this case. Method 1 by contrast found the optimal solution for this problem in less than 1 s. On the other 16 problems, CPLEX was unable to terminate naturally within a 10-h time limit while the UQP approach of method 1 obtained solutions within 50 SPAN cycles in an average of less than 9 s. For these problems, the best solutions found in the 10-h time limit by CPLEX had objective function values on average 20 % inferior to those produced by the method 1 tabu search approach. Full details of these results are given in [42].

4.3 The Group Technology Problem

The group technology (GT) problem is concerned with clustering machines and parts together in a manner that facilitates economies in time and cost. From a graph theoretic point of view, nodes in a graph can be taken to represent machines and parts, which are connected by edges denoting the association of each pair of nodes in the network. The GT problem then becomes one of partitioning the nodes into cliques with similar characteristics. Thus, the GT problem can be modeled by the standard IP formulation for clique partitioning:

$$\begin{aligned} & \max \sum_{(i,j) \in E} w_{ij} x_{ij} \\ \text{st} \quad & x_{ij} + x_{ir} - x_{jr} \leq 1 \quad \forall \text{ all distinct } i, j, r \in V \\ & x_{ij} \in \{0, 1\} \text{ for all } \{i, j\} \in E \end{aligned}$$

The variable x_{ij} is equal to 1 if machine (or part) i and machine (or part) j are assigned in a cell and is equal to 0 otherwise. The coefficient w_{ij} is the weight of the edge (i, j) in the graph.

Since the variables are associated with edges in the graph, the model has many variables and constraints. In practice even modest-sized GT problems give rise to extraordinarily large IP models and often are beyond the capability of modern MIP solvers to handle. As an alternative to the standard IP model for clique partitioning, it is shown in [6] that the GT problem can be solved by employing the following quadratic model for clique partitioning:

$$\begin{aligned} & \max \sum_{i=1}^{n-1} \sum_{j=i+1}^n w_{ij} \sum_{k=1}^{K_{\max}} x_{ik} x_{jk} \\ \text{st} \quad & \sum_{k=1}^{K_{\max}} x_{ik} = 1 \quad \text{for } i = 1, n \end{aligned}$$

In this formulation n is the number of nodes, w_{ij} again denotes the weight of edge (i, j) , K_{\max} is an upper bound on the number of groups to be formed, and $x_{ik} = 1$

if node i is assigned to clique k and otherwise equals zero. It is to be noted that variables are associated with nodes rather than edges in this model. Thus, in spite of being nonlinear, the model is much smaller than the standard IP model and is readily approached via the unified framework. Specifically, the problem has the form

$$\begin{aligned} \max \quad & x'Qx \\ \text{st} \quad & \\ & Ax = b \end{aligned}$$

and thus can be recast in the form of UQP using Transformation 1 of [Sect. 1.2](#).

To test the effectiveness of the UQP approach for modeling GT, experiments were conducted on 36 standard test problems of modest size ranging from graphs with 46 nodes and 1,035 edges to 71 nodes and 2,485 edges. Each problem was solved with method 1 running for 100 SPAN cycles, with the largest problem taking slightly more than 1 min. To provide a benchmark for comparison, each problem was solved in its standard MIP formulation by CPLEX 6.5. CPLEX's time performance on these problems was very erratic. The average solution time was 1.2 days, and 4 problems required more than 4 days. Nonetheless, CPLEX terminated naturally on all 36 problems and thus confirmed the optimality of the solutions obtained by the UQP approach although requiring running times that were 1,500–6,000 times longer. Complete details are given in Wang et al. [61].

4.4 The Set Packing Problem

[Example 1](#) of [Sect. 1](#) of this chapter presented a small example of a set packing problem, illustrating how this class of problems can easily be reformulated as an unconstrained quadratic binary program. In general, this class of problems is given by

$$\begin{aligned} \max \quad & \sum_{j=1}^n w_j x_j \\ \text{st} \quad & \\ & \sum_{j=1}^n a_{ij} x_j \leq 1 \quad \text{for } i = 1, \dots, m \end{aligned}$$

where the a_{ij} are 0/1 coefficients, the w_j are weights, and the x_j variables are binary. Each constraint can alternatively be enforced by subtracting one or more quadratic penalties from the objective function, thus producing an equivalent problem in the form of an unconstrained quadratic binary program. Note that this recasting takes place without the introduction of new variables. Moreover, the size of the equivalent UQP model depends only on the number of original variables and is independent of the number of constraints in the original set packing problem.

Computational results are given in [4] for applying method 1 to a set of publicly available set of 16 test problems containing 1,000–2,000 variables and 2,000–10,000 constraints. Best known results for these problems reported in the literature

are obtained by a leading heuristic (see [19]) specially designed for set packing problems. This special procedure was allowed to run for 5 h on each problem. For each problem, method 1 was run for 1,000 SPAN cycles, taking 20 min for the largest instance and 7 min on average. For the 16 test problems, method 1 quickly found the best known solutions for 13 of the 16 problems and produced solutions whose objective function values were more than 99 % of the best known values for the other 3 problems. To provide additional comparison, these problems were also solved using CPLEX 8.1 which consumed on average 38 h of computer run time before terminating due to reaching memory limitations. The reader is referred to [4] for complete details of these runs as well as additional computational experience.

4.5 The Set Partitioning Problem

As noted in [Example 2](#) of [Sect. 1](#), the set partitioning problem can be formulated as

$$\begin{aligned} \max \quad & \sum_{j=1}^n c_j x_j \\ \text{st} \quad & \sum_{j=1}^n a_{ij} x_j = 1 \quad \text{for } i = 1, \dots, m \end{aligned}$$

where the a_{ij} are 0/1 coefficients, the c_j are objective function coefficients, and the x_j variables are binary. Computational experience with a wide variety of solution methods has shown that even modest-sized instances of these problems are extraordinarily difficult to solve and become increasingly difficult as density grows. Applying Transformation 1, the set partitioning problem becomes a UQP problem without introducing new variables and whose size is independent of the number of constraints in the original problem.

An equivalent but simpler alternative to Transformation 1 is presented in [47] for re-expressing set partitioning problems as UQP models. Computational experience is reported with a set of 31 test problems ranging in size from 600 to 15,000 variables and 100 to 5,000 constraints. CPLEX 8.1, applied to the classical formulation given above, was used to provide a benchmark for comparison with the basic tabu search and path relinking approach of method 2. For each problem, CPLEX was run until optimality was proven or until reaching 12 h of processing time (or until the run was terminated by an “out of memory” error). For the first 21 problems, CPLEX terminated naturally with an optimal solution. Method 2 quickly found an optimal solution to these problems as well with a time performance that was on average 270 times faster than CPLEX. (This excludes the time it took CPLEX to prove the optimality of the solutions it found).

For the largest problem (15,000 variables and 5,000 constraints), CPLEX terminated with a memory fault and no solution while method 2 found a feasible solution in less than 20 min. For the other problems, CPLEX terminated due to the

12-h time limit, giving a “best solution found so far.” On these problems, method 2 obtained solutions with objective functions approximately 2 % superior to those obtained by CPLEX while running more than 100 times faster.

4.6 The Linear Ordering Problem

The linear ordering problem (LOP) is a particularly hard problem defined by an n-by-n matrix of weights $C = \{c_{ij}\}$ where the goal is to find a permutation, p , of columns (and rows) that maximizes the sum of the weights in the upper triangular matrix. Such problems arise in a variety of settings (such as finding an acyclic tournament of maximum weight, or the aggregate ordering of paired observations) but are most often associated with the triangulation of input-output matrices in economics where the data in question often refer to sectors. The problem can be modeled utilizing the decision variable: $x_{ij} = 1$ if sector i goes before sector j in the permutation; and $x_{ij} = 0$ otherwise. Taking advantage of the fact that $x_{ij} + x_{ji} = 1$ for all i and j , a standard integer programming formulation for the problem is given by

$$\begin{aligned} & \text{Max } \sum_{i < j} c_{ij} x_{ij} + \sum_{j < i} c_{ij} (1 - x_{ji}) \\ & \text{st} \\ & x_{ij} + x_{jk} - x_{ik} \leq 1 \quad \forall (i, j, k) : i < j < k \\ & x_{ij} + x_{jk} - x_{ik} \geq 0 \quad \forall (i, j, k) : i < j < k \\ & x_{ij} \in \{0, 1\} \quad \forall (i, j) : i < j \end{aligned}$$

This model can be recast into the form of UQP by employing a specially crafted quadratic penalty, a penalty that is unique to the structure of the linear ordering problem. To see how this special penalty arises, note that for a particular set $i < j < k$, the pair of constraints shown above allows 6 of the 8 possible solutions, excluding only $x_{ij} = 1$, $x_{jk} = 1$, and $x_{ik} = 0$ and $x_{ij} = 0$, $x_{jk} = 0$, and $x_{ik} = 1$. It is easy to see that an exact quadratic penalty that precludes these same two solutions, while allowing the others, is given by

$$P \{x_{ik} + x_{ij}x_{jk} - x_{ij}x_{ik} - x_{jk}x_{ik}\}$$

Thus, without introducing additional variables, this special penalty can be used to easily transform the linear ordering problem into an equivalent UQP. For a problem with n sectors, both the IP formulation and the equivalent UQP model will have $n(n - 1)/2$ variables. Again the size of the UQP model is independent of the number of constraints in the original IP model.

An application of this UQP approach is reported in [48] for a set of 11 test problems ranging in size from 190 variable and 2,280 constraints (a LOP instance with $n = 20$) to 19,900 variables and 2,626,800 constraints (a LOP instance with $n = 200$). As before, experience with CPLEX was reported on these problems to provide a benchmark of comparison. Both CPLEX and method 2 were given a time

limit of 1 h. Best known solutions, obtained from a specially crafted Scatter Search method for LOP problems, are also available for purposes of benchmarking.

Due to the large size of the IP models, CPLEX was able to find and prove optimality for only the two smallest problems. Method 2 found these same optimal solutions within the 1-h time limit. For the four largest problems (more than 4,000 variables and 234,960 constraints), CPLEX was unable to find a feasible solution within the 1-h time limit while the UQP approach readily found feasible solutions. For the other problems, CPLEX reported solutions that were inferior to those produced by the tabu search/path-relinking approach of method 2. Across the entire set of test problems, solutions were produced using the UQP model whose objective function values were at least 95 % of the best known values obtained by any method in the literature specialized for solving LOP problems. See [48] for a complete discussion of these results.

4.7 The Max-Cut Problem

Given an undirected graph $G(V,E)$ with edge weights w_{ij} , the Max-Cut (MC) problem seeks a partition $S_1 \subset V$ and $S_2 = V|S_1$ such that the weight of the cut, defined as the sum of the weights on the edges connecting the two sets, is maximized. MC is another classic problem in combinatorial optimization. This problem has a natural quadratic structure and with a simple change of variables, it can be readily put into the form of UQP. The common formulation from the literature is

$$\max \frac{1}{2} \sum_{1 \leq i < j \leq n} w_{ij}(1 - y_i y_j)$$

subject to

$$y_i \in \{-1, 1\} \quad \forall i \in V$$

The change of variables $y_i = 2x_i - 1$ yields the unconstrained quadratic binary program

$$\max \sum_{i < j} w_{ij}(x_i + x_j - 2x_i x_j); \quad x_j \in \{0, 1\}$$

which is of the form $\max x'Qx$. This class of problems is described in detail in [28] together with reporting computational experience for 69 test problems from the literature ranging in size from 800 variables to 10,000 variables. Most articles in the literature addressing MC only consider problems with up to a few thousand variables.

This study employed the tabu search approach of method 3. Depending on the problem size, run time limits ranged from one half hour for small problems to 24 h for the 10,000 variable problems. The results obtained were highly attractive compared to previously published results in the literature, as evidenced by the fact that, over the entire test bed of 69 problems, method 3 matched best known solutions on 19 problems, found new best known solutions on 46 problems, and failed to find best known solutions on just 4 problems.

4.8 Comments on Computational Experience

The computational experience reported above is intended to demonstrate the viability and breadth of applicability of the unified framework. This framework has been successfully applied to many other problem classes as well. While the intention of the present chapter is to disclose the general applicability of the unified modeling and solution methodology, and not to provide a comprehensive comparison of this approach with the best-performing specialized methods for each class of problems at this time, it should nonetheless be emphasized that the results presented clearly establish that the reformulation approach not only works across a wide array of problem classes but works very well. The approach finds best known solutions for many problems, regardless of problem class, in modest computer times. As future studies take advantage of the opportunity to apply more advanced UQP algorithms such as method 3 across a wider range of applications, additional records will undoubtedly be set for finding best known solutions to test problems from a variety of sources.

4.9 Important Alternative Model for Assignment Problems

Many important classes of problems have assignment constraints where, generally, agents of some kind are assigned to tasks. For such problems, Transformation 1 can be used to enforce the assignment constraints via quadratic penalties as illustrated earlier in this chapter. For such problems, however, a slightly different manner of constructing the quadratic penalty matrix, Q , has proven to be attractive in certain cases provided that resulting quadratic optimization problem is carried out subject to a cardinality constraint rather than being unconstrained. Recall that Transformation 1 results in an additive constant and a modification of the elements on the main diagonal of the Q matrix. With the alternative method, neither the additive constant nor modified diagonal elements are employed. The idea is illustrated by the following example:

Suppose the goal is to obtain solutions that satisfy

$$\begin{aligned}x_{11} + x_{12} + x_{13} &= 1 \\x_{21} + x_{22} + x_{23} &= 1 \\x_{31} + x_{32} + x_{33} &= 1\end{aligned}$$

Clearly exactly three variables will be equal to one and the other six variables will be zero. Finding solutions that satisfy the above assignment constraints is equivalent to solving the problem

$$\begin{aligned}\min x_0 &= x'Qx \\ \text{st} \\ \sum_{i=1}^3 \sum_{j=1}^3 x_{ij} &= 3\end{aligned}$$

where

$$\mathbf{x} = (x_{11}, x_{12}, x_{13}, x_{21}, \dots, x_{33})$$

and the Q matrix is given by

$$Q = \begin{bmatrix} 0 & P & P & 0 & 0 & 0 & 0 & 0 & 0 \\ P & 0 & P & 0 & 0 & 0 & 0 & 0 & 0 \\ P & P & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P & P & 0 & 0 & 0 \\ 0 & 0 & 0 & P & 0 & P & 0 & 0 & 0 \\ 0 & 0 & 0 & P & P & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & P & P & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P & P & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & P & 0 & P \\ 0 & 0 & 0 & 0 & 0 & 0 & P & P & 0 \end{bmatrix}$$

where P is a positive scalar penalty. Note that the block diagram structure along the main diagonal makes this penalty matrix particularly easy to construct.

The cardinality constraint requires exactly three of the nine variables to be equal to 1. The penalty structure of Q, given the goal of forcing x_0 to zero, will not allow, say, x_{12} or x_{13} to be 1 in the event that x_{11} is equal to one. In this manner, all three assignment constraints are enforced via the penalties.

In applications involving assignment constraints, there are typically additional constraints (besides the assignment constraints), and these need to be “folded” into the Q matrix as well. This approach is particularly well suited for quadratic assignment problems where facilities are assigned to locations, clustering problems where data are assigned to clusters, coloring problems where colors are assigned to nodes, clique partitioning problems where nodes are assigned to cliques, and so forth.

5 Promising New Application Areas

This combined modeling/solution approach provides a unifying theme that can be applied in principle to all linearly constrained quadratic and linear programs in bounded integer variables, and the computational findings for a broad spectrum of problem classes raises the possibility that similarly successful results may be obtained for even wider ranges of problems. The generality of the approach of modeling and solving problems using the UQP formulation invites additional applications to be pursued via this unified framework. Promising applications that are part of current work in progress include:

1. *The variable (feature) selection problem:* This important problem in linear regression concerns choosing a set of independent variables that are strongly correlated with the independent variable and weakly correlated with one another. One model for this, given by Eksioglu et al. [20], takes the form of a

two-objective IP model that can be easily recast into the unified $x'Qx$ framework. Tests are currently underway applying this alternative approach to a variety of data sets and making comparisons with standard statistical approaches.

2. *Clustering*: Clustering is an important data mining tool with applications in many important areas from medicine to marketing. Taking a graph-theoretical perspective, clustering can be viewed as a clique partitioning problem, and thus the quadratic model presented in Sect. 4.3 can be used to cluster data. In this setting too, tests are in process to extend the encouraging early results from this approach reported in Kochenberger et al. [43] and in Wang et al. [62].
3. *Computational biology*: Several important problems arising in biology can be modeled and solved as combinatorial optimization problems. Of particular interest are the multiple sequence alignment problem, the lattice protein folding problem, the rotamer assignment problem and the contact map optimization problem. Each of these problems, as explained in Forrester and Greenberg [21], can be modeled as a constrained quadratic optimization problem in zero-one variables, thus permitting them to be transformed into the $x'Qx$ framework and solved by solution methods designed for the unified framework. Early testing of this approach is underway.
4. *General linear 0/1 programming*: The general 0/1 linear programming problem can be represented by

$$\begin{aligned} & \max cx \\ & \text{st} \\ & Ax = b \\ & x \text{ binary} \end{aligned}$$

By using Transformation 1 it is possible to recast the problem in the form of

$$\begin{aligned} & \max x_0 = x^t Qx \\ & \text{st } x \text{ binary} \end{aligned}$$

For problems with inequality constraints, slack variables, via a binary expansion, can always be introduced to create the system of constraints $Ax = b$. This procedure is illustrated by the following example:

$$\begin{aligned} & \max 6x_1 + 4x_2 + 8x_3 + 5x_4 + 5x_5 \\ & \text{st} \\ & 2x_1 + 2x_2 + 4x_3 + 3x_4 + 2x_5 \leq 7 \\ & 1x_1 + 2x_2 + 2x_3 + 1x_4 + 2x_5 = 4 \\ & 3x_1 + 3x_2 + 2x_3 + 4x_4 + 4x_5 \geq 5 \\ & x \in \{0, 1\} \end{aligned}$$

Adding slack variables for the 1st and 3rd constraints

$$\begin{aligned} 0 \leq s_1 \leq 3 &\Rightarrow s_1 = 1x_6 + 2x_7 \\ 0 \leq s_3 \leq 6 &\Rightarrow s_3 = 1x_8 + 2x_9 + 4x_{10} \end{aligned}$$

gives the system $Ax = b$ with A given by

$$A = \begin{bmatrix} 2 & 2 & 4 & 3 & 2 & 1 & 2 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 2 & 4 & 4 & 0 & 0 & -1 & -2 & -4 \end{bmatrix}$$

For a scalar penalty $P=10$, applying Transformation 1 gives the equivalent problem

$$\max x_0 = x'Qx$$

with an additive constant of -900 and a Q matrix

$$Q = \begin{bmatrix} 526 & -150 & -160 & -190 & -180 & -20 & -40 & 30 & 60 & 120 \\ -150 & 574 & -180 & -200 & -200 & -20 & -40 & 30 & 60 & 120 \\ -160 & -180 & 688 & -220 & -200 & -40 & -80 & 20 & 40 & 80 \\ -190 & -200 & -220 & 645 & -240 & -30 & -60 & 40 & 80 & 160 \\ -180 & -200 & -200 & -240 & 605 & -20 & -40 & 40 & 80 & 160 \\ -20 & -20 & -40 & -30 & -20 & 130 & -20 & 0 & 0 & 0 \\ -40 & -40 & -80 & -60 & -40 & -20 & 240 & 0 & 0 & 0 \\ 30 & 30 & 20 & 40 & 40 & 0 & 0 & -110 & -20 & -40 \\ 60 & 60 & 40 & 80 & 80 & 0 & 0 & -20 & -240 & -80 \\ 120 & 120 & 80 & 160 & 160 & 0 & 0 & -40 & -80 & -560 \end{bmatrix}$$

Solving $\max x_0 = x'Qx$ gives the nonzero values

$$x_1 = x_4 = x_5 = x_9 = x_{10} = 1$$

for which $x_0 = 916$. Adjusting for the additive constant gives an objective function value of 16 which is optimal. Note that any linear problem in bounded integer variables, through a binary expansion, could be converted into $\max x_0 = x'Qx$ as illustrated here. Note also, though, that the elements of the Q matrix can, for some problems, get unacceptably large and may require suitable scaling to mitigate this problem.

5. *The Max 3-Sat Problem:* Section 4.2 discusses the Max 2-Sat problem, showing how it can be reformulated and solved as $\min x^t Qx$. The penalty approach adopted for the Max 2-Sat problem can be expanded to address the Max 3-Sat problem. In this case, however, the penalty function that results is a cubic rather than a quadratic function. Nonetheless, by a simple transformation, the cubic function can be recast as a quadratic and thus the Max 3-Sat problem, like

the Max 2-Sat problem, can also be modeled and solved using the $\min x^T Qx$ formulation.

In particular, the Max 3-Sat problem gives rise to four possible clause types. A linear constraint is associated with each which can be carried into the problem objective by a cubic penalty function which is equal to zero when the constraint is satisfied and otherwise equal to one. Finding solutions that maximize the number of clauses satisfied then corresponds to minimizing the penalty function that results by summing the individual penalties. The procedure is illustrated below.

The four clause types together with their associated constraint and penalty are:

1. *No negations*:

Linear constraint: $x_i + x_j + x_k \geq 1$

Cubic penalty: $(1 - x_i x_j x_k - x_i - x_j - x_k + x_i x_j + x_i x_k + x_j x_k)$

2. *One negation*:

Linear constraint: $x_i + x_j + \bar{x}_k \geq 1$

Cubic penalty: $(x_k - x_i x_k - x_j x_k + x_i x_j x_k)$

3. *Two negations*:

Linear constraint: $x_i + \bar{x}_j + \bar{x}_k \geq 1$

Cubic penalty: $(x_j x_k - x_i x_j x_k)$

4. *Three negations*:

Linear constraint: $\bar{x}_i + \bar{x}_j + \bar{x}_k \geq 1$

Cubic penalty: $(x_i x_j x_k)$

The conversion of the cubic penalty function to an equivalent quadratic function is carried out by the reduction procedure of Boros and Hammer [11] by replacing a product term xy by a new binary variable z and adding the penalty term $P(xy - 2xz - 2yz + 3z)$ to the objective function as illustrated in the following example with $n=5$ variables and 12 clauses.

Clause #	Clause
1	$x_1 \vee x_2 \vee x_3$
2	$x_1 \vee \bar{x}_2 \vee x_3$
3	$\bar{x}_1 \vee x_2 \vee \bar{x}_3$
4	$x_2 \vee \bar{x}_3 \vee x_4$
5	$\bar{x}_2 \vee x_3 \vee x_4$
6	$\bar{x}_2 \vee \bar{x}_3 \vee \bar{x}_4$
7	$x_2 \vee x_4 \vee x_5$
8	$\bar{x}_2 \vee x_3 \vee x_5$
9	$x_2 \vee \bar{x}_3 \vee x_5$
10	$x_3 \vee x_4 \vee x_5$
11	$x_3 \vee \bar{x}_4 \vee \bar{x}_5$
12	$\bar{x}_3 \vee \bar{x}_4 \vee \bar{x}_5$

Introducing the penalty functions for these clauses gives the following cubic penalty function to minimize:

$$\begin{aligned} x_0 = & 3 - x_1 + x_2 - 2x_4 - 2x_5 + 2x_1 x_3 - 4x_2 x_3 + 3x_4 x_5 \\ & - x_1 x_2 x_3 + 3x_2 x_3 x_4 - x_2 x_4 x_5 - x_3 x_4 x_5 + 2x_2 x_3 x_5 \end{aligned}$$

Note that this function has five cubic terms. Introducing two new binary variables, $x_6 = x_2x_3$ and $x_7 = x_4x_5$ along with the associated quadratic penalties yield the equivalent quadratic penalty function:

$$\begin{aligned} x_0 = & 3 - x_1 + x_2 - 2x_4 - 2x_5 + 2x_1x_3 - 4x_2x_3 + 3x_4x_5 \\ & - x_1x_6 + 3x_4x_6 - x_2x_7 - x_3x_7 + 2x_5x_6 \\ & + P(x_2x_3 - 2x_2x_6 - 2x_3x_6 + 3x_6) \\ & + P(x_4x_5 - 2x_4x_7 - 2x_5x_7 + 3x_7) \end{aligned}$$

which is of the form

$$x_0 = 3 + (xQx)/2$$

Taking (arbitrarily) the penalty P to be 10, the 7-by-7 Q matrix is

$$Q = \begin{bmatrix} -2 & 0 & 2 & 0 & 0 & -1 & 0 \\ 0 & 2 & 6 & 0 & 0 & -20 & -1 \\ 2 & 6 & 0 & 0 & 0 & -20 & -1 \\ 0 & 0 & 0 & -4 & 13 & 3 & -20 \\ 0 & 0 & 0 & 13 & -4 & 2 & -20 \\ -1 & -20 & -20 & 3 & 2 & 60 & 0 \\ 0 & -1 & -1 & -20 & -20 & 0 & 60 \end{bmatrix}$$

Minimizing xQx yields $x_1 = x_2 = x_3 = x_6 = 1$, $x_4 = x_5 = x_7 = 0$ for which $xQx = -6$. Thus, $x_0 = 0$ implying that all 12 clauses are satisfied at this solution. Note that in carrying out the reduction from the cubic penalty function to the quadratic several choices for variable substitutions were available. In general it is desirable to make these choices in a manner that minimizes the number of new variables that are introduced.

The preceding discussion and illustrations disclose that the unified unconstrained approach offers a fresh and promising alternative method of attack for these important problems. As additional research is conducted to provide enhanced methods for solving the UQP model, the benefits of recasting diverse problems into this general framework will become even greater.

6 Conclusion

A variety of disparate combinatorial optimization problems can be treated by first reexpressing them within the common modeling framework of the unconstrained quadratic binary program. Once in this unified form, the problems can be solved effectively by recently developed solution approaches for UQP.

The empirical findings from extensive testing challenge the conventional wisdom that places high priority on preserving linearity and exploiting specific structure. Although the merits of such a priority are well-founded in many cases, experience with a large variety of problem classes suggests that an unflinching adoption of

the conventional *linear* approach may preclude obtaining the best outcomes in a number of cases. In making use of the UQP model, any linearity that the original problem may have exhibited is destroyed. Moreover, any exploitable structure that may have existed originally is “folded into” the \hat{Q} matrix, and a general UQP solution procedure takes no advantage of it. Nonetheless, the use of such procedures has been remarkably successful, yielding results that rival the effectiveness of the best specialized methods.

Acknowledgments The authors would like to acknowledge the contributions of their coworkers Drs. Bahram Alidaee, Jin-Kao Hao, Mark Lewis, Karen Lewis, Zhipeng Lu, and Cesar Rego whose work contributed to the results appearing in this chapter.

Cross-References

- ▶ [Algorithms for the Satisfiability Problem](#)
 - ▶ [Combinatorial Optimization in Data Mining](#)
 - ▶ [Graph Searching and Related Problems](#)
 - ▶ [On Coloring Problems](#)
 - ▶ [Tabu Search](#)
-

Recommended Reading

1. B. Alidaee, G. Kochenberger, A. Ahmadian, 0-1 quadratic programming approach for the optimal solution of two scheduling problems. *Int. J. Syst. Sci.* **25**, 401–408 (1994)
2. B. Alidaee, G. Kochenberger, F. Glover, C. Rego, A new modeling and solution approach for the number partitioning problem. *J. Appl. Math. Decis. Sci.* **9**(2), 113–121 (2005)
3. B. Alidaee, F. Glover, G. Kochenberger, H. Wang, Solving the maximum edge weight clique problem via unconstrained quadratic programming. *Eur. J. Oper. Res.* **181**, 592–587 (2007)
4. B. Alidaee, G. Kochenberger, K. Lewis, M. Lewis, H. Wang, A new approach for modeling and solving set packing problems via unconstrained quadratic binary programming. *Eur. J. OR* **186**(#2), 504–512 (2008)
5. B. Alidaee, G. Kochenberger, K. Lewis, M. Lewis, H. Wang, Computationally attractive non-linear models for combinatorial optimization. *Int. J. Math. Oper. Res.* **1**(1 and 2), 9–20 (2009)
6. T.M. Alkhamis, M. Hasan, M.A. Ahmed, Simulated annealing for the unconstrained binary quadratic pseudo-boolean function. *Eur. J. Oper. Res.* **108**, 641–652 (1998)
7. M. Amini, B. Alidaee, G. Kochenberger, A scatter search approach to unconstrained quadratic binary programs, in *New Methods Optimization*, ed. by D. Corne, M. Dorigo, F. Glover (McGraw-Hill, England, 1999), pp. 317–330
8. J.E. Beasley, Heuristic algorithms for the unconstrained binary quadratic programming problem. Working Paper, Imperial College, 1999
9. A. Billionnet, A. Sutter, Minimization of a quadratic pseudo-boolean function. *Eur. J. OR* **78**, 106–115 (1994)
10. B. Borchers, J. Furman, A two-phase exact algorithm for Max-Sat and weighted Max Sat. *J. Comb. Optim.* **2**, 299–306 (1999)
11. E. Boros, P. Hammer, The Max-cut problem and quadratic 0-1 optimization, polyhedral aspects, relaxations and bounds. *Ann. OR* **33**, 151–225 (1991)
12. E. Boros, A. Prekopa, Probabilistic bounds and algorithms for the maximum satisfiability problem. *Ann. OR* **21**, 109–126 (1989)

13. E. Boros, P. Hammer, Pseudo-boolean optimization. *Discret. Appl. Math.* **123**(1–3), 155–225 (2002)
14. E. Boros, P. Hammer, X. Sun, The DDT method for quadratic 0-1 minimization, RUTCOR Research Center, RRR 39–89, 1989
15. J.M. Bourjolly, P. Gill, G. Laporte, H. Mercure, A quadratic 0/1 optimization algorithm for the maximum clique and stable set problems. Working paper, University of Montreal, 1994
16. P. Chardaire, A. Sutter, A decomposition method for quadratic zero-one programming. *Manage. Sci.* **41**(4), 704–712 (1994)
17. G. Cornuejols, *Combinatorial Optimization: Packing and Covering*. CBMS-NSF (SIAM, Philadelphia, 2001)
18. C.M. De Simone, M. Diehl, P. Junger, Mutzel, G. Reinelt, G. Rinaldi, Exact ground State4s of Ising spin glasses: new experimental results with a branch and cut algorithm. *J. Stat. Phys.* **80**, 487–496 (1995)
19. X. Delorme, X. Gandibleau, J. Rodrigues, GRASP for set packing. *EJOR* **153**, 564–580 (2004)
20. B. Eksioglu, R. Demirer, I. Capar, Subset selection in multiple linear regression: a new mathematical programming approach. *Comput. Ind. Eng.* **49**, 155–167 (2005)
21. R. Forrester, H. Greenberg, Quadratic binary programming models in computational biology. *Algorithmic Oper. Res.* **3**, 110–129 (2008)
22. F. Glover, G. Kochenberger, *Handbook of Metaheuristic* (Kluwer Academic, Boston/Dordrecht/London, 2003)
23. F. Glover, M. Laguna, *Tabu Search* (Kluwer Academic, Boston, 1997)
24. F. Glover, G. Kochenberger, B. Alidaee, Adaptive memory tabu search for binary quadratic programs. *Manage. Sci.* **44**(3), 336–345 (1998)
25. F. Glover, G. Kochenberger, B. Alidaee, M.M. Amini, Tabu with search critical event memory: an enhanced application for binary quadratic programs, in *MetaHeuristics: Advances and Trends in Local Search Paradigms for Optimization*, ed. by S. Voss, S. Martello, I. Osman, C. Roucairol (Kluwer Academic, Boston, 1999)
26. F. Glover, B. Alidaee, C. Rego, G. Kochenberger, One-pass heuristics for large-scale unconstrained binary quadratic programs. *EJOR* **137**, 272–287 (2002)
27. F. Glover, G. Kochenberger, B. Alidaee, M. Amini, Solving quadratic Knapsack problems by reformulation and tabu search: single constraint case, in *Combinatorial and Global Optimization*, ed. by P.M. Pardalos, A. Migdalas, R. Burkard (World Scientific, River Edge, 2002), pp. 111–121
28. F. Glover, J.K. Hao, G. Kochenberger, Z. Lu, H. Wang, Solving large scale max cut problems via tabu search. Working Paper, University of Colorado at Denver, 2010
29. F. Glover, Z. Lu, J.K. Hao, Diversification-driven tabu search for unconstrained binary quadratic programming. *4OR* **8**(3), 239–253 (2010)
30. M. Grötschel, M. Junger, G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design. *Oper. Res.* **36**(#3), 493–513 (1988)
31. P. Hammer, S. Rudeanu, *Boolean Methods in Operations Research* (Springer, New York, 1968)
32. P.B. Hansen, Methods of nonlinear 0–1 programming. *Ann. Discret. Math.* **5**, 53–70 (1979)
33. P. Hansen, B. Jaumard, Algorithms for the maximum satisfiability problem. *Computing* **44**, 279–303 (1990)
34. P. Hansen, B. Jaumard, V. Mathon, Constrained nonlinear 0-1 programming. *INFORMS J. Comput.* **5**(2), 97–119 (1993)
35. L.D. Iasemidis, D.S. Shihau, J.C. Sackellares, P. Pardalos, Transition to epileptic seizures: optimization, in *DIMACS Series in Discrete Math and Theoretical Computer Science*, American Mathematical Society Publishing, Providence, RI, vol. 55 (2000), pp. 55–73
36. A. Joseph, A concurrent processing framework for the set partitioning problem. *Comput. Oper. Res.* **29**, 1375–1391 (2002)

37. K. Katayama, M. Tani, H. Narihisa, Solving large binary quadratic programming problems by an effective genetic local search algorithm, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'00)* (Morgan Kaufmann, San Francisco, 2000)
38. G. Kochenberger, F. Glover, A unified framework for modeling and solving combinatorial optimization problems: a tutorial, in *Multiscale Optimization Methods and Applications*, ed. by W. Hager, S.-J. Huang, P. Pardalos, O. Prokopyev (Springer, New York/Boston, 2006)
39. G. Kochenberger, F. Glover, B. Alidaee, C. Rego, A unified modeling and solution framework for combinatorial optimization problems. *OR Spectr.* **26**, 237–250 (2004)
40. G. Kochenberger, F. Glover, B. Alidaee, C. Rego, Solving combinatorial optimization problems via reformulation and adaptive memory metaheuristics, in *Revolutionary Visions in Evolutionary Computation*, ed. by A. Menon, D. Goldberg (Kluwer, Boston, MA, 2004)
41. G. Kochenberger, F. Glover, B. Alidaee, C. Rego, An unconstrained quadratic binary approach to the vertex coloring problem. *Ann. OR* **139**, 229–241 (2005)
42. G. Kochenberger, F. Glover, B. Alidaee, K. Lewis, Using the unconstrained quadratic program to model and solve max 2-Sat problems. *Int. J. Oper. Res.* **1**(1 and 2), 89–100 (2005)
43. G. Kochenberger, F. Glover, B. Alidaee, H. Wang, Clustering of microarray data via clique partitioning. *J. Comb. Optim.* **10**, 77–92 (2005)
44. G. Kochenberger, B. Alidaee, H. Wang, An effective modeling and solution approach for the generalized independent set problem. *Optim. Lett.* **1**, 111–117 (2007). Springer-Verlag
45. D.J. Laughunn, Quadratic binary programming. *Oper. Res.* **14**, 454–461 (1970)
46. M. Lewis, B. Alidaee, G. Kochenberger, Using xQx to model and solve the uncapacitated task allocation problem. *OR Lett.* **33**, 176–182 (2005)
47. M. Lewis, G. Kochenberger, B. Alidaee, A new modeling and solution approach for the set partitioning problem. *Comput. OR* **35**, 807–813 (2008)
48. M. Lewis, B. Alidaee, F. Glover, G. Kochenberger, A note on xQx as a modeling and solution framework for the linear ordering problem. *Int. J. OR* **5**(#2), 152–162 (2009)
49. A. Lodi, K. Allemand, T.M. Liebling, An evolutionary heuristic for quadratic 0-1 programming. Technical Report OR-97-12, D.E.I.S., University of Bologna, 1997
50. P. Merz, B. Freisleben, Genetic algorithms for binary quadratic programming, in *Proceedings of the 1999 International Genetic and Evolutionary Computation Conference (GECCO'99)* (Morgan Kaufmann, San Francisco, 1999), pp. 417–424
51. A. Mingozzi, M. Boschetti, S. Ricciardelli, L. Blanco, A set partitioning approach to the crew scheduling problem. *Oper. Res.* **47**(6) 873–888 (1999)
52. M. Padberg, On the facial structure of set packing polyhedra. *Math. Program.* **5**, 199–215 (1973)
53. G. Palubeckis, A heuristic-branch and bound algorithm for the unconstrained quadratic zero-one programming problem. *Computing* **54**, 284–301 (1995)
54. P. Pardalos, G.P. Rodgers, Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing* **45**, 131–144 (1990)
55. P. Pardalos, G.P. Rodgers, A branch and bound algorithm for maximum clique problem. *Comput. OR* **19**, 363–375 (1992)
56. P. Pardalos, J. Xue, The maximum clique problem. *J. Glob. Optim.* **4**, 301–328 (1994)
57. A. Pekec, M. Rothkopf, Combinatorial auction design. *Manage. Sci.* **49**(#11), 1485–1503 (2003)
58. M. Ronnqvist, Optimization in forestry. *Math. Program. B* **97**, 267–284 (2003)
59. L. Schrage, *Optimization Modeling with LINDO* (Duxbury, Pacific Grove, 1997)
60. R. Vemuganti, Applications of set covering, set packing and set partitioning models: a survey, in *Handbook of Combinatorial Optimization*, ed. by D. Zhu, P. Pardalos (Kluwer Academic, Boston, MA, 1998)
61. H. Wang, B. Alidaee, F. Glover, G. Kochenberger, Solving group technology problems via clique partitioning. *Int. J. Flex. Manuf.* **18**, 77–97 (2006)
62. H. Wang, T. Obremski, B. Alidaee, G. Kochenberger, Clique partitioning for clustering: a comparison with K-Means and latent class analysis. *Commun. Stat.* **37**(1), 1–13 (2008)

Combinatorial Optimization Algorithms

Elisa Pappalardo, Beyza Ahlatcioglu Ozkok and Panos M. Pardalos

Contents

1	Introduction	560
2	Microarrays and Applications	561
3	Probe Design and Selection	562
3.1	Specificity	562
3.2	Sensitivity	563
3.3	Homogeneity	563
3.4	Literature Review for Probe Design and Selection	564
4	Probe Selection Problems	567
5	Oligonucleotide Fingerprinting	567
5.1	Clone Classification	568
5.2	Probe Selection	571
6	The Nonunique Probe Selection Problem	576
6.1	State-of-the-Art Methods	577
7	Datasets and Experimental Comparisons for the Nonunique Probe Selection Problem	586
8	Conclusion	588
	Recommended Reading	589

E. Pappalardo (✉)

Dipartimento di Matematica e Informatica, Universita di Catania, Catania, Italy

e-mail: epappalardo@dmi.unict.it

B. Ahlatcioglu Ozkok

Department of Mathematics, Faculty of Arts and Science, Yildiz Technical University, Istanbul, Turkey

e-mail: beyzaahlatcioglu@gmail.com; bahlat@yildiz.edu.tr

P.M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA

e-mail: pardalos@ufl.edu

Abstract

Identification of targets, generally virus or bacteria, in a biological sample is a relevant problem in medicine. Biologists can use hybridization experiments to determine whether a specific DNA fragment, that represents the virus, is present in a DNA solution. A probe is a segment of DNA or RNA, labeled with a radioactive isotope, dye, or enzyme, used to find a specific target sequence on a DNA molecule by hybridization. Selecting unique probes through hybridization experiments is a difficult task, especially when targets have a high degree of similarity, for instance, in case of closely related viruses.

The nonunique probe selection problem is a challenging problem from a biological and computational point of view; a plethora of methods have been proposed in literature ranging from evolutionary algorithms to mathematical programming approaches.

In this study, we conducted a survey of the existing computational methods for probe design and selection. We introduced the biological aspects of the problem and examined several issues related to the design and selection of probes: oligonucleotide fingerprinting, maximum distinguishing probe set, minimum cost probe set, and nonunique probe selection.

1 Introduction

The discovery of the DNA has been defined as the most important biological research in the last century, thus a new era started for biology, medicine, and all genetic researches. These advances in genomic research showed the necessity of implementing efficient methodologies for genome analysis and classification. In this scenario, *in silico* design has become an effective approach to model complex systems, due to the several advantages that computer-aided design holds; in particular, computational approaches for biological problems allow one to reduce the time and cost required for performing experiments, by decreasing the number of wet experiments needed or by speeding up the experimentation process.

Deoxyribonucleic acid (DNA) is a nucleic acid that carries genetic information in the cells that can be passed from one generation to the next [4]. DNA was first observed by Frederich Miescher in the late 1800s, and its structure was discovered in the 1950s by James Watson, Francis Crick, Maurice Wilkins, and Rosalind Franklin. In 1953 James Watson and Francis Crick presented the double-helix model of DNA [83]. Nine years later, Watson, Crick, Wilkins, and Franklin shared the Nobel Prize for their discoveries concerning the molecular structure of nucleic acids and its significance for information transfer in living material.

The structure of the DNA helix consists of two antiparallel complementary strands of simple units, called nucleotides, which represent the genetic material of all living organisms. Each nucleotide comprises three components:

two-deoxyribose, which is a five-carbon sugar; a phosphate group; and a nitrogen-containing base attached to the sugar. In particular, DNA nucleotides have four bases: adenine (A), cytosine (C), guanine (G), and thymine (T).

The sugars in nucleic acids are linked one another by phodiester bridges between the 3' and 5' carbon atoms of adjacent sugar rings. The asymmetric ends of DNA are called the 5' end and 3' end: the 5' end contains a terminal phosphate group, and the 3' end contains a terminal hydroxyl group [4].

According to the Watson-Crick base pairing, the end of one strand in a DNA molecule lines up with the beginning of the other: each adenine on one strand pairs with a thymine on the other and in the same way each guanine pairs with a cytosine base [40].

Ribonucleic acid (RNA) consists of a long chain of nucleotide units. The structure of RNA is usually single stranded; each nucleotide in a RNA chain consists of three chemical units: a nitrogenous base, a ribose sugar, and a phosphate. RNA nucleotides contain ribose while DNA contains deoxyribose and the base uracil rather than the thymine. Some viruses use RNA as their genetic material.

After the description of the double helix by Watson and Crick [83], it was shown that the two DNA strands could be separated by heat or treatment with alkali. Marmur and Doty [47] described the reconstitution process of double-stranded DNA molecule, which underlies all the methods based on DNA renaturation or molecular hybridization. These methods exploit the technology offered by DNA microarrays to monitor thousands of genes at the same time: the basic mechanism behind microarrays is hybridization between two DNA strands, according to the Watson-Crick complementary pairing of nucleotide bases. The results of hybridization experiments can be used to infer the presence of specific targets, and find applications in many areas of biomedicine, such as gene expression profiling, disease diagnosis, drug discovery and development, and therapy planning.

2 Microarrays and Applications

One of the most important recent development in biology is the success of The Human Genome Project (HGP). The principal goal of this project was to sequence the entire human genome. This project was completed in 2003 after 13 years effort. Nevertheless, due to the great amount of data collected, the data analysis will continue for years.

The first step towards mapping a genome is to break down the DNA into segments: this is achieved by making a genomic library. A *DNA library* can be made by fragmenting large pieces of DNA with a restriction enzyme and cloning these fragments randomly into a suitable vector [72]. Given a DNA library, the problem is how to identify specific probes in a clone.

A *probe* is a segment of DNA or RNA, labeled with a radioactive isotope, dye, or enzyme and used to find a specific sequence of nucleotides or gene on a DNA

molecule by hybridization. A clone is said to be positive if it contains a given probe; otherwise, it is negative.

DNA microarray technology provides a fast approach to monitor the entire genome on a single chip and to perform hybridization experiments. A DNA microarray or DNA chip is a glass or nylon slide, onto which single strands of DNA sequences are fixed, called probes. The foundation of microarray technology lies in the Watson-Crick base pairing: two DNA strands, the probe and the target, hybridize if they are complementary to each other. The amount of hybridized target represents the gene expression level.

The two most important DNA microarray platforms are the c-DNA based-platform, and the oligonucleotide microarrays [11].

Some important DNA microarray applications are [76]:

- *Gene identification:* microarrays are used to identify functionality and phenotypes from DNA sequences. Levels and patterns obtained from microarray experiments provide a measure of cell- and tissue-specific gene expression.
- *Diagnosis of genetic diseases:* Genetic diseases are caused by an abnormality in an individual's genome. Common aberrations involved in cancer disease include gene amplifications, nonreciprocal translocations, and interstitial deletions. Microarray technology (i.e., comparative genomic hybridization) can be used to detect and map these changes in copy number of DNA sequences [1].
- *Study of cell cycle variation:* Microarrays are well suited for the analysis of temporal changes in gene expression during cellular events [20].
- *Drug discovery and toxicological research:* DNA microarrays can be used to identify appropriate targets for therapeutic intervention and to monitor changes in gene expression in response to drug treatments [15].
- *Forensic identification:* Microarrays can also be used to identify individuals within forensics [33]. Scientists can use microarrays to find markers in a DNA sample in order to create a profile of an individual.

3 Probe Design and Selection

Probe design involves the selection of probes to be used in microarray experiments. To be effective, each DNA microarray probe should meet the following criteria: specificity, sensitivity, and homogeneity. Additionally, “space and time search” efficiency should be taken into account.

3.1 Specificity

The specificity criterion identifies probes that are unique to each gene in the genome, minimizing the chance cross hybridization [76]. Specificity can be measured by

the Hamming distance metric¹ and BLAST (Basic Local Alignment Search Tool) search.

Libson et al. [44] developed a specificity search algorithm to handle the probe specificity property, under a stochastic model assumption.

Sung and Lee [76] used the Hamming distance metric to check for specificity criterion: if the Hamming distance between a probe and every subsequence in the genome is greater than some constant, the probe is said to be specific enough.

3.2 Sensitivity

According to the sensitivity criterion, probes with high self-complementariness and minimal secondary structure should not appear in microarrays.

Gasieniec et al. [22, 23] adopted free energy to measure sensitivity, where the total difference in the free energy of the folded and unfolded states of a DNA duplex is approximated by using a nearest-neighbor model.

3.3 Homogeneity

Temperature is one of the most important experimental conditions to take into account when performing hybridization experiments. The homogeneity filter aims to discard probes which hybridize at a temperature out of the experiment temperature range [76]. Several studies have derived accurate equations for the melting temperature T_m .

In 1996 SantaLucia et al. [71] calculated the T_m for individual melting curves from the fitted parameters.

In their model each probe should be in a predefined melting temperature range. The melting temperature T_m for a length- m probe p is given by

$$T_m(p) = \frac{\Delta H(p)}{\Delta S(p) + R \times \log(C_T)} - 273.15 + 16.6 \log(\text{Na}^+) \quad (1)$$

where R is the molar gas constant ($1.987 \text{ cal/C}^\circ \times \text{mol}$) and C_T is the total molar concentration of the annealing oligonucleotides when oligonucleotides are self-complementary. For non-self-complementary oligonucleotides, C_T is replaced by $\frac{C_T}{4}$. Na^+ is the salt concentration of the solution in which the oligomers are dissolved. $\Delta H(p)$ and $\Delta S(p)$ are the enthalpy and entropy for helix formation of p respectively. They then calculated the optimal hybridization temperature T_h of p as

$$T_h(p) = T_m(p) - 25 - 0.62(C_F) \quad (2)$$

where C_F is the formamide concentration of the solution.

¹For two strings s and t , the Hamming distance $H(s, t)$ is the number of positions where the characters of the two strings differ.

Wang and Seed [82] calculated the median temperature (T_m) for probes as follows:

$$\text{Median } T_m = 64.6 + 41 \times \frac{\text{gcCount}}{\text{probeLength}} - \frac{600}{\text{probeLength}} \quad (3)$$

where gcCount is the number of all *G*s and *C*s in a probe and the molar sodium concentration is taken to be 0.1 M. A probe candidate is discarded if its T_m is not within 5 C° of the median T_m .

In [22, 23], the melting temperature T_m is calculated as

$$T_m = \frac{\Delta H}{\Delta S + R \times \ln(c/4)} - 273.15 \quad (4)$$

where ΔH and ΔS are the enthalpy and entropy for helix formation respectively, R is the molar gas constant (1.987 cal/C° × mol), and c is the total molar concentration of the annealing oligonucleotides when oligonucleotides are not self-complementary.

3.4 Literature Review for Probe Design and Selection

A plethora of methods and softwares has been presented to design and select “effective” probes for microarrays. In this paragraph, we provide the reader a short summary of some of the most known methodologies.

In 1996 Lockhart et al. [45] proposed a program for designing oligo probes. Their approach is based on hybridization to small, high-density arrays containing tens of thousands of synthetic oligonucleotides.

Xu et al. [86] developed a method useful in selecting gene-specific fragments, which can be used as the input to design PCR primer pair for PCR amplification and microarrays.

Li and Stormo [42] developed a heuristic approach to optimize the selection of specific probe for each gene in an entire genome.

Rouillard et al. [69] presented the Oligoarray software that computes gene-specific and secondary structure-free oligonucleotides for genome-scale oligonucleotide microarray construction. They used BLAST for avoiding cross hybridization.

Kaderali and Schliep [35] proposed an algorithm for the probe design problem. Melting temperatures are calculated for all possible probe target interactions using an extended nearest-neighbor model.

Libson et al. [44] addressed the probe specificity properties under a stochastic model assumption.

Mrowka et al. [53] presented a web-based system for interactive design of suitable oligos, for transcription profiling experiments with user-defined parameters. The web interface allows the user to retrieve oligos for human transcripts in a

flexible manner. Criteria for selecting the oligos are low similarity to all other known human genes, the absence of stable secondary structures, the length of the oligo, and the absence of low-complexity regions. The user may select oligos corresponding to the desired position in the 5'-3' direction of the transcript and based on the melting temperature.

Chen and Sharp [10] provided a suite of Perl scripts that facilitates the search for gene-specific oligonucleotides for microarray experiments. Genes of interest are first identified in the form of UniGene clusters. The sequences of these clusters are extracted and assembled into contigs to increase their accuracy. The 3'-untranslated region (3'UTR) of the contig is parsed. Then, multiple 50 mer oligonucleotide sequences with similar melting temperature are obtained from each 3'UTR. These sequences are analyzed for gene specificity.

Sung and Lee [76] proposed an algorithm to select good probes based on the criteria of homogeneity, sensitivity, and specificity. They used the pigeonhole principle to speed up the specificity filter. Their algorithm is based on probe elimination and includes three steps: first, it filters oligo probes in the genome which fail to satisfy homogeneity criterion, then, it filters oligo probes in the genome which fail to satisfy sensitivity criterion, and finally filters oligo probes based on the specificity criterion using the pigeonhole principle. This is done to remove probes that might cross hybridize with any of the subsequences in the whole genome.

Wang and Seed [82] described the results of studies on the impact of various design criteria on the fraction of candidate probes qualified for microarray interrogation of RNA coding regions. They built the OligoPicker design software that evaluates specificity primarily for the occurrences of contiguous perfect matches. Moreover, they developed an algorithm to cluster redundant mouse and human genes; the resulting unique sequences were used for probe design.

Emrich et al. [19] designed a software tool that enables optimal oligos to be designed for a single or a group of target nucleic acid sequences.

Bozdech et al. [6] developed a software package, ArrayOligoSelector, to design an open reading frame (ORF)-specific DNA microarray using the publicly available *P. falciparum* genome sequence.

Tolstrup et al. [78] presented, OligoDesign, which provides optimal design of LNA (locked nucleic acid)-substituted oligonucleotides for functional genomics applications. OligoDesign features recognition and filtering of the target sequence by genome-wide BLAST analysis in order to minimize cross hybridization with nontarget sequences, and it includes routines for predicting melting temperature, self-annealing, and secondary structure for LNA-substituted oligonucleotides, as well as secondary structure prediction of the target nucleotide sequence. Individual scores for all these properties are calculated for each possible LNA oligonucleotide in the query gene, and the OligoDesign program ranks the LNA probes according to a combined fuzzy logic score, finally returns the top scoring probes to the user. OligoDesign has been used to design a *Caenorhabditis elegans* LNA oligonucleotide microarray, which allows monitoring the expression of a set of 120 potential marker genes for a variety of stress and toxicological processes and toxicologically relevant pathways.

Nielsen et al. [55] presented a program for designing optimal oligonucleotides for microarrays, OligoWiz. The program implements the design of highly specialized arrays, as well as general use arrays. OligoWiz evaluates and graphically presents all the potential oligonucleotides in the input sequences, according to a collection of parameters, each of which can be assigned a different weight.

Gordon and Sensen [26] developed a software package called Osprey for the calculation of optimal oligonucleotides for DNA sequencing and the creation of microarrays based on either PCR-products or directly spotted oligomers. It incorporates a novel use of position-specific scoring matrices, for the sensitive and specific identification of secondary binding sites, anywhere in the target sequence. Osprey consists of a module for target site selection based on user input, novel utilities for dealing with problematic sequences such as repeats, and a common code base for the identification of optimal oligonucleotides from the target list.

Chou et al. [12] implemented an oligo microarray design tool for large genomes, which integrates novel computer science techniques and the best-known nearest-neighbor parameters to quickly identify sequence similarities, and estimate their hybridization properties. The designed oligos are computationally optimized to guarantee the best specificity, sensitivity and uniformity, under the given design constrains.

Reymond et al. [67] developed a software to design optimal oligonucleotide probe sets for microarrays. Selected probes show no significant cross hybridization and no stable secondary structures, and their melting temperature (T_m) is chosen to minimize the T_m variability of the probe set.

Li et al. [43] presented a tool that uses sequence identity, free energy, and continuous stretch for selection of optimal oligonucleotide probes. This approach implements a series of filters to check every oligonucleotide. A sequence identity based on gapped global alignments is identified, and a traversal algorithm is used to generate alignments for free energy calculation. The optimal T_m interval determined is based on probe candidates that have passed all other filters. Final probes are picked using a combination of user-configurable piecewise linear functions and an iterative process. The thresholds for identity, stretch, and free energy filters are automatically determined from experimental data by an accessory software tool, CommOligo-PE (CommOligo Parameter Estimator).

Rimour et al. [68] compared the existing oligo design softwares under certain experimental conditions. They proposed a new approach to design oligonucleotides, which combines good specificity with a potentially high sensitivity. They also presented an experimental validation of their strategy by comparing results obtained with standard oligos and with their composite oligos.

Charbonnier et al. [9] described a procedure for selecting 40–60 mer oligonucleotide probes combining optimal thermodynamic properties with high target specificity, suitable for genomic studies of microbial species. The algorithm for filtering probes from extensive oligonucleotide libraries includes positional information of predicted target-probe binding regions. The validity of potential microarray probes is tested by considering the possibility of cross hybridization

with nontarget coding sequences. This analysis can be performed within a single genome or between several different organisms.

Nordberg [56] presented an application for assisting biological researchers in selecting signature sequences. It incorporates a custom sequence similarity search to find potential cross hybridizing non-target sequences. The tool supports multiple probe design goals, including single-genome, multiple genome, pathogen host, and species/strain identification.

4 Probe Selection Problems

The choice of the oligonucleotide probe sets plays an important role in hybridization experiments: the optimal probe set should contain a minimal number of oligonucleotides in order to minimize the number of hybridization experiments, since this affects the experimental costs [5]. Due to the importance of an efficient design and selection, this subject has attracted increasing attention in the last 20 years. Therefore, a plethora of approaches has been proposed in literature, ranging from specific software tools and web-based applications to select oligonucleotides having specific features, such as low similarity, absence of secondary structure, as already seen in the previous sections, [6, 12, 19, 26, 53, 55, 56, 67, 69, 78, 82], to algorithms that optimize the design [9, 35, 43, 45, 68, 76].

Informally, the probe selection problem involves the selection of a small set of probes to analyze a population of unknown clones [59]. The basic (binary) probe selection problem can be formally stated as follows [5]:

Definition 1 Let $C = \{c_1, \dots, c_m\}$ be a set of unknown clones (or *targets*) and $P = \{p_1, \dots, p_n\}$ a set of oligonucleotide probes of preselected length l . The objective is to find a *smallest* set P of probes such that any two distinct clones $c, d \in C$ are distinguished by at least one probe $p \in P$, where a probe p is said to *distinguish* two clones c, d if it is a substring of exactly one of c or d .

This problem can be classified into two main categories: oligonucleotide fingerprinting and nonunique probe selection problem.

5 Oligonucleotide Fingerprinting

Oligonucleotide fingerprinting is a technique to identify unique probes by performing hybridization experiments: hybridization signals are evaluated and a vector of numerical values, the *fingerprint*, is assigned to each probe. Essentially, a fingerprint is a sequence of m numbers, that are signals from an image, which represents the interaction of the clone sequence with the probe set. For the sake of simplicity, it can be assumed that a fingerprint is a binary vector where the value 1 at a given position j , $1 \leq j \leq m$, means that the j th probe hybridizes to that clone, whereas a 0 means

that it does not. The fundamental hypothesis in this approach is the uniqueness of the fingerprint for each probe [29, 30].

The oligonucleotide fingerprinting method was first developed in 1986 for analyzing the mammalian genome. From then on, since many computational challenges arise in this method, several algorithmic approaches have been developed; in particular, two main issues have been handled: clone classification and probe selection.

5.1 Clone Classification

The classification of oligonucleotide fingerprints finds application in analysis of bacterial community composition and expression levels of genes. This process comprises several steps, starting with the creation of clones to perform hybridization, to the analysis of the fingerprints resulting from the experiment. Once this analysis is completed, clones are classified according to their characteristics and the results of the hybridization; two classes are therefore constructed, representing the clones that hybridize to probes and those that do not [21]. Unfortunately, the step of translating hybridization signals into two groups is not easy to perform, especially due to the intensity values of hybridization array [21]. Most of the existing approaches to classify clones are based on clustering methods.

Clustering can be informally defined as the classification of patterns into groups, the clusters, according to a similarity measure. Intuitively, patterns belonging to a specific cluster present more similarity to each other than to patterns within a different cluster. Typically, a clustering procedure comprises several steps [34]: pattern representation, definition of the similarity measure used to cluster the domain into groups, the clustering of data, and the evaluation of the output.

One of the most known clustering method is the k -means algorithm [46], where the underlying idea is to partition the pattern into k clusters, by minimizing the sum of squares of distances between data and the corresponding cluster centroid. Briefly, the k -means procedure works as follows: k cluster centers are chosen; therefore, the patterns are assigned to the closest cluster centroid. The centroids are recomputed using the current cluster memberships, and at each iteration, a new binding is done between the same dataset points and the nearest new centroid. This process ends when a specific convergence criterion is met.

In [29], Herwig et al. proposed a clustering procedure based on a sequential k -means algorithm to classify clone fingerprints. The clustering algorithm groups similar fingerprints together, according to a pairwise similarity measure. This measure is based on mutual information and takes into account the total number of matched similarities, whereas the commonly used metric distances, like the Euclidean distance, do not. An important assumption is that hybridization signals in clones are independent; mutual information measures the mutual dependence of two signals: it tends to zero if there is no correlation between the fingerprints $x = (x_1, \dots, x_p)$ and $y = (y_1, \dots, y_p)$; it is maximal if x and y are equal (perfect correlation). Given two fingerprints x and y , the mutual information is given by

$$H(x; y) = \sum_{i=1}^K \sum_{j=1}^K \frac{n_{ij}^{xy}}{n^{xy}} \log_2 \frac{n_{ij}^{xy} n^{xy}}{n_i^x n_j^y} \quad (5)$$

where K represents the number of intervals used to digitalize signals, n_{ij}^{xy} is the number of pairs where x falls in interval i and y falls in interval j , n_i^x and n_j^y are the respective marginal frequencies of x and y , and n^{xy} is the number of pairs where signals are present in both vectors [29]. To select pairs that are close to each other, the value $t(x, y) = \sum_{i=1}^K (n_{ii}^{xy}) / (n^{xy})$ is computed for each pair of data points.

In order to compare different pairwise clone similarities, the value of mutual information is normalized as follows:

$$s(x, y) = \frac{2H(x; y)}{H(x) + H(y)} \quad (6)$$

where $H(x) = -\sum_{i=1}^K K \frac{n_i^x}{n^{xy}} \log_2 \frac{n_i^x}{n^{xy}}$ represents the entropy value for x and $H(y)$ the entropy value for y .² Since

$$H(x; y) \leq \min\{H(x), H(y)\} \quad (7)$$

as reported in [14], s lies in the interval $[0,1]$. If $s = 1$, x and y are perfectly correlated; otherwise, no correlation is found.

For clustering fingerprints, two parameters are introduced: γ is the minimal similarity threshold to merge two clusters, and ρ is the maximal admissible similarity of a data point to a cluster centroid. A random sample of data pairs is selected to adjust the threshold according to the dataset, and a distribution of pairwise similarity is derived from this sample. The median m and the standard deviation s computed from the obtained distribution are used to compute the threshold values γ and ρ .

The clustering procedure guarantees that cluster centroids are sufficiently separated, because their pairwise similarity at each step is less than γ ; moreover, centroids that have similarity with a data point greater or equal than ρ are replaced by this point. To validate the clustering, a measure of quality is computed: let us assume to know the true clustering T for a data structure of N data points, x_1, \dots, x_n . Let $t_{ij} = 1$ if x_i and x_j are in the same cluster; otherwise, $t_{ij} = 0$, where $1 \leq i, j \leq N$. For a calculated cluster C , c_y is defined in the same way: it is equal to 1 if x_i and x_j belongs to the same cluster, 0 if not. To evaluate the clustering quality, a 2×2 contingency table is computed (see Table 1), where columns correspond to the true clustering and rows correspond to the calculate clustering.

²Entropy measures the information content of a probe with respect to the set of sequences. Shannon entropy is introduced in [74].

Table 1 Contingency table for clustering validation

	0	1	Total
0	N_{00}	N_{01}	$N_{0\cdot}$
1	N_{10}	N_{11}	$N_{1\cdot}$
Total	$N_{\cdot 0}$	$N_{\cdot 1}$	$N_{\cdot \cdot}$

In **Table 1**, $N_{kl} = \#\{(i, j); t_{ij} = l, c_{ij} = k, l \leq i, j \leq N\}$, $0 \leq k, l \leq 1$. $N_{\cdot k}$ and $N_{l\cdot}$ represent the respective marginal frequencies. The number of diagonal pairs in the table ($N_d = N_{00} + N_{11}$) indicates the number of correct clustered data pairs. To measure the clustering quality, the relative mutual information coefficient (RMIC) is computed as

$$\rho(C, T) = \text{sgn}(N_d - N^2/2) \frac{H(C; T)}{H(T)} \quad (8)$$

where C and T represent, respectively, calculated and true clustering; $\text{sgn}(y) = 1$ if $y \geq 0$, -1 otherwise. The number K of intervals considered for $H(C; T)$ and $H(T)$ is 2. RMIC represents the amount of information that calculated clustering contains about true clustering, and its value is negative if there are more pairs incorrectly clustered than correctly.

In a later work [30], the described clustering method is implemented in a data-analysis pipeline, where the probe set is a simulation parameter. An initial set of sequences is divided into two parts: a training set, and a part used to derive test sets for clustering. Probe sets are constructed by iteratively choosing probes that partition the training sequences into subsets that are as equal in size as possible. Binary fingerprints are computed from sequences in the test sets. Since the number of possible fingerprints increases as 2^p for p probes, computing all the possibilities is computationally infeasible. Then an approximation is applied [48]: in the first step, the probe that best partitions the sequences into two groups having the same size is found. Then, the algorithm selects the second probe that, together with the first one, partitions the training set into four groups that are as equal in size as possible. This process is iterated until the number of selected probes is equal to a threshold, or each partition contains only one probe, condition defined as complete partitioning.

Another clustering procedure is introduced in [28]. The proposed algorithm works on a similarity graph, constructed from a similarity matrix S as follows: given an $n \times p$ matrix H , where n is the number of clones and p the number of probes, H_{ij} represents the intensity level of hybridization of clone i with probe j , for $i \leq n, j \leq p$. The similarity matrix S , where $S_{ij} = \sum_k H_{ik} \cdot H_{jk}$, is constructed from H . The similarity graph G_θ is a graph with vertices corresponding to the clones and whose edges connect clones having pairwise similarity greater or

equal to the threshold θ [52]. The idea behind this approach is to identify cliques³ in the similarity graph (because they represent similar fingerprints), by recursively searching for highly connected subgraphs.⁴

5.2 Probe Selection

Selecting optimal probes for oligonucleotide fingerprinting is a difficult task, since the “quality” of probes affects the final design. From this perspective, effective and efficient computational methods have been developed to provide a fast and accurate set of probes. These approaches range from exact methods to heuristic algorithms. In particular, two variations of the original probe selection problem have been introduced by Borneman et al. in [5], where the objectives are to maximize the number of clones distinguished (*maximum distinguishing probe set*) and to minimize the number of probes needed to distinguish all the clones (*minimum cost probe set*). We will discuss these problems later in this section.

A randomized algorithm for probe selection is proposed in [23], whose underlying idea is to explore randomization to reduce the computational time needed for selecting probes. In this approach, every substring of a gene of a specific length m is referred to as a *candidate*. For each candidate, the algorithm verifies if it satisfies the following criteria: quantitative criteria, concerning the size of the bases and the length contiguous regions of bases [45]; homogeneity, concerning the melting temperature; and sensitivity. Candidates that do not meet such standards are filtered out whereas candidates that pass the filtering step are called probes. Once the probes are produced, the optimal among them are selected from this pool by applying a randomized procedure, that guarantees a faster selection and assures unique probes for up 99 % of genes in the whole genome. The time complexity of the algorithm is $\mathcal{O}(kmn)$, where k (usually smaller than n) is the number of genes in the whole genome, m is the length of probes, and n the length of the whole genome, versus the $\mathcal{O}(mn^2)$ complexity of the brute-force approach.

In [65], Rash and Gusfield introduced the barcoding problem, where the notion of barcode is equivalent to the concept of oligonucleotide fingerprint. In their approach, the problem is formulated as an integer linear programming (ILP) model, and suffix trees are used to reduce the number of variables by selecting substrings, the probes, from strings that represent the target sequences.

Let c_1, \dots, c_k represent the set of all substrings for each pair of clones (C_1, C_2). For each substring c_i , $1 \leq i \leq k$, a corresponding binary variable v_i is assigned. The goal of the mathematical formulation is to minimize the sum of v_i , subject to

³We recall that a clique in a graph $G = (V, E)$ is a subset $C \subseteq V$, such that each pair of vertices in C is connected by an edge.

⁴A graph G with $n > 1$ vertices is said highly connected if the minimum size of a cut, $k(G)$, is $> n/2$. A highly connected subgraph is an induced subgraph that is highly connected.

$$v_1 + \dots + v_k \geq 1. \quad (9)$$

In this approach, a method to identify virus mutations is introduced: instead of requiring that each pair of clones is distinguished by only one probe, the constraint of having pairs distinguished by at least r probes is imposed. Therefore, (9) becomes

$$v_1 + \dots + v_k \geq r. \quad (10)$$

Using $r = 5$ ensures with high confidence that a signature would remain valid under reasonably high mutation rates and short generation time. A second constraint imposes a minimum edit distance between pairs of clones:

$$v_i + v_j \leq 1. \quad (11)$$

According to (11), if a pair of substrings v_i and v_j does not satisfy the constraint, only one of the strings is allowed to be in the solution set. The result of the analysis shows that using 2 or 4 as edit distance results in added confidence that mutations in nature not invalidate the signatures.

The choice of addressing an optimal design, by maximizing the quality of probes or minimizing the cost of selecting probe, is addressed in two variations of the oligonucleotide fingerprinting problem introduced in [5]: the *maximum distinguishing probe set (MDPS, for short)* and the *minimum cost probe set (MCPS, for short)*. Some basic notations are now introduced to define the problems.

Definition 2 Let $C = \{c_1, \dots, c_m\}$ be the set of clones (or *targets*) and $P = \{p_1, \dots, p_n\}$ the set of probes of preselected length l .

Let $C^2 = \{(c, d) : c, d \in C, c < d\}$ be the set of all pairs of different clones of C , where “ $<$ ” denotes an arbitrary ordering of C ; $\text{occ}(c, p)$ is the number of occurrences of the probe p in the target c .

R denotes the threshold for the number of occurrences of each probe in a clone.

The P -fingerprint of c , denoted by $\text{fingerprint}_P(c)$, is a vector of length m which contains one entry for each probe p ; $\text{fingerprint}_P(c)$ is the value $\min\{R, \text{occ}(p, c)\}$, $\forall p \in P$.

If $R = 1$, the fingerprint is just a binary vector with a 1 in position i if probe p_i is a substring of clone c , and a 0 otherwise.

A set of probes P distinguishes two clones c and d , if $\text{fingerprint}_P(c) \neq \text{fingerprint}_P(d)$.

$\Delta_P \subseteq C^2$ is the set of pairs of clones distinguished by P .

In *MDPS*, the problem is to find a set of k probes, where k is a known parameter, that maximizes the number of distinguished pairs of clones. Formally,

Problem 1 (Maximum Distinguishing Probe Set) Given a set C of clones, a set P of probes, and an integer k , find a subset $S \subseteq P$, with $|S| = k$, where $|\Delta_S|$ is maximized.

In *MCPS*, the objective is to find a minimal number of probes that distinguishes all the clones. Formally,

Problem 2 (Minimum Cost Probe Set) Given a set C of clones and a set P of probes, find a subset $S \subseteq P$, such that $\Delta_P = C^2$ and $|S|$ is minimized.

In particular, the *MCPS* problem can be considered as a special case of the set-cover problem [31], and *MDPS* as a special case of maximum coverage [31]. *MCPS* and *MDPS* are NP-hard, when the length of probes is unbounded [5]. To overcome such NP-hardness, some heuristic methods have been introduced in literature: in [5], a heuristic method based on a Lagrangian relaxation is applied to solve the *MCPS* problem, since this approach performs good for the set cover problem [7]. The *MCPS* problem is formulated as a constrained integer linear program, as follows:

$$\min |S| = \sum_{p \in P} x_p \quad (12)$$

subject to

$$\sum_{p \in P} \delta_{p,c,d} \cdot x_p \geq 1 \quad \forall (c, d) \in C^2 \quad (13)$$

$$x_p \in \{0, 1\} \quad \forall p \in P \quad (14)$$

where $x_p \in \{0, 1\}$ indicates whether the probe p is in a feasible solution S or not; $\delta_{p,c,d} = 1$ if $(c, d) \in \Delta_p$. This formulation is solved by using a Lagrangian relaxation algorithm. Since the solutions obtained can be infeasible, a greedy strategy is applied to ensure their feasibility.

The *MDPS* problem is addressed by applying a simulated annealing algorithm: it starts with an initial random solution and moves to neighbors characterized by values that are better than the current one. Two sets of probes are said neighbors if one can be obtained from the other by substituting exactly one probe.

At each iteration of the algorithm, a new random neighbor S' is generated from the current solution S . S' is chosen as new solution according to a probability $\text{pr} = \min\{1, \exp \frac{|\Delta_S| - |\Delta_{S'}|}{t}\}$, where t represents the temperature parameter. The temperature value is decreased at each iteration according to $t = \beta t / (\beta + t)$; the value of β has been set empirically to 2,000.

Experimental results show that both methods presented in [5] are able to find satisfactory probe sets for real instances.

In [8], the *MDPS* several approaches for solving the *MDPS* problems have been proposed. The quality of the results is estimated in [8] by using four different measures: entropy; number of sets that measure the number of sets of undistinguishable clones; largest set, which considers the cardinality of the largest set; and number of pairs, counting the number of clones distinguished by the probe set.

Entropy is defined as follows:

$$E = - \sum_{i=1}^k \frac{|C_i|}{m} \log_2 \left(\frac{|C_i|}{m} \right), \quad (15)$$

where C_i , $1 \leq i \leq k$, represent the sets of undistinguishable clones for the given probe set and m is the number of clones. The entropy value for the optimal solution is $\log_2 m$. Two kinds of neighbors are exploited: augmented and pivot neighbors. Two solutions S_i , S_{i+1} are said to be augmented neighbors if and only if $|S_i| + 1 = |S_{i+1}|$ and $S_i \subset S_{i+1}$.

S_i , S_{i+1} are pivot neighbors if and only if $|S_i| = |S_{i+1}| = k$ and $|S_i \cap S_{i+1}| = k - 1$, that is, S_i and S_{i+1} differ for exactly one element. k is the parameter representing the number of probes.

The first approach presented in [8] is an affinity-based algorithm: a square matrix of the affinity is computed for each pair of probes, and the two probes having the best value are selected. At each step the probe that together with the former ones generates a better affinity score is picked, until the number of selected probes is equal to k . The greedy algorithm proposed in [8] moves from a solution set to its augmented neighbors, by selecting a probe that together with the previously selected ones partitions the clone set better, according to the quality measures, until the number of selected probes is equal to k . The pivot-last algorithm and pivot-greedy algorithm combine the greedy strategy with pivot steps. Furthermore, two heuristic approaches, a genetic algorithm [32] and a simulated annealing method [36], are proposed.

All the proposed methods for the *MDPS* problem are tested on real and synthetic data; it turns out that the best approach is the pivot greedy algorithm. Moreover, the entropy measure is recognized as the best estimation measure, since it represents the most global and exhaustive criterion with respect to the others.

In [51], an efficient Asynchronous Teams technique [17] is applied to find near-optimal solutions for the *maximum distinguishing probe set* and the *minimum cost probe set* problems. An A-Team can be viewed as a collection of autonomous agents, each of which representing a heuristic strategy. Agents can read and write to shared memory without any synchronization among them. The details of this approach for the *MDPS* and the *MCPS* problems can be found in Figs. 1 and 2, respectively. $C1$, $C2$, and $C3$ are independent agents that create the probe set, compute the fingerprint for each probe, and construct a feasible solution. Agents $CB1$ and $CB2$ work together with $C2$ for creating new solutions starting from the “old” ones. $M1$ and $M2$ represent the shared memory. Agents $I1$ and $I2$ improve the current solutions, by implementing a 1-neighbor and 2-neighbors strategy for *MDPS* and a 2,1-exchange for *MCPS*, where, if possible, two probes are substituted by one.

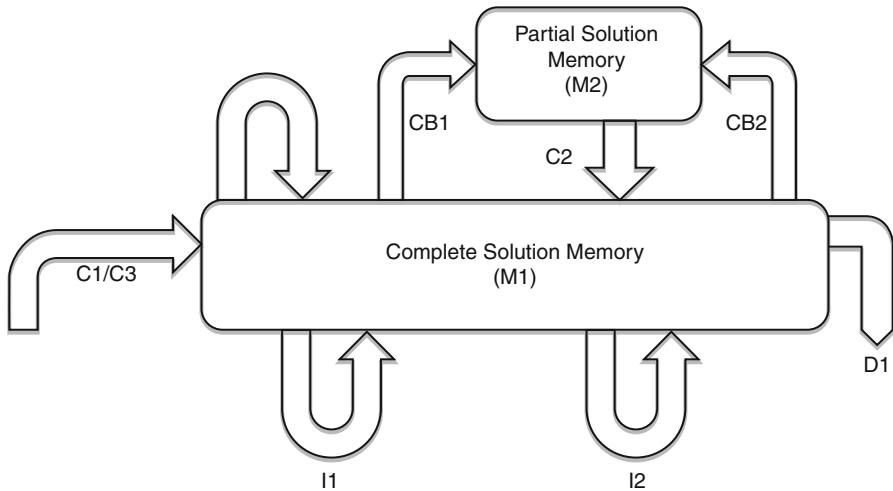


Fig. 1 A-Team for the *MDPS*: agents and shared memories

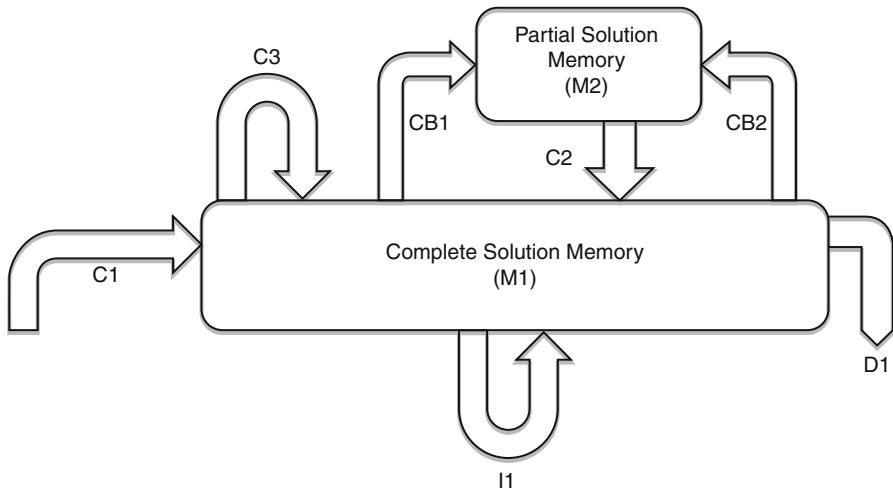


Fig. 2 A-Team for the *MCPS*: agents and shared memories

The A-Team strategy produces results comparable to [5]; moreover, this approach is able to find near-optimal probe sets with probes of both fixed and variable lengths.

6 The Nonunique Probe Selection Problem

Finding probes that are unique through hybridization experiments is difficult, especially when targets have a high degree of similarity, as in the case of closely related viruses. An alternative approach consists on the selection of *nonunique* probes, characterized by the hybridization to more than one target. In addition to the intrinsic complexity of this task, the presence of errors can affect the results of the experiments: false hybridizations can occur (*false positive*), or experiments might not report their presence (*false negative*). A remedy is introducing redundancy into the design, by requiring that targets have to be separated by more than one probe (*separation constraint*) and that each target has to hybridize to more than one probe (*coverage constraint*). The formal definition of the Nonunique probe selection problem (*NUPS*, for short) requires some additional definitions.

Definition 3 (Target-Probe Incidence Matrix) Given m target sequences t_1, \dots, t_m , and n candidate probes p_1, \dots, p_n , a target-probe incidence matrix $H = (h_{ij})$ is defined by $h_{ij} = 1$ if target t_i hybridizes to probe candidate p_j , and $h_{ij} = 0$ otherwise.

An example of target-probe incidence matrix can be found in [Table 2](#).

Definition 4 (Probe Coverage) Given m target sequences t_1, \dots, t_m , and n candidate probes p_1, \dots, p_n , a probe p_j , $1 \leq j \leq n$, is said to be covered by the target t_i , $1 \leq i \leq m$, if the target t_i hybridizes to the probe p_j or, equivalently, if $h_{ij} = 1$.

Definition 5 (Probe Separation) Given m target sequences t_1, \dots, t_m , and n candidate probes p_1, \dots, p_n , a probe p_j , $1 \leq j \leq n$, is said to distinguish or separate two target sequences t_a and t_b , if it is a substring of exactly one of t_a or t_b , that is, if $|h_{aj} - h_{bj}| = 1$.

As an example, given the targets $t_a = AGGCAATT$ and $t_b = CCATATTGG$, for the probe $p_j = GCAA$, $h_{aj} = 1$, $h_{bj} = 0$; therefore, p_j distinguishes t_a and t_b . On the contrary, for the probe $p_k = ATT$, we have $h_{ak} = h_{bk} = 1$, so that it follows p_k does not distinguish the target-pair t_a, t_b .

Informally, the goal of the nonunique probe selection problem is to select a minimum set of probes such that the presence or absence of a single target can be univocally determined. In order to reduce the presence of experimental errors, each pair of targets should be distinguished by at least h_{\min} probes; these requirements are called *Hamming distance constraints*⁵ or *separation constraints*. The *coverage constraint* requires that each target has at least c_{\min} probes hybridizing to it.

⁵Recall that the Hamming distance between two strings of equal length is defined as the number of positions at which the strings differ.

Table 2 Target-probe incidence matrix

	p_1	p_2	p_3	p_4	p_5	p_6
t_1	1	1	0	1	0	1
t_2	1	0	1	0	0	1
t_3	0	1	1	0	1	1
t_4	0	0	1	1	1	0

It can be shown that the problem is NP-hard, using a reduction from the set-cover problem as a special case for $h_{\min} = 1$ and $c_{\min} = 0$ [37, 38].

6.1 State-of-the-Art Methods

Due to its importance in genomic applications, the nonunique probe selection problem has been broadly studied, and a great number of mathematical and algorithmic approaches has been proposed. In particular, we distinguish between exact and heuristic methods.

6.1.1 Exact Methods

Many combinatorial formulations of the nonunique probe selection problem, have been presented in literature. In [37, 38], after a preliminary reduction of the numbers of candidate probes, the problem is modeled as a variation of a set-cover integer linear programming problem.

Let $N = \{p_1, \dots, p_n\}$ denote the set of candidate probes, $M = \{t_1, \dots, t_m\}$ denote the set of targets, and $P = \{(i, k) | 1 \leq i < k \leq m\}$ denote the set of all combinations of target indices. Let x_j , $j \in N$, be the binary-valued decision variables, with $x_j = 1$ if the probe j is chosen, 0 otherwise.

The problem is formulated as follows:

$$\min \sum_{j=1}^n x_j \quad (\text{master ILP}) \quad (16)$$

subject to

$$\sum_{j=1}^n h_{ij} x_j \geq c_{\min} \quad \forall i \in M \quad (17)$$

$$\sum_{j=1}^n |h_{ij} - h_{kj}| x_j \geq h_{\min} \quad \forall (i, k) \in P \quad (18)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, n. \quad (19)$$

The objective function (16) minimizes the number of probes; the constraints in (17) require that at least c_{\min} selected probes hybridize to each target, and (18) represents the Hamming distance constraints. Finally, (19) restricts the variables to binary values.

When h -separation cannot be ensured for all target pairs with the given set of probes, the solutions of the above ILP formulation are empty. As a remedy, a large number $l = m \cdot h$ of unique virtual probes are added to the initial set of available probes. It is important to note that such “artificial” probes are chosen only if the separation constraint does not hold the original set of candidate probes. In this scenario, the objective function coefficients of the virtual probes are set to a large number M ; the original master ILP is hence reformulated as follows:

$$\min \sum_{j=1}^n x_j + M \sum_{k=n+1}^{n+l} x_k, \quad (20)$$

with n replaced by $n + l$ in the constraints of the original master ILP formulation.

The master ILP formulation presented in (16)–(19) guarantees separation between pairs of single targets. In some circumstances, however, separation between groups of targets is desirable, since this approach can take into account cross hybridization and error tolerance. A group testing approach for this case has been already introduced in [73], where separation is defined also among pairs of small target groups.

For the mathematical formulation, group-separability is considered in a new model, the slave ILP. Given a set of targets S , let $\omega_j^S = \max_{i \in S} h_{ij}$ the signature of S , resulting from applying the logical *OR* to the rows in S , with $j = 1, \dots, n$. S and T are d -separable if and only if the Hamming distance between ω^S and ω^T is at least d . A cutting-plane approach is proposed to solve this problem: the idea is to iteratively construct a most violated group constraint by looking at the current solution.

Let x^* be a solution vector and $X = \{j | x_j^* = 1\}$ the index set of probes included in x^* . For a target set S , let $\omega_{|X}^S$ be the restriction of ω^S to the columns in X . The following slave ILP is solved to find target groups S and T such that the Hamming distance between $\omega_{|X}^S$ and $\omega_{|X}^T$ is less or equal than d :

$$\max \sum_{j \in X} (\sigma_j^0 + \sigma_j^1) \quad (\text{slave ILP}) \quad (21)$$

subject to

$$\sigma_j^0 \leq 1 - s_i \quad \forall (i, j) \in M \times X \quad \text{with } h_{ij} = 1 \quad (22)$$

$$\sigma_j^0 \leq 1 - t_i \quad \forall (i, j) \in M \times X \quad \text{with } h_{ij} = 1 \quad (23)$$

$$\sigma_j^1 \leq \sum_{i \in M} h_{ij} s_i \quad \forall j \in X \quad (24)$$

$$\sigma_j^1 \leq \sum_{i \in M} h_{ij} t_i \quad \forall j \in X \quad (25)$$

$$\sum_{i \in M} s_i \geq 1 \quad (26)$$

$$\sum_{i \in M} t_i \geq 1 \quad (27)$$

$$s_i + t_i \leq 1 \quad \forall i \in M \quad (28)$$

$$\sum_{i \in M} s_i \leq c \quad (29)$$

$$\sum_{i \in M} t_i \leq c \quad (30)$$

$$0 \leq \sigma_j^0 \leq 1 \quad \forall j \in X \quad (31)$$

$$0 \leq \sigma_j^1 \leq 1 \quad \forall j \in X \quad (32)$$

$$s_i \in \{0, 1\} \quad \forall i \in M \quad (33)$$

$$t_i \in \{0, 1\} \quad \forall i \in M \quad (34)$$

Variables σ_j^0 and σ_j^1 model the similarity between S and T for the probe j : $\sigma_j^0 = 1$ if and only if ω_j^S and ω_j^T are equal to 0, and $\sigma_j^1 = 1$ if and only if both variables are equal to 1. Constraints (22) and (23) express that $\sigma_j^0 = 0$ if S or T contains a target that hybridizes to probe j ; due to the constraints (24) and (25), $\sigma_j^1 = 1$ if at least one target in both S and T hybridizes to probe j .

Finally, constraints (26) and (27) avoid that $S = \emptyset$ and $T = \emptyset$, while (28) and (29) control the size of S and T .

A new formulation that does not require virtual probes is modeled in [38]. Here, the maximal possible separation between two groups of targets S and T is defined as

$$h(S, T) = \sum_{j \in N} |\omega_j^S - \omega_j^T|. \quad (35)$$

Therefore, the following model is introduced:

$$\min \sum_{j=1}^n x_j \quad (36)$$

subject to

$$x_j \in \{0, 1\} \quad \forall j \in N \quad (37)$$

$$\sum_{j \in N} |\omega_j^S - \omega_j^T| \cdot x_j \geq \min\{d, h(S, T)\} \quad \forall S, T \subseteq M, \quad (38)$$

$$|S| \leq c, \quad |T| \leq c, \quad S \neq T.$$

Inequalities (38) ensure that coverage constraints are satisfied; since the number of constraints in (38) is exponential, a branch-and-bound approach is applied to solve the relaxed version of the problem.

In [16], an algorithm based on the previous ILP formulations is proposed to solve the nonunique probe selection problem using a d -disjunct matrix. A matrix is d -disjunct if and only if the union of any d columns does not contain any other column, formally:

Definition 6 (d -Disjunct Matrix) H is a d -disjunct matrix if and only if for any $(d + 1)$ columns t_0, t_1, \dots, t_d , there exists a row p_i such that $H[i, 0] = 1$ and $H[i, j] = 0, \forall j = 1, \dots, d$.

Row p_i is said to cover the pair $(t_0, \langle t_1, \dots, t_d \rangle)$. The problem is hence reformulated as follows:

Problem 3 (Minimum- d -Disjunct Submatrix (MIN- d -SD)) Given m nonunique probe candidates and an $m \times n$ target-probe incidence matrix M , select a minimum set of the probe candidates such that the $h \times n$ submatrix H is d -disjunct, where $h \leq n$.

Naturally, the new defined problem is also NP-hard [77]. This problem is addressed by reformulating the ILP models proposed in [38]. The first new ILP formulation is based on the following definition:

Definition 7 H is called (d, k) -disjunct if each column has at least $(k + 1)$ 1-entries not contained in the union of other d columns.

The following mathematical formulation is used to construct a $(1, d - 1)$ -disjunct submatrix.

$$\min \sum_{i=1}^m x_j \quad (39)$$

subject to

$$\sum_{i \in M} |M_{ij} - M_{ij} M_{ik}| x_i \geq d \quad \forall j, k \in N, \quad j \neq k \quad (40)$$

$$x_j \in \{0, 1\} \quad \forall i \in M, \quad (41)$$

where M is the target-probe incidence matrix; m and n are, respectively, the number of probes and targets; and $x_i = 1$ if probe p_i is chosen for the submatrix H , otherwise is 0. The first constraint guarantees that any column in M has at least a number d of one-components not contained in any other column.

A second ILP formulation recalls the slave ILP proposed in [38]; it finds the target sets R and S that violate d -disjunctness.

In order to detect errors in hybridization experiments, a new ILP problem is modeled that is able to identify at most d targets when at most k errors occur, by constructing a $(d, 2k)$ -disjunct matrix [18].

$$\min \sum_{i=1}^m x_j \quad (42)$$

subject to

$$\sum_{i \in M} |M_{ij} - M_{ij} M_{ik}| x_i \geq d + 2k \quad \forall j, k \in N, \quad j \neq k \quad (43)$$

$$x_j \in \{0, 1\} \quad \forall i \in M \quad (44)$$

The algorithm for the error tolerance case uses this formulation to construct the $(1, d + 2k - 1)$ -disjunct matrix and the second one to find the target sets that violate $(d, 2k)$ -disjunctness.

6.1.2 Heuristic Methods

Though exact methods are able to locate solutions with a guarantee of optimality, these approaches require an exact knowledge of the domain; moreover, in some cases, finding a solution via integer programming software is intractable, due to the large number of variables and constraints within the problem [49]. Heuristic algorithms allow to overcome these difficulties, by implementing search strategies that speed up the location of optimal solutions, even if without any guarantee of optimality.

A two-phases heuristic algorithm for the nonunique probe selection problem has been presented in [50]. It consists of a construction phase, which builds a feasible solution when possible, and a reduction phase that reduces the number of probes while maintaining feasibility. To begin with, candidate probes are selected according to the number of targets to which each probe binds, until the minimum coverage constraint is satisfied for each target. The algorithm next focuses on satisfying the minimum Hamming distance constraint: probes that are not included in the current solution are sorted in decreasing order by the number of target pairs they separate; then probes are selected from the ordered list and added to the solution until the number of Hamming distance violated constraints is reduced to a predefined threshold. The remaining probes are found using a local search. The reduction phase

iteratively deletes two probes at a time and verifies if the solution is still feasible or if it can be made feasible by adding just one probe to the solution.

A cutting-plane algorithm is presented in [60]: this method relaxes a large subset of Hamming distance constraints, in order to find and improve the lower bound on the number of probes required in an optimal solution. Relaxed constraints are reinstated only if it is needed to maintain feasibility. Results provided by this approach are compared to the values computed by the heuristic algorithm implemented in [50] and the ILP approach presented in [37]: the cutting-plane algorithm outperforms the two approaches; in particular it is able to find solutions that are about 20 % better than the previous best ones provided by the other two methods.

A model-based approach is discussed in [79], where a greedy heuristic uses a Bayesian model [57] for guiding the search towards the probes that satisfy separation and coverage constraints. In general, a Bayesian optimization algorithm generates an initial population of solutions randomly, according to a uniform distribution. Therefore, the current population is updated for a number of iterations, by selecting the best solutions using methods inspired to genetic algorithms. This allows one to construct a Bayesian network representing the population of promising solutions, and new solutions are generated by sampling the defined probabilistic model. The new candidate solutions replace some of the old ones in the original population, in order to obtain in-silicon evolutionary pressure.

The model-based approach proposed in this work is similar to a Bayesian algorithm, but it does not consider any evolution mechanism. It represents a modification of the heuristic presented in [50]: the idea is to prefer probes that appear in the largest number of minimal sets, by distinguishing between *good* and *bad* ones. Good probes are those with the highest degree of contribution to minimal solutions, whereas bad probes have the lowest degree of contribution. In order to include good probes in a feasible candidate solution, a coverage and a separation probabilistic models are implemented as follows.

Given the target-probe incidence matrix H , a candidate probe set $P = \{p_1, \dots, p_n\}$, and a target set $T = \{t_1, \dots, t_m\}$, let the function $\text{cov}_{\text{drc}} : P \times T \rightarrow [0, 1]$ be defined as follows:

$$\text{cov}_{\text{drc}}(p_j, t_i) = h_{ij} \times \frac{c_{\min}}{|P_{t_i}|}, \quad p_j \in P_{t_i}, \quad t_i \in T \quad (45)$$

where P_{t_i} is the set of probes hybridizing to target t_i . Notice that $\text{cov}_{\text{drc}}(p_j, t_i)$ is the amount that p_j contributes to satisfy the coverage constraint for target t_i . Additionally, we have $0 \leq \text{cov}_{\text{drc}}(p_j, t_i) \leq 1$. The coverage function $C_{\text{drc}} : P \rightarrow [0, 1]$ is then defined as follows:

$$C_{\text{drc}}(p_j) = \max_{t_i \in T_{p_j}} \{\text{cov}_{\text{drc}}(p_j, t_i) \mid 1 \leq j \leq n\}, \quad (46)$$

where T_{p_j} is the set of targets covered by p_j . $C_{\text{drc}}(p_j)$ is the maximum amount that p_j can contribute to satisfy the minimum coverage constraints. Function C_{drc}

favors the selection of probes that c_{\min} -cover targets t_i that have the smallest subsets P_{t_i} ; these are the essential or near-essential covering probes. In particular, C_{drc} guarantees the selection of near-essential covering probes that c_{\min} -cover *dominated targets*, where t_i dominates t_k if $P_{t_k} \subset P_{t_i}$.

Similarly, the function $\text{sep}_{\text{drc}} : P \times T^2 \rightarrow [0, 1]$ is defined as

$$\text{sep}_{\text{drc}}(p_j, t_{ik}) = |h_{ij} - h_{kj}| \times \frac{h_{\min}}{|P_{t_{ik}}|}, \quad p_j \in P_{t_{ik}}, \quad t_{ik} \in T^2 \quad (47)$$

where $P_{t_{ik}}$ is the set of probes separating target-pair t_{ik} ; $\text{sep}_{\text{drc}}(p_j, t_{ik})$ is what p_j can contribute to satisfy the separation constraint for target-pair t_{ik} . We have $0 \leq \text{sep}_{\text{drc}}(p_j, t_{ik}) \leq 1$. The separation function $S_{\text{drc}} : P \rightarrow [0, 1]$ is

$$S_{\text{drc}}(p_j) = \max_{t_{ik} \in T_{p_j}^2} \{\text{sep}_{\text{drc}}(p_j, t_{ik}) \mid 1 \leq j \leq n\}, \quad (48)$$

where $T_{p_j}^2$ is the set of target pairs separated by p_j . $S_{\text{drc}}(p_j)$ is the maximum amount that p_j can contribute to satisfy the minimum separation constraints. Function S_{drc} favors the selection of essential or near-essential probes that h_{\min} -separate *dominated target pairs*; these probes h_{\min} -separate target-pairs t_{ik} that have the smallest subsets $P_{t_{ik}}$.

Now, cover and selection functions are combined together into the selection function that allows to choose the minimum number of probes such that coverage and separation constraints are satisfied:

$$D_{\text{drc}}(p_j) = \max\{(C_{\text{drc}}(p_j), S_{\text{drc}}(p_j)) \mid 1 \leq j \leq n\}. \quad (49)$$

$D_{\text{drc}}(p_j)$ represents the degree of contribution of p_j , that is, the maximum amount required for p_j to satisfy all constraints; D_{drc} ensures that all essential probes p_j will be selected for inclusion in the subsequent candidate solution, since $C_{\text{drc}}(p_j) = 1$ or $S_{\text{drc}}(p_j) = 1$.

The model-based algorithm uses the above-introduced functions to create a feasible set of probes, according to the constraints of coverage and separation. This algorithm consists of three phases: the initialization phase creates an initial solution; since this solution can be infeasible, the construction phase repeatedly inserts high-degree probes into the initial solution in order to maintain feasibility; finally, the reduction phase removes low-degree probes to reduce the cardinality of the solution set. This heuristic is called *dominated row covering* heuristic. Results provided by the described model are compared with the ones obtained by applying the ILP method [37] and the heuristic algorithm presented in [50]: the Bayesian algorithm obtains comparable results for the artificial data and better results on real instances.

The selection strategy introduced in [79] is combined with an evolutionary approach in [80], where a genetic algorithm, already proposed for the set-cover problem in [3], is adapted to the nonunique probe selection problem. After the creation of a random population, two solutions are selected, by using fitness scaling and binary tournament. Therefore, a fusion crossover is used to create a

new solution, to which a mutation operator is applied. In order to guarantee the feasibility, the heuristic presented in [79] is applied to the generated solution. At this point, a new solution has been generated from the pool of “parents,” and it is used to replace a randomly selected solution with an above-average fitness in the population. This process will be iterated until a stopping criterion is met.

Each solution is modeled as an n -bit binary string $s = s_1 \dots s_n$, where $s_j = 1$ if $p_j \in S$ and $s_j = 0$ if $p_j \notin S$, $1 \leq j \leq n$. The fitness of an individual s is directly related to its objective value, which corresponds to the number of probes in its associated subset S . That is, the fitness of s is defined as

$$f(s) = \sum_{j=1}^n s_j. \quad (50)$$

To create a new population from the current one, crossover and mutation operators are applied. The crossover operator used to create a new individual from two parental ones is a generalized fitness-based crossover: it considers the differences between the parents and is more capable of generating new solutions when the parents are similar; also, the fittest parent influences more the “child” solution. Let f_{P_1} and f_{P_2} be the fitness values of the parents P_1 and P_2 respectively, and let C denote the child solution. Then for $1 \leq j \leq n$, $C_j = P_{1j} = P_{2j}$, provided that $P_{1j} = P_{2j}$; otherwise, $C_j = P_{1j}$, with probability $p = f_{P_2}/(f_{P_1} + f_{P_2})$, and $C_j = P_{2j}$, with probability $1 - p$.

After the crossover, the algorithm applies the mutation operator by randomly altering a number of bit positions, in order to introduce diversity and prevent the stagnation into local optima. The number of positions to mutate for a given solution depends on the mutation rate, which is correlated to the convergence rate of the algorithm; it is defined as

$$B = \left\lceil \frac{m_f}{1 + \exp \frac{-4m_g(t-m_c)}{m_f}} \right\rceil, \quad (51)$$

where t is the current number of child solutions generated, m_f specifies the final stable mutation rate, m_c is the number of solutions that should be generated such that the mutation rate is $\frac{m_f}{2}$, and m_g specifies the gradient at $t = m_c$. The value of m_f is user defined and the values of m_c and m_g are problem-dependent parameters.

The initial population is generated with a high probability of being feasible, by putting

$$s_{cj} = \begin{cases} 1 & \text{if } r \leq D_{\text{drc}}(p_j) \\ 0 & \text{otherwise} \end{cases} \quad 1 \leq j \leq n \quad (52)$$

where $r \in [0, 1]$ is randomly chosen.

If the solution is infeasible, a heuristic feasibility operator, consisting of the construction and reduction phase of the model presented in [79], is applied to maintain feasibility. In order to replace solutions in the current population, a steady-state replacement strategy is implemented: a new solution replaces a randomly chosen individual of the population having a higher fitness value. This strategy is elitist since the best solutions are picked and they replace always the worst ones; in this way the population will converge quicker. This method allows one to create high quality solutions; in fact it outperforms all the previous implemented methods, by constructing probe set with minimum cardinality.

Though the dominated row covering heuristic presented in [79] guarantees the selection of near-essential probes that c_{\min} -cover and h_{\min} -separate dominated targets, it is not able to distinguish among targets having the same values for C_{drc} or S_{drc} . To address this task, a dominated probe selection heuristic is introduced in [81]. Here, the coverage and separation functions penalize each probe p by an amount that is proportional, respectively, to the number of targets that it covers or separates.

In order to guarantee separation not only between target pairs but also between pairs of small groups of targets, a subset selection criterion is introduced in [81]. This criterion is used by a sequential forward algorithm [58] to select the best subset of probes; in particular, it iteratively selects the probe that maximizes the dominant probe selection criterion and adds it to the current probe set. When a feasible solution is found, a reduction phase based on [79] is applied. This method performs better than the approaches proposed in [50, 73]; nevertheless, the genetic algorithm proposed in [80] and the cutting-plane method developed in [60] provide better results for all the tested datasets.

In [54], the two heuristics presented so far, the dominated row covering and the dominated probe selection, are modified in order to consider, at each step, the knowledge about the probes that are already selected in the previous iterations. In fact, these methods compute coverage and separation values during the initialization phase, and their values remain unchanged for the rest of the computation. Three new heuristics are introduced: the dynamic dominated row covering heuristic, the dynamic dominant probe selection heuristic and the normalized dynamic dominant probe selection heuristic. The first takes into account the information about selected probes: if a probe q is selected for a given target or target-pair, coverage and separation are recomputed only for those rows affected by this selection.

In a further work [24], the dominated row covering heuristic, the dominated probe selection heuristic, and a new one, the sum of dominated row covering, are combined with a Bayesian optimization algorithm [41]. The new heuristic introduced in [41] is capable to distinguish among probes that have the same score for the coverage of the dominated targets and the same score for the separation of the dominated target pairs. This new approach outperforms the other methods in some test instances.

In [25], a multiobjective optimization method is combined with the Bayesian algorithm presented in [24] to solve the nonunique probe selection problem for

the multiple targets case. The two objectives for this problem are minimizing the number of probes involved in a solution and maximizing the ability of recognizing multiple targets. In order to evaluate the fitness functions, a modified version of the weighted average ranking (WAR, for short) is computed. In WAR, each objective is evaluated separately and the fitness values of the solutions for each objective are sorted. Solutions are then ranked according to their fitness; finally the ranks obtained by sorting each list of objectives are averaged. This approach guarantees successful results for the simple cases of 5 and 10 targets in the sample, and good results for the case of 15 in almost all the instances considered in the dataset. When 20 targets are considered, it is able to find good solutions for some of the real and artificial datasets.

7 Datasets and Experimental Comparisons for the Nonunique Probe Selection Problem

The experiments for all the methods presented in the previous section were conducted on the same groups of data: it includes ten artificial datasets ($a_1, \dots, a_5, b_1, \dots, b_5$) and three real datasets (*meiobenthos*, *HIV-1*, and *HIV-2*).

To generate the artificial data, the Random Evolutionary FORest Model (REFORM) software was used with two different forest models. From each model, five independent test sets were generated. The test sets generated from the first model (a_1-a_5) contain 256 targets each; from the second one, five test sets (b_1-b_5) with 400 targets each are constructed.

The described two model topologies were chosen because they were able to produce sets of target sequences having a high degree of similarity, so that they cannot be easily separated using unique probes.

The probe candidates for each of the ten artificial test sets were generated using Promide software [61, 62].

For the real datasets, three groups of sequences are constructed. Benthos are organisms that live in the sea floor. They are categorized according to their size; meiobenthos are those having size between 100 and 500 μm [2]. In order to obtain the target sequences, 1,230 28S rDNA sequences from different organisms present in the meiobenthos were clustered in 149 groups. Clusters were representative of approximately 56 % of all *meiobenthos* sequences [73]. A test set of 679 targets was obtained by arbitrarily selecting a representative from each cluster. The corresponding probe set contained 15,139 probes.

The second and the third group of real data consisted of 200 each *HIV-1* and *HIV-2* sequences downloaded from the National Center for Biotechnology Information, having high similarity [50]. Candidate probes for the sequences were generated using Primer3 [70] with default input parameters. The default parameters include length between 18 and 27 nucleotides, melting temperature between 57 and 63°C, and GC content between 20 and 80 % for each HIV sequence.

Properties of the datasets are presented in Table 3: the second and third columns are the number of targets and number of probes of each dataset, respectively. Since

Table 3 Properties of the datasets used for experiments

Set	$ T $	$ P $	$ V $
a1	256	2,786	6
a2	256	2,821	2
a3	256	2,871	16
a4	256	2,954	2
a5	256	2,968	4
b1	400	6,292	0
b2	400	6,283	1
b3	400	6,311	5
b4	400	6,223	0
b5	400	6,285	3
M	679	15,139	75
HIV-1	200	4,806	20
HIV-2	200	4,686	35

Table 4 $|P_{\min}|$ value of the proposed method

	GrdS [73]	ILP [37, 38]	GDRM [50]	DRC [79]	OCP [60]	GA [80]	SFPS [81]	BOA [75]
a1	1,163	503	568	549	509	502	530	502
a2	1,137	519	560	552	494	490	516	490
a3	1,175	516	613	590	543	534	557	533
a4	1,169	540	597	579	539	537	557	537
a5	1,175	504	605	583	529	528	558	528
b1	1,908	879	961	974	830	839	883	834
b2	1,885	938	976	1,013	842	852	890	846
b3	1,895	891	951	953	827	835	896	829
b4	1,888	915	1,001	1,019	873	879	920	875
b5	1,876	946	1,022	1,019	874	890	933	879
M	3,851	3,158	2,336	2,084	1,962	1,962	2,036	–
HIV-1	–	–	531	487	451	450	468	450
HIV-2	–	–	578	506	479	476	492	474

most of the datasets used are not able to satisfy the minimum Hamming distance constraint for every pair of targets, even if the entire set of candidate probes is chosen for the solution, artificial probes are added to the final solution set. Column $|V|$ represents the number of these virtual probes.

All the experiments were performed with parameters $c_{\min} = 10$ and $s_{\min} = 5$.

Tables 4 and **5** show the number of probes included in the final solution ($|P_{\min}|$), for the methods described in the previous section. The best solutions found are reported in bold face. As shown in **Tables 4** and **5**, the genetic and the Bayesian approaches guarantee to find the best solutions for $a1, \dots, a5$ datasets and on the real instances *HIV-1* and *HIV-2*, although they are computationally expensive and slower than the other methods: genetic algorithm runs for at least 2 h for the a datasets [80] on the other hand, the optimal cutting-plane algorithm runs for at most 851 s [60].

Table 5 $|P_{\min}|$ value of the proposed method

	DDRC [54]	DDPS [54]	DDPSn [54]
a1	523	519	511
a2	510	502	501
a3	543	544	542
a4	552	548	547
a5	551	543	537
b1	884	880	875
b2	892	887	880
b3	879	881	868
b4	919	905	905
b5	929	918	921
M	1,996	2,016	1,986
HIV-1	459	461	460
HIV-2	487	488	487

For the *Meiobenthos* dataset, genetic and optimal cutting-plane methods find the optimal solution.

Genetic and Bayesian algorithms have difficulties in optimizing the sets $b1, \dots, b5$; for these datasets, the optimal cutting-plane method performs better.

As described in the previous section, genetic and Bayesian algorithms use the DRC heuristic presented in [79] for selecting the probes. It can be noticed that $a1-a5$ contain a greater percentage of probes having a higher heuristic value respect to $b1-b5$ sets. Moreover, $b5$ contains many probes with the same heuristic value, and the DRC heuristic is not able to distinguish between these probes. On the other hand, the optimal cutting plane algorithm focuses first on probes that cover the largest number of rows [80]; hence, it does better on the $b1-b5$ sets.

8 Conclusion

The complexity in designing and selecting optimal probes for hybridization experiments makes this subject an interesting problem from a computational perspective. Specifically, a plethora of methods have been proposed in literature to address several issues related to the selection and design of probes, ranging from exact approaches to heuristic algorithms. After introducing the biological aspects of hybridization procedures, this chapter illustrates the state-of-the-art methods for the described problems.

In particular, the nonunique probe selection problem represents a challenging problem from a biological and computational point of view.

It is our opinion that there is still a lot of work to be done on this emerging and interesting research field: firstly, target-group separation for the nonunique probe selection problem represents a new field, that has not yet been well investigated.

From a biological point of view, it is important to note that the experiments conducted for the nonunique probe selection problem are focused on identifying

five targets in the sample; it can be interesting to extend the existing approaches for the case of more targets since this is more realistic. A new approach for the problem could concern the reformulation of separation and coverage constraints in a unique metric, since the resulting problem could be easier to solve. Moreover, the intrinsic characteristics of the datasets used for the experiments are not well explored. Since they affect the computational results, it would be desirable taking into account the biological information contained in the sequences, in order to better explore the search space and deeply understand the results. An interesting idea could be the application of a multiobjective optimization strategy for the problem, where the constraints are treated as conflicting objective functions; in this way we aim to provide to biologists a better way to analyze the computational results from a biological perspective. Also, a crucial point for understanding the effectiveness of the existing methods for this class of problem is to provide new real-world datasets; for this sake, interdisciplinary efforts between biologists and computer scientists should be encouraged in the future.

Acknowledgements Research was partially supported by grants from NSF, Air Force, DTRA, and DURIP.

Recommended Reading

1. D.G. Albertson, D. Pinkel, Genomic microarrays in human genetic disease and cancer. *Hum. Mol. Genet.* **12**(Review Issue 2), R145 (2003)
2. R.S.K. Barnes, R.N. Hughes, *An Introduction to Marine Ecology* (Wiley, Oxford/Malden, 1999)
3. J.E. Beasley, P.C. Chu, A genetic algorithm for the set covering problem. *Eur. J. Oper. Res.* **94**(2), 392–404 (1996)
4. J.M. Berg, J.L. Tymoczko, L. Stryer, *Biochemistry (Looseleaf)* (Freeman, San Francisco 2008)
5. J. Borneman, M. Chrobak, G. Della Vedova, A. Figueroa, T. Jiang, Probe selection algorithms with applications in the analysis of microbial communities. *Bioinformatics* **17**(Suppl 1), S39 (2001)
6. Z. Bozdech, J. Zhu, M.P. Joachimiak, F.E. Cohen, B. Pulliam, J.L. DeRisi, et al., Expression profiling of the schizont and trophozoite stages of *Plasmodium falciparum* with a long-oligonucleotide microarray. *Genome Biol.* **4**(2), R9 (2003)
7. A. Caprara, M. Fischeretti, P. Toth, A heuristic method for the set covering problem. *Oper. Res.* **47**(5), 730–743 (1999)
8. D. Cazalís, T. Milledge, G. Narasimhan, Probe selection problem: structure and algorithms, in *Proceedings of the 8th Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2004)* (Citeseer, 2004), pp. 124–129
9. Y. Charbonnier, B. Gettler, P. François, M. Bento, A. Renzoni, P. Vaudaux, W. Schlegel, J. Schrenzel, A generic approach for the design of whole-genome oligoarrays, validated for genomotyping, deletion mapping and gene expression analysis on *Staphylococcus aureus*. *BMC Genomics* **6**(1), 95 (2005)
10. H. Chen, B.M. Sharp, Oliz, a suite of Perl scripts that assist in the design of microarrays using 50 mer oligonucleotides from the 3' untranslated region. *BMC Bioinform.* **3**(1), 27 (2002)
11. C.C. Chou, C.H. Chen, T.T. Lee, K. Peck, Optimization of probe length and the number of probes per gene for optimal microarray analysis of gene expression. *Nucl. Acids Res.* **32**(12), e99 (2004)

12. H.H. Chou, A.P. Hsia, D.L. Mooney, P.S. Schnable, Picky: oligo microarray design for large genomes. *Bioinformatics* **20**(17), 2893–2902 (2004)
13. F.S. Collins, M. Morgan, A. Patrinos, The Human Genome Project: lessons from large-scale biology. *Science* **300**(5617), 286–290 (2003)
14. T.M. Cover, J.A. Thomas, *Elements of Information Theory* (Wiley, New York, 2006)
15. C. Debouck, P.N. Goodfellow, DNA microarrays in drug discovery and development. *Nat. Genet.* **21**(1 suppl), 48–50 (1999)
16. P. Deng, M. Thai, Q. Ma, W. Wu, Efficient non-unique probes selection algorithms for DNA microarray. *BMC Genomics* **9**(Suppl 1), S22 (2008)
17. P.S. de Souza, S.N. Talukdar, Asynchronous organizations for multi-algorithm problems, in *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice* (ACM, New York, 1993), p. 293
18. D.Z. Du, F.K. Hwang, Pooling Designs: Group Testing in Molecular Biology (World Scientific, New Jersey, 2006)
19. S.J. Emrich, M. Lowe, A.L. Delcher, PROBEmer: a web-based software tool for selecting optimal DNA oligos. *Nucl. Acids Res.* **31**(13), 3746 (2003)
20. C.B. Epstein, R.A. Butow, Microarray technology—enhanced versatility, persistent challenge. *Curr. Opin. Biotechnol.* **11**(1), 36–41 (2000)
21. A. Figueroa, J. Borneman, T. Jiang, Clustering binary fingerprint vectors with missing values for dna array data analysis. *J. Comput. Biol.* **11**(5), 887–901 (2004)
22. L. Gasieniec, C.Y. Li, P. Sant, P.W.H. Wong, Efficient probe selection in microarray design, in *2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology, 2006. CIBCB'06*, Toronto (IEEE, 2007), pp. 1–8
23. L. Gasieniec, C.Y. Li, P. Sant, P.W.H. Wong, Randomized probe selection algorithm for microarray design. *J. Theor. Biol.* **248**(3), 512–521 (2007)
24. L.S. Ghoraie, R. Gras, L. Wang, A. Ngom, Bayesian optimization algorithm for the non-unique oligonucleotide probe selection problem, in *Proceedings of the 4th IAPR International Conference on Pattern Recognition in Bioinformatics*, Sheffield (Springer, 2009), pp. 365–376
25. L.S. Ghoraie, R. Gras, L. Wang, A. Ngom, Optimal decoding and minimal length for the non-unique oligonucleotide probe selection problem. *Neurocomputing* **73**(13), 2407–2418 (2010)
26. P.M.K. Gordon, C.W. Sansen, Osprey: a comprehensive tool employing novel methods for the design of oligonucleotides for DNA sequencing and microarrays. *Nucl. Acids Res.* **32**(17), e133 (2004)
27. E. Halperin, S. Halperin, T. Hartman, R. Shamir, Handling long targets and errors in sequencing by hybridization. *J. Comput. Biol.* **10**(3–4), 483–497 (2003)
28. E. Hartuv, A.O. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, R. Shamir, An algorithm for clustering cDNA fingerprints. *Genomics* **66**(3), 249–256 (2000)
29. R. Herwig, A.J. Poustka, C. Muller, C. Bull, H. Lehrach, J. O'Brien, Large-scale clustering of cDNA-fingerprinting data. *Genome Res.* **9**(11), 1093 (1999)
30. R. Herwig, A.O. Schmitt, M. Steinfath, J. O'Brien, H. Seidel, S. Meier-Ewert, H. Lehrach, U. Radelof, Information theoretical probe selection for hybridisation experiments. *Bioinformatics* **16**(10), 890 (2000)
31. D.S. Hochba, Approximation algorithms for NP-hard problems. *ACM SIGACT News* **28**(2), 40–52 (1997)
32. J.H. Holland, *Adaptation in Natural and Artificial Systems* (MIT, Cambridge, 1992)
33. N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J.V. Pearson, D.A. Stephan, S.F. Nelson, D.W. Craig, Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genet.* **4**(8), e1000167 (2008)
34. A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, Englewood Cliffs, 1988)
35. L. Kaderali, A. Schliep, Selecting signature oligonucleotides to identify organisms using DNA arrays. *Bioinformatics* **18**(10), 1340 (2002)

36. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing. *Science* **220**(4598), 671–680 (1983)
37. G.W. Klau, S. Rahmann, A. Schliep, M. Vingron, K. Reinert, Optimal robust non-unique probe selection using integer linear programming. *Bioinformatics* **20**(suppl 1), i186 (2004)
38. G.W. Klau, S. Rahmann, A. Schliep, M. Vingron, K. Reinert, Integer linear programming approaches for non-unique probe selection. *Discret. Appl. Math.* **155**(6–7), 840–856 (2007)
39. D.P. Kreil, R.R. Russell, S. Russell, [4] microarray oligonucleotide probes. *Methods Enzymol.* **410**, 73–98 (2006)
40. J.K. Lanctot, M. Li, B. Ma, S. Wang, L. Zhang, Distinguishing string selection problems, in *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, 1999, pp. 633–642
41. P. Larranaga, J.A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation* (Springer, Berlin, 2002)
42. F. Li, G.D. Stormo, Selection of optimal DNA oligos for gene expression arrays. *Bioinformatics* **17**(11), 1067 (2001)
43. X. Li, Z. He, J. Zhou, Selection of optimal oligonucleotide probes for microarrays using multiple criteria, global alignment and parameter estimation. *Nucl. Acids Res.* **33**(19), 6114 (2005)
44. D. Lipson, P. Webb, Z. Yakhini, Designing specific oligonucleotide probes for the entire *S. cerevisiae* transcriptome, in *Algorithms in Bioinformatics* (2002), pp. 491–505
45. D.J. Lockhart, H. Dong, M.C. Byrne, M.T. Follettie, M.V. Gallo, M.S. Chee, M. Mittmann, C. Wang, M. Kobayashi, H. Horton, et al., Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat. Biotechnol.* **14**(13), 1675 (1996)
46. J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability: Statistics*, vol. 1 (University of California Press, Berkeley, California, 1967), p. 281
47. J. Marmur, P. Doty, Thermal renaturation of deoxyribonucleic acids. *J. Mol. Biol.* **3**(5), 585–594 (1961)
48. S. Meier-Ewert, Global expression mapping of mammalian enomes, PhD thesis, University College, London, 1994
49. C.N. Meneses, Z. Lu, C.A.S. Oliveira, P.M. Pardalos, et al., Optimal solutions for the closest-string problem via integer programming. *INFORMS J. Comput.* **16**(4), 419–429 (2004)
50. C.N. Meneses, P.M. Pardalos, M.A. Ragle, A new approach to the non-unique probe selection problem. *Ann. Biomed. Eng.* **35**(4), 651–658 (2007)
51. C.N. Meneses, P.M. Pardalos, M. Ragle, Asynchronous teams for probe selection problems. *Discret. Optim.* **5**(1), 74–87 (2008)
52. B.G. Mirkin, *Mathematical Classification and Clustering* (Springer, Heidelberg, 1996)
53. R. Mrowka, J. Schuchhardt, C. Gille, Oligodb—interactive design of oligo DNA for transcription profiling of human genes. *Bioinformatics* **18**(12), 1686 (2002)
54. A. Ngom, L. Rueda, L. Wang, R. Gras, Selection based heuristics for the non-unique oligonucleotide probe selection problem in microarray design. *Pattern Recognit. Lett.* **31**(14), 2113–2125 (2010)
55. H.B. Nielsen, R. Wernersson, S. Knudsen, Design of oligonucleotides for microarrays and perspectives for design of multi-transcriptome arrays. *Nucl. Acids Res.* **31**(13), 3491 (2003)
56. E.K. Nordberg, YODA: selecting signature oligonucleotides. *Bioinformatics* **21**(8), 1365 (2005)
57. M. Pelikan, Bayesian optimization algorithm, in *Hierarchical Bayesian Optimization Algorithm* (Springer, Berlin/New York, 2005), pp. 31–48
58. P. Pudil, J. Novovicová, J. Kittler, Floating search methods in feature selection. *Pattern Recognit. Lett.* **15**(11), 1119–1125 (1994)
59. M.A. Ragle, Computational methods for the design and selection of hybridization probes, PhD thesis, D., University of Florida, 2007

60. M.A. Ragle, J.C. Smith, P.M. Pardalos, An optimal cutting-plane algorithm for solving the non-unique probe selection problem. *Ann. Biomed. Eng.* **35**(11), 2023–2030 (2007)
61. S. Rahmann, Fast and sensitive probe selection for dna chips using jumps in matching statistics, in *Proceedings of the 2003 IEEE Bioinformatics Conference, 2003. CSB 2003*, Stanford (IEEE, Washington, DC, 2003), pp. 57–64
62. S. Rahmann, Fast large scale oligonucleotide selection using the longest common factor approach. *Int. J. Bioinform. Comput. Biol.* **1**(2), 343–362 (2003)
63. S. Rahmann, Algorithms for probe selection and DNA microarray design. University of Berlin, Berlin, 208, 2004
64. J.B. Rampal, *DNA Arrays: Methods and Protocols* (Humana Press, Totowa, 2001)
65. S. Rash, D. Gusfield, String barcoding: uncovering optimal virus signatures, in *Proceedings of the Sixth Annual International Conference on Computational Biology* (ACM, New York, 2002), pp. 254–261
66. A. Relógio, C. Schwager, A. Richter, W. Ansorge, J. Valcárcel, Optimization of oligonucleotide-based DNA microarrays. *Nucl. Acids Res.* **30**(11), e51 (2002)
67. N. Reymond, H. Charles, L. Duret, F. Calevro, G. Beslon, J.M. Fayard, ROSO: optimizing oligonucleotide probes for microarrays. *Bioinformatics* **20**(2), 271 (2004)
68. S. Rimour, D. Hill, C. Milton, P. Peyret, GoArrays: highly dynamic and efficient microarray probe design. *Bioinformatics* **21**(7), 1094 (2005)
69. J.M. Rouillard, C.J. Herbert, M. Zuker, OligoArray: genome-scale oligonucleotide design for microarrays. *Bioinformatics* **18**(3), 486–487 (2002)
70. S. Rozen, H. Skaletsky, Primer3 on the WWW for general users and for biologist programmers. *Methods Mol. Biol.* **132**(3), 365–386 (2000)
71. J. SantaLucia Jr., H.T. Allawi, P.A. Seneviratne, Improved Nearest-Neighbor Parameters for Predicting DNA Duplex Stability. *Biochemistry* **35**(11), 3555–3562 (1996)
72. M.P. Sawicki, G. Samara, M. Hurwitz, E. Passaro Jr., Human genome project. *Am. J. Surg.* **165**(2), 258–264 (1993)
73. A. Schliep, DC Torney, S. Rahmann, Group testing with dna chips: generating designs and decoding experiments. *Proc. IEEE Comput. Soc. Bioinform. Conf.* **2**, 84–91 (2003)
74. C.E. Shannon, W. Weaver, *The Mathematical Theory of Communication* (Citeseer, 1959)
75. L. Soltan Ghoraie, R. Gras, L. Wang, A. Ngom, Bayesian optimization algorithm for the non-unique oligonucleotide probe selection problem. *Pattern Recognit. Bioinform.*, 365–376 (2009)
76. W.K. Sung, W.H. Lee, Fast and accurate probe selection algorithm for large genomes, in *Proceedings/IEEE Computer Society Bioinformatics Conference*, Stanford, vol. 2, 2003, p. 65
77. M.T. Thai, T. Znati, On the complexity and approximation of non-unique probe selection using d-disjunct matrix. *J. Comb. Optim.* **17**(1), 45–53 (2009)
78. N. Tolstrup, P.S. Nielsen, J.G. Kolberg, A.M. Frankel, H. Vissing, S. Kauppinen, OligoDesign: optimal design of LNA (locked nucleic acid) oligonucleotide capture probes for gene expression profiling. *Nucl. Acids Res.* **31**(13), 3758 (2003)
79. L. Wang, A. Ngom, A model-based approach to the non-unique oligonucleotide probe selection problem, in *Second International Conference on Bio-Inspired Models of Network, Information, and Computing Systems, 2007. Bionetics 2007*, Budapest (IEEE, 2007), pp. 209–215
80. L. Wang, A. Ngom, R. Gras, L. Rueda, An evolutionary approach to the non-unique oligonucleotide probe selection problem. *Trans. Comput. Syst. Biol. X*, **10**, 143–162 (2008)
81. L. Wang, A. Ngom, L. Rueda, Sequential forward selection approach to the non-unique oligonucleotide probe selection problem. *Pattern Recognit. Bioinform.*, 262–275 (2008)
82. X. Wang, B. Seed, Selection of oligonucleotide probes for protein coding sequences. *Bioinformatics* **19**(7), 796 (2003)
83. J.D. Watson, F.H.C. Crick, Molecular structure of nucleic acids. *Nature* **171**(4356), 737–738 (1953)
84. R. Wernersson, A.S. Juncker, H.B. Nielsen, Probe selection for DNA microarrays using OligoWiz. *Nat. Protoc.* **2**(11), 2677–2691 (2007)

85. R. Wernersson, H.B. Nielsen, OligoWiz 2.0 integrating sequence feature annotation into the design of microarray probes. *Nucl. Acids Res.* **33**(suppl 2), W611 (2005)
86. D. Xu, Y. Xu, G. Li, J. Zhou, A computer program for generating gene-specific fragments for microarrays, in *Currents in Computational Molecular Biology* (Universal Academy Press, Tokyo, 2000), pp. 3–4

Combinatorial Optimization in Data Mining

Samira Saedi and O. Erhun Kundakcioglu*

Contents

1	Introduction	596
2	Supervised Learning.....	597
2.1	Support Vector Machines.....	597
2.2	Consistent Biclustering.....	604
2.3	Other Classification Approaches.....	607
3	Unsupervised Learning.....	608
3.1	Latent Variable Models.....	609
3.2	Network-Based Models.....	612
3.3	Biclustering.....	613
4	Semi-supervised Learning.....	614
4.1	Semi-supervised Support Vector Machines (S^3VM).....	615
4.2	Expectation-Maximization (EM) Method.....	619
4.3	Graph-Based Methods.....	619
4.4	Co-training.....	620
5	Ranking.....	620
6	Feature Selection.....	621
7	Conclusion.....	623
	Cross-References.....	623
	Recommended Reading.....	624

*This work is supported by University of Houston New Faculty Research Grant.

S. Saedi (✉) • O.E. Kundakcioglu

Department of Industrial Engineering, University of Houston, Houston, TX, USA

e-mail: ssaedi@uh.edu; erhun@uh.edu

Abstract

This chapter presents data mining techniques that are formulated as combinatorial optimization problems together with their applications. There are a number of cases where fundamental data mining tool is not combinatorial in nature, yet widely used special-purpose combinatorial extensions exist. For the sake of completeness, these fundamental tools are also discussed in detail before the extensions with underlying combinatorial optimization problems. A number of computationally challenging data mining algorithms that have non-convex formulations are also explored.

1 Introduction

Data mining is defined as the practice of searching through large amounts of computerized data to find useful patterns or trends [101]. Being closely related to statistics and machine learning, data mining has been referred to as statistical learning and learning from the data. Different classifications of data mining methods exist in the literature [24], but one of the most commonly used classifications is based on the type of input data. Other than the features (attributes), input data can contain information on classes data instances belong to or responses of the underlying system, or it may contain no additional prior information. If information on classes or responses are available, the data is considered as *labeled*, and *supervised learning techniques* would be appropriate. Conversely, *unlabeled* data consists of only features with no class or response information, where *unsupervised learning techniques* attempt to identify patterns in data. *Semi-supervised learning*, which can be considered as halfway between supervised and unsupervised learning, utilizes both labeled and unlabeled data.

Regardless of the data mining tool employed, a data instance belongs to one of the two sets: *training set* or *test set*. Training set contains data instances that are used to train the data mining tool. That includes discovering patterns and relationships. Test set contains data instances that are used to assess the success of the predictions on relationships, which shed a light on the generalization performance of the employed tool.

Generalization performance of a model is its capability to find a more general classification based on given instances, i.e., how accurate the model can classify independent sample sets. To evaluate the generalization performance of a mining model, *cross validation* methods are used. Two widely used cross validation methods are *k-fold cross validation* and *leave-one-out cross validation (jackknife)*. In *k*-fold cross validation, all available data is randomly partitioned into *k* approximately equal subsets. Then, one of the *k* subsets is chosen as test set and the rest of the subsets are used as training set. This process is repeated *k* times in a way that each subset is chosen as test set once. The *k* results obtained will be averaged to find the generalization performance of the classification method. Leave-one-out cross validation (LOOCV), also known as jackknife, is a special case of *k*-fold where *k* is equal to the number of data instances.

In this chapter, methods to learn from data are explored in three main sections: supervised, unsupervised, and semi-supervised, with an emphasis on underlying combinatorial optimization problems. Discrete decision making based on analysis of data is the goal of fundamental problems in data mining (e.g., class assignment, feature selection, data categorization, identifying outlier instances). These problems are combinatorial in nature and can be formulated and solved by combinatorial approaches [33]. Last two sections are devoted to combinatorial ranking and feature selection problems.

2 Supervised Learning

Supervised learning refers to data mining tools that utilize *labeled* data. These labels can be *discrete* categories or *continuous* responses where the corresponding techniques are called *classification* or *regression*, respectively. Labels are input to the data mining tool during the training stage by a *supervisor*, hence the name supervised learning. In this section, combinatorial optimization problems are investigated in the context of classification and regression.

2.1 Support Vector Machines

Support vector machines (SVMs) are the state-of-the-art supervised machine learning methods that are initially introduced to classify pattern vectors that belong to two different classes (see [136]). Although there are multi-class generalizations, hyperplane-based methods have major drawbacks in classifying data from multiple classes (see [19]). Besides, classification of two classes has a wide variety of applications in image and voice recognition, text mining, healthcare, and clinical data mining.

The SVM classification function is a hyperplane that separates given two classes. The desired hyperplane maximizes the distance from the convex hulls of both classes. This problem can be formulated as a quadratic (and convex) optimization problem. SVM classifiers' success relies on strong fundamentals from the statistical learning theory, implementation advantages due to regularization (hence sparsity), and their generalization performance. When misclassified instances are penalized in the linear form, SVM classifiers are proven to be universally consistent (see [130]). A classifier is *consistent* if the probability of misclassification (in expectation) converges to a Bayes optimal rule when the number of data instances increases. A classifier is *universally consistent* if it is consistent for all distributions of data. SVMs can also perform nonlinear classification utilizing separating curves by implicitly embedding the original data in a nonlinear space using *kernel functions* (see, e.g., [121]).

The training is performed by minimizing a quadratic convex function that is subject to linear constraints. Although minimizing a convex function has a polynomial worst-case complexity, the general purpose methods are not practical for

large problems. SVM Light [78] and LIBSVM [71] are among the most frequently used implementations that use chunking [107] and decomposition [110] methods efficiently and use subsets of points to find a near-optimal hyperplane. Recently, Shalev-Shwartz et al. [120] propose a stochastic sub-gradient descent algorithm (Pegasos), whose run-time does not depend directly on the size of the training set but the bound on the number of nonzero features in each data instance. Experimental results show that Pegasos is especially successful for linear kernels.

SVM classifiers have a wide spectrum of application areas ranging from pattern recognition [92] and text categorization [77] to biomedicine [26, 43, 106, 109], cell death [112], nanotoxicology [113], brain-computer interface [60, 89], and financial applications [72, 134].

Based on SVM classifiers, support vector regression (SVR) is an optimization-based regression framework for solving machine learning problems. SVR approach is based on estimation of a linear function in a kernel-induced feature space. The objective is to optimize a certain boundary to the optimal regression line; therefore, errors within a certain distance (ε) of predicted value are disregarded. The learning algorithm minimizes a convex functional with sparse solution comparable to classification technique. For improved illustration, this can be considered a hyper-tube (insensitive band) about a linear function in the kernel-induced nonlinear space, such that pattern vectors in this tube are assumed not to contribute any error. This form of regression is called ε -insensitive because any point within ε distance of the anticipated regression function does not contribute an error. An important advantage for considering the ε -insensitive loss function is the sparseness of the dual variables similar to the case with SVM classifiers. Representing the solution by a small subset of training points has computational advantages. Furthermore, ε -insensitive regression ensures the existence of a global minimum and minimization of a reliable generalization error bound (see [45]).

SVR has various applications in numerous technological [15, 116], analytical [73, 91], and scientific fields [131, 145]. Wu et al. [142] perform location estimation using the Global System for Mobile communication (GSM) based on an SVR approach which demonstrates promising performances, especially in terrains with local variations in environmental factors. SVR method is also used in agricultural schemes in order to enhance output production and reduce losses [42, 108, 143]. Based on statistical learning theory, SVR has been used to deal with forecasting problems. Performing structural risk minimization rather than minimizing the training errors, SVR algorithms have better generalization ability than the conventional artificial neural networks [70].

2.1.1 SVM Classifiers: Mathematical Formulation

In a typical *binary classification* problem, class S^+ and S^- are composed of pattern vectors $\mathbf{x}_i \in \mathbb{R}^d$, $i = 1, \dots, n$. If $\mathbf{x}_i \in S^+$, it is given the label $y_i = 1$; if $\mathbf{x}_i \in S^-$, then it is given the label $y_i = -1$. The ultimate goal is to determine which class a new pattern vector $\mathbf{x}_i \notin \{S^+ \cup S^-\}$ belongs to. SVM classifiers solve this problem by finding a hyperplane (\mathbf{w}, b) that separates instances in classes S^+ and S^- with the maximum interclass margin.

A hyperplane in the feature space $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ is represented as the normal vector \mathbf{w} and the offset parameter b . The *geometric distance* to be maximized between a data point \mathbf{x}_i and the hyperplane is given by $(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) / \|\mathbf{w}\|$. Instead of maximizing the geometric distance fixing $\|\mathbf{w}\|$, the approach is minimizing $\|\mathbf{w}\|$ fixing $\langle \mathbf{w}, \mathbf{x}_i \rangle + b$, which is referred to as the *functional distance*.

In practice, many real-life problems are composed of nonseparable data which is generally due to noise. In this case, *slack variables* ξ_i are introduced for each pattern vector \mathbf{x}_i in the training set. Slack variables allow misclassifications for each pattern vector, but they are subject to a penalty to avoid trivial solutions. Therefore, the SVM classifier formulation is given as

$$[\text{SVM}] \quad \min_{\mathbf{w}, b, \xi} \quad \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2} \sum_{i=1}^n \xi_i^2 \quad (1a)$$

$$\text{subject to } y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad i = 1, \dots, n, \quad (1b)$$

where nonnegativity of the slack variables is assured implicitly since the solution cannot be optimal when $\xi_i < 0$ for any pattern vector.

Using the optimal solution (\mathbf{w}^*, b^*) for (1), a new pattern vector \mathbf{x}' can be classified as positive if $\langle \mathbf{w}^*, \mathbf{x}' \rangle + b^* > 0$ and negative if $\langle \mathbf{w}^*, \mathbf{x}' \rangle + b^* < 0$ (see Fig. 1). It is common to penalize the two-norm of the slack in the objective of SVM classifiers. Alternative formulations exist with one-norm penalization of the slack vector in the objective function, which is commonly referred to as the *hinge loss function* [45].

Lagrangian dual formulation of (1) and optimality conditions lead to an optimization problem where input vectors only appear in the form of dot products. Therefore, *kernel trick* can be introduced for nonlinear classification [45]. The dual problem is a concave maximization problem, which can also be solved efficiently. However, when the number of data points increases for better generalization performance, the dual tends to get harder to solve compared to the primal. The dual for two-norm soft margin formulation in (1) is given as

$$[\text{Dual - SVM}] \quad \max_{\alpha} \quad \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \frac{1}{2C} \sum_{i=1}^n \alpha_i^2 \quad (2a)$$

$$\text{subject to } \sum_{i=1}^n y_i \alpha_i = 0 \quad (2b)$$

$$\alpha_i \geq 0 \quad i = 1, \dots, n. \quad (2c)$$

When the dual formulation is used, b^* is calculated as follows using Karush-Kuhn-Tucker complementarity conditions:

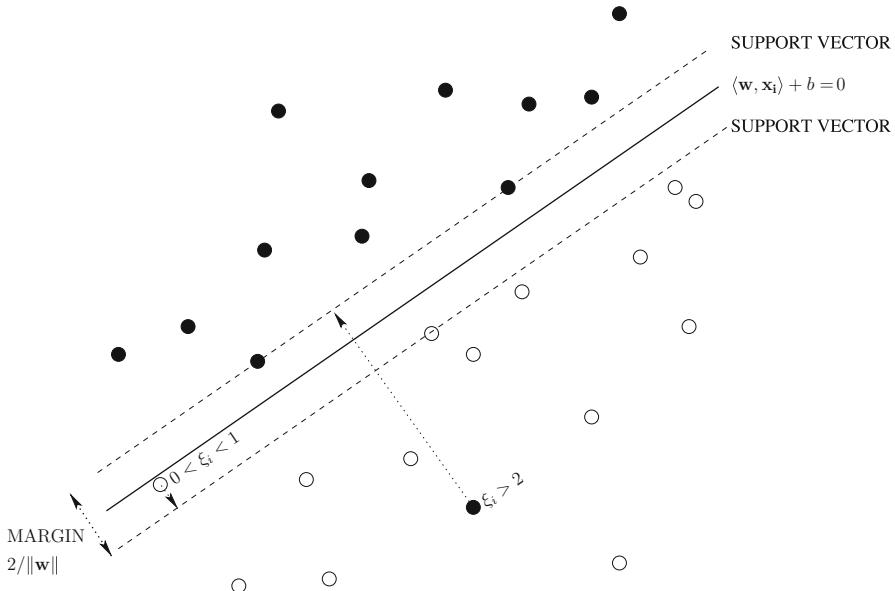


Fig. 1 Illustration of linear support vector machine classifiers

$$b^* = \sum_{i:\alpha_i^*>0} y_i - \sum_{j=1}^n y_j \alpha_j^* \langle \mathbf{x}_j, \mathbf{x}_i \rangle, \quad (3)$$

and a new pattern vector, \mathbf{x}' , can be classified as positive if $\sum_{j=1}^n y_j \alpha_j^* \langle \mathbf{x}_j, \mathbf{x}' \rangle + b^* > 0$, and negative otherwise. Note that dot products in (2) and (3) can be substituted with a suitable kernel function. The kernel function transforms the original *input space*, \mathcal{X} to a usually higher dimensional dot product space \mathcal{H} called the *feature space*, with a map $\Phi : \mathcal{X} \rightarrow \mathcal{H}$, such that $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle$. The function must satisfy conditions for Mercer's theorem which are equivalent to the requirement that the corresponding matrix is positive semidefinite for any finite subset of \mathcal{X} . Further information on the kernel trick can be found in [45, 117].

2.1.2 SVM Regressors: Mathematical Formulation

There are many reasonable choices of loss function for regression. SVR uses ε -insensitive loss function to ensure that the solution is characterized as the minimum of a convex functional. Another motivation for considering the ε -insensitive loss function is that it will ensure sparseness of the dual variables similar to SVM classifiers.

The linear ε -insensitive loss function $L^\varepsilon(\mathbf{x}, y, f)$ is defined by

$$L^\varepsilon(\mathbf{x}, y, f) = |y - f(\mathbf{x})|_\varepsilon = \max(0, |y - f(\mathbf{x})| - \varepsilon),$$

where f is a real-valued function on a domain X , $\mathbf{x} \in X$, and $y \in \mathbb{R}$. The quadratic ε -insensitive loss function $L_2^\varepsilon(\mathbf{x}, y, f)$ is similarly

$$L_2^\varepsilon(\mathbf{x}, y, f) = |y - f(\mathbf{x})|_\varepsilon^2.$$

Let \mathbf{X} be a set of pattern vectors $\mathbf{x}_i \in \mathbb{R}^d$, with dependent variable values (i.e., real-valued response) $y_i \in \mathbb{R}$. The problem of finding a regression function $f(\cdot)$ for these pattern vectors is minimizing the sum of ε -insensitive losses over all pattern vectors \mathbf{x} . Let the distance between the regression hyperplane and the closest pattern vector \mathbf{x}^* be $|(\langle \mathbf{w}, \mathbf{x}^* \rangle) + b|$. Then, the solution to the following quadratic programming problem finds the regression hyperplane with the minimum sum of quadratic ε -insensitive losses. Similar to the goal programming approach in SVM classifiers, there is an associated penalty of C for outliers. Therefore, the primal SVR problem is as follows:

$$[\text{SVR}] \quad \min_{\mathbf{w}, b, \xi, \hat{\xi}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2} \sum_{i=1}^n (\xi_i^2 + \hat{\xi}_i^2) \quad (4a)$$

$$\text{subject to } (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - y_i \leq \varepsilon + \xi_i \quad i = 1, \dots, n \quad (4b)$$

$$y_i - (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \leq \varepsilon + \hat{\xi}_i \quad i = 1, \dots, n. \quad (4c)$$

The main idea is to create a linear function in the kernel-induced space such that the quadratic loss function for regression from the generalization theory is minimized. First component of the objective function deals with minimizing the squared norm of the regression hyperplane ($\|\mathbf{w}\|^2$). The goal here is to enclose all pattern vectors with the ε -insensitive band of the regression hyperplane. Second component of the objective introduces penalty cost for instances outside the ε boundaries. Note that both ξ_i and $\hat{\xi}_i$ will be zero for all points inside the limits. The objective function seeks to minimize this penalty cost to reveal the best regression fit to the model. Constraints (4b)–(4c) imply that the pattern vectors are allowed to be ε below or above the target value without penalty. All pattern vectors outside the ε range are still allowed; however, they incur a cost of C [121].

The dual can be found using the Lagrangian function for the primal problem, differentiating this function with respect to the primal variables, and substituting equivalent expression for the primal variables back in the Lagrangian function. The resulting dual formulation is given as

$$[\text{Dual - SVR}] \quad \max_{\alpha, \hat{\alpha}} \quad -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_i - \alpha_i) \langle \mathbf{x}_i, \mathbf{x}_j \rangle \quad (5a)$$

$$-\varepsilon \sum_{i=1}^n (\alpha_i + \hat{\alpha}_i) + \sum_{i=1}^n y_i (\hat{\alpha}_i - \alpha_i)$$

$$\text{subject to } \sum_{i=1}^n (\alpha_i - \hat{\alpha}_i) = 0 \quad (5b)$$

$$0 \leq \alpha_i, \hat{\alpha}_i \leq C \quad i = 1, \dots, n. \quad (5c)$$

From the solution α^* and $\hat{\alpha}^*$, the regression function can be written as $f(\mathbf{x}) = \sum_{i=1}^n (\hat{\alpha}_i^* - \alpha_i^*) \langle \mathbf{x}_i, \mathbf{x} \rangle + b^*$, where b^* is chosen such that $f(\mathbf{x}_i) - y_i = -\varepsilon$ for any i with $0 < \hat{\alpha}_i^* < C$.

2.1.3 Combinatorial Extensions of Support Vector Machines

Recently proposed combinatorial optimization problems related to SVMs are based on penalization of different loss functions and generalizations of traditional classification problem. Integer programming formulations for SVM classifiers with the ramp loss (6) and hard margin loss (7) are proposed in [25]. Facet-defining inequalities are presented and both ramp loss and hard margin loss SVM classifiers are proven to be universally consistent. The idea behind these combinatorial formulations is to obtain classifiers that are more robust to the outliers compared to hinge or smooth loss functions:

$$[\text{RL - SVM}] \quad \min_{\mathbf{w}, b, \xi, \mathbf{z}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \left(\sum_{i=1}^n \xi_i + 2 \sum_{i=1}^n z_i \right) \quad (6a)$$

$$\text{subject to } y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i \quad \text{if } z_i = 0, \quad i = 1, \dots, n \quad (6b)$$

$$z_i \in \{0, 1\} \quad i = 1, \dots, n \quad (6c)$$

$$0 \leq \xi_i \leq 2 \quad i = 1, \dots, n \quad (6d)$$

$$[\text{HML - SVM}] \quad \min_{\mathbf{w}, b, \mathbf{z}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n z_i \quad (7a)$$

$$\text{subject to } y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \text{if } z_i = 0, i = 1, \dots, n \quad (7b)$$

$$z_i \in \{0, 1\} \quad i = 1, \dots, n. \quad (7c)$$

Seref et al. [118] introduce novel *selective* linear and nonlinear classification methods, in which sets of pattern vectors sharing the same label are given as input. One pattern vector is *selected* from each set in order to maximize the classification margin with respect to the selected positive and negative pattern vectors. The problem of selecting the best pattern vectors is referred to as the *hard selection* problem. Kernelized hard selection problems are also developed for classification. However, these combinatorial problems cannot be solved in polynomial time unless $\mathcal{P} = \mathcal{NP}$ [119]. Alternative approaches are proposed with relaxed formulations. The selective nature of these formulations is satisfied by the restricted free slack

concept. The intuition behind this concept is to reverse the combinatorial selection problem by detecting influential pattern vectors which require free slack to decrease their effect on the classification functions. Iteratively removing problematic pattern vectors, better classification results can be found.

Two variations of the free slack method, namely, pooled free slack (PFS) and free slack per set (FSS), are introduced for selective linear classification together with kernelized dual formulations for selective nonlinear classification. These methods are further extended to direct separation by increasing the total free slack to diminish the effect of multiple pattern vectors per set and provide more flexibility for the hyperplane to reorient itself with respect to well-separated pattern vectors. The performance of iterative elimination and direct selection algorithms is compared with each other, as well as with a naïve elimination algorithm that uses standard SVM method and ideas from the proposed methods. Results are reported for linear and nonlinear simulated data.

Kundakcioglu et al. [88] consider the margin maximization problem within the multiple instance learning (MIL) context. Training data is composed of labeled bags of instances. Despite the large number of margin maximization-based classification methods, there are only a few methods that consider the margin for MIL problems in the literature. A combinatorial margin maximization problem (8) is formulated for multiple instance classification which is proven to be \mathcal{NP} -hard. Kernel trick is applied on this formulation to classify nonlinear MIL data. A branch and bound algorithm is proposed that outperforms a leading commercial solver in terms of the best integer solution and optimality gap in a majority of image annotation and molecular activity prediction test cases. The major difference between the MIL setting and the selective setting is the interpretation of negative bags. In selective learning, a selection is performed on negative bags as well as positive bags. In MIL, on the other hand, only actual positives are to be discovered where all negative instances must be kept. The mixed-integer nonlinear programming (MINLP) formulation for this problem is as follows:

$$[\text{MI - SVM}] \quad \min_{\mathbf{w}, b, \xi, \eta} \quad \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{2} \sum_{i=1}^n \xi_i^2 \quad (8a)$$

$$\text{s.t.} \quad \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 1 - \xi_i - M(1 - \eta_i) \quad i \in I^+ \quad (8b)$$

$$- \langle \mathbf{w} \cdot \mathbf{x}_i \rangle - b \geq 1 - \xi_i \quad i \in I^- \quad (8c)$$

$$\sum_{i \in I_j} \eta_i \geq 1 \quad j \in J^+ \quad (8d)$$

$$\eta_i \in \{0, 1\} \quad i \in I^+. \quad (8e)$$

In this formulation, $I^+ = \{i : i \in I_j \wedge y_j = 1\}$ is the index set for instances that are in a positive bag, $I^- = \{i : i \in I_j \wedge y_j = -1\}$ is the index set for instances that are in a negative bag (negative instances), and $J^+ = \{j : y_j = 1\}$ is the index set for positive bags. Note that M is a sufficiently large number that ensures that

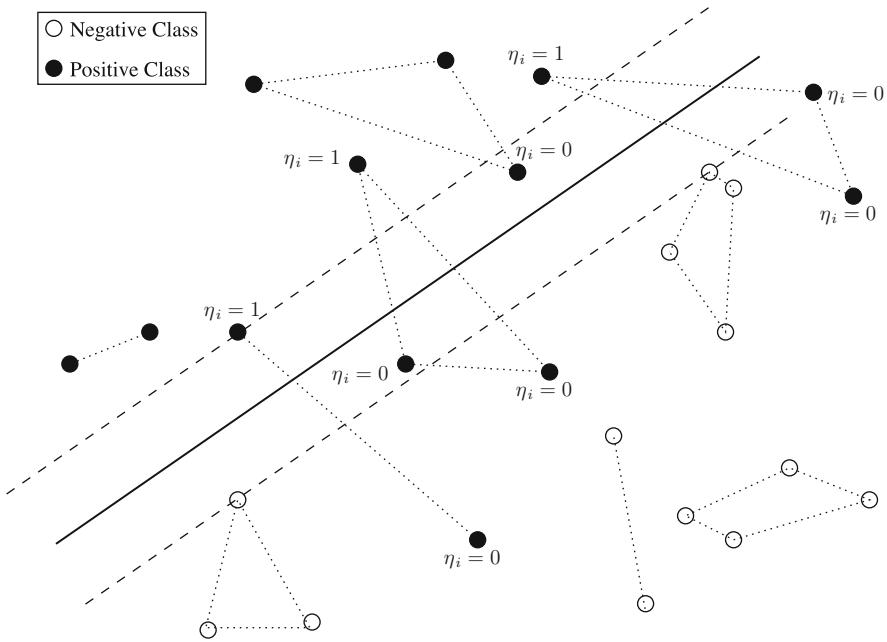


Fig. 2 Multiple instance support vector machine classifiers

the constraint is active if and only if $\eta_i = 1$. η_i is a binary variable that is 1 if i th instance is one of the actual positive examples of its bag (see Fig. 2).

Recently, Poursaeidi and Kundakcioglu [111] propose a hard margin loss formulation for multiple instance learning. The main idea is to avoid the dependency of the classifier in (Eq. 8) to the number of negative instances and make it dependent on the number of negative bags. Proposed formulations in [111] are also less sensitive to outliers, showing better generalization performance.

Next, *consistent biclustering* is introduced, another classification technique that employs combinatorial optimization formulations.

2.2 Consistent Biclustering

The concept of *consistent biclustering* is introduced by Busygin et al. [29]. In this method, a classification of features as well as data instances are needed as input. Formally, a biclustering \mathcal{B} is consistent if in each instance (feature) from any set S_r (set F_r), the average expression of features (instances) that belong to the same class r is greater than the average expression of features (instances) from other classes. The model for supervised biclustering involves solution of a special case of fractional 0–1 programming problem whose consistency is achieved by feature

selection. Computational results on microarray data mining problems are obtained by reformulating the problem as a linear mixed 0–1 programming problem.

An improved heuristic procedure is proposed in [103], where a linear programming problem with continuous variables is solved at each iteration. Numerical experiments on the data, which consists of instances from patients diagnosed with *acute lymphoblastic leukemia (ALL)* or *acute myeloid leukemia (AML)* diseases (see [10, 11, 64, 141, 144]), confirm that the algorithm outperforms the previous results in the quality of solution as well as computation time. Busygin et al. [30] use consistent biclustering to analyze scalp EEG data obtained from epileptic patients undergoing treatment with a vagus nerve stimulator (VNS).

Given an $m \times n$ data matrix A , each column represents a data instance and each row represents a feature. Formally, $A = (a_{ij})_{m \times n}$, where a_{ij} is the expression of i th feature of j th instance. A classification of instances is provided through a 0–1 matrix $S = (s_{jr})_{n \times k}$, where $s_{jr} = 1$ if instance j is classified as a member of the class r (i.e., $a^j \in S_r$), and $s_{jr} = 0$ otherwise. Similarly, given a classification of the features, F_r , let $F = (f_{ir})_{m \times k}$ denote a 0–1 matrix where $f_{ir} = 1$ if feature i belongs to class r (i.e., $a_i \in F_r$), and $f_{ir} = 0$ otherwise. Corresponding *centroids* are constructed as follows:

$$C_S = AS(S^T S)^{-1} = (c_{i\xi}^S)_{m \times r}, \quad (9)$$

$$C_F = A^T F(F^T F)^{-1} = (c_{j\xi}^F)_{n \times r}. \quad (10)$$

The elements of the matrices, $c_{i\xi}^S$ and $c_{j\xi}^F$, represent the average expression of the corresponding instance and feature in class ξ , respectively:

$$c_{i\xi}^S = \frac{\sum_{j=1}^n a_{ij} s_{j\xi}}{\sum_{j=1}^n s_{j\xi}} = \frac{\sum_{j|a^j \in S_\xi} a_{ij}}{|S_\xi|},$$

and

$$c_{j\xi}^F = \frac{\sum_{i=1}^m a_{ij} f_{i\xi}}{\sum_{i=1}^m f_{i\xi}} = \frac{\sum_{i|a_i \in F_\xi} a_{ij}}{|F_\xi|}.$$

Using the elements of matrix C_S , one can assign a feature to a class where it is overexpressed. Therefore, feature i is assigned to class \hat{r} if $c_{i\hat{r}}^S = \max_\xi \{c_{i\xi}^S\}$. Similarly, one can use the elements of matrix C_F to classify the instances. Data instance j is assigned to class \hat{r} if $c_{j\hat{r}}^F = \max_\xi \{c_{j\xi}^F\}$. Formally,

$$a_i \in \hat{F}_{\hat{r}} \implies c_{i\hat{r}}^S > c_{i\xi}^S, \quad \forall \xi, \xi \neq \hat{r}, \quad (11)$$

$$a^j \in \hat{S}_{\hat{r}} \implies c_{j\hat{r}}^F > c_{j\xi}^F, \quad \forall \xi, \xi \neq \hat{r}. \quad (12)$$

Note that the constructed classification of the features and instances \hat{F}_r and \hat{S}_r are not necessarily the same as classifications F_r and S_r , respectively.

Biclustering \mathcal{B} is referred to as a *consistent biclustering* if relations (11) and (12) hold for all elements of the corresponding classes, where matrices C_S and C_F are defined according to (9) and (10), respectively. A data set is *biclustering-admitting* if some consistent biclustering for it exists. Furthermore, the data set is called *conditionally biclustering-admitting* with respect to a given (partial) classification of some instances and/or features if there exists a consistent biclustering preserving the given (partial) classification. Busygin et al. [29] prove conic separability which also indicates convex hulls of classes do not intersect.

By definition, a biclustering is consistent if $F_r = \hat{F}_r$ and $S_r = \hat{S}_r$. However, a given data set might not have these properties. In such cases, one can remove a set of features and/or instances from the data set so that there is a consistent biclustering for the truncated data. This feature selection process may incorporate various objective functions depending on the desirable properties of the selected features. One general choice is to select the maximal possible number of features in order to lose minimal amount of information provided by the training set:

$$[\text{CB}] \quad \max_{\mathbf{x}} \quad \sum_{i=1}^m x_i \quad (13a)$$

$$\text{subject to} \quad \frac{\sum_{i=1}^m a_{ij} f_{i\hat{r}} x_i}{\sum_{i=1}^m f_{i\hat{r}} x_i} > \frac{\sum_{i=1}^m a_{ij} f_{i\xi} x_i}{\sum_{i=1}^m f_{i\xi} x_i} \quad \hat{r}, \xi = 1, \dots, k, \hat{r} \neq \xi, j \in S_{\hat{r}} \quad (13b)$$

$$x_i \in \{0, 1\} \quad i = 1, \dots, m. \quad (13c)$$

In this formulation, x_i , $i = 1, \dots, m$ are the decision variables. $x_i = 1$ if i th feature is selected, and $x_i = 0$ otherwise. $f_{ik} = 1$ if feature i belongs to class k , and $f_{ik} = 0$ otherwise. The objective is to maximize the number of features selected and (13b) ensures that the biclustering is consistent with respect to the selected features.

The goal of the CB problem is to find the largest set of features that can be used to construct a consistent biclustering. A problem with selecting the most representative feature set is the following. Assume that there is a consistent biclustering for a given data set, and there is a feature, i , where the difference between the two largest values of c_{ir}^S is negligible. For such cases, *additive* and *multiplicative consistent biclustering* are introduced in [103] by relaxing (11)–(12) with (14)–(15) and (16)–(17), respectively:

$$a_i \in F_{\hat{r}} \implies c_{i\hat{r}}^S > \alpha_i^S + c_{i\xi}^S, \quad \forall \xi, \xi \neq \hat{r} \quad (14)$$

$$a^j \in S_{\hat{r}} \implies c_{j\hat{r}}^F > \alpha_j^F + c_{j\xi}^F, \quad \forall \xi, \xi \neq \hat{r} \quad (15)$$

$$a_i \in F_{\hat{r}} \implies c_{i\hat{r}}^S > \beta_i^S c_{i\xi}^S, \quad \forall \xi, \xi \neq \hat{r} \quad (16)$$

$$a^j \in S_{\hat{r}} \implies c_{j\hat{r}}^F > \beta_j^F c_{j\xi}^F, \quad \forall \xi, \xi \neq \hat{r}. \quad (17)$$

Using these relaxations, α -consistent and β -consistent biclustering problem formulations are obtained as follows:

$$[\alpha\text{-CB}] \quad \max_{\mathbf{x}} \quad \sum_{i=1}^m x_i \quad (18a)$$

$$\text{subject to} \quad \frac{\sum_{i=1}^m a_{ij} f_{i\hat{r}} x_i}{\sum_{i=1}^m f_{i\hat{r}} x_i} > \alpha_j + \frac{\sum_{i=1}^m a_{ij} f_{i\xi} x_i}{\sum_{i=1}^m f_{i\xi} x_i} \quad \forall \hat{r},$$

$$\xi = 1, \dots, k, \hat{r} \neq \xi, j \in S_{\hat{r}} \quad (18b)$$

$$x_i \in \{0, 1\} \quad i = 1, \dots, m \quad (18c)$$

$$[\beta\text{-CB}] \quad \max_{\mathbf{x}} \quad \sum_{i=1}^m x_i \quad (19a)$$

$$\text{subject to} \quad \frac{\sum_{i=1}^m a_{ij} f_{i\hat{r}} x_i}{\sum_{i=1}^m f_{i\hat{r}} x_i} > \beta_j \frac{\sum_{i=1}^m a_{ij} f_{i\xi} x_i}{\sum_{i=1}^m f_{i\xi} x_i} \quad \forall \hat{r},$$

$$\xi = 1, \dots, k, \hat{r} \neq \xi, j \in S_{\hat{r}} \quad (19b)$$

$$x_i \in \{0, 1\} \quad i = 1, \dots, m. \quad (19c)$$

The information obtained from these solutions can be used to classify additional instances in the *test set*. These solutions are also useful for adjusting the values of vectors α and β to produce more characteristic features and decrease the number of misclassifications. Feature selection for consistent (13), α -consistent (18), and β -consistent biclustering (13) is proven to be \mathcal{NP} -hard [86].

2.3 Other Classification Approaches

A special case of biclustering introduced by Ben-Dor et al. [12] is the order-preserving submatrix problem, which is proven to be \mathcal{NP} -hard. The goal of this problem is to select a subset of rows and columns from the original data matrix in which there exists a permutation of columns such that in each row the values are strictly increasing. Trapp and Prokopyev [135] propose a general linear mixed 0–1 programming formulation and an iterative algorithm that makes use of a smaller linear 0–1 programming formulations. The proposed solution algorithm enhanced by a number of enhancements including valid inequalities and bounding schemes is able to solve problems with approximately 1,000 rows and 50 columns to optimality.

Bertsimas and Shioda [16] introduce mixed-integer linear methods to the classical statistical problems of classification and regression and construct a software

package called *classification and regression via integer optimization* (CRIo). CRIo separates data points into different polyhedral regions. In classification, each region is assigned a class, while in regression, each region has its own distinct regression coefficients. Computational experimentations show that CRIo is comparable to the leading methods in classification and regression.

Logical analysis of data is a technique that is used for risk prediction in medical applications [2]. This method is based on combinatorial optimization and boolean logic. The goal is essentially classifying groups of patients at low and high mortality risk, and LAD is shown to outperform standard methods used by cardiologists.

Kundakcioglu and Ünlüyurt [87] consider the problem of generating the sequence of tests required to reach a diagnostic conclusion with minimum average cost, which is also known as a test-sequencing problem. In this setting, the training data consists of asymmetrical tests (i.e., a probabilistic outcome for a feature or test) and the decision rule is obtained using an optimal binary AND/OR decision tree. An efficient bottom-up tree construction algorithm is developed based on fundamental ideas of Huffman coding. Computational results show that the algorithm outperforms previously proposed heuristic algorithms in reasonable time.

3 Unsupervised Learning

Unsupervised learning can be defined as finding patterns in unlabeled data. The goal is finding similarities among instances that belong to same class, where no class information is available. In this group of methods, N instances (x_1, x_2, \dots, x_N) of a random p -vector \mathbf{X} having joint density $Pr(\mathbf{X})$ are given and a decision is to be made based on representation of instances. Unlike the supervised methods that are trying to minimize some external error criterion, in unsupervised methods, calculating the difference between the target and input data instances is not possible. This results in different methods that are not relying on any outside information. It should be noted that unsupervised methods are particularly useful since unlabeled data is usually less costly compared to labeled data. Two widely used applications of unsupervised learning are clustering and dimensionality reduction.

Clustering (or cluster analysis) is defined as finding a convenient and valid organization of the data to establish rules for separating future data into categories [75]. Clustering algorithms can be divided into two major groups: *partitional* and *hierarchical* [74]. In partitional clustering, all clusters are found simultaneously as partitions of the data. In hierarchical clustering, nested clusters are found recursively. This can be done by one of the following two approaches. *Agglomerative methods* initially consider each data point as the head of its own cluster and try to merge the most similar pair of clusters successively to form a cluster hierarchy. On the contrary, *divisive methods* initially assign all data instances in one cluster and divide each cluster into smaller clusters (see [62]).

Another useful application of unsupervised learning is dimensionality reduction. These techniques aim to find a lower dimensional representation of the original data, which captures the content of the original data based on some criterion. A lower

dimensional representation of data is desired mainly because the computational advantage of handling a smaller data set outweighs the loss of information. Frequently used in the preprocessing stage, dimensionality reduction helps mitigate computational limitations of data mining algorithms for large data sets.

3.1 Latent Variable Models

In this section, probabilistic methods that utilize *latent variables* are introduced. Latent variable (or hidden variable) is a statistical variable that is not observed directly but can be estimated based on its relation with an observed variable. Latent variable models can be used to perform dimensionality reduction and clustering, the two cornerstones of unsupervised learning. Clustering problems that are addressed in this section are hard clustering problems and fuzzy clustering problems (FCP). In hard clustering problems, each data point needs to be assigned to one and only one cluster. On the other hand, in fuzzy clustering, each data point is a member of all clusters and it has a membership grade for each cluster showing its likelihood of belonging to that cluster.

3.1.1 k -Means

k -means is a hard clustering problem that has been introduced by several researchers across different disciplines, most notably Lloyd [95], Forgy [54], Friedman and Rubin [58], Ball and Hall [6], and MacQueen [96]. Given a data set of n vectors $x_j \in \mathbb{R}^d$, $j = 1, \dots, n$, the goal is to find a partition $S = \{S_1, S_2, \dots, S_k\}$ that minimizes the expected loss (see [96]). A random point $z \in \mathbb{R}^d$ with a known distribution p generates a loss proportional to the square of the error resulting from the choice of clusters, formally $\|z - \hat{z}\|^2$, where \hat{z} is the cluster centers or *exemplars*. The goal is to minimize the expected loss over possible choices of clusters, which is formally defined as

$$w^2(S) = \sum_{i=1}^k \int_{S_i} \|z - \hat{z}\|^2 dp(z). \quad (20)$$

Given a selection of cluster members from the data, it is easy to see that the cluster mean minimizes the squared error term. Therefore, the problem reduces to selection of cluster members, which can be formulated as the following mixed-integer programming (MIP) problem:

$$[k-\text{means}] \quad \min_{\mu, r} \quad \sum_{i=1}^k \sum_{j=1}^n r_{ij} \|x_j - \mu_i\|^2 \quad (21a)$$

$$\text{subject to} \quad \sum_{i=1}^k r_{ij} = 1 \quad j = 1, \dots, n \quad (21b)$$

$$r_{ij} \in \{0, 1\} \quad i = 1, \dots, k, \quad j = 1, \dots, n. \quad (21c)$$

This problem is proven to be \mathcal{NP} -hard [3]. One of the most widely used iterative heuristic approaches is the k -means algorithm, also known as Lloyd's algorithm [95], which can be considered as an expectation-maximization algorithm. Initially, k initial exemplars are randomly selected. Next, by repeating the following two steps, the clusters will be iteratively refined. In the first (i.e., *expectation*) step, each instance will be temporarily associated with the cluster whose exemplar is the closest. In the second (i.e., *maximization*) step, based on the temporary cluster assignment, each exemplar is updated as the mean of instances in its cluster. The algorithm terminates when no exemplar can be further updated. The convergence is guaranteed since the objective is finite and an improvement is assured in each step.

It should be noted that the term k -means is used to define the problem of minimizing the within-cluster sum of squares (i.e., (21)) as well as Lloyd's expectation-maximization algorithm defined above.

One observation is that k -means algorithm is sensitive to the initial set of exemplars especially when k is large. Thus, random initialization of exemplars is acceptable for relatively small values of k . There are alternative approaches, especially useful when number of clusters is increased, including, but not limited to, heuristic initialization schemes and parallel implementations with multiple initial exemplars.

Typically, k -means problem considers spherical clusters by minimizing Euclidean distance between points and cluster centers, as presented in formulations (20) and (21). Other distances that are used in k -means are mahalanobis distance metric [99], Itakura-Saito distance [93], L1 distance [81], and Bregman distance [7].

It has been proven that the worst-case complexity of the k -means algorithm is $O(kn^2\Delta^2)$ [4]. Although it is introduced over 50 years ago, k -means is still one of the most widely used algorithms for clustering. This algorithm is particularly useful due to its simplicity, ease of implementation, efficiency, and success in practice.

3.1.2 Message Passing

In order to overcome difficulties associated with the sensitivity of k -means to the initialization process, Frey and Dueck [57] propose a message-passing method where every data instance is a potential exemplar. The problem to be solved is the \mathcal{NP} -hard k -median problem. For computational tractability, an *affinity propagation* scheme is proposed, where data instances are nodes of a network, real-valued messages are transmitted along edges, and the magnitude of each message is updated based on the current affinity that one data instance has for choosing another data instance as its exemplar. Two types of messages are transmitted: First, *responsibility*, $r(i, j)$, sent from data instance i to candidate exemplar j , is a cumulative variable for how appropriate instance j is to be the exemplar

for instance i . Second, *availability*, $a(i, j)$, sent from potential exemplar j to data instance i , reflects the accumulated evidence for how good it is for instance i to select instance j as its exemplar. Availability includes support from other instances that instance j is an exemplar. Formally, responsibility and availability are defined as

$$r(i, j) \leftarrow s(i, j) - \max_{j' \text{ s.t. } j' \neq i} \{a(i, j') + s(i, j')\} \quad (22)$$

$$a(i, j) \leftarrow \min \left\{ 0, r(j, j) + \sum_{i' \text{ s.t. } i' \notin \{i, j\}} \max\{0, r(i', j)\} \right\}, \quad (23)$$

where $s(i, j)$ is the similarity between instances i and j . The objective is to minimize the total squared error; thus, $s(i, j)$ is defined as $-\|x_i - x_j\|^2$. Since availabilities are zero in the first iteration, $r(i, j)$ is equal to the input similarity between instance i and instance j as its exemplar minus the largest of the similarities between instance i and other potential exemplars. In the subsequent iterations, when an instance is effectively assigned to an exemplar, its updated availabilities will become below zero. This negative availability removes some of the exemplars from candidate exemplars of that data instance. If availability of some other instances becomes negative for that exemplar, the exemplar is out of competition.

When $j = i$, the responsibility $r(j, j)$ equals $s(j, j)$, minus the largest of the similarities between instance i and all other candidate exemplars. If $r(j, j)$ (i.e., self-responsibility) is negative at any iteration, it is more appropriate for instance j to belong to another exemplar rather than being an exemplar itself. This transmission continues until a good set of exemplars and corresponding clusters are obtained.

3.1.3 Fuzzy Clustering

Fuzzy clustering is an extension of k -means problem, in which instances have a degree of membership for *all* clusters. In this setting, a data instance is not designated to a specific cluster, but it belongs to each cluster to a certain extent, referred to as the *membership grade*. To solve this problem, different nonexact (i.e., soft computing) approaches have been introduced in the literature (see [8, 17, 82, 102]). An exact reformulation-linearization technique (RLT)-based approach has been introduced by Sherali and Desai [123], where the objective is to minimize the total degree-2 fuzzifier weighted squared Euclidean distance. Formally, given a data set of n vectors $a_j \in \mathbb{R}^d$, $j = 1, \dots, n$, the problem is

$$[\text{FCP}] \quad \min_{\mathbf{w}, \mathbf{z}} \quad \sum_{i=1}^n \sum_{j=1}^c w_{ij}^2 \|a_i - z_j\|^2 \quad (24a)$$

$$\text{subject to } \sum_{j=1}^c w_{ij} = 1 \quad \forall i = 1, \dots, n \quad (24b)$$

$$w_{ij} \geq 0 \quad \forall (i, j), \quad (24c)$$

which is essentially a weighted version of (21). Since the objective function (24a) is nonlinear, RLT is used to relax (24) as a linear programming problem and a specialized branch and bound algorithm is employed. To further speed up the process, data reduction is performed on instances by replacing group examples with their centroids [53].

3.2 Network-Based Models

In the last few decades, network-based data mining models have received increasing attention for their capability of extracting useful information from large-scale data sets that are widely available in network structure. Boginski [22] presents a wide variety of applications for network-based data mining models.

One use of networks is finding the structural properties of a data set by *degree* (i.e., number of edges emanating from a node) distribution of constructed graphs, which is a representation of large-scale pattern of connections in the graph. This leads to the fact that graphs that are related to completely different data sets might have a similar well-defined power-law structure. Power-law structure states that the probability that a vertex of a graph has degree k is $P(k) \propto k^{-\gamma}$ [22].

Network-based models are widely used in clustering. Finding *cliques* and *independent sets* in a network is synonymous with discovering clusters in that data set. Given a graph $G = (V, E)$, a subgraph whose vertices are pairwise adjacent is called a clique or complete subgraph. On the contrary, an independent set or stable set is a set of vertices none of which are adjacent. Therefore, maximum clique in a graph gives the maximum possible size of a group of similar objects, whereas maximum independent set leads to the largest group of essentially different objects. It should be noted that a clique in graph is an independent set in the complement graph and vice versa. Maximum clique problem and the maximum independent set problem are proven to be \mathcal{NP} -hard [61].

These two problems are extended to find the minimum number of distinct cliques for clustering purposes [22]. There also exist relaxations of clique problems such as quasi-clique problems ensuring a lower bound on the total number of edges within the subgraph [1] and k -plex problems ensuring a lower bound on the number of adjacent nodes for each selected node within the subgraph [5]. See [125] for a detailed survey on combinatorial optimization techniques for network-based data mining.

3.3 Biclustering

Biclustering is simultaneous partitioning of data instances and their features into classes. Instances and features classified together are expected to have a high relevance with each other (i.e., similar data values).

Given an $m \times n$ data matrix A , each column represents a data instance and each row represents a feature. Formally, $A = (a_{ij})_{m \times n}$, where a_{ij} is the expression of i th feature of j th instance. Biclustering is applied by simultaneous partitioning of the instances and features into k classes. Let S_1, S_2, \dots, S_k denote the classes of the instances (columns) and F_1, F_2, \dots, F_k denote the classes of features (rows). Biclustering is defined as a collection of pairs of instance and feature subsets $\mathcal{B} = \{(S_1, F_1), (S_2, F_2), \dots, (S_k, F_k)\}$ such that

$$S_1, S_2, \dots, S_k \subseteq \{a^j\}_{j=1,\dots,n},$$

$$\bigcup_{r=1}^k S_r = \{a^j\}_{j=1,\dots,n},$$

$$S_\zeta \cap S_\xi = \emptyset \Leftrightarrow \zeta \neq \xi,$$

$$F_1, F_2, \dots, F_k \subseteq \{a_i\}_{i=1,\dots,m},$$

$$\bigcup_{r=1}^k F_r = \{a_i\}_{i=1,\dots,m},$$

$$F_\zeta \cap F_\xi = \emptyset \Leftrightarrow \zeta \neq \xi,$$

where $\{a^j\}_{j=1,\dots,n}$ and $\{a_i\}_{i=1,\dots,m}$ denote the set of columns and rows of the matrix A , respectively. The ultimate goal in a biclustering problem is to find a partitioning scheme for which instances from the same class have *similar* values for that class' characteristic features. One of the early algorithms to obtain an appropriate biclustering is [69], which is known as *block clustering*. Given a biclustering \mathcal{B} , the variability of the data in the block (S_r, F_r) is used to measure the quality of partitions. A lower variability in the resulting problem is desired. The number of classes should be fixed in order to avoid a trivial, zero variability solution in which each class consists of only one instance. A more sophisticated approach for biclustering is introduced in [40], where the objective is to minimize the mean squared residual. In this setting, the problem is proven to be \mathcal{NP} -hard and a greedy algorithm is proposed to find an approximate solution. A simulated annealing algorithm is proposed in [27]. Dhillon [49] proposes another biclustering method for text mining using a bipartite graph. In the graph, the nodes represent features and instances, and each feature i is connected to an instance j with a link (i, j) ,

which has a weight a_{ij} . The total weight of all links connecting features and instances from different classes is used to measure the quality of a biclustering. A lower value corresponds to a better biclustering. A similar method for microarray data is suggested in [83].

Dhillon et al. [50] treat input data as a joint probability distribution between two discrete sets of random variables. The goal of the method is to find disjoint classes for both variables. A Bayesian biclustering technique based on Gibbs sampling can be found in [122]. A detailed survey on biclustering techniques can be found in [31, 97].

4 Semi-supervised Learning

Semi-supervised learning is a branch of machine learning that utilizes both labeled and unlabeled data to improve generalization performance. Since unlabeled data is relatively easier to collect, these methods are less expensive than supervised learning methods. Moreover, these methods are more accurate than unsupervised learning because they are using information obtained from labeled data as well.

Data used in these methods are comprised a set to be used for training and a set to be used for testing. Let $X = (x_1, \dots, x_n)$ be a set that consists of l labeled examples $\{(x_i, y_i)\}_{i=1}^l$, $y_i = \pm 1$ and of u unlabeled examples $\{x_i\}_{i=l+1}^n$. In the literature, u unlabeled examples are referred to as the *working set* and l labeled examples are referred to as the *training set* [14]. Both training set and working set in this definition are used during the training process. Test set is used to test the accuracy of the learning method as usual.

Based on definition of semi-supervised learning, two types of data are available: the similarity distances for the training and working set and class labels of training set. According to the capability of an algorithm to handle unseen data instances (in the test set), semi-supervised methods can be classified as *transductive* or *inductive*. In transductive learning, training, and working sets are processed but test set cannot be used during the algorithm. The early graph-based methods in semi-supervised learning are transductive. On the other hand, inductive learning algorithms can handle test set as well. Semi-supervised learning algorithms can be divided into three main groups based on the data representation considered. These groups are *manifold assumption*, *cluster assumption*, and *manifold-cluster assumption*. In manifold assumption, the data instances lie on a low-dimensional manifold in the input space. These algorithms generally represent data as a graph, instances as vertices, and pairwise similarities between instances as edge weights. In cluster assumption, decision boundaries between classes must lie in low-density regions as the aim of these algorithms is placing instances with high similarities in the same cluster. Manifold-cluster assumption proposed by Mallapragada et al. [98] uses both manifold and cluster assumptions to overcome weaknesses of both approaches.

One of the earliest semi-supervised learning methods proposed in the literature is *self-training*. This method initially constructs a temporary classifier using only

labeled data. Using this classifier, labels are predicted for the instances in the working set and the classifier is progressively updated [150]. Next, some of the most notable methods for semi-supervised learning are introduced.

4.1 Semi-supervised Support Vector Machines ($S^3\text{VM}$)

The idea of using SVMs to solve semi-supervised learning problems is introduced by Vapnik and Sterin [138]. The combinatorial formulation for semi-supervised support vector machines is as follows (see [37]):

$$[\text{S}^3\text{VM}] \quad \min_{\mathbf{w}, b, \xi, \mathbf{y} \in \mathbb{R}^u} \frac{1}{2} \|\mathbf{w}\|^2 + C \left[\sum_{i=1}^l \xi_i \right]^p + C^* \left[\sum_{j=l+1}^{l+u} \xi_j \right]^p \quad (25a)$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) + \xi_i \geq 1 \quad i = 1, \dots, l+u \quad (25b)$$

$$\xi_i \geq 0 \quad i = 1, \dots, l+u \quad (25c)$$

$$\frac{1}{u} \sum_{i=l+1}^{l+u} y_i = 2r - 1 \quad (25d)$$

$$y_i \in \{-1, 1\} \quad i = l+1, \dots, l+u. \quad (25e)$$

In this formulation, y_i are given as input for $i = 1, \dots, l$ (i.e., training set) and are decision variables for $i = l+1, \dots, l+u$ (i.e., working set). Equation (25d) is called the *balancing constraint* since it enforces the solutions to be balanced by assigning specified percentages of unlabeled data to positive and negative classes [133]. r is the ratio of positive labels to be assigned to the number of instances in the working set.¹ Usually, r is hard to predict; thus, it is assumed to be equal to the ratio of positive labels in the training set. C and C^* represent the penalty weight for labeled (training) and unlabeled (working) data, respectively. To obtain a good generalization performance, ideally C and C^* should be different (see [38]); however, they are usually assumed equal for the sake of simplicity. Some common penalty functions for the objective are linear (i.e., $p = 1$) and quadratic (i.e., $p = 2$) functions.

Bennett and Demiriz [14] introduce a 0–1 variable d_j for each instance x_j in working set to formulate (25) as a MINLP problem. An instance belongs to positive class if $d_j = 1$ and negative class if $d_j = 0$. With this change, $S^3\text{VM}$ can be formulated as

¹An exact equality for balancing constraint is likely to lead to infeasible solutions depending on the number of instances and ratios. Therefore, a subtle adjustment is usually necessary to ensure feasibility.

$$[\mathbf{S}^3\mathbf{VM - MINLP}] \quad \min_{\mathbf{w}, b, \eta, \xi, \mathbf{z}} \quad C \left[\sum_{i=1}^l \eta_i + \sum_{j=l+1}^{l+u} (\xi_j + z_j) \right] + \|\mathbf{w}\| \quad (26a)$$

$$\text{subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i - b) + \eta_i \geq 1 \quad i = 1, \dots, l \quad (26b)$$

$$\mathbf{w} \cdot \mathbf{x}_j - b + \xi_j + M(1 - d_j) \geq 1 \quad j = l+1, \dots, l+u \quad (26c)$$

$$-(\mathbf{w} \cdot \mathbf{x}_j - b) + z_j + Md_j \geq 1 \quad j = l+1, \dots, l+u \quad (26d)$$

$$\eta_i \geq 0 \quad i = 1, \dots, l \quad (26e)$$

$$\xi_j \geq 0 \quad j = l+1, \dots, l+u \quad (26f)$$

$$z_j \geq 0 \quad j = l+1, \dots, l+u \quad (26g)$$

$$d_j \in \{0, 1\} \quad j = l+1, \dots, l+u \quad (26h)$$

where M is a large enough number that ensures $\xi_j = 0$ when $d_j = 0$ and $z_j = 0$ when $d_j = 1$. It should be noted that Bennett and Demiriz [14] assume $C = C^*$, ignore balancing constraint, and use one-norm of \mathbf{w} in the objective function.

A number of methods have been introduced to solve non-convex problem (25) (see [38]). These methods typically utilize continuous relaxations or heuristics except one hard approach solving (25) directly.

4.1.1 Branch and Bound

Proposed briefly as a framework in [137], this method is used to find the global optimum for (25). Chapelle et al. [37] implemented a branch and bound algorithm that performs a search over y_u 's. This approach is particularly useful for relatively smaller data sets due to its complexity. Based on empirical evidence, the generalization performance of the global optimum solution can be significantly better than nonexact solutions.

The initial solution for the root node of branch and bound tree is the SVM solution for labeled training data. Branching is performed on unlabeled instances in the working set. To achieve optimal solution quickly, unlabeled instance x^* to be labeled next is selected in a way that its label (y^*) assignment has a high potential to improve the objective function. To formulate this concept, let $s(L)$ be the SVM objective function trained on labeled set that is formally defined as

$$s(L) = \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{(x_i, y_i)} \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b))^2. \quad (27)$$

The branching is performed on the following unlabeled instance:

$$\arg \max_{x \in U, y \in \pm 1} s(L \cup \{x, y\}). \quad (28)$$

The lower bound for each node will be the objective function value after optimizing a standard SVM on instances that have been labeled so far, i.e., $s(L)$ introduced in (27). The incumbent solution will set the upper bound; thus, exploration strategy in the tree is depth first search. This approach promotes reaching leaves frequently and having tight upper bounds to perform aggressive pruning.

4.1.2 Continuous Relaxations

To solve the problem by continuous optimization methods, integer variables y_i for $i = l+1, \dots, l+u$ should be relaxed. Chapelle et al. [36, 38] introduce a soft computing approach that balances the distance of unlabeled instances from the hyperplane instead of the number of instances as in (25d). The problem is formulated as

$$[\text{S}^3\text{VM} - \mathbf{R}] \quad \min_{\mathbf{w}, b, \xi, z} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l \xi_i^p + C^* \sum_{j=l+1}^{l+u} z_j^p \quad (29a)$$

$$\text{subject to} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \quad i = 1, \dots, l \quad (29b)$$

$$\xi_i \geq 0 \quad i = 1, \dots, l \quad (29c)$$

$$|\mathbf{w} \cdot \mathbf{x}_j + b| \geq 1 - z_j \quad j = l+1, \dots, l+u \quad (29d)$$

$$z_j \geq 0 \quad j = l+1, \dots, l+u \quad (29e)$$

$$\frac{1}{u} \sum_{j=l+1}^{l+u} \mathbf{w}^T \mathbf{x}_j = \tilde{r} - 1. \quad (29f)$$

Note that (29f) balances the sum of distances for the working set, given \tilde{r} . This constraint is first proposed in [35], and the problem with and without the balancing constraint is solved in [36, 38], respectively.

The problem can be solved via a concave convex-procedure, where the non-convex objective function is rewritten as the sum of a convex component and a concave component [148]. At each iteration of this algorithm, concave part is estimated by its tangent and the convex part is minimized using traditional algorithms. This type of methods to minimize the concave function is discussed by Fung and Mangasarian [59] and improved in [44, 139] to handle larger data sets.

$\nabla \text{S}^3\text{VM}$ method [35] minimizes the objective function by gradient descent. Since the objective function is not smooth, z_j^p in the objective function is replaced by an exponential function $\exp(-s(\mathbf{w} \cdot \mathbf{x}_j + b)^p)$, where s is a parameter. Furthermore, an annealing procedure is performed by increasing C^* at each iteration. Chapelle et al. [36] propose a continuation technique to solve the S^3VM problem. In this method, a system of nonlinear equations is solved through a simpler system of equations

by smoothing the objective function. Annealing is performed without increasing C^* , but instead keeping it fixed, and continuation technique is used to transform the objective function. Chapelle [34] later introduces Newton S³VM that is more efficient compared to ∇ S³VM and Continuation S³VM. This technique utilizes a new loss function for unlabeled instances in the primal problem and an associated Newton method.

Bie and Cristianini [18] propose a convex relaxation using the dual of (25) and solve through semidefinite programming (SDP). The dual formulation also allows kernel trick to be used for nonlinear classification and classification of nonvectorial data. The dual problem can be formulated as

$$\min_{\mathbf{Y}} \max_{\alpha} 2\alpha^T \mathbf{1} - \alpha^T (K \odot \mathbf{Y}\mathbf{Y}^T)\alpha \quad (30a)$$

$$\text{subject to } 0 \leq \alpha_i \leq C \quad i = 1, \dots, l+u \quad (30b)$$

$$\mathbf{Y}_u \in \{-1, 1\}^u. \quad (30c)$$

In this formulation, $\alpha = (\alpha_1, \dots, \alpha_{l+u})$ is a vector of dual variables α_i , K is the complete kernel matrix, and $\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_l \\ \mathbf{Y}_u \end{bmatrix}$ is the complete label vector. The optimization problem (30) can be reformulated as follows by introducing the outer product of labels $\Gamma = \mathbf{Y}\mathbf{Y}^T$:

$$\min_{\Gamma} \max_{\alpha} 2\alpha^T \mathbf{1} - \alpha^T (K \odot \Gamma)\alpha \quad (31a)$$

$$\text{subject to } 0 \leq \alpha_i \leq C \quad i = 1, \dots, l+u \quad (31b)$$

$$\mathbf{Y}_u \in \{-1, 1\}^u. \quad (31c)$$

In (31), $\Gamma = \mathbf{Y}\mathbf{Y}^T = \begin{bmatrix} \mathbf{Y}_l \mathbf{Y}_l^T & \mathbf{Y}_l \mathbf{Y}_u^T \\ \mathbf{Y}_u \mathbf{Y}_l^T & \mathbf{Y}_u \mathbf{Y}_u^T \end{bmatrix}$, objective function is linear in Γ , concave in α , and constraints are linear. The problem is not convex due to (31c); therefore, a relaxation is proposed as

$$[\mathbf{S}^3\mathbf{VM} - \mathbf{SDP}] \quad \min_{\Gamma} \max_{\alpha} 2\alpha^T \mathbf{1} - \alpha^T (K \odot \Gamma)\alpha \quad (32a)$$

$$\text{subject to } 0 \leq \alpha_i \leq C \quad i = 1, \dots, l+u \quad (32b)$$

$$\text{diag}(\Gamma) = 1 \quad (32c)$$

$$\Gamma \succeq 0, \quad (32d)$$

where $\Gamma = \begin{pmatrix} \mathbf{Y}_l \mathbf{Y}_l^T & \Gamma_{lu} \\ \Gamma_{ul} & \Gamma_{uu} \end{pmatrix}$. In (32), \mathbf{Y}_u 's lie in the interval $[-1, 1]$. Furthermore, rank of Γ is not necessarily 1. Although the problem is convex, this method is not widely used due to its high time complexity $O((1 + u^2)^2(l + u)^{2.5})$ [38].

4.1.3 Heuristics

Introduced by Thorsten [133], S^3VM^{light} is an S^3VM implementation within the SVM^{light} software package. It is based on local combinatorial search guided by a label-switching procedure. In this method, the labels of test instances are switched in a way that improves the objective function. In an outer loop, the value of C^* is gradually increased. This temporary adjustment of C^* helps algorithm avoid suboptimal local minima.

Deterministic annealing (DA) is a metaheuristic that has been used to solve hard combinatorial or non-convex problems. Sindhiani and Keerthi [127] use DA for S^3VM s by relaxing discrete label variables y_u to real-valued variables $p_u = (p_{l+1}, \dots, p_{l+u})$, where p_i is interpreted as the probability that $y_i = 1$. Based on these new variables, an easier reformulation is solved.

4.2 Expectation-Maximization (EM) Method

The expectation-maximization (EM) algorithm is first introduced by Dempster et al. [48] as an iterative procedure for estimating parameter values that maximize the likelihood function, when there is missing data. This method has two steps: *expectation* (E-step) and *maximization* (M-step). E-step calculates the expected values of the sufficient statistics given the current parameter estimates. M-step sets parameters to their maximum likelihood estimates given the estimated values of the sufficient statistics. Starting with a randomly generated set of parameter estimates, these steps are repeated either for a predefined number of steps or more commonly until the difference between parameter estimates in two consecutive iterations is negligible. In this setting, labels for working set are considered as missing data and EM method attempts to label these instances [100].

4.3 Graph-Based Methods

Graph-based semi-supervised methods define an empirical graph $G = (V, E)$. In this graph, nodes $V = 1, \dots, n$ are instances given in training and working sets, and edges E represent the similarity between these instances. A weight matrix $W = [w_{ij}]$ is also defined in a way that $w_{ij} \neq 0$ if there is an edge between i and j . Weight matrix can be introduced in a number of different ways. For example, W can be a k -nearest neighbor matrix, i.e., $w_{ij} = 1$ if i is among the k -nearest neighbors of j . The way the weight matrix is defined affects the performance of graph-based semi-supervised learning [41].

These methods rely on the geometry of data induced by labeled and unlabeled instances and usually assume a smooth distribution of labels over the graph. Graph-based methods are nonparametric, discriminative, and transductive in nature, yet they can be modified to be inductive. Goal of a graph-based method can be *label propagation* on a similarity graph or minimization of a *quadratic cost criterion*.

In label propagation, known labels are used to spread information through the graph in order to label unlabeled nodes [151]. Szummer and Jaakkola [132] propose a Markov random walk algorithm based on label propagation on similarity graph. In this method, labeling is performed based on how easy it gets to move from one node to the other via a random walk.

In quadratic cost criterion-based methods, rapid changes of estimated labels (i.e., \hat{Y}) between close instances are penalized. The penalty term is defined as follows:

$$\begin{aligned} \frac{1}{2} \sum_{i,j=1}^n W_{i,j} (\hat{y}_i - \hat{y}_j)^2 &= \frac{1}{2} \left(2 \sum_{i=1}^n \hat{y}_i^2 \sum_{j=1}^n W_{ij} - 2 \sum_{i,j=1}^n W_{ij} \hat{y}_i \hat{y}_j \right) \\ &= \hat{Y}^T (D - W) \hat{Y} \\ &= \hat{Y}^T L \hat{Y}. \end{aligned} \quad (33)$$

In (33), $L = D - W$ is an un-normalized graph laplacian. In [153], the penalty term is minimized over \hat{Y}_u . Other methods considering quadratic cost criterion can be found in [9, 47, 79]. Bengio et al. [13] show that minimizing quadratic cost criterion is equivalent to label propagation.

4.4 Co-training

In co-training, two learners are iteratively combining their outputs to increase size of the training set, which is reused for training and generating more labeled data automatically. Co-training is proposed by Blum and Mitchell [21] for semi-supervised learning. This technique assumes features are separable into two sets that are conditionally independent given the classes. In first step, two classifiers are trained with the labeled data using subfeature sets. Then, each classifier is used to designate labels to unlabeled instances. Each classifier is retrained by the most confident predictions given by the other classifier. This process is repeated until all instances in the working set are labeled [149].

5 Ranking

Ranking is one of the fastest growing areas in machine learning. In ranking, a group of alternatives are ranked based on aggregate scores assigned by voters. Some of the most popular applications are internet databases, search engines, and e-commerce sites. A number of algorithms have been proposed for ranking in the literature [28, 52, 56] with different limitations.

Recently, Jiang et al. [76] propose a model called HodgeRank using graph Helmholzian. Alternatives to be ranked by voters are vertices of a graph, where preferences are quantified and aggregated into an edge flow. Hodge theory yields an orthogonal decomposition of the edge flow known as the Hodge or

Helmholtz decomposition and analyzes pairwise rankings represented as edge flows. HodgeRank uses combinatorial Hodge theory to reduce rank aggregation to a linear least squares regression and avoids usual \mathcal{NP} -hard combinatorial optimization problems. HodgeRank outperforms classical ranking methods due to its ability to handle incomplete and imbalanced data as well as large complex network data. Furthermore, although this method is designed for cardinal data (i.e., each alternative is given a score), it can also give insights about ordinal data sets (i.e., a set of alternatives is compared and ranked by each voter).

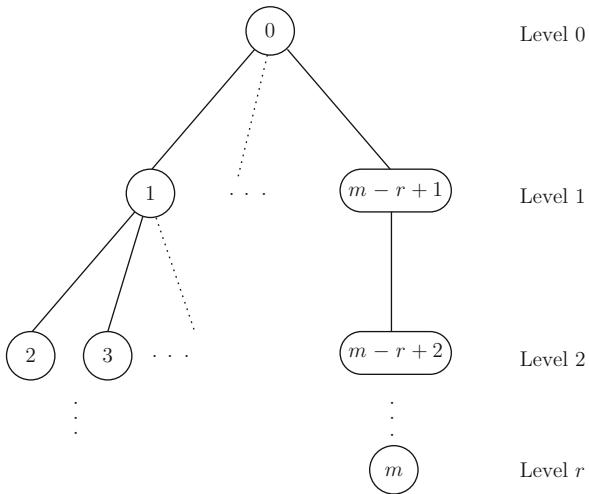
6 Feature Selection

Feature selection has been studied since the 1970s in statistical pattern recognition, machine learning, and data mining [94]. Feature selection is applied in a variety of fields such as text categorization, gene expression array analysis, combinatorial chemistry, and bioinformatics [66, 115] and is one of the key techniques in data preprocessing for data mining [20]. Feature selection is the process of removing irrelevant (features that do not affect the underlying target concept) and redundant features (features that do not add any information to the target concept) [46, 80]. Feature selection's role is more significant in real-world problems in which the data set is large (i.e., larger than the desired learning tool can handle). The idea with feature selection is to decrease running time of the learning process without decreasing the accuracy of the result significantly. It is needed because of generalization performance, running time requirements, constraints, and interpretational issues imposed by the problem itself. Further benefits of feature selection include facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, and defying the curse of dimensionality to improve prediction performance [66]. Problem of selecting the optimal subset of features is proven to be an \mathcal{NP} -complete problem [63].

Based on availability of class information, feature selection methods can be divided into three groups: supervised feature selection (feature selection for classification), unsupervised feature selection (feature selection for clustering), and semi-supervised feature selection. Feature selection methods can also be categorized based on role of the mining algorithm: the filter model, the wrapper model, and the hybrid model. *Filter models* start with an empty set of features and add features until a combination consistent with the training data is found. Mining method is applied next based on the selected features. On the other hand, *wrapper models* need a predefined mining method to use as evaluation criterion. Generally, wrapper methods are more accurate than filter methods due to the interaction between a mining algorithm and its training data [84, 90]. Wrapper methods are also computationally more expensive as they run the mining algorithm repeatedly during the evaluation process.

Feature selection in classification is defined as methods that select minimal sized subset of features based on two criteria. First, classification accuracy should

Fig. 3 Illustration of a branch and bound algorithm for feature selection



not notably decrease. Second, the resulting class distribution based on selected features should be as close as possible to the original class distribution based on all features. There are a number of studies in the literature (e.g., [51, 126]) that explore stages of feature selection methods. Dash and Liu [46] breaks down a typical feature selection method into four basic steps: generation, evaluation, stopping criterion, and validation. Generation procedure is a search approach that generates subsets for evaluation. When number of features is N , order of search space is $O(2^N)$. The search approach can be complete, heuristic, or random. Using complete methods guarantees optimality of the feature subset because they do backtracking. Backtracking is done by using techniques such as branch and bound, best first search, and beam search. Stopping criterion can be based on generation (predefined number of features/iterations) or evaluation procedure (optimal subset is obtained or addition/deletion of any feature does not result in a better subset).

Basic branch and bound algorithm omits r out of m features by producing r levels as shown in Fig. 3. At each level, one of the features is discarded. The level number indicates the number of features that have been omitted at that level. The root of the tree (at level 0) refers to the set of all m features. The leaves at the bottom of the tree correspond to all possible subsets of size $m - r$ features. The paths in this tree denote all combinations of r -eliminated features. Numbers in the nodes correspond to omitted features. The problem is selecting the best path through the tree that yields the best criterion function result. Ideally, this path should be achieved with the fewest number of calculations.

Branch and bound is first used for solving feature selection problem by Narendra and Fukunaga [105]. This algorithm assumes *monotonicity* of criterion function to find optimal feature subset without evaluating all possible feature subsets. Monotonicity property states that a subset of features is not better than any larger set

that contains the subset. A criterion function that satisfies the monotonicity property helps to cut off some subtrees and decreases the search area in the branch and bound algorithm. Generally, all branch and bound algorithms used for feature selection fall into two groups based on monotonicity assumption. Hamamoto et al. [68] show that criterion function is not monotonic, branch and bound still can result in a good recognition rate. A more efficient branch and bound algorithm with same assumption is introduced by Yu and Yuan [147]. This method can handle high-dimensional data by dynamically searching for the optimal solution on a minimum solution tree which is a subtree of the traditional solution tree. Another high-dimensional branch and bound algorithm (HDBB) that assumes monotonicity is introduced in [32]. To improve the speed of branch and bound, right-left search strategy is employed in addition to top-down strategy and backtracking in [39]. Frank et al. [55] propose a branch and bound algorithm using Bhattacharyya distance which is monotonic and additive. This method finds a feature subset of a given size with the lowest Bayesian classification error. A branch and bound method that can give a large initial bound using a floating search method is introduced in [104]. This model can order the tree nodes by the significance of features and has a jump search strategy to avoid redundant criterion function calculations. There are a number of methods that utilize monotonicity assumption (see [85, 128, 140, 146]). Recently, Ris et al. [114] introduce a branch and bound algorithm, which bases the representation and exploration of the search space on new lattice properties and the criterion function is not monotonic.

Another widely used approach in feature selection is the use of SVMs. Combined with filter methods, SVMs can be used after features are selected. SVMs can also be used in wrapper strategies, where one-norm SVM is used for automatic feature selection [23, 129, 152]. It is shown that the one-norm SVM has advantages over the two-norm SVM when there are features that present noise [152]. Having more spare solutions implies w has more zero components; thus, less features could be chosen. An extension of one-norm SVM is used in [154]. This method generates groups among features by clustering and uses F_∞ -norm SVM. Shi et al. [124] introduce a feature selection strategy using l_p -norm support vector classification (l_p -SVC) and l_p -norm proximal support vector machine (l_p -PSVM) where $0 < p < 1$. Guyon et al. [67] have proposed a recursive feature elimination (RFE) method. Adaptive scaling methods can be used for feature selection in SVMs [65]. For scaling, a linear transformation is done within the input space.

7 Conclusion

This review presents the numerous applications in data mining, which have benefited from the theory of combinatorial optimization. Combinatorial optimization improves quality and robustness of numerous applications and will certainly continue to support the constantly growing field of data mining.

Cross-References

► Combinatorial Optimization Techniques for Network-Based Data Mining

Recommended Reading

1. J. Abello, M.G.C. Resende, S. Sudarsky, Massive quasi-clique detection, in *LATIN 2002: Theoretical Informatics* (Springer, Berlin/New York, 2002), pp. 598–612
2. S. Alexe, E. Blackstone, P. Hammer, H. Ishwaran, M. Lauer, C. Snader, Coronary risk prediction by logical analysis of data. *Ann. Oper. Res.* **119**, 15–42 (2003)
3. D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean sum-of-squares clustering. *Mach. Learn.* **75**, 245–248 (2009)
4. D. Arthur, S. Vassilvitskii, How slow is the k-means method? in *Proceedings of the 22nd Annual Symposium on Computational Geometry* (ACM, New York, 2006), pp. 144–153
5. B. Balasundaram, S. Butenko, I.V. Hicks, Clique relaxations in social network analysis: the maximum k-plex problem. *Oper. Res.* **59**, 133–142 (2011)
6. G.H. Ball, D.J. Hall, ISODATA, a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, CA, 1965
7. A. Banerjee, S. Merugu, I.S. Dhillon, J. Ghosh, Clustering with Bregman divergences. *J. Mach. Learn. Res.* **6**, 1705–1749 (2005)
8. A. Baraldi, P. Blonda, A survey of fuzzy clustering algorithms for pattern recognition – part II. *IEEE Trans. Syst. Man Cybern. B* **29**(6), 786–801 (1999)
9. M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs. *Learn. Theory* **3120**, 624–638 (2004)
10. A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, Z. Yakhini, Tissue classification with gene expression profiles, in *Proceedings of the 4th Annual International Conference on Computational Biology (RECOMB)*, Tokyo, 2000, pp. 54–64
11. A. Ben-Dor, N. Friedman, Z. Yakhini, Class discovery in gene expression data, in *Proceedings of the 5th Annual International Conference on Computational Biology (RECOMB)*, New York, NY, USA (ACM, 2001), pp. 31–38
12. A. Ben-Dor, B. Chor, R. Karp, Z. Yakhini, Discovering local structure in gene expression data: the order-preserving submatrix problem. *J. Comput. Biol.* **10**(3–4), 373–384 (2003)
13. Y. Bengio, O. Delalleau, N. Le Roux, Label propagation and quadratic criterion, in *Semi Supervised Learning* (MIT, Cambridge, 2006)
14. K.P. Bennett, A. Demiriz, Semi-supervised support vector machines. *Adv. Neural Inf. Process. Syst.* **11**, 368–374 (1999)
15. C. Bergeron, F. Cheriet, J. Ronsky, R. Zernicke, H. Labelle, Prediction of anterior scoliotic spinal curve from trunk surface using support vector regression. *Eng. Appl. Artif. Intell.* **18**(8), 973–983 (2005)
16. D. Bertsimas, R. Shioda, Classification and regression via integer optimization. *Oper. Res.* **55**(2), 252–271 (2007)
17. J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms* (Kluwer Academic, Norwell, 1981)
18. T.D. Bie, N. Cristianini, Semi-supervised learning using semi-definite programming, in *Semi-Supervised Learning* (MIT, Cambridge, 2006), pp. 119–135
19. C.M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)* (Springer, New York, 2006)
20. A.L. Blum, P. Langley, Selection of relevant features and examples in machine learning. *Artif. Intell.* **97**(1–2), 245–271 (1997)
21. A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in *Proceedings of the 11th Annual Conference on Computational Learning Theory* (ACM, New York, 1998), pp. 92–100

22. V. Boginski, Network-based data mining: operations research techniques and applications, in *Encyclopedia of Operations Research and Management Science* (Wiley, Hoboken, 2010) pp. 3498–3508
23. P.S. Bradley, O.L. Mangasarian, Feature selection via concave minimization and support vector machines, in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML)*, Madison, 1998, pp. 82–90
24. P.S. Bradley, U.M. Fayyad, O.L. Mangasarian, Mathematical programming for data mining: formulations and challenges. *INFORMS J. Comput.* **11**, 217–238 (1999)
25. J.P. Brooks, Support vector machines with the ramp loss and the hard margin loss. *Oper. Res.* **59**(2), 467–479 (2011)
26. M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugne, T. Furey, M. Ares, D. Haussler, Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci.* **97**(1), 262–267 (2000)
27. K. Bryan, Biclustering of expression data using simulated annealing, in *Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems (CBMS)* Washington, DC, USA, 2005, pp. 383–388
28. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender, Learning to rank using gradient descent, in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, 2005, pp. 89–96
29. S. Busygin, O.A. Prokopyev, P.M. Pardalos, Feature selection for consistent biclustering. *J. Comb. Optim.* **10**, 7–21 (2005)
30. S. Busygin, N. Boyko, P.M. Pardalos, M. Bewernitz, G. Ghacibeh, Biclustering EEG data from epileptic patients treated with vagus nerve stimulation, in *Data Mining, Systems Analysis and Optimization in Biomedicine*, vol. 953, ed. by O. Seref, O.E. Kundakcioglu, P.M. Pardalos (American Institute of Physics, Melville, 2007), pp. 220–231
31. S. Busygin, O. Prokopyev, P.M. Pardalos, Biclustering in data mining. *Comput. Oper. Res.* **35**(9), 2964–2987 (2008)
32. D. Casasent, X.W. Chen, Waveband selection for hyperspectral data: optimal feature selection, in *Proceedings of SPIE*, vol. 5106, Orlando, FL, 2003, pp. 259–270
33. W. Chaovallitwongse, Novel quadratic programming approach for time series clustering with biomedical application. *J. Comb. Optim.* **15**, 225–241 (2008)
34. O. Chapelle, Training a support vector machine in the primal. *Neural Comput.* **19**, 1155–1178 (2007)
35. O. Chapelle, A. Zien, Semi-supervised classification by low density separation, in *Proceeding of International Conference on Artificial Intelligence and Statistics (AISTAT)*, Barbados, 2005, pp. 57–64
36. O. Chapelle, M. Chi, A. Zien, A continuation method for semi-supervised SVMs, in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, New York, NY, USA (ACM, 2006), pp. 185–192
37. O. Chapelle, V. Sindhwani, S.S. Keerthi, Branch and bound for semi-supervised support vector machines. *Adv. Neural Inform. Process. Syst.* **19**, 217–224 (2007)
38. O. Chapelle, V. Sindhwani, S.S. Keerthi, Optimization techniques for semi-supervised support vector machines. *J. Mach. Learn. Res.* **9**, 203–233 (2008)
39. X. Chen, An improved branch and bound algorithm for feature selection. *Pattern Recognit. Lett.* **24**(12), 1925–1933 (2003)
40. Y. Cheng, G.M. Church, Biclustering of expression data, in *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology* (AAAI, Menlo Park, 2000) pp. 93–103
41. H. Cheng, Z. Liu, J. Yang, Sparsity induced similarity measure for label propagation, in *Proceedings of 12nd IEEE International Conference on Computer Vision*, Kyoto, Japan, 2010, pp. 317–324
42. K.Y. Choy, C.W. Chan, Modeling of river discharges and rainfall using radial basis function networks based on support vector regression. *Int. J. Syst. Sci.* **34**(14–15), 763–773 (2003)

43. C. Cifarelli, G. Patrizi, Solving large protein folding problem by a linear complementarity algorithm with 0–1 variables. *Optim. Methods Softw.* **22**(1), 25–49 (2007)
44. R. Collobert, F. Sinz, J. Weston, L. Bottou, T. Joachims, Large scale transductive SVMs. *J. Mach. Learn. Res.* **7**, 2006 (2006)
45. N. Cristianini, J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods* (Cambridge University Press, Cambridge, 2000)
46. M. Dash, H. Liu, Feature selection for classification. *Intell. Data Anal.* **1**(3), 131–156 (1997)
47. O. Delalleau, Y. Bengio, N. Le Roux, Efficient non-parametric function induction in semi-supervised learning, in *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics (AISTAT 2005)*, Barbados, 2005
48. A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. B* **39**(1), 1–38 (1977)
49. I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA (ACM, 2001), pp. 269–274
50. I.S. Dhillon, S. Mallela, D.S. Modha, Information-theoretic co-clustering, in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, New York, NY, USA (ACM, 2003), pp. 89–98
51. J. Doak, An evaluation of feature selection methods and their application to computer security. Technical report, University of California, 1992
52. C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in *Proceedings of the 10th International Conference on World Wide Web*, New York, NY, USA (ACM, 2001), pp. 613–622
53. S. Eschrich, J. Ke, L.O. Hall, D.B. Goldgof, Fast accurate fuzzy clustering through data reduction. *IEEE Trans. Fuzzy Syst.* **11**(2), 262–270 (2003)
54. E. Forgy, Cluster analysis of multivariate data: efficiency vs. interpretability of classifications. *Biometrics* **21**(3), 768 (1965)
55. A. Frank, D. Geiger, Z. Yakhini, A distance-based branch and bound feature selection algorithm, in *Proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, Acapulco, 2003, pp. 241–248
56. Y. Freund, R. Iyer, R.E. Schapire, Y. Singer, An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.* **4**, 933–969 (2003)
57. B.J. Frey, D. Dueck, Clustering by passing messages between data points. *Sci.* **315**(5814), 972–976 (2007)
58. H.P. Friedman, J. Rubin, On some invariant criteria for grouping data. *J. Am. Stat. Assoc.* **62**(320), 1159–1178 (1967)
59. G. Fung, O.L. Mangasarian, Semi-supervised support vector machines for unlabeled data classification. *Optim. Methods Softw.* **15**, 29–44 (2001)
60. G.N. Garcia, T. Ebrahimi, J.M. Vesin, Joint time-frequency-space classification of EEG in a brain-computer interface application. *J. Appl. Signal Process* **7**, 713–729 (2003)
61. M.R. Garey, D.S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness* (W. H. Freeman, New York, 1979)
62. Z. Ghahramani, Unsupervised learning, in *Advanced Lectures on Machine Learning* (Springer, Berlin/New York, 2003), pp. 72–112
63. I.A. Gheyas, L.S. Smith, Feature subset selection in large dimensionality domains. *Pattern Recognit.* **43**(1), 5–13 (2010)
64. T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, E.S. Lander, Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286**(5439), 531–537 (1999)
65. Y. Grandvalet, S. Canu, Adaptive scaling for feature selection in SVMs, in *NIPS*, Vancouver, 2002, pp. 553–560
66. I. Guyon, A. Elisseeff, An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)

67. I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines. *Mach. Learn.* **46**, 389–422 (2002)
68. Y. Hamamoto, S. Uchimura, Y. Matsuura, T. Kanaoka, S. Tomita, Evaluation of the branch and bound algorithm for feature selection. *Pattern Recognit. Lett.* **11**(7), 453–456 (1990)
69. J.A. Hartigan, Direct clustering of a data matrix. *J. Am. Stat. Assoc.* **67**(337), 123–129 (1972)
70. W.C. Hong, P.F. Pai, Potential assessment of the support vector regression technique in rainfall forecasting. *Water Res. Manage.* **21**(2), 495–513 (2007)
71. C.W. Hsu, C.C. Chang, C.J. Lin, A practical guide to support vector classification (2004), <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>
72. Z. Huang, H. Chen, C.J. Hsu, W.H. Chenb, S. Wuc, Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decis. Support Syst.* **37**, 543–558 (2004)
73. K. Hyunsoo, Z.X. Jeff, M.C. Herbert, P. Haesun, A three-stage framework for gene expression data analysis by L1-norm support vector regression. *Int. J. Bioinformatics Res. Appl.* **1**(1), 51–62 (2005)
74. A.K. Jain, Data clustering: 50 years beyond k-means. *Pattern Recognit. Lett.* **31**(8), 651–666 (2010)
75. A.K. Jain, R.C. Dubes, *Algorithms for Clustering Data* (Prentice-Hall, Upper Saddle River, 1988)
76. X. Jiang, L.H. Lim, Y. Yao, Y. Ye, Statistical ranking and combinatorial hodge theory. *Mathematical Programming* **127**, 1–42 (2010)
77. T. Joachims, Text categorization with support vector machines: learning with many relevant features, in *Proceedings of the European Conference on Machine Learning*, Berlin, ed. by C. Nédellec, C. Rouveiro (Springer, 1998), pp. 137–142
78. T. Joachims, Making large-scale SVM learning practical, in *Advances in Kernel Methods – Support Vector Learning*, Cambridge, MA, ed. by B. Schölkopf, C.J.C. Burges, A.J. Smola (MIT, 1999), pp. 169–184
79. T. Joachims, Transductive learning via spectral graph partitioning, in *Proceedings of 20th International Conference on Machine Learning (ICML)*, Washington, DC, USA, vol. 20, 2003, pp. 290–297
80. G.H. John, R. Kohavi, K. Pfleger, Irrelevant features and the subset selection problem, in *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, vol. 129, 1994, pp. 121–129
81. H. Kashima, J. Hu, B. Ray, M. Singh, K-means clustering of proportional data using L1 distance, in *Proceedings of 19th International Conference on Pattern Recognition (ICPR)*, Tampa, FL, 2009, pp. 1–4
82. F. Klawonn, A. Keller, Fuzzy clustering based on modified distance measures, in *IDA '99 Proceedings of the Third International Symposium on Advances in Intelligent Data Analysis* (Springer, Berlin, 1999), pp. 291–302
83. Y. Kluger, R. Basri, J.T. Chang, M. Gerstein, Spectral biclustering of microarray data: coclustering genes and conditions. *Genome Res.* **13**(4), 703–716 (2003)
84. R. Kohavi, G.H. John, Wrappers for feature subset selection. *Artif. Intell.* **97**(1–2), 273–324 (1997)
85. M. Kudo, J. Sklansky, Comparison of algorithms that select features for pattern classifiers. *Pattern Recognit.* **33**(1), 25–41 (2000)
86. O.E. Kundakcioglu, P.M. Pardalos, The complexity of feature selection for consistent biclustering, in *Clustering Challenges in Biological Networks* (World Scientific, Hackensack, 2009), pp. 257–266
87. O.E. Kundakcioglu, T. Ünlüyurt, Bottom-up construction of minimum-cost AND/OR trees for sequential fault diagnosis. *IEEE Trans. Syst. Man Cybern. A* **37**(5), 621–629 (2007)
88. O.E. Kundakcioglu, O. Seref, P.M. Pardalos, Multiple instance learning via margin maximization. *Appl. Numer. Math.* **60**(4), 358–369 (2010)

89. T.N. Lal, M. Schroeder, T. Hinterberger, J. Weston, M. Bogdan, N. Birbaumer, B. Schölkopf, Support vector channel selection in BCI. *IEEE Trans. Biomed. Eng.* **51**(6), 1003–1010 (2004)
90. P. Langley, Selection of relevant features in machine learning, in *Proceedings of the AAAI Fall Symposium on Relevance* (AAAI, 1994), New Orleans, LA, pp. 140–144
91. F. Lauer, G. Bloch, Incorporating prior knowledge in support vector regression. *Mach. Learn.* **70**, 89–118 (2008)
92. S. Lee, A. Verri (eds.), *Pattern Recognition with Support Vector Machines*, Niagara Falls, Canada (Springer, New York/Berlin, 2002)
93. Y. Linde, A. Buzo, R. Gray, An algorithm for vector quantizer design. *IEEE Trans. Commun.* **28**(1), 84–95 (1980)
94. H. Liu, L. Yu, Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.* **17**(4), 491–502 (2005)
95. S. Lloyd, Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**, 129–137 (1982). Original paper was published as a technical note in 1957, Bell Labs
96. J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in *Fifth Symposium on Math, Statistics and Probability* (University of California Press, Berkeley, 1967), pp. 281–297
97. S. Madeira, A. Oliveira, Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* **1**, 24–45 (2004)
98. P.K. Mallapragada, R. Jin, A.K. Jain, Y. Liu, SemiBoost: boosting for semi-supervised Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(11), 2000–2014 (2009)
99. J. Mao, A.K. Jain, A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Trans. Neural Netw.* **7**(1), 16–29 (2002)
100. G.J. McLachlan, T. Krishnan, *The EM algorithm and extensions* Wiley-Interscience, Hoboken, New Jersey (LibreDigital, 2008)
101. Merriam-Webster, Dictionary and Thesaurus – Merriam-Webster Online (2011), http://www.merriam-webster.com/dictionary/data_mining
102. B.G. Mirkin, *Mathematical Classification and Clustering*, Kluwer Academic Publishers, Dordrecht, Netherland, (Springer, 1996)
103. A. Nahapetyan, S. Busygina, P.M. Pardalos, An improved heuristic for consistent biclustering problems, in *Mathematical Modelling of Biosystems* (Springer, Berlin, 2008), pp. 185–198
104. S. Nakariyakul, D.P. Casasent, Adaptive branch and bound algorithm for selecting optimal features. *Pattern Recognit. Lett.* **28**(12), 1415–1427 (2007)
105. P.M. Narendra, K. Fukunaga, A branch and bound algorithm for feature subset selection. *IEEE Transact. Comput.* **100**(9), 917–922 (1977)
106. W.S. Noble, Support vector machine applications in computational biology, in *Kernel Methods in Computational Biology* (MIT, Cambridge MA, 2004), New York, NY, pp. 71–92
107. R.F.E. Osuna, F. Girosi, An improved training algorithm for support vector machines, in *IEEE Workshop on Neural Networks for Signal Processing*, New York, NY, 1997, pp. 276–285
108. P.F. Pai, W.C. Hong, A recurrent support vector regression model in rainfall forecasting. *Hydrol. Process.* **21**(6), 819–827 (2007)
109. P.M. Pardalos, E. Romeijn (eds.), *Handbook of Optimization in Medicine* (Springer, Newyork/London, 2009)
110. J. Platt, Fast training of SVMs using sequential minimal optimization, in *Advances in Kernel Methods: Support Vector Learning* (MIT, Cambridge MA, 1999), pp. 185–208
111. M.H. Poursaeidi and O.E. Kundakcioglu, Robust support vector machines for multiple instanceclassification, *Annals of Operations Research*, published online. doi:10.1007/s10479-012-1241-z
M.H. Poursaeidi, O.E. Kundakcioglu, Robust support vector machines for multiple instance classification (2011, under revision)

112. G. Pyrgiotakis, O.E. Kundakcioglu, K. Finton, P.M. Pardalos, K. Powers, B.M. Moudgil, Cell death discrimination with Raman spectroscopy and support vector machines. *Ann. Biomed. Eng.* **37**(7), 1464–1473 (2009)
113. G. Pyrgiotakis, O.E. Kundakcioglu, P.M. Pardalos, B.M. Moudgil, Raman spectroscopy and support vector machines for quick toxicological evaluation of titania nanoparticles. *J. Raman Spectrosc.* (2011, accepted). doi:10.1002/jrs.2839
114. M. Ris, J. Barrera, D.C. Martins Jr., U-curve: a branch-and-bound optimization algorithm for u-shaped cost functions on boolean lattices applied to the feature selection problem. *Pattern Recognit.* **43**(3), 557–568 (2010)
115. Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics. *Bioinformatics* **23**(19), 2507 (2007)
116. N.A. Sakhnenko, G.F. Luger, Shock physics data reconstruction using support vector regression. *Int. J. Mod. Phys.* **17**(9), 1313–1325 (2006)
117. B. Schölkopf, A.J. Smola, *Learning with Kernels* (MIT, Cambridge MA, 2002)
118. O. Seref, O.E. Kundakcioglu, P.M. Pardalos, Selective linear and nonlinear classification, in *CRM Proceedings and Lecture Notes*, vol. 45, ed. by P.M. Pardalos, P. Hansen (American Mathematical Society, Providence, 2008), pp. 211–234
119. O. Seref, O.E. Kundakcioglu, O.A. Prokopyev, P.M. Pardalos, Selective support vector machines. *J. Comb. Optim.* **17**(1), 3–20 (2009)
120. S. Shalev-Shwartz, Y. Singer, N. Srebro, A. Cotter, Pegasos: primal estimated sub-gradient solver for SVM. *Math. Program. B* **127**, 3–30 (2011)
121. J. Shawe-Taylor, N. Cristianini, *Kernel Methods for Pattern Analysis* (Cambridge University Press, Cambridge, 2004)
122. Q. Sheng, Y. Moreau, B. DeMoor, Bioclustering microarray data by Gibbs sampling. *Bioinformatics* **19**, 196–205 (2003)
123. H.D. Sherali, J. Desai, A global optimization RLT-based approach for solving the fuzzy clustering problem. *J. Glob. Optim.* **33**(4), 597–615 (2005)
124. Y. Shi, Y. Tian, G. Kou, Y. Peng, J. Li, *Optimization Based Data Mining: Theory and Applications* (Springer, New York, 2011)
125. O. Shirokikh, V. Stozhkov, V. Boginski, Combinatorial optimization techniques for network-based data mining, in *Handbook of Combinatorial Optimization*, 2nd Edition, (Springer, 2013)
126. W. Siedlecki, J. Sklansky, On automatic feature selection. *Intern. J. Pattern Recognit. Artif. Intell.* **2**(2), 197–220 (1988)
127. V. Sindhwani, S.S. Keerthi, Large scale semi-supervised linear SVMs, in *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (ACM, New York, 2006), pp. 477–484
128. P. Somol, P. Pudil, J. Kittler, Fast branch & bound algorithms for optimal feature selection. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(7), 900–912 (2004)
129. M. Song, C.M. Breneman, J. Bi, N. Sukumar, K.P. Bennett, S. Cramer, N. Tugcu, Prediction of protein retention times in anion-exchange chromatography systems using support vector regression. *J. Chem. Inf. Comput. Sci.* **42**(6), 1347–1357 (2002)
130. I. Steinwart, Support vector machines are universally consistent. *J. Complex.* **18**, 768–791 (2002)
131. Y.F. Sun, Y.C. Liang, C.G. Wu, X.W. Yang, H.P. Lee, W.Z. Lin, Estimate of error bounds in the improved support vector regression. *Prog. Nat. Sci.* **14**(4), 362–364 (2004)
132. M. Szummer, T. Jaakkola, Partially labeled classification with Markov random walks. *Adv. Neural Inf. Process. Syst.* **2**, 945–952 (2002)
133. J. Thorsten, Transductive inference for text classification using support vector machines, in *Proceedings of 16th International Conference on Machine Learning* (Morgan Kaufmann, San Francisco, 1999), pp. 200–209
134. T.B. Trafalis, H. Ince, Support vector machine for regression and applications to financial forecasting, in *Proceedings of International Joint Conference on Neural Networks (IJCNN)*, Como, 2002

135. A.C. Trapp, O.A. Prokopyev, Solving the order-preserving submatrix problem via integer programming. *INFORMS J. Comput.* **22**(3), 387–400 (2010)
136. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer, New York, 1995)
137. V. Vapnik, A. Chervonenkis, *Theory of Pattern Recognition* (Naula/Moscow, Russia, 1974)
138. V. Vapnik, A. Sterin, On structural risk minimization or overall risk in a problem of pattern recognition, in *Automation and Remote Control*, vol. 10, 1977, pp. 1495–1503
139. J. Wang, On transductive support vector machines, in *Prediction and Discovery* (American Mathematical Society, Providence, Snowbird, Utah, 2007)
140. Z. Wang, J. Yang, G. Li, An improved branch & bound algorithm in feature selection, in *Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, Chongqing, 2003, pp. 549–556
141. J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, Feature selection for SVMs, in *Proceeding of NIPS*, Denver, 2000, pp. 668–674
142. Z.L. Wu, C.H. Li, J.K.Y. Ng, K.R.P.H. Leung, Location estimation via support vector regression. *IEEE Trans. Mob. Comput.* **6**(3), 311–321 (2007)
143. X.S. Xie, W.T. Liu, B.Y. Tang, Space based estimation of moisture transport in marine atmosphere using support vector regression. *Remote Sens. Environ.* **112**(4), 1846–1855 (2008)
144. E.P. Xing, R.M. Karp, CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Bioinformatics Discov. Note* **17**, 306–315 (2001)
145. K. Yamamoto, F. Asano, T. Yamada, N. Kitawaki, Detection of overlapping speech in meetings using support vector machines and support vector regression. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **E89-A**(8), 2158–2165 (2006)
146. S. Yang, P. Shi, Bidirectional automated branch and bound algorithm for feature selection. *J. Shanghai Univ. (English Edition)* **9**(3), 244–248 (2005)
147. B. Yu, B. Yuan, A more efficient branch and bound algorithm for feature selection. *Pattern Recognit.* **26**(6), 883–889 (1993)
148. A.L. Yuille, A. Rangarajan, The concave-convex procedure. *Neural Comput.* **15**(4), 915–936 (2003)
149. X. Zhu, Semi-supervised learning with graphs. PhD thesis, Carnegie Mellon University, 2005, CMU-LTI-05-192
150. X. Zhu, Semi-supervised learning literature survey (2006), Available online at <http://pages.cs.wisc.edu/~jerryzhu>
151. X. Zhu, Z. Ghahramani, Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002
152. J. Zhu, S. Rosset, T. Hastie, R. Tibshirani, 1-norm support vector machines, in *Proceedings of Advances in Neural Information Processing Systems*, Vancouver, 2003
153. X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in *Proceedings of 21st International Conference on Machine Learning (ICML)*, Washington, DC, USA, vol. 20, 2003, p. 912
154. H. Zou, M. Yuan, The f_∞ -norm support vector machine. *Stat. Sin.* **18**, 379–398 (2008)

Combinatorial Optimization Techniques for Network-Based Data Mining

Oleg Shirokikh, Vladimir Stozhkov and Vladimir Boginski

Contents

1	Introduction	632
2	Similarity Measures Used in Graph-Based Data Mining.....	634
3	Basic Concepts from Graph Theory and Their Data Mining Interpretation.....	642
3.1	Connected Components and Degree Distributions.....	643
3.2	Cliques and Independent Sets.....	646
3.3	Clustering via Clique Partitioning.....	648
3.4	Using Clique Relaxations for Graph-Based Clustering.....	650
4	Examples of Real-World Applications.....	655
4.1	Biological Networks.....	655
4.2	Chemical Networks.....	659
4.3	Brain Networks.....	661
4.4	Telecommunication/Information Exchange Networks.....	662
4.5	Financial Networks.....	664
4.6	Social Networks.....	667
5	Conclusion.....	668
	Cross-References.....	668
	Recommended Reading	669

O. Shirokikh • V. Stozhkov

Industrial and Systems Engineering Department, University of Florida, Gainesville, FL, USA
e-mail: olegshirokikh@ufl.edu; vstozhkov@ufl.edu

V. Boginski

Industrial and Systems Engineering Department, University of Florida, Shalimar, FL, USA
e-mail: boginski@reef.ufl.edu

Abstract

The matters of discussion are combinatorial optimization aspects, concepts, and applications arising in the broad area of network-based data mining. The approach of representing real-world datasets as large-scale networks (graphs) has become increasingly popular during recent years. The purpose of this chapter is to briefly review the graph-theoretic and combinatorial optimization concepts that are important in the context of data mining, as well as to discuss the interpretation of these concepts from mathematical modeling perspective.

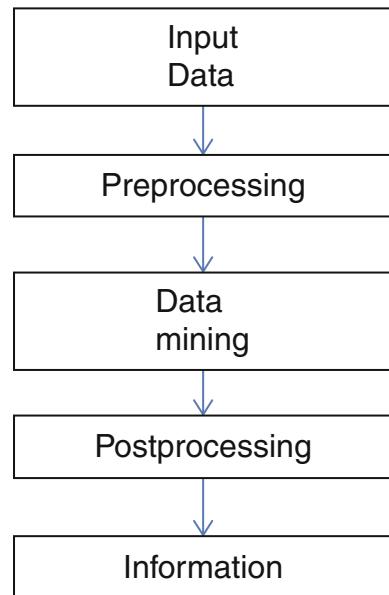
1 Introduction

The process of studying real-life complex systems often deals with large datasets arising in diverse applications including telecommunications, biotechnology, medicine, finance, physics, ecology, and geographical information systems [2, 21]. Understanding the structural properties of a certain dataset is in many cases the task of the crucial importance. To extract useful information from data, one often needs to apply advanced mathematical modeling techniques to summarize, visualize, and process the information contained in a dataset. *Data mining* problems deal with multiple aspects of information retrieval from datasets, and a variety of techniques has been developed to address different types of tasks arising in this area. Data mining is a part of *knowledge discovery in databases (KDD)* [81]. This is the process of converting data into useful information, which includes several main steps summarized in Fig. 1. The purpose of preprocessing is to transform input data to a proper format for further analysis. An example of postprocessing is visualization, which allows to research and explore data and data mining results from different viewpoints. One can also apply statistical measures and hypothesis testing techniques at the postprocessing step to potentially eliminate inconsistent data mining results. However, data mining itself is a crucial step in the KDD process. The purpose of this chapter is to review possible methods of utilizing combinatorial optimization and network-based tools to extract information from real-world datasets.

During the past several years, *optimization-based approaches* have received increased attention in the context of data mining problems [27]. Moreover, along with classical data mining techniques (e.g., support vector machines, artificial neural networks, decision trees, K-means/K-median clustering, and fuzzy C-means clustering), many recent studies deal with *network (graph)-based representations* of large real-world datasets and utilize graph-theoretic and network optimization concepts to extract useful and nontrivial information from these datasets. In this approach, a dataset is represented as a *graph (network)* with certain attributes associated with its vertices and edges. As it will be discussed later, combinatorial optimization problems often arise in this analysis, and solving these problems is an integral part of network-based data mining.

Studying the structure of a graph representing a dataset is often important for understanding the internal properties of the application it represents, as well as for

Fig. 1 Knowledge discovery in databases



improving storage organization and information retrieval. One can visualize a graph as a set of dots and links connecting them, which often makes this representation convenient and easily understandable.

The main concepts of graph theory were founded several centuries ago, and many network optimization algorithms have been developed since then. However, the trend of applying graph models to represent various real-life massive datasets has started relatively recently. Graph theory and related operations research techniques are quickly becoming a field with a strong practical impact. The expansion of graph-theoretical approaches in various applications gave birth to the terms “graph practice” and “graph engineering” [47]. Graph (network)-based data mining techniques have gained significant popularity in the recent years. Various aspects of this emerging field are addressed in [32, 84]. The term “network science” referring to a broad variety of network-based modeling approaches has also been widely used recently. In [10] potential relevance between “network science” and operations research is discussed. Although potential applications of operations research in this broad field can be diverse, network-based data mining is among the areas where optimization-based techniques can provide valuable contributions.

Network-based models allow one to extract information from real-world datasets using various standard concepts from graph theory. In many cases, one can investigate specific properties of a dataset by detecting special formations in the corresponding graph, for instance, *connected components*, *spanning trees*, *cliques*, and *independent sets*. In particular, connected components, cliques, and independent sets can be used to address the important *clustering* problems arising in many data mining applications. This problem essentially represents partitioning the set of elements of a certain dataset into a number of subsets (clusters) of objects according

to some *similarity* (or *dissimilarity*) criterion. These concepts are associated with a number of network optimization problems that will be discussed below.

Another aspect of investigating network models of real-world datasets is studying the global organization and structural properties of these networks, which are reflected by *degree distributions* of the constructed graphs. The degree distribution represents the large-scale pattern of connections in the graph, which reflects the global properties of the dataset. One of the important results discovered during the last several years is the observation that many graphs representing the datasets from diverse areas (Internet, telecommunications, biology, sociology) obey the *power-law* model [5]. The fact that graphs representing completely different datasets have a similar well-defined power-law structure has been widely reflected in the literature [4, 8, 16, 17, 21, 31, 47]. It indicates that global organization and evolution of datasets arising in various spheres of life follow similar laws and patterns. This fact served as a motivation to introduce a concept of “self-organized networks.” The information about the degree distribution of a graph that represents a certain dataset can be significant in the data mining context, since it may allow one to analytically predict the size of certain types of clusters in this graph (e.g., connected components, cliques, and other structures). In particular, the size of connected components (clusters) in very large power-law graphs has been studied in [5].

Further in this chapter, the main issues and concepts arising in network-based data mining will be addressed: specifically, the process of representing a dataset as a graph using appropriate similarity (proximity) measures, which serves as a basis for applying graph-theoretic and network optimization techniques. These concepts and techniques are discussed later in this chapter. Finally, there will be given several examples of popular recent application areas of network-based data mining.

2 Similarity Measures Used in Graph-Based Data Mining

As indicated above, the essence of network-based data mining is representing a dataset under consideration as a network (graph) where vertices and edges are constructed according to certain principles that may depend on the structure and the origin of this dataset. The vertices of the graph usually represent data records (objects), whereas the edges that connect the vertices are constructed using an appropriate *similarity criterion*. That is, two data objects (vertices) are connected if they are “similar enough” to each other. However, there are a lot of different ways to quantify the degree of similarity. In this section, possible types of *similarity measures* (also referred to as *proximity measures*) will be discussed. More precisely, those are proximity measures between data objects that can be used for constructing the corresponding graph representations of datasets.

Usually, the *similarity* between two objects is a numerical measure of the degree to which the two objects are alike. It can often be scaled between 0 (no similarity) and 1 (complete similarity). The *dissimilarity* between two objects is a numerical measure of the degree to which the two objects are different. Also, the term *distance* is used to measure the degree of similarity/dissimilarity. Dissimilarities sometimes occupy the interval $[0, 1]$, but it is usual for them to belong to the range $[0, \infty)$.

Transformations are used to convert similarity to dissimilarity and vice versa. First of all, proximity measures should be converted to the interval $[0, 1]$. For instance, if the grade has a range from 1 to 5, it makes sense to use the formula $\hat{s} = (s - 1)/4$ to replace the initial proximity measure s by standard \hat{s} that belongs to the interval $[0, 1]$. Generally, the transformations of similarities to the interval $[0, 1]$ are given by the formula

$$\hat{s} = \frac{s - \min s}{\max s - \min s}.$$

Analogously, dissimilarity measures with a finite range can be mapped to the interval $[0, 1]$ by transformation

$$\hat{d} = \frac{d - \min d}{\max d - \min d}.$$

In other cases, when values of the original proximity measure fill the interval $[0, \infty)$, a nonlinear conversion is necessary. As an example, the formula $\hat{d} = \frac{d}{1+d}$ can be utilized. The dissimilarities $0, \frac{1}{3}, \frac{2}{3}, 1, 5, 10, 100, 1,000$ will be converted to the new dissimilarities $0, 0.25, 0.4, 0.5, 0.833, 0.909, 0.99, 0.999$. Transforming dissimilarities into similarities and vice versa is rather straightforward too. If the similarity falls in the interval $[0, 1]$, then the dissimilarity can be defined by the formula $d = 1 - s$. For $d \in [0, \infty)$, various nonlinear transformations can be employed. Consistent examples are $s = 1/(d + 1)$ and $s = e^{-d}$.

In the standard setup of many data mining problems, one deals with a dataset of N data records that can be represented by vectors $x^i = (x_1^i, x_2^i, \dots, x_n^i)$, $i = 1, \dots, N$, where every vector x^i has n components that are referred to as the *features*, or *attributes*, of the data record. The similarity/dissimilarity measure between any two data records would be defined by the corresponding values of their attributes. The following similarity/dissimilarity measures are commonly used in practice.

- *Distance metrics:* Generally it can be expressed in the form of Minkowski distance metric

$$d(x, y) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{1/r},$$

where r is a parameter that defines different types of metrics. The most commonly used distance metrics are L_1 norm for $r = 1$ (*Manhattan distance*, *city block distance*), L_2 norm for $r = 2$ (*Euclidean distance*), and L_∞ norm for $r = \infty$ (*supremum distance*) that signifies

$$d(x, y)_{L_\infty} = \max_{i=1, \dots, n} |x_k - y_k|.$$

A common particular instance of Manhattan distance is the *Hamming distance*, which is the number of bits that are different between two objects that have only binary attributes (i.e., between two binary vectors). In Fig. 2, the difference

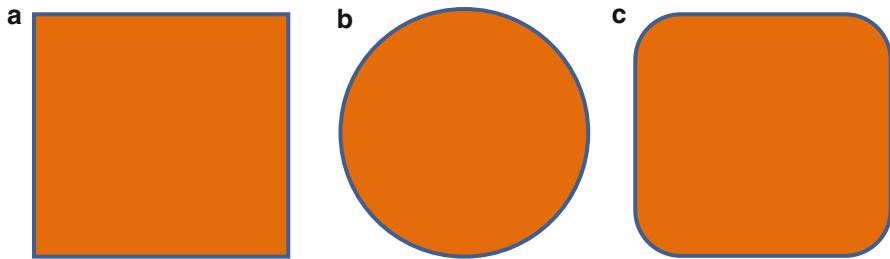


Fig. 2 Circumferences in different L_p norms. (a) Standard circle in L_1 norm. (b) Standard circle in L_2 norm. (c) Standard circle in L_4 norm

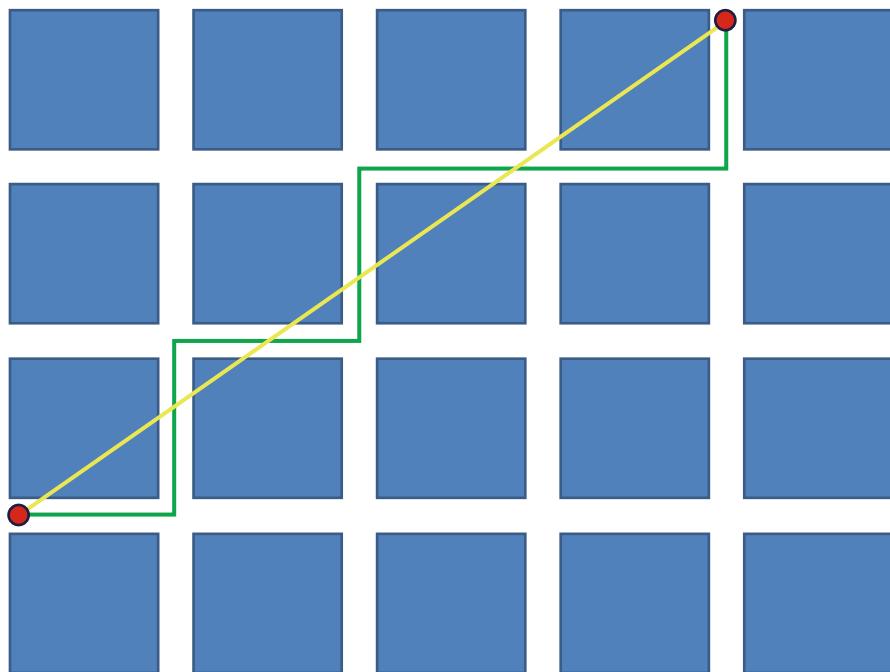


Fig. 3 Comparison of city block distance and Euclidean distance

between L_1 , L_2 , and L_4 can be observed graphically on the example of the standard circle presented in all these metrics. The norms L_1 and L_2 are compared visually in Fig. 3.

Supremum (L_{\max} or L_{∞} norm) distance is the maximum difference between any attribute of two objects. More rigorously, L_{∞} is defined by the limit

$$d(x, y) = \lim_{r \rightarrow \infty} \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{1/r}.$$

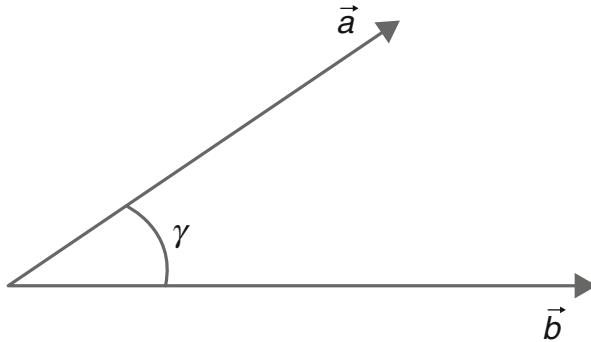


Fig. 4 Geometric representation of cosine similarity

To add more flexibility in the situations when the values of some attributes have substantially different importance levels or orders of magnitude, the formulas of proximity can be modified by weighting the contribution of each attribute. In the case of the distance metrics, the Minkowski distance can be modified as follows:

$$d(x, y) = \left(\sum_{k=1}^n \omega_k |x_k - y_k|^r \right)^{1/r},$$

where ω_k is the relative weight of the k th component in the considered vector space.

- *Cosine similarity:* This measure is often used to characterize text document similarity (e.g., documents represented as vectors with attributes denoting the frequency of each word in the considered document) and can be expressed as

$$\cos(a, b) = \frac{a \cdot b}{\|a\| \|b\|},$$

where $a \cdot b$ is a scalar product of vectors a and b . Even though documents have thousands of attributes, each document is sparse because there is a few nonzero attributes. Thus, similarity should not depend on the number of shared zero values. Therefore, a proximity measure for documents must ignore 0–0 matches like *Jaccard measure*. The cosine similarity is one of the useful measures in this case (Fig. 4).

- *Correlation:* The standard correlation measure is commonly used to characterize the similarity between different types of data, for instance, time series data, where the attributes represent the values of a certain parameter for different time moments. A well-known example that the correlation measure can be effectively used is the analysis of stock market data, where high correlation would mean high degree of similarity between a given pair of stocks.

The *general correlation coefficient* is given by the formula

$$\bar{r} = \frac{\sum_{i < j} c_{ij}(x)c_{ij}(y)}{\sqrt{\sum_{i < j} c_{ij}^2(x) \cdot \sum_{i < j} c_{ij}^2(y)}},$$

where x and y are two samples of comparable data and c_{ij} is some function measuring relative distance between x_i and x_j (i th and j th coordinates of the sample vector x). Three particular examples of the general correlation coefficient are given below.

Pearson's correlation coefficient is the most widely used correlation measure. It is defined by the formula

$$\text{corr}(x, y) = \frac{\text{covariance}(x, y)}{\text{standard deviation}(x) * \text{standard deviation}(y)},$$

where from statistics theory it is known that

$$\begin{aligned} \text{covariance}(x, y) &= \sum_{k=1}^n (x_k - \bar{x})(y_k - \bar{y}), \\ \text{standard deviation}(x) &= \sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2}, \\ \text{standard deviation}(y) &= \sqrt{\frac{1}{n-1} \sum_{k=1}^n (y_k - \bar{y})^2}. \end{aligned}$$

Using standard notations, \bar{x} and \bar{y} are means of two samples x and y , respectively. Notice that Pearson's correlation coefficient could be easily derived from the general correlation coefficient if it is assigned that $c_{ij} = x_j - x_i$. Also, Spearman's rank correlation coefficient [79] and Kendall's tau (τ) rank correlation coefficient [55] are frequently utilized for peculiar statistical tests and hypotheses.

If it is assigned that $c_{ij} = R_j - R_i$ where R_k is the rank of x_k in the given sample vector $x = (x_1, \dots, x_n)$ for each $k = 1, \dots, n$, then the formula below for *Spearman's rank correlation coefficient* can be derived:

$$\rho_S = \frac{\sum_{i=1}^n (R_i - \bar{R})(S_i - \bar{S})}{\sqrt{\sum_{i=1}^n (R_i - \bar{R})^2 \sum_{i=1}^n (S_i - \bar{S})^2}},$$

where R_k and S_k are the ranks of x_k and y_k in the samples x and y , correspondingly. Also, it should be pointed out that always $\bar{R} = \bar{S} = (n+1)/2$.

Having assigned $c_{ij} = \text{sgn}(x_j - x_i) = \text{sgn}(R_j - R_i)$, the formula for *Kendall's tau (τ) rank correlation coefficient* is obtained:

$$\tau = \sum_{i < j} \text{sgn}(R_i - \bar{R}) \cdot \text{sgn}(S_i - \bar{S}),$$

where R_k and S_k are ranks of the given samples as in the previous notations.

Regarding binary data, other definitions of similarity (proximity) measures are also used in practice [81]. For instance, simple matching coefficient, Jaccard coefficient, and extended Jaccard coefficient can be efficiently used to measure the similarity between data objects of aforementioned kind. Let x and y be two objects that consist of n binary attributes. Comparing of binary data x and y induces the following qualities:

- f_{00} = the number of attributes where $x_i = 0$ and $y_i = 0$
- f_{01} = the number of attributes where $x_i = 0$ and $y_i = 1$
- f_{10} = the number of attributes where $x_i = 1$ and $y_i = 0$
- f_{11} = the number of attributes where $x_i = 1$ and $y_i = 1$.

Simple matching coefficient (SMC) is commonly used similarity measure for binary data. It does not pay attention to elimination of 0–0 matches from consideration. SMC reflects both presences and absences equally. For instance, it can be utilized to find respondents who had answered questions on a test similarly.

$$SMC = \frac{f_{00} + f_{11}}{f_{00} + f_{01} + f_{10} + f_{11}}$$

Jaccard coefficient can be employed for sparse data. The idea is to compare nonzero elements belonging to two samples. Consequently, it makes little sense to use SMC coefficient in such a case because samples will be always similar because of the large number of zeros. In this case, 0–0 matches must not be counted. That is why under such conditions, Jaccard coefficient is more reasonable than simple matching coefficient.

$$J = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

Extended Jaccard coefficient is an expansion of Jaccard coefficient to real data. In the binary case, it is equal to a regular Jaccard coefficient. The extended Jaccard coefficient can be used for document data and also known as Tanimoto coefficient.

$$EJ(x, y) = \frac{x \cdot y}{||x||^2 + ||y||^2 - x \cdot y}$$

A significant issue related to distance measures is how to handle the situation when attributes do not have the same range of values. Often there is some correlation between some of attributes. In this case, a generalization of Euclidean distance referred to as *Mahalanobis distance* can be applied to measure similarity.

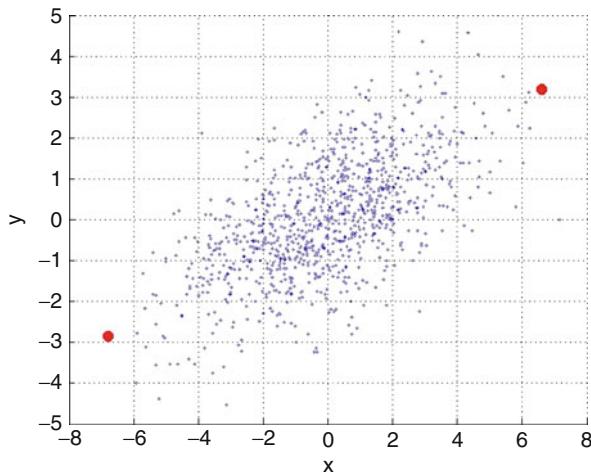


Fig. 5 Mahalanobis distance versus Euclidean distance [81]. The Mahalanobis distance between these two points are represented by *large dots* is 6. The corresponding Euclidean distance is 14.7. The correlation between axes causing distortion of data is equal to 0.6

The Mahalanobis distance between two objects x and y is defined as follows:

$$\text{mahalanobis}(x, y) = (x - y)\Sigma^{-1}(x - y)^T,$$

where Σ is the covariance matrix of the given data that reflects dependence between some attributes. The good example [81] is provided in Fig. 5. In practice, computing the Mahalanobis distance is worth doing it for correlated points in spite of its expensive cost.

Moreover, besides defining the similarity between data objects, in some cases, it is also needed to quantify the *dissimilarity*. The aforementioned proximity measures can be also used to define the pairwise dissimilarity. Therefore, all further discussion is applicable to clustering applications based on either similarity or dissimilarity. Without loss of generality, the term “similarity” will mostly be used throughout this chapter.

Determining an appropriate similarity measure is often a nontrivial task. However, if a suitable similarity measure is determined for a given dataset, this gives one the information about pairwise proximities for all pairs of data objects. Based on this information, one can easily construct the $N \times N$ proximity matrix and then use the proximity matrix to create a graph with N vertices and $N(N - 1)/2$ weighted edges, where the weights $w_{ij} = d(x^i, x^j)$ are equal to the corresponding proximity measures between data objects x^i and x^j .

Note that this simple procedure would produce a complete weighted graph (with all possible edges), which can be challenging to analyze from both theoretical and computational perspectives. To reduce the edge density of the obtained graph, one can use an appropriate *sparsification* procedure that can substantially reduce the number of edges in the graph while still preserving valuable information about the dataset. The following simple principles can be used to achieve this:

- *k-nearest neighbor approach*: For every vertex i , keep only the edges that connect it to its k -nearest neighbors, that is, preserve k edges (i, j) with the smallest values of w_{ij} (as determined by an appropriate proximity measure) and delete the remaining edges.
- *Threshold approach*: For every edge (i, j) , delete it if the corresponding similarity measure w_{ij} is below (or above) a specified threshold.

Applying one or both of these approaches can, in many cases, eliminate most of the edges in the constructed graph and make sure that the remaining edges preserve the most meaningful similarity relationships between the data objects in the considered dataset. It should be noted that the similarity measures between each pair of vertices can be further updated to incorporate the idea of *shared nearest neighbors*. That is, if two vertices i and j are both connected to certain number of other vertices, the weight of the edge (i, j) is increased by the number of these shared neighbors. This idea can be used for constructing the *shared nearest neighbor (SNN) similarity graph*. In particular, the concept of SNN graph has been used in the Jarvis-Patrick clustering algorithm, which essentially performs clustering on the considered dataset by identifying connected components in the corresponding SNN graph [49].

Finalizing this section, it should be emphasized that in many real-world situations, one of the most crucial problems is to choose a suitable proximity measure which is the best fit for the considered type of data. For many cases with dense continuous data, metric distance measures such as Euclidean distance are appropriate and often used. On the contrary, for sparse data, measures that disregard 0–0 matches are frequently utilized. The cosine similarity, Jaccard, and Extended Jaccard measures are typically used for such data. For comparing time series, Euclidean distance or various correlation coefficients depending on the given data should be employed. If the time series contain similar quantities, then Euclidean distance can be used. On the contrary, if the time series represent different quantities (for instance, quantities of various substances in human organism in different biomedical studies), then one should use the one of aforementioned definitions of correlation coefficients.

In general, after the graph representing a dataset has been constructed and sparsified, one can use a variety of graph-theoretic concepts and techniques to extract useful information about the considered dataset and divide the dataset into a number of meaningful clusters according to the specified similarity criterion. The next section discusses relevant concepts, definitions, and optimization problems arising in this analysis.

3 Basic Concepts from Graph Theory and Their Data Mining Interpretation

To facilitate further discussion, several formal basic definitions and notations from graph theory will be presented, and the interpretation of the introduced concepts from the perspective of data mining and information retrieval will be discussed.

Let $G = (V, E)$ be an undirected graph with the set of N vertices V and the set of edges $E = \{(i, j) : i, j \in V\}$. Directed graphs, where the head and tail of each edge are specified, are considered in some applications. The concept of a *multigraph* is also sometimes used. A multigraph is a graph where multiple edges connecting a given pair of vertices may exist. A directed multigraph is a directed graph in which loops and multiple edges between any two vertices are permitted. Examples of different types of graphs are given in Fig. 6.

One of the important characteristics of a graph is its *edge density*: the ratio of the number of edges in the graph to the maximum possible number of edges. A *dense graph* is a graph in which the number of edges is close to the maximum possible number of edges. A *sparse graph* is a graph in which the number of edges is substantially smaller than maximum possible number of edges. In undirected graphs, edge density formally defined as

$$D = \frac{2|E|}{|V|(|V| - 1)}, \quad (1)$$

where $|E|$ denotes the number of edges and $|V|$ denotes the number of vertices in graph G . Clearly, the maximum number of edges is $\frac{1}{2}|V|(|V| - 1)$, so the highest possible edge density is 1 (for complete graphs) and the lowest edge density is 0. As indicated in the previous section, one often needs to eliminate a substantial number of edges from a graph representing a real-life dataset, so that the edge density is reduced and only the most meaningful connections are preserved.

Often, a real-world dataset can be represented as a graph with the particular structure, and the examined properties of such graphs can be used for data mining. Some examples of well-known graph structures are:

- *Complete graph*: a graph, which contains all possible edges, that is, any two vertices are adjacent.
- *Bipartite graph*: a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent.
- *Complete bipartite graph*: a bipartite graph such that every pair of graph vertices in the two sets are adjacent.
- *Linear (path) graph*: a graph, which vertices can be listed in order, v_0, v_1, \dots, v_n , so that the edges are $(v_{i-1}, v_i) \in E$ for each $i = 1, 2, \dots, n$. If a linear graph occurs as a subgraph of another graph, it is a *path* in that graph.
- *Cycle graph*: a graph containing a single cycle through all nodes.
- *Planar graph*: a graph whose vertices and edges can be drawn in a plane such that no two of the edges intersect.

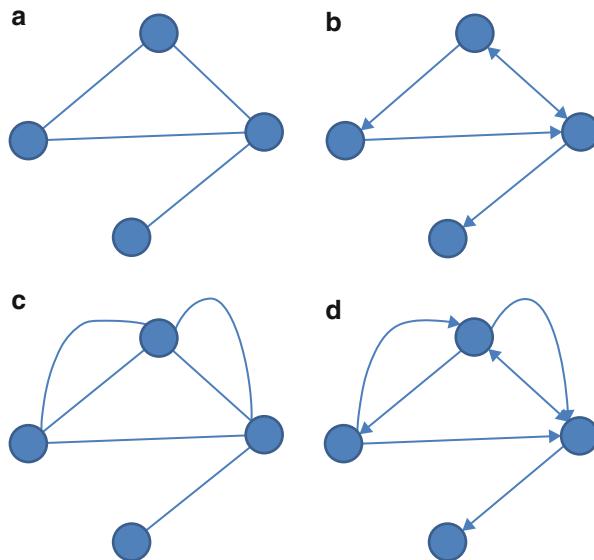


Fig. 6 Examples of different concepts of graphs. (a) Undirected graph. (b) Directed graph. (c) Multigraph. (d) Directed multigraph

- *Tree*: a connected graph with no cycles.
- *Forest*: a graph with no cycles, that is, the disjoint union of one or more trees.

3.1 Connected Components and Degree Distributions

The graph $G = (V, E)$ is *connected* if there is a path from any vertex to any vertex in the set V . If the graph is disconnected, it can be decomposed into several connected subgraphs, which are referred to as the *connected components* of G . A graph is called *K-vertex-connected* or *K-edge-connected* if there is no set of $K - 1$ vertices (edges, respectively) that disconnects the graph. A *K-vertex-connected* graph is often simply called *K-connected*. In many clustering techniques, distinct connected components are interpreted as distinct *clusters* in the corresponding dataset. It should be noted that although using connected components for clustering is rather common in a variety of applications, there are alternative graph-theoretic concepts that can be utilized in clustering problems. These concepts will be discussed in further sections.

The *degree* of a vertex is the number of edges emanating from it. For every integer k , one can calculate the number of vertices $n(k)$ with a degree equal to k , and then get the estimate of the probability that a vertex has the degree k as $P(k) = n(k)/n$, where n is the total number of vertices. The function $P(k)$ is referred to as the *degree distribution* of the graph. In the case of a directed graph, the concept of degree distribution is generalized: one can distinguish the distribution of *in-degrees*

and *out-degrees*, which deal with the number of edges ending at and starting from a vertex, respectively.

Degree distribution is an important characteristic of a dataset represented by a graph. It reflects the overall pattern of connections in the graph, which in many cases reflects the global properties of the dataset this graph represents. As mentioned above, many real-world graphs representing the datasets coming from diverse areas (Internet, telecommunications, finance, biology, chemistry, sociology) have degree distributions that follow the *power-law* model, which states that the probability that a vertex of a graph has a degree k (i.e., there are k edges emanating from it) is $P(k) \propto k^{-\gamma}$. Equivalently, one can represent it as $\log P \propto -\gamma \log k$, which demonstrates that this distribution forms a straight line in the logarithmic scale, and the slope of this line equals the value of the parameter γ . Figures 7 and 8 illustrate the power-law distribution in a regular and logarithmic scale.

An important characteristic of the power-law model is its *scale-free* property. This property implies that the power-law structure of a certain network should not depend on the size of the network. Clearly, real-world networks dynamically grow over time; therefore, the growth process of these networks should obey certain rules in order to satisfy the scale-free property. The necessary properties of the evolution of the real-world networks are *growth* and *preferential attachment* [17]. The first property implies the obvious fact that the size of these networks grows continuously (i.e., new vertices are added to a network, which means that new elements are added to the corresponding dataset). The second property represents the idea that new vertices are more likely to be connected to old vertices with high degrees. It is intuitively clear that these principles characterize the evolution of many real-world complex networks and massive datasets they represent.

From another perspective, some properties of graphs that follow the power-law model (and the corresponding datasets) can be predicted theoretically. Interesting properties of power-law graphs were studied using the theoretical *power-law random graph model* [5, 31]. Among these results, one can mention the existence of a giant connected component (i.e., a giant cluster that contains $\Theta(N)$ vertices) in a power-law graph with $\gamma < \gamma_0 \approx 3.47875$ and the fact that all connected components (clusters) are small ($o(N)$ vertices) otherwise.¹ The emergence of a giant connected component (or a giant connected cluster) at the point $\gamma_0 \approx 3.47875$ is often referred to as a *phase transition*.

The size of connected components of the graph may provide useful information about the structure of the corresponding dataset, since, as indicated above, the connected components would normally represent clusters of “similar” objects. In many applications, decomposing the graph into a set of connected components can provide a reasonable solution to the clustering problem. From the computational perspective, identifying all connected components in a graph is a *polynomially*

¹These results are valid *asymptotically almost surely* (a.a.s.), which means that the probability that a given property takes place tends to 1 as the number of vertices N goes to infinity.

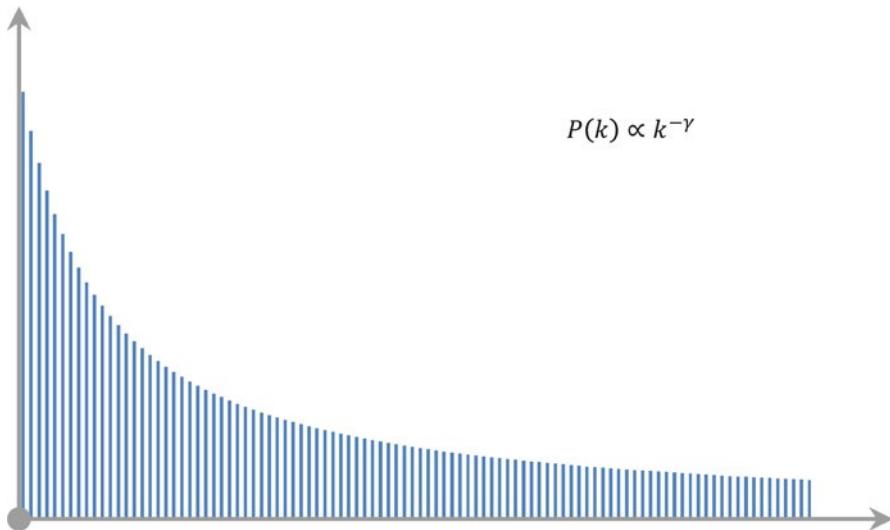


Fig. 7 A power-law distribution in the regular scale. The *horizontal axis* represents node degrees, and the *vertical axis* represents the corresponding probabilities (frequencies)

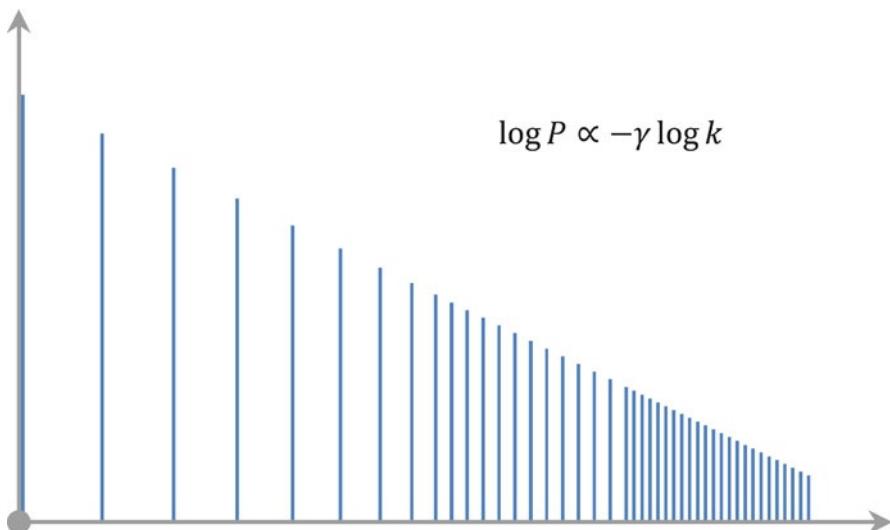


Fig. 8 A power-law distribution in the log-log scale. The interpretation of the axes is the same as in the previous figure

solvable problem, which is especially important when the considered graphs and datasets are very large.

However, in some situations, clusters based on connected components can be extremely large (e.g., comparable with the size of the whole graph, as indicated above),

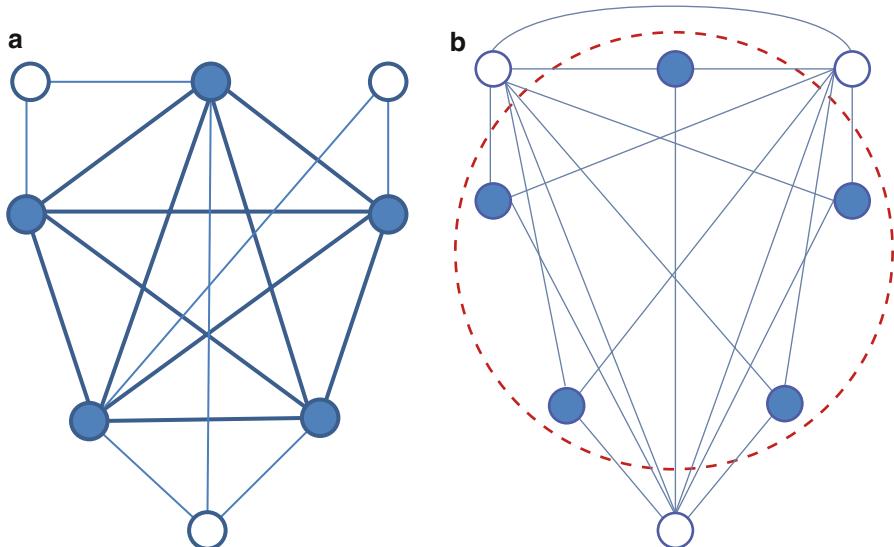


Fig. 9 Correspondence between the maximum clique and the maximum independent set in the complementary graph. (a) Clique in the graph (size 5). (b) Independent set in the complement graph (size 5)

and/or they may not be “tight enough,” that is, the degree of connectivity within a cluster may not be sufficient to make reliable conclusions regarding the similarity of the data objects. This motivates one to look for substantially more “robust” clusters that have specific properties of their connectivity patterns. The corresponding graph structures that address these issues are discussed in the next subsections.

3.2 Cliques and Independent Sets

Given a subset $S \subseteq V$, $G(S)$ is denoted as the subgraph induced by S . A subset $C \subseteq V$ is a *clique* if $G(C)$ is a complete graph (i.e., it has all possible edges). Figure 9a shows an example of a clique in the graph.

One of the most well-known problems in combinatorial optimization, the *maximum clique problem* [26], is to find the largest clique in a graph. The maximum clique problem has several equivalent formulations as an integer programming (IP) problem, or as a continuous nonconvex optimization problem.

The simplest formulation of maximum clique problem is the following edge formulation [70]:

$$\max \sum_{i=1}^N x_i \quad (2)$$

s.t.

$$\begin{aligned} x_i + x_j &\leq 1, \forall (i, j) \in \bar{E} \\ x_i &\in \{0, 1\}, i = 1, \dots, N. \end{aligned}$$

In the above formulation, \bar{E} is the set of edges in the *complementary graph* $\bar{G}(V, \bar{E})$, defined as follows. If an edge $(i, j) \in E$, then $(i, j) \notin \bar{E}$, and if $(i, j) \notin E$, then $(i, j) \in \bar{E}$. Decision variables x_i defined as

$$x_i = \begin{cases} 1, & \text{if vertex } i \text{ is in the clique} \\ 0, & \text{otherwise.} \end{cases}$$

The objective (2) is to maximize the number of vertices, which are in the clique. The maximum clique problem finds a clique of maximum cardinality. It can be easily transformed into the *maximum-weight clique problem*, which finds a clique of maximum weight, by introducing positive weights w_i associated with node i . Clearly, the objective of formulation (2) in the weighted case will be of the form

$$\max \sum_{i=1}^N w_i x_i$$

with the same set of constraints as in (2).

An *independent set* is a subset $I \subseteq V$ such that the subgraph $G(I)$ has no edges. Figure 9b shows an example of an independent set in the graph. The maximum independent set problem can be easily reformulated as the maximum clique problem in the *complementary graph* $\bar{G}(V, \bar{E})$. A maximum clique in \bar{G} is a maximum independent set in G , so the maximum clique and maximum independent set problems can be easily transformed to each other and are essentially equivalent. Let \mathcal{I} denote the set of all maximal independent sets of G . An alternative formulation is the following independent set formulation [70]:

$$\max \sum_{i=1}^N x_i \tag{3}$$

s.t.

$$\begin{aligned} \sum_{i \in S} x_i &\leq 1, \forall S \in \mathcal{I} \\ x_i &\in \{0, 1\}, i = 1, \dots, N, \end{aligned}$$

where $S \subseteq V$ is a subset of graph G .

The advantage of formulation (3) over (2) is the fact that it is a tighter formulation, that is, the gap between the optimal values of (3) and its linear programming relaxation is smaller [70].

Clearly, locating cliques and/or independent sets in a graph representing a dataset provides important information in terms of identifying completely connected (or, on the contrary, completely disconnected) clusters. Therefore, cliques would naturally represent very *dense* clusters of similar objects. On the contrary, independent sets can be treated as groups of objects that differ from every other object in the group. This information may be important in some applications. In particular, it is often useful to find a *maximum clique or independent set* in the graph, since it would give the maximum possible size of the groups of “similar” or “different” objects.

Although finding large cliques in a graph representing a dataset may be useful in terms of identifying large tightly connected clusters, an important computational disadvantage of this approach is that the maximum clique problem (as well as the maximum independent set problem) is known to be NP-hard [44]. Moreover, it turns out that these problems are difficult to approximate [11, 46]. This makes these problems especially challenging in large graphs.

3.3 Clustering via Clique Partitioning

The problem of locating cliques and independent sets in a graph can be naturally extended to finding an optimal *partition* of a graph into a minimum number of distinct cliques or independent sets. These problems are referred to as *minimum clique partition* and *graph coloring* [43], respectively. Pardalos et al. [71] give various mathematical programming formulations of these problems. Clearly, as in the case of maximum clique and maximum independent set problems, minimum clique partition and graph coloring are reduced to each other by considering the complimentary graph, and both of these problems are NP-hard [44]. The example of optimal graph coloring is represented in Fig. 10. The simplest formulation of graph coloring for graph $G(V, E)$ is given below. Solving these problems for graphs representing real-life datasets is important from a data mining perspective, especially for addressing the clustering problem.

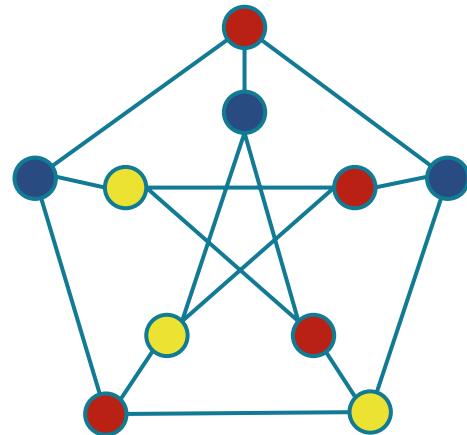
$$\min \sum_{k=1}^N y_k \quad (4)$$

s.t.

$$\sum_{k=1}^N x_{ik} = 1, \forall i \in V \quad (5)$$

$$x_{ik} + x_{jk} \leq 1, \forall (i, j) \in E \quad (6)$$

Fig. 10 Vertex coloring of the Petersen graph (minimum number of colors – 3)



$$y_k \geq x_{ik}, \forall i \in V, k = 1, \dots, N \quad (7)$$

$$y_k, x_{ik} \in \{0, 1\}, \forall i \in V, k = 1, \dots, N. \quad (8)$$

In the above model, decision variables x_{ik} and y_k defined as

$$x_{ik} = \begin{cases} 1, & \text{if color } k \text{ is assigned to vertex } i \\ 0, & \text{otherwise} \end{cases}$$

$$y_k = \begin{cases} 1, & \text{if color } k \text{ is used} \\ 0, & \text{otherwise.} \end{cases}$$

Constraints (5) ensure that exactly one color is assigned to each vertex. Constraints (6) prevent adjacent vertices from having the same color. Constraints (7) ensure that no x_{ik} can be 1 unless color k is used. The optimal objective value gives the chromatic number of a graph ($\chi(G)$), and the sets $S_k = \{i | x_{ik} = 1\}, \forall k$, comprise a partition of the vertices the minimum number of independent sets.

Since the data elements assigned to the same cluster should be similar to each other, the goal of clustering is achieved by finding a clique partition of the graph, and the number of clusters will equal the number of cliques in the partition.

Similar arguments hold for the case of the graph coloring problem which should be solved when a dataset needs to be decomposed into the clusters of “different” objects (i.e., each object in a cluster is different from all other objects in the same cluster) that can be represented as independent sets in the corresponding graph. The number of independent sets in the optimal partition of graph G is referred to as the *chromatic number* ($\chi(G)$) of the graph.

3.4 Using Clique Relaxations for Graph-Based Clustering

Although cliques and independent sets address the issue of “robust” tightly connected clusters, a significant practical drawback of these structures is that they are often *too restrictive*, that is, all pairs of vertices need to be connected (disconnected). On the contrary, as mentioned above, a simple connectivity requirement is not restrictive enough to guarantee sufficient tightness of a cluster. In an attempt to provide a tradeoff between the clique-based and sparse connected component-based clusters, the concepts of *clique relaxations* are introduced.

Several concepts of clique relaxations first appeared in studying cohesive subgroups in social networks and were based on relaxing some of the desirable properties that a clique idealizes. For instance, instead of cliques and independent sets, one can consider *quasi-cliques* and *quasi-independent sets* and partition the graph on this basis. Quasi-cliques are subgraphs that are *dense enough* (i.e., they have a sufficiently high edge density, but the edge density does not have to be 1 as in the case of cliques). Formal definitions of a quasi-clique and other clique relaxations will be given below.

It is often reasonable to relate clusters to quasi-cliques, since they still represent sufficiently dense clusters of similar objects [50, 88]. Obviously, in the case of partitioning a dataset into clusters of “different” objects, one can use quasi-independent sets (i.e., subgraphs that are sparse enough) to define these clusters.

Other types of clique relaxations have also been introduced. Many of these definitions come from the studies of social networks; however, these concepts clearly have important data mining interpretations. The main idea behind these concepts is to “relax” certain properties of a clique while still maintaining sufficient connectivity and robustness characteristics of the obtained network structures. Note that in many cases, the size of these clique relaxations is substantially larger than the size of cliques, which provides a significant advantage in situations when a *large tightly connected cluster* needs to be identified. In addition, an important factor that motivates the use of these clusters instead of cliques is the fact that they are not as sensitive to potential errors in data collection and measurement. For instance, if one link in a clique-based cluster is missing because of an error, the structure of the clique is violated, and the corresponding cluster may not be adequately reflected in data mining results. However, if one uses quasi-clique-based clusters, the absence of several edges due to such errors is usually not as critical, since it will likely not have a significant effect on the required edge density of the cluster.

The ideas for the clique relaxation concepts discussed below originally come from the study of social networks; however, these definitions can be efficiently utilized in the data mining context. More specifically, there are three main directions for possible relaxations of the clique definition:

1. *Density-based relaxations* – relaxing the requirement for the edge density of a clique to be 1: *quasi-cliques* (γ -dense subgraphs) [3]

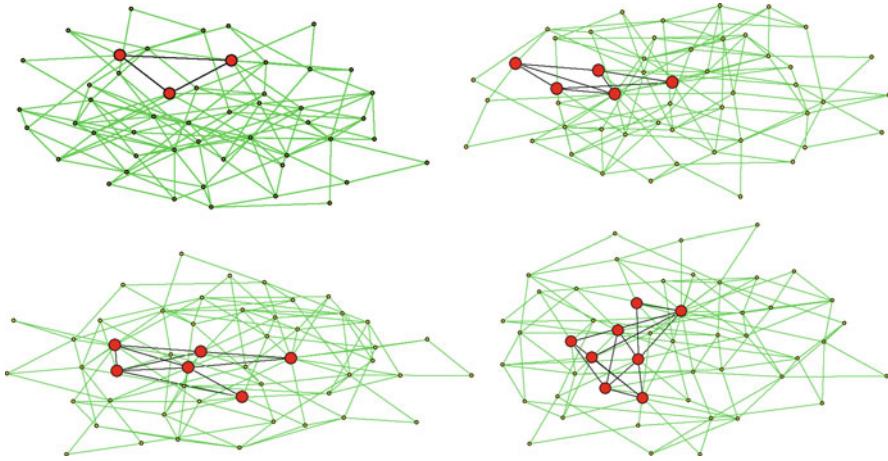


Fig. 11 Visual comparison of maximum clique ($\gamma = 1$) versus maximum quasi-clique ($\gamma = 0.7, 0.6, 0.5$) in the same uniform random graph $G(50, 0.1)$

2. *Degree-based relaxations* – relaxing the requirement that the degree of each node in a clique of size q to be $q - 1$: k -plexes [78]
3. *Path (diameter)-based relaxations* – relaxing the requirement that the length of the path between any two nodes in a clique to be 1: k -cliques [59], k -clubs, and k -clans [66]

As mentioned above, a *quasi-clique* (also referred to as a γ -clique or a γ -dense subgraph) is a subgraph that has the edge density of at least γ , where $\gamma \in (0, 1]$. Clearly, a quasi-clique becomes a clique when $\gamma = 1$. A more rigorous definition is provided below. Let $G = (V, E)$ be the graph with the set of vertices G and the set of nodes E . Denote the graph induced by the vertex subset $S \subseteq G$ by G_S . Under these notations, G_S is a quasi-clique (γ -dense subset) if $|E(G_S)| \geq \gamma \binom{|V(G)|}{2}$ (recalling that $\binom{|V(G)|}{2} = \frac{|V(G)|(|V(G)|-1)}{2}$) and it is connected. The illustrative example is shown in Fig. 11. The given graph $G(50, 0.1)$ is a random graph on 50 vertices with probability $p = 0.1$ that each edge exists. The size of the maximum clique is equal to 3, and the size of the maximum quasi-clique with $\gamma = 0.7$ is equal to 5, with $\gamma = 0.6$ is equal to 6 and with $\gamma = 0.5$ is equal to 8. As one can observe, the sizes of quasi-cliques are substantially larger than the size of the maximum clique, whereas the density of these clusters is still high. Therefore, clusters based on quasi-cliques may be more meaningful as they include a larger number of data elements, which are still “similar enough” to be included in the same cluster.

A k -plex is a subgraph in which the degree of each node is at least $q-k$ (assuming that q is the number of nodes in this subgraph). Figure 12 presents an illustrative example of k -plexes in a small graph on 7 vertices.

A k -clique is a subgraph where the length of the path between any two nodes is at most k (note that other nodes in this path are *not required* to belong to the k -clique). Consider the graph $G = (V, E)$ as in the previous definition and the subset $S \subseteq G$

Fig. 12 Maximum k -plexes in the graph for different k .

- (a) Maximum 1-plex (clique) subgraph in the given graph.
- (b) Maximum 2-plex subgraph in the given graph.
- (c) Maximum 3-plex subgraph in the given graph.
- (d) Maximum 4-plex subgraph in the given graph

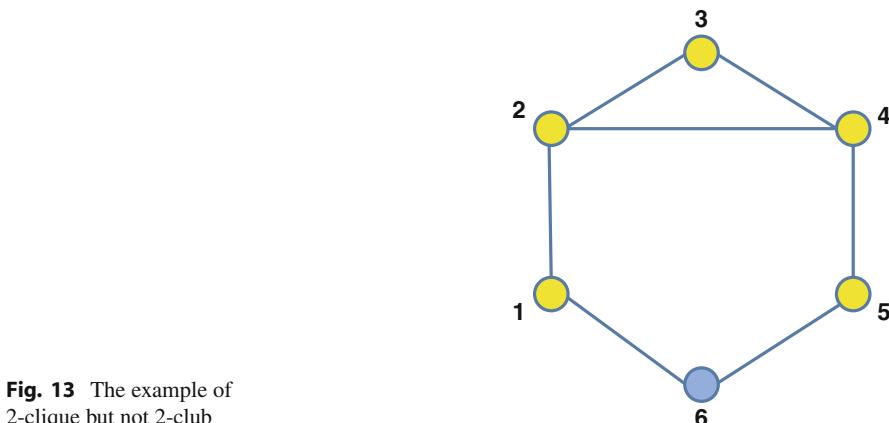
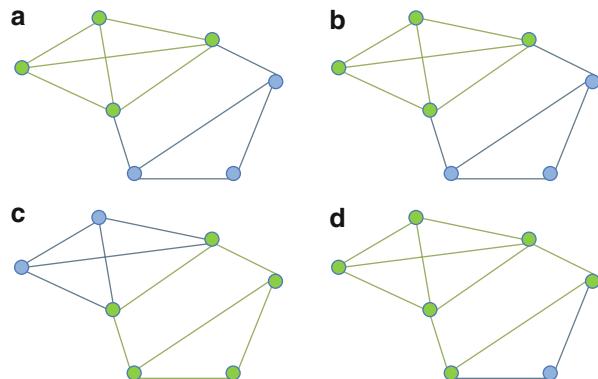


Fig. 13 The example of 2-club but not 2-clique

denoted by G_S . Then, G_S is a k -clique if $\forall i, j \in V$ it is given that $d_G(i, j) \leq k$. Note that distances between any two points in k -clique can be greater than k because some paths might pass through the complementary graph $G \setminus G_S$ to the k -clique. An example illustrating this fact is mentioned in [7] and given in Fig. 13. As one can see, there is the set $L = \{1, 2, 3, 4, 5\}$ that forms a 2-clique, but the diameter of this cluster is equal to 3. Therefore, the concept of k -clique does not necessarily embody the idea of “tightness” of the corresponding cluster of points in a graph. In order to ensure a higher level of tightness and low diameter of a cluster, the concept of a k -clique can be “upgraded” to a k -club (Fig. 14).

A k -club is a subgraph that has a *diameter* of at most k (note that in this definition, all the nodes in the shortest path that connects any pair of nodes within a k -club have to also belong to this k -club). Using the same notations as in previous definitions, it can be pointed out that G_S is a k -club if $\forall i, j \in V$ it is given that $d_{G_S}(i, j) \leq k$. Thus, k -club can be viewed as a “tighter” structure than a k -clique. It is more applicable for connectivity and clustering problems on networks where cluster “cohesiveness” and “robustness” issues play an important role.

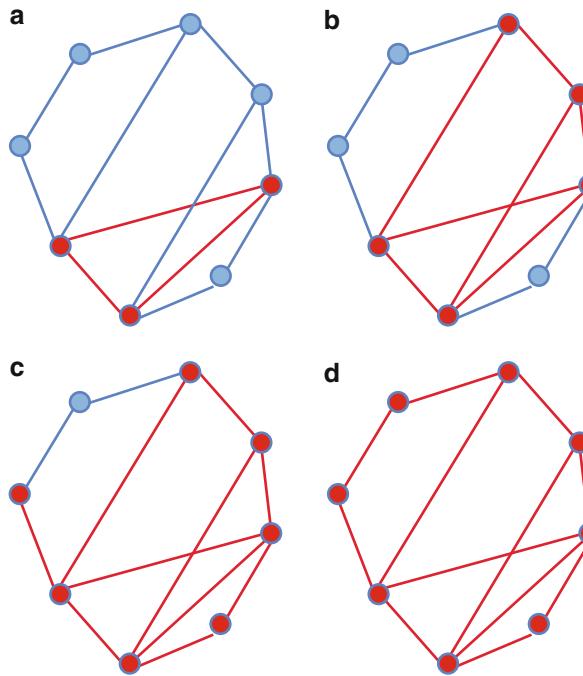


Fig. 14 Maximum k -clubs in the graph for different k . (a) Maximum 1-club (clique) subgraph in the given graph. (b) Maximum 2-club subgraph in the given graph. (c) Maximum 3-club subgraph in the given graph. (d) Maximum 4-club subgraph in the given graph

Although the given definitions are rather straightforward and intuitively clear, mathematically rigorous studies on related optimization aspects (e.g., mathematical programming formulations for finding the largest clique relaxation-based clusters in graphs) have started to appear only within the past few years.

For instance, an important combinatorial optimization problem associated with quasi-cliques is the maximum γ -clique problem. Pattillo et al. [72] proved that the decision version of the γ -clique problem is NP-complete for any fixed positive $\gamma \in (0, 1)$.

Furthermore, linear 0–1 formulations for the maximum γ -clique problem were proposed in [72]. Consider the problem of finding a maximum γ -clique in the graph $G = (V, E)$. If for some subgraph G_S one wants to check whether this subgraph is a γ -clique, one can define $x = (x_1, \dots, x_N)$ as a 0–1 vector with $x_i = 1$ if node i belongs to G_S , or zero otherwise. Since $x_i^2 = x_i \quad \forall i$, then the subgraph G_S is a γ -clique if it consists of $\frac{1}{2}\gamma \sum_{i,j=1, i \neq j}^N x_j x_i$ arcs. The number of arcs in the subgraph G_S can be calculated as

$$\frac{1}{2}x^T Ax = \sum_{i,j=1,i \neq j}^N a_{ij} x_j x_i.$$

Therefore, the problem of finding the maximum γ -clique in the graph G can be formulated as follows:

$$\max \sum_{i=1}^N x_i.$$

subject to

$$\sum_{i,j=1,i \neq j}^N a_{ij} x_j x_i \geq \gamma \sum_{i,j=1,i \neq j}^N x_j x_i.$$

This is the problem with a linear objective and one quadratic constraint. In [72], two possible linearizations of this problem were proposed.

The simplest linearization is rather straightforward with the main idea of introducing new variables $w_{ij} = x_i x_j$. The constraint $w_{ij} = x_i x_j$ is equivalent to

$$\begin{aligned} w_{ij} &\leq x_i, \\ w_{ij} &\leq x_j, \\ w_{ij} &\geq x_i + x_j - 1. \end{aligned}$$

Now, the problem can be formulated as the following linear 0–1 problem:

$$\max \sum_{i=1}^N x_i$$

subject to

$$\begin{aligned} \sum_{i,j=1,i < j}^N a_{ij} w_{ij} &\geq \gamma \sum_{i,j=1,i < j}^N w_{ij}, \\ w_{ij} &\leq x_i, \\ w_{ij} &\leq x_j, \\ w_{ij} &\geq x_i + x_j - 1. \end{aligned}$$

This problem contains $N(N - 1)/2$ variables and $\frac{3}{2}N(N - 1) + 1$ constraints, that is, it has $O(N^2)$ entities. Besides this formulation, the problem of finding a maximum γ -clique can also be represented by the linear 0–1 formulation with $O(N)$ variables and constraints as described in [72], although this linearization is not as straightforward as the one mentioned above, since it explicitly uses the special structure of the considered problem. Despite the fact that the second linearization is more compact, it does not always provide better computational performance

compared to the $O(N^2)$ linearization, although it does perform faster on sparse graphs and relatively high values of γ , which is a typical setup in many situations that deal with finding dense clusters in sparse networks representing real-world datasets.

Besides the maximum quasi-clique problem, similar problems for other types of clique relaxations representing tight clusters (in particular, k -plexes and k -clubs) have been considered in the recent literature from the combinatorial optimization perspective. The maximum k -plex problem has been addressed by Balasundaram et al. in [15], where they provide the most compact formulation with N variables and constraints. It has also been proven in [15] that the decision version of this problem is NP -complete for any fixed positive integer k .

The maximum k -club problem was considered in several previous studies. The first mathematical programming formulations (linear 0–1 formulations containing $O(N^k)$ variables and constraints) and complexity analysis of this problem were addressed in [14]. In a recent study, Veremyev and Boginski in [82] developed a new compact linear 0–1 programming formulation for finding maximum k -clubs that substantially reduces the number of entities compared to the known formulations with $O(kN^2)$ entities and allows one to find exact solutions to problems with larger values of k , which can provide more flexibility in terms of modeling k -club-based clusters.

In conclusion, utilizing clique relaxations in the context of clustering in data mining appears to be a promising approach with attractive characteristics, since it provides sufficient flexibility in terms of the tightness and size of the obtained clusters. The chapter describes more details on clique relaxation models in graph theory.

4 Examples of Real-World Applications

The aforementioned network-based data mining techniques have been used in a variety of real-world applications during recent years. Although it is impossible to describe all the applications in detail within one chapter, several examples of networks and datasets that have been addressed in the recent literature will be briefly discussed.

4.1 Biological Networks

Nowadays, representing complex biological systems as networks and studying the properties of these networks has become a very popular tool in computational biology and biomedical applications. For instance, an overview of graph-based molecular data mining approaches is presented in [41]. An extensive recent volume [30] covers many topics related to clustering problems in biological network applications, such as the analysis of regulatory and interaction networks from clusters of

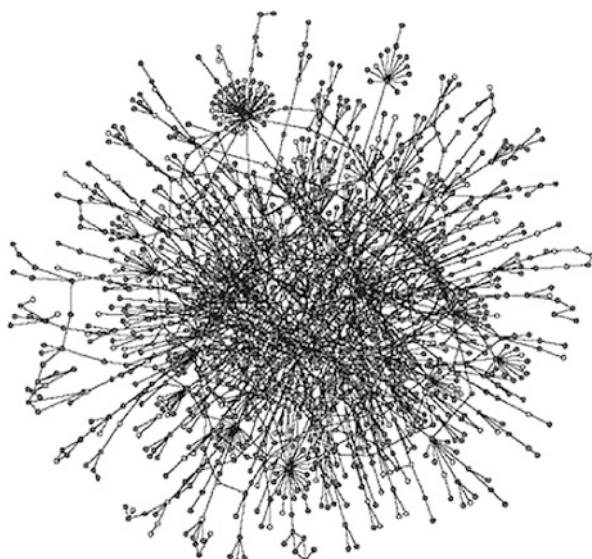
co-expressed genes, graph-based approaches for motif discovery, identifying critical nodes in protein-protein interaction networks, genetic graph partitioning algorithms, and so on. The most recent review of network-based approaches in biomedical applications is presented in [18].

As new advances in experimental design produce a large amount of data to describe biological interactions at a genome scale, this data is increasingly being deposited into biological databases. These interaction networks include, among others, protein interaction networks [54, 77], metabolic networks [33, 69], gene regulatory networks [6], and signal transduction networks [80]. Since there is evidence that preserved interaction pathways exist across organisms, efficient algorithms to analyze common pathways in these networks can make significant contributions to understanding of these networks. The most common representation of such networks is a graph with vertices representing biological entities and edges representing interactions between them. With this representation, one can look for the presence of special substructures in these graphs to answer various questions. By studying paths in these graphs, one can investigate the properties of pathways and their relationships, and by finding similar subgraphs, one can search for *network motifs* and study common substructures embedded in these networks. While path matching allows the study of individual “linear” paths or chains, graph matching allows one to investigate entire “nonlinear” pathways or functional modules. Efficient algorithms for solving these problems give biologists an opportunity to study evolutionary mechanisms such as pathway conservation, duplication, and specialization. Thus, one important strategy is to start from a given pathway of interest and find related pathways within the same organism. By comparing proximities and differences in the returned pathways, it is possible to evaluate various hypotheses concerning evolution. Although the path matching problem models linear interaction chains well, many biological pathways are more complex and may consist of multiple interacting components [86].

To solve the problem of finding similar paths in two graphs, a combined graph from the two given graphs should be constructed, so that each vertex in the combined graph represents a pair of related vertices, one from each of the two given graphs, and each pathway alignment is represented as a simple path in the combined graph. A randomized algorithm for finding simple paths by imposing acyclic edge orientations may be useful in this case. The main difficulty here arises from the presence of cycles in the given graphs. To avoid substantial repetitions of vertices in a path, the problem of finding similar paths in two graphs has been reduced to the NP-hard problem of finding high-scoring simple paths of a given length in a combined graph (see [54]).

In contemporary post-genomic era, protein interactions are well captured by so-called protein-protein interaction networks. In particular, protein-protein interaction networks (see Fig. 15) have been experimentally constructed for many known organisms. A protein interaction network is represented by a graph with the proteins as nodes, and an arc exists between two nodes if the proteins are known to interact. Also, it has been discovered that the degree distribution of many of these complex protein interaction networks follows a power law. For instance, the protein

Fig. 15 A cluster in the protein-protein interaction network of the yeast [87]



interaction network of *S. cerevisiae*, as well as many other networks of this type, has been experimentally shown to have a power-law degree distribution. Due to the power-law structure, an average node degree is no longer representative, and the majority of nodes have few neighbors, while a smaller number of nodes have very high degrees. Many problems for biological networks can be solved via identification of large clusters or functional modules. Cliques have formed the basis for several studies that attempt to decompose a protein interaction network into functional modules and protein complexes. Protein complexes are groups of proteins that interact with each other at the same time and place, while functional modules are groups of proteins that are known to have pairwise interactions by binding with each other to participate in different cellular processes at different times. Clique models have been popular in this area as they represent tight clusters in a network, but sometimes they are too restrictive, as indicated earlier in this chapter. Therefore, employing clique relaxations instead of cliques may be useful in this case. Network-based clustering techniques utilizing clique relaxations (e.g., k -clubs) have been applied to studying those networks in [13] and [14].

There are several computational methods to measure sequence similarities between proteins. Some homology search algorithms capture similar database sequences to a query from a database and calculate the statistical significance of their similarities. However, retrieved proteins with evident similarities are often uninformative from a practical perspective. To cope with this problem, it is important to consider alternative approaches, such as signature identification and sequence clustering. Functionally or structurally related proteins often have locally conserved regions, referred to as functional motifs or signatures. Proteins sharing the same signatures do not always exhibit apparent similarities between

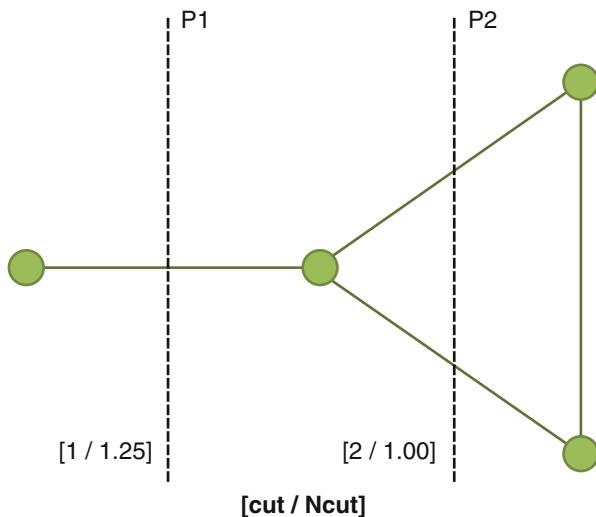


Fig. 16 An instance of a graph representation of protein sequence similarities. Graph partitions are shown by *dashed lines* P_1 and P_2 relatively. Each of these lines partitions vertices of the graph into two distinct sets. Numbers below the *dashed lines* show the capacities of their cuts and normalized cuts

their sequences. Therefore, they are referred to as *distantly related proteins*. Massive amounts of signature data have been accumulated in special signature databases. These databases enable one to accurately find proteins that share signatures and their relative positions on the protein sequences. Clustering protein sequences is an important approach to finding distant relationships. Distantly related proteins may be grouped together into a cluster even though they do not show apparent similarities. The most well-known and simplest method for clustering proteins is *single linkage* (SL). A detailed description of SL method can be found in [57]. Although the method is simple, fast, and widely used, it has difficulty in detecting an appropriate cutoff score for sequence similarity. More advanced algorithms employ a graph representation of a given set of proteins. Such algorithms are *SL-based clustering method*, named SL-KL (see [52]), and *p-quasi complete linkage algorithm* (see [61]). They allow some proteins to overlap between two or more groups. However, such clustering techniques usually carry high computational costs. Therefore, in order for the algorithms to be applicable to a large number of proteins, attention should be focused on approximate distantly related proteins without overlapping groups. This approach to clustering proteins is based on iterative partitioning. The key concept of graph partitioning is a “cut,” which is a set of edges between two distinct sets of nodes (or proteins). When the sum of edge weights in a cut (referred to as the cut capacity) is small, the two sets are considered to be dissimilar.

Figure 16 shows a very simple example of graph representation of protein similarities, where for simplicity all edges have unit weights. The proteins are

naturally grouped into two sets on the left and right side of the cut P2. However, P1 is a partition with the lowest capacity of the cut, called the *minimum cut*, due to the sparseness of the set of edges (e.g., one edge) connecting the proteins on the left side of P1. Unfortunately, it is difficult for applications to find distant relationships in a protein set containing various kinds of signatures using minimum-cut partitioning. The *normalized cut partitioning* is proposed to avoid the tendency of inaccurate clustering by the minimum-cut partitioning. It uses the normalized cut capacity (N_{cut}) criterion, which is the ratio of the capacity of a cut to the sum of similarities in each of the two distinct sets. In the context of the graph in Fig. 16, P2 is a partition with the minimum normalized cut. In addition, a locally minimal cut criterion can also be used. All the aforementioned methods are described in detail in [53].

The genome is also a highly interactive system, and the expression of a gene depends on the activity of other genes. Gene expression data is often represented in a network format. Cliques are widely used for gene co-expression networks' clustering. In the case of gene expression data, nodes represent the genes and arcs represent the relations between co-expressed genes with correlation higher than a specified threshold (see [51]) based on microarray experiments. Quasi-clique-based clusters can also be identified in these networks [73].

The main challenge in the analysis of this data is to understand biological functions from the topology of the network. There is a rather useful approach based on efficient *sequence alignment* algorithms and a statistical theory to assess the significance of the results (see [35]). Another way to address these challenges leads to an algorithmic procedure referred to as *local graph alignment*, which is conceptually similar to sequence alignment. It is based on a scoring function measuring the statistical significance for families of mutually similar subgraphs. This scoring involves quantifying the significance of the individual subgraphs, as well as their mutual similarity. As a computational problem, graph alignment is more challenging than sequence alignment. Sequences can be aligned in polynomial time by using dynamic programming algorithms. For graph alignment, the existence of a polynomial-time algorithm is unlikely. Thus, an important issue for graph alignment is the construction of efficient heuristic search techniques, such as those described in [19].

4.2 Chemical Networks

Chemical datasets are often represented as graphs, since they are natural representations of chemical compounds. Such graphs can be used to model topological and geometric characteristics of chemical structures. The nodes correspond to atoms, and the edges correspond to bonds connecting the atoms. For instance, Fig. 17 shows ball-and-stick diagrams of chemical compounds pentacene (Fig. 17a) and lysozyme (Fig. 17b).

Chemical graph mining techniques have many applications in the drug discovery procedure that include structure-activity-relationship (SAR) model construction and bioactivity classification. Many data mining algorithms are based on the assumption

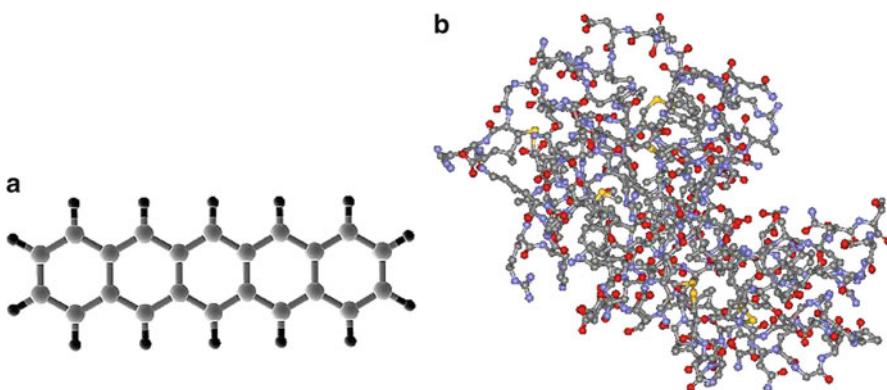


Fig. 17 Representation of chemical data. (a) Pentacene molecule. (b) Lysozyme molecule

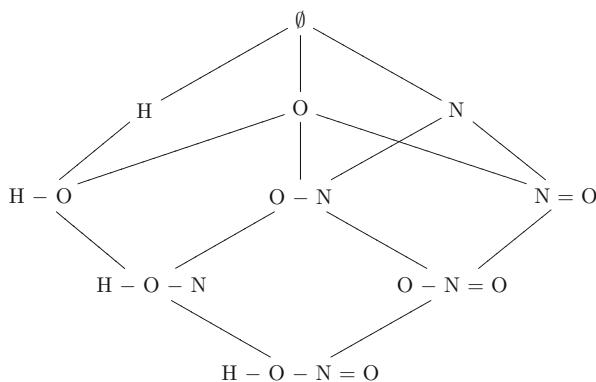
that the properties of a chemical element are related to its structure. For further details, see [83]. Studying and mining chemical graphs can provide new perspectives to chemistry, biology, and toxicology.

In recent years, a lot of algorithms on frequent graph pattern mining and related chemical graph mining problems have been published. The amount and complexity of the available data is steadily increasing. Within molecular databases, it is interesting to find patterns (fragments) that appear at least in a certain percentage of graphs. Another problem is to find fragments that are frequent in one part of the database but infrequent in the other. Recently developed approaches for chemical graph data mining include *Subdue* (see [32]), *inductive logic programming* (see [40]), and *Gaston* (see [68]).

Chemical graph data mining can be considered in the context of search in the lattice of all possible subgraphs. In Fig. 18, a small example is shown based on a small molecule of nitrous acid (shown in the bottom of the figure). The four nodes are labeled corresponding to their atom types (H, N, and O), and the edges are labeled with their corresponding bond types (single and double bonds). All possible subgraphs of this small graph are shown in the figure. At the top, the empty graph is denoted by \emptyset . In the next row, all possible fragments containing just one atom are listed and so on. At the bottom of the figure, the complete molecule with three bonds is given. Many chemical graph mining algorithms are based on the common idea of performing search this subgraph lattice. They are interested in finding a subgraph (or several subgraphs) that is the most frequent in the considered graph. Building this lattice of frequent subgraphs involves two main steps: *candidate generation*, where new subgraphs are created out of smaller ones, and *support computation* where the frequency or support of the new subgraphs in the database is determined. Both steps are computationally challenging, and various algorithms and techniques have been developed to find frequent subgraphs in reasonable time. For more details, see [42].

Another approach utilized in chemical graph data mining is based on kernels. Kernel methods have emerged as an important class of machine-learning methods

Fig. 18 The lattice of all subgraphs in nitrous acid HNO_2



suitable for variable-size structured chemical data (see [76]). Applications of kernel methods to molecular bond graphs require the construction of graph kernels, that is, functions that measure the similarity between graphs with labeled nodes and edges. Also, kernel methods can be applied to molecular clustering and regression problems, such as predicting the boiling point of alkanes and other organic substances [12].

4.3 Brain Networks

One more interesting application of network-based data mining is associated with studying human brain. It is a significant practical task since the results of this analysis are important in medical practice, especially studying brain diseases. The analysis of connectivity patterns of the brain function is extremely challenging because of the complex structure with the huge numbers of neurons and dynamic nature of connections between them. According to [67], the number of neurons is estimated to be 8.3×10^9 , and the number of connections is approximately 6.6×10^{13} . The empirical analysis of such structures represented as a graph with neurons as nodes is computationally prohibitive.

Several aspects of the analysis of brain connectivity are studied in [48]. One can potentially use the following technique to analyze the graph representing the brain. Different functional units of the brain containing a large number of neurons can be considered as a nodes in the graph. This approach enables the use of different algorithms proposed for much smaller graphs. This method can be applied in order to study properties of the connections between these functional units. In [37], the authors applied this approach to study the cortical visual system of a macaque monkey represented by a graph.

A similar approach can also be applied to study some properties of the human brain. The major areas of the human brain and their functions are represented on Fig. 19. In [36], the authors investigated a graph corresponding to 147,456

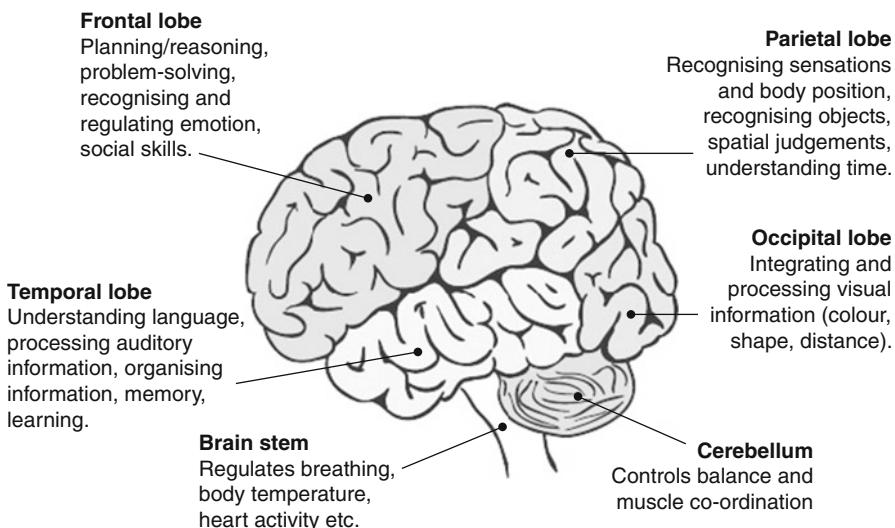


Fig. 19 Functional areas of the human brain

functional units of the human brain, which were selected by dividing the entire brain into a set of $64 \times 64 \times 36$ voxels of a small size. The set of edges of this graph were constructed using the time series defined by the recorded signals representing the activity of each functional unit. The correlation coefficients were calculated according to the formula similar to (9) using the time series representing the signals obtained from functional units, and the corresponding nodes were connected if the correlation exceeded a certain threshold value. It turned out that the resulting brain graphs also follow the power-law distribution with the parameter $\gamma \approx 2$ [36].

Investigating different characteristics of brain networks is a crucial practical task. One can study various structures (network motifs) in the brain networks as spanning trees, cliques/independent sets, clique relaxations, etc., which may provide a new insight into the process of signal propagation between the functional brain units and neurons. This information could potentially be very useful in studying brain disorders. More detailed information on network models in brain dynamics is given in the chapters [75].

4.4 Telecommunication/Information Exchange Networks

Large-scale information exchange networks arise in a great variety of areas, including large-scale wireless communication/sensor networks, the Internet/World Wide Web, and telephone traffic networks. The Internet, the World Wide Web, and the telephone call networks (also referred to as call graphs) were experimentally shown to have a power-law structure. Studying the structural properties and the global

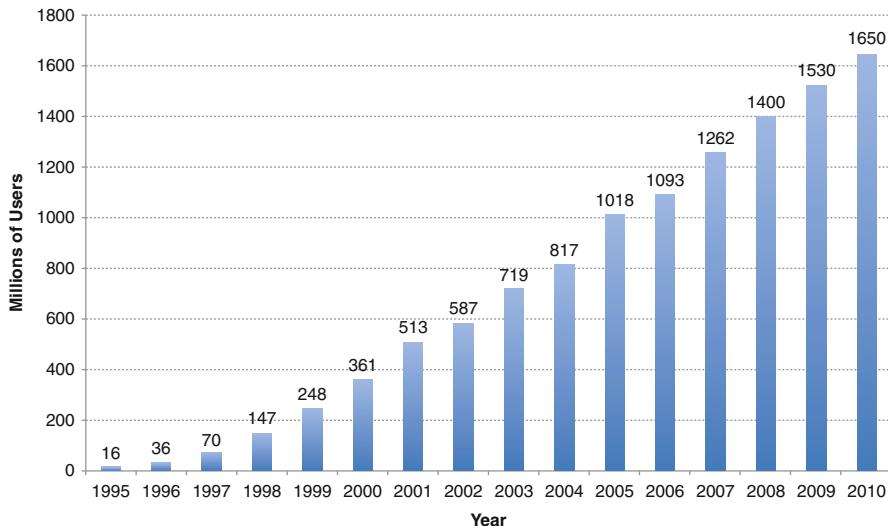


Fig. 20 Internet users in the world. Growth 1995–2010 [65]

organization of these networks has been addressed in recent literature. Graph theory has been applied for web search [28, 56], web mining [63, 64], and other problems arising in the Internet and World Wide Web [9]. In an interesting experimental study of the World Wide Web, Broder et al. [29] used two Al tavista crawls, each with about 200 million pages and 1.5 billion links and identified certain properties of the connected components of this graph. Studying the structural properties of the Internet is an exciting area of ongoing research [85]. The necessity of such research is justified because the usage of the Internet increases very rapidly in the world over years. The graph on the Fig. 20 represents the growth of number of Internet users through 1995–2010 [65]. The overall structural characteristics of the Internet are sometimes referred to as “robust yet fragile” [34], which implies the overall robustness with respect to random disruptions/errors, but potential vulnerability to targeted attacks on “hubs” (i.e., high-degree vertices).

Another well-known experimental study of a large communication network was conducted on the “call graph” representing daily telephone traffic data from AT&T telephone billing records [1]. The call graph is also very large, and the considered instance contained 53,767,087 vertices and over 170 million edges. The size of cliques and quasi-cliques in this graph was investigated, and it turned out that the size of the largest clique did not exceed 32, and the size of the largest quasi-clique with $\gamma \geq 0.5$ did not exceed 96. Since the size of the considered graphs is extremely large, the problems of identifying tight clusters, such as cliques and quasi-cliques, cannot be solved exactly due to computational challenges. Therefore, appropriate heuristic algorithms need to be applied to tackle these problems. For instance, in [1], the greedy randomized adaptive search procedure (GRASP) [38, 39] was successfully used.

4.5 Financial Networks

Stock market data can also be represented as a network and analyzed using combinatorial optimization techniques. One can represent each traded stock by a node in a network, and a given pair of nodes will be connected by an edge if the corresponding stocks exhibit a similar behavior over a certain period of time (this similarity can be measured as a correlation between the corresponding time series, and a threshold can be set, so that an edge would be placed if this threshold is exceeded) [23].

Boginski et al. [24, 25] have previously done the work on studying the behavior of the US stock market and modeling the corresponding massive dataset as a large-scale network (referred to as the *market graph*).

It is possible to construct the market graph in a relatively simple way [23]. As it was mentioned, the set of nodes of such graphs corresponds to the set of stocks. For each pair of stocks i and j , the correlation coefficient C_{ij} is calculated using the following procedure. Let $P_i(t)$ denote the price of the instrument i at time t . Then,

$$R_i(t, \Delta t) = \log \frac{P_i(t + \Delta t)}{P_i(t)}$$

defines the natural logarithm of return of the stock i over the certain period of time $[t, t + \Delta t]$.

Correlation coefficients C_{ij} between all pairs of stocks i and j are calculated as

$$C_{ij} = \frac{\mathbb{E}[R_i R_j] - \mathbb{E}[R_i] \mathbb{E}[R_j]}{\sqrt{\text{Var}[R_i] \text{Var}[R_j]}}, \quad (9)$$

where $\mathbb{E}[R_i]$ is the average return of the instrument i over T considered time units [58, 60, 74]:

$$\mathbb{E}[R_i] = \frac{1}{N} \sum_{t=1}^T R_i(t).$$

If one defines a threshold $\theta \in [-1, 1]$, then an undirected edge between the nodes i and j is added to the graph if the corresponding coefficient $C_{ij} \geq \theta$. Usually, the value of θ is chosen to be significantly larger than zero, and therefore, an edge between two nodes reflects the fact that stocks i and j are significantly correlated.

The values of the correlation threshold θ can be changed in order to construct market graphs where the edges between the nodes reflect different correlation between the corresponding stocks. As the threshold value θ increases, the number of edges in the market graph decreases.

The edge density (as defined in Sect. 3) of the market graph is a measure of the fraction of pairs of stocks exhibiting a similar behavior over some certain time period. One can define different “levels” of this similarity by specifying different

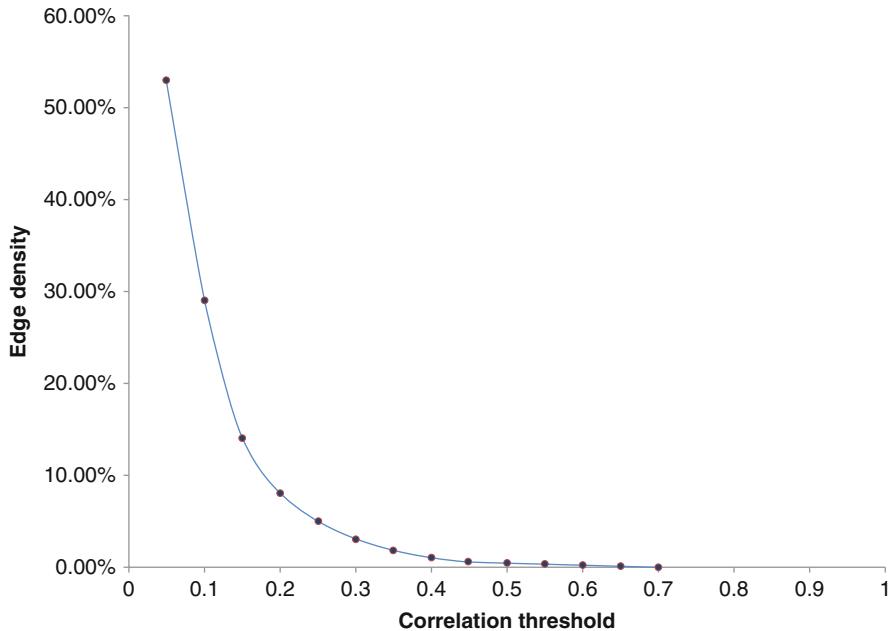


Fig. 21 Edge density of the market graph for different values of the correlation threshold

values of θ . Figure 21 represents the plot of the edge density of the market graph as a function of θ [23].

In [20], the authors also looked at the changes of the edge density of the market graph over time. The authors analyzed these dynamics for 11 overlapping 500-day period in 2000–2002 (the 1st period was the earliest, and the 11th period was the latest). A relatively large value of θ ($\theta = 0.5$) was chosen in order to consider only highly correlated stocks. It turned out that the edge density of the market graph corresponding to the 11th period was more than 8 times higher than for the first period. The corresponding plot is shown in Fig. 22. This jump of the edge density suggests that there is a trend to the globalization of the modern stock market. It means that the number of stocks that significantly affect the behavior of the others was consistently increasing during the considered time period, and it is possible to derive some regularities in the structure of the market.

In order to define the pattern of connections between stocks, the concept of degree distribution defined above in the previous sections can be utilized. It turns out that the degree distribution of the market graph has a well-defined power-law structure [23].

In [22], the authors proposed to relate combinatorial properties of the market graph to some properties of the stock market. For instance, one can apply algorithms for finding cliques or quasi-cliques in the market graph and relate them to groups of highly correlated stocks (the market graph should be constructed using a relatively

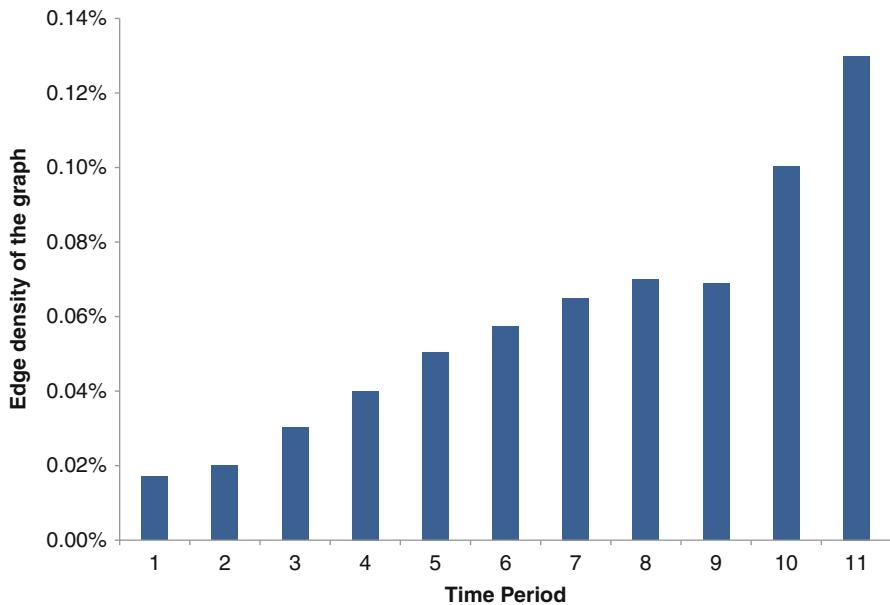


Fig. 22 Evolution of the edge density of the market graph during 2000–2002

large value of θ). A clique and quasi-clique (as defined in Sects. 3.2 and 3.4) of the graph are subsets of the this graph which are complete or “almost” complete graphs itself, respectively (i.e., has all possible edges or needed edge density is satisfied). Also, one can define the minimum clique partition problem for the market graphs, which solution (the minimum number of distinct cliques that the market graph) would represent the minimum number of clusters in the partition of the set of financial instruments.

It is worth mentioning that even though the problem of finding maximum clique in the graph is NP-hard, the exact solutions of this problem were found for different instances of the market graph, which was possible because the market graph is clustered (i.e., it contains dense groups of connected nodes).

Recall from Sect. 3.2 that an independent set is a subset of nodes which are not connected with any other node in this set. Therefore, this notion can be utilized for the market graphs in order to partition these graphs into clusters with stocks whose price fluctuations are not correlated or negatively correlated. It is rather important because finding independent sets in market graphs gives a method of choosing fully diversified portfolios. Partitioning the market graph in such a way that the number of clusters containing distinct diversified portfolios is minimal would represent the optimization problem of partitioning this graph to minimal number of independent sets. The optimal solution to this problem is a chromatic number corresponding to the graph coloring problem. As it was mentioned above, modern stock markets are usually highly correlated, so it is

much more likely to observe a relatively large cliques and quasi-cliques which represent correlated sets of the stocks in the market graph than a large independent sets which represent diversified portfolios in the modern stock market. These results are interesting since they support the idea of the globalization of the stock market.

Overall, finding cliques (as well as quasi-cliques, k -plexes, and other clique relaxations) and independent sets, along with the extensions similar to clique relaxations, in the market graph provides an efficient tool of performing data mining based on the stock market data, that is, partitioning the set of stocks into clusters of “similar” or “dissimilar” objects.

4.6 Social Networks

The process of formation of new communities, their integration and disintegration, attracting new members and their interaction with each other, and growth of the network and its progress, is among the most important research problems in social sciences. Political, religious, professional, and other forms of social organizations provide examples of such communities. A social network is a structure of people related directly or indirectly to each other through a common relation or interest. Therefore, it is very natural to represent these systems as graphs: the nodes are the people, groups, organizations, or other social entities and links represent relationships or flows between the nodes.

A number of recent studies have focused on the statistical properties of social networks. Researchers have concentrated particularly on a few properties that seem to be common to many social networks: the small-world property, power-law degree distribution (which was mentioned earlier in this chapter), and network transitivity. “Small-world effect” is the name given to the finding that the average distance between vertices in a network is short, usually scaling logarithmically with the total number of vertices. A characteristic that many networks have in common is clustering, or network transitivity, which is the property that two vertices that are both neighbors of the same third vertex have a higher probability of also being neighbors of each other. In the language of social networks, two of one’s friends will have a greater probability of knowing each other than two people chosen at random from the population. Another important property that appears to be common in many networks is the property of community structure. Consider the case of social networks of friendships or other acquaintances between individuals. It is intuitively clear that such networks would contain “communities”: subsets of vertices, within which vertex-to-vertex connections are dense, but between which connections are not as dense. A visual sketch of a network with such a community structure is shown in Fig. 23. The ability to detect community structures in a network clearly has practical applications. Detection of community structures can be performed using different approaches. A traditional method is hierarchical clustering that constructs a measure that tells one which edges are most central to communities. An alternative approach to the detection of communities is focused on the edges

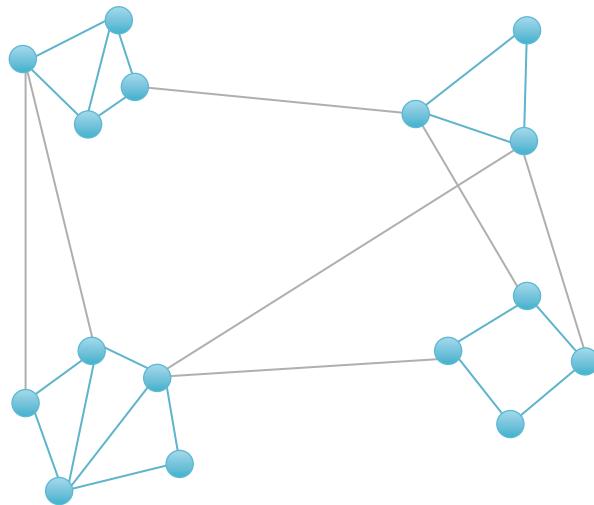


Fig. 23 A representation of a network with community structure. In this network, there are four communities of densely connected nodes (*circles with solid blue lines*), with a much lower density of connections (*gray lines*) between them

that are most “between” communities and strongly connected cores of communities (see [45]). In addition, the aforementioned concepts of clique relaxations (k -cliques, k -clubs, k -plexes) have their origins in the context of communities (tightly knit clusters) in social networks; therefore, efficient combinatorial algorithms for identifying these structures in networks have significant implications in this important application area.

Nowadays, social online communities are growing extensively, and the number of people using such services is increasing very fast; therefore, the necessity of social science research and related data mining techniques is unquestionable. More details on this emerging application of network-based data mining techniques can be found in [62].

5 Conclusion

Network-based data mining and related combinatorial optimization techniques are promising in a variety of applications, which have been demonstrated by multiple recent studies. In this chapter, several graph-theoretic concepts and problems have been discussed in the context of their interpretation from both data mining and combinatorial optimization perspectives. In addition, important real-world application areas have been briefly described, including biological/medical, chemical, telecommunication, financial, and social networks. Clearly, the research in this broad area is far from complete, and new methodological and application aspects will likely continue to be developed in the near future.

Cross-References

- [Combinatorial Optimization in Data Mining](#)
- [Graph Theoretic Clique Relaxations and Applications](#)
- [Small World Networks in Computational Neuroscience](#)

Recommended Reading

1. J. Abello, P.M. Pardalos, M.G.C. Resende, On maximum clique problems in very large graphs, in *External Memory Algorithms* (American Mathematical Society, Providence, 1999), pp. 119–130
2. J. Abello, P.M. Pardalos, M.G.C. Resende (eds.), *Handbook of Massive Data Sets* (Kluwer, Dordrecht, 2002)
3. J. Abello, M.G.C. Resende, S. Sudarsky, Massive quasi-clique detection, in *LATIN 2002: Theoretical Informatics*. Lecture Notes in Computer Science (Springer, Berlin/New York, 2002), pp. 598–612
4. W. Aiello, F. Chung, L. Lu, A random graph model for power law graphs. *Exp. Math.* **10**, 53–66 (2001)
5. W. Aiello, F. Chung, L. Lu, Random evolution in massive graphs, in *Handbook on Massive Data Sets*, ed. by J. Abello, P. Pardalos, M. Resende (Kluwer, Dordrecht, 2002)
6. T. Akutsu, S. Kuhara, O. Maruyama, Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions, in *Proceedings of the 9th Annual ACM-SIAM Symposium Discrete Algorithms* (SODA 1998), San Francisco, CA, 1998, pp. 695–702
7. R.D. Alba, A graph-theoretic definition of a sociometric clique. *J. Math. Sociol.* **6**, 113–26 (1973)
8. R. Albert, A.-L. Barabasi, Statistical mechanics of complex networks. *Rev. Mod. Phys.* **74**, 47–97 (2002)
9. R. Albert, H. Jeong, A.-L. Barabási, Diameter of the World-Wide Web. *Nature* **401**, 130–131 (1999)
10. D. Alderson, Catching the “Network Science” bug: insight and opportunity for the operations researcher. *Oper. Res.* **56**, 1047–1065 (2008)
11. S. Arora, S. Safra, Approximating clique is NP-complete, in *Proceedings of the 33rd IEEE Symposium on Foundations on Computer Science*, Pittsburg, PA, 24–27 Oct 1992, pp. 2–13
12. C.-A. Azencott, A. Ksikes, S.J. Swamidass, J.H. Chen, L. Ralaivola, P. Baldi, One- to four-dimensional kernels for virtual screening and the prediction of physical, chemical, and biological properties. *J. Chem. Inf. Model.* **47**, 965–974 (2007)
13. B. Balasundaram, S. Butenko, Network clustering, in *Analysis of Biological Networks*, ed. by B.H. Junker, F. Schreiber (Wiley, Hoboken, 2008), pp. 113–138
14. B. Balasundaram, S. Butenko, S. Trukhanov, Novel approaches for analyzing biological networks. *J. Comb. Optim.* **10**, 23–39 (2005)
15. B. Balasundaram, S. Butenko, I. Hicks, Clique relaxations in social network analysis: the maximum k-plex problem. *Oper. Res.* **59**(1), 133–142 (2011)
16. A.-L. Barabasi, *Linked* (Perseus Publishing, New York, 2002)
17. A.-L. Barabasi, R. Albert, Emergence of scaling in random networks. *Science* **286**, 509–511 (1999)
18. A.L. Barabasi, N. Gulbahce, J. Loscalzo, Network medicine: a network-based approach to human disease. *Nat. Rev. Genet.* **12**(1), 56–68 (2011)
19. J. Berg, M. Lassig, Local graph alignment and motif search in biological networks. *Natl. Acad. Sci. USA* **101**(41), 14689–14694 (2004)

20. V. Boginski, S. Butenko, P.M. Pardalos, Network-based techniques in the analysis of the stock market, in *Supply Chain and Finance* (World Scientific, Singapore, 2003), pp. 1–14
21. V. Boginski, S. Butenko, P.M. Pardalos, Modeling and optimization in massive graphs, in *Novel Approaches to Hard Discrete Optimization*, ed. by P.M. Pardalos, H. Wolkowicz (American Mathematical Society, Providence, 2003), pp. 17–39
22. V. Boginski, S. Butenko, P.M. Pardalos, On structural properties of the market graph, in *Innovations in Financial and Economic Networks*, ed. by A. Nagurney (Edward Elgar, Cheltenham/Northampton, 2003), pp. 29–45
23. V. Boginski, S. Butenko, P.M. Pardalos, Network models of massive datasets. *Comput. Sci. Inf. Syst.* **1**, 79–93 (2004)
24. V. Boginski, S. Butenko, P.M. Pardalos, Statistical analysis of financial networks. *Comput. Stat. Data Anal.* **48**(2), 431–443 (2005)
25. V. Boginski, S. Butenko, P.M. Pardalos, Mining market data: a network approach. *Comput. Oper. Res.* **33**, 3171–3184 (2006)
26. I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo, The maximum clique problem, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos (Kluwer, Dordrecht, 1999), pp. 1–74
27. P.S. Bradley, U.M. Fayyad, O.L. Mangasarian, Mathematical programming for data mining: formulations and challenges. *INFORMS J. Comput.* **11**(3), 217–238 (1999)
28. S. Brin, L. Page, The anatomy of a large scale hypertextual web search engine, in *Proceedings of the 7th World Wide Web Conference*, Brisbane, Australia, 1998, pp. 107–117
29. A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, J. Wiener, Graph structure in the Web. *Comput. Netw.* **33**, 309–320 (2000)
30. S. Butenko, W.A. Chaovalltwongse, P.M. Pardalos, Clustering challenges in biological networks (World Scientific, New Jersey, 2009)
31. F. Chung, L. Lu, *Complex Graphs and Networks*. CBMS Lecture Series (American Mathematical Society, Providence, 2006)
32. D.J. Cook, L.B. Holder, Graph-based data mining. *IEEE Intell. Syst.* **15**(2), 32–41 (2000)
33. T. Dandekar, S. Schuster, B. Snel, Pathway alignment: application to the comparative analysis of glycolytic enzymes. *Biochem. J.* **343**, 115–124 (1999)
34. J.C. Doyle, D.L. Alderson, L. Li, S. Low, M. Roughan, S. Shalunov, R. Tanaka, W. Willinger, The “robust yet fragile” nature of the internet. *Proc. Natl. Acad. Sci.* **102**(41), 14497–14502 (2005)
35. R. Durbin, S.R. Eddy, A. Krogh, G. Mitchison, *Biological Sequence Analysis* (Cambridge University Press, Cambridge, 1998)
36. V.M. Eguiluz, D.R. Chialvo, G. Cecchi, M. Baliki, A.V. Apkarian, Scale-free structure of brain functional networks. *Phys. Rev. Lett.* **94**, 018102 (2005)
37. D.J. Felleman, D.C. Van Essen, Distributed hierarchical processing in the primate cerebral cortex. *Cereb. Cortex* **1**, 1–47 (1991)
38. T.A. Feo, M.G.C. Resende, A greedy randomized adaptive search procedure for maximum independent set. *Oper. Res.* **42**, 860–878 (1994)
39. T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6**, 109–133 (1995)
40. P. Finn, S. Muggleton, D. Page, A. Srinivasan, Phannacophore discovery using the inductive logic programming system Progol. *Mach. Learn.* **30**, 241–271 (1998)
41. I. Fischer, T. Meinl, Graph based molecular data mining – an overview, in *Proceedings of the 2004 IEEE International Conference on Systems, Man and Cybernetics* (IEEE, Piscataway, 2004), pp. 4578–4582
42. I. Fischer, T. Meinl, Graph based molecular data mining – an overview, in *IEEE International Conference on Systems, Man and Cybernetics*, The Hague, The Netherlands, 2004
43. M.R. Garey, D.S. Johnson, The complexity of near-optimal coloring. *J. ACM* **23**, 43–49 (1976)
44. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness* (Freeman, New York, 1979)
45. M. Girvan, M.E.J. Newman, Community structure in social and biological networks. *Natl. Acad. Sci.* **99**, 7821–7826 (2002)

46. J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.* **182**, 105–142 (1999)
47. B. Hayes, Graph theory in practice. *Am. Sci.* **88**, 9–13 (Part I), 104–109 (Part II) (2000)
48. C.C. Hilgetag, R. Kotter, K.E. Stephen, O. Sporns, Computational methods for the analysis of brain connectivity, in *Computational Neuroanatomy* (Humana, Totowa, 2002)
49. R.A. Jarvis, E.A. Patrick, Clustering using a similarity measure based on shared nearest neighbors. *IEEE Trans. Comput.* **C-22**(11), 1025–1034 (1973)
50. D. Jiang, J. Pei, Mining frequent cross-graph quasi-cliques. *ACM Trans. Knowl. Discov. Data* **2**(4), 1–42 (2009)
51. D. Jiang, C. Tang, A. Zhang, Cluster analysis for gene expression data: a survey. *16*(11), 1370–1386 (2004)
52. H. Kawaji, Y. Yamaguchi, H. Matsuda, A. Hashimoto, A graph based clustering method for a large set of sequences using a graph partitioning algorithm. *Genome Inform.* **12**, 93–102 (2001)
53. H. Kawaji, Y. Takenaka, H. Matsuda, Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics* **20**(2), 243–252 (2004)
54. B.P. Kelley, R. Sharan, R.M. Karp, Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proc. Natl. Acad. Sci. USA* **100**, 11394–11399 (2003)
55. M.G. Kendall, *Rank Correlation Methods* (Griffin, Oxford, 1948)
56. J. Kleinberg, Authoritative sources in a hyperlinked environment. *J. ACM* **46**, 604–632 (1999)
57. E.V. Koonin, R.L. Tatusov, K.E. Rudd, Sequence similarity analysis of Escherichia coli proteins: functional and evolutionary implications. *Proc. Natl. Acad. Sci. USA* **92**, 11921–11925 (1995)
58. L. Laloux, P. Cizeau, J.-P. Bouchad, M. Potters, Noise dressing of financial correlation matrices. *Phys. Rev. Lett.* **83**(7), 1467–1470 (1999)
59. R.D. Luce, Connectivity and generalized cliques in sociometric group structure. *Psychometrika* **15**, 169–190 (1950)
60. R.N. Mantegna, H.E. Stanley, *An Introduction to Econophysics: Correlations and Complexity in Finance* (Cambridge University Press, Cambridge/New York, 2000)
61. H. Matsuda, T. Ishihara, A. Hashimoto, Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theor. Comput. Sci.* **210**, 305–325 (1999)
62. N. Memon, J.J. Xu, L.L. Hicks, H. Chen, *Data Mining for Social Network Data* (Springer, New York/London, 2010)
63. A. Mendelzon, P. Wood, Finding regular simple paths in graph databases. *SIAM J. Comput.* **24**, 1235–1258 (1995)
64. A. Mendelzon, G. Mihaila, T. Milo, Querying the World Wide Web. *J. Digit. Libr.* **1**, 68–88 (1997)
65. Miniwatts Marketing Group, Internet growth statistics (2008), <http://www.internetworldstats.com/emarketing.htm>
66. R.J. Mokken, Cliques, clubs and clans. *Qual. Quant.* **13**, 161–173 (1979)
67. J.M. Murre, D.P. Sturdy, The connectivity of the brain: multi-level quantitative analysis. *Biol. Cybern.* **73**, 529–545 (1995)
68. S. Nijssen, J.N. Kok, A quickstart in frequent structure mining can make a difference. LIACS, Leiden University, The Netherlands, Tech. Rep., April 2004
69. H. Ogata, W. Fujibuchi, S. Goto, A heuristic graph comparison algorithm and its application to detect functionally related enzyme clusters. *Nucleic Acids Res.* **28**, 4021–4028 (2000)
70. P.M. Pardalos, J. Xue, The maximum clique problem. *J. Glob. Optim.* 301–328 (1994)
71. P.M. Pardalos, T. Mavridou, J. Xue, The graph coloring problem: a bibliographic survey, in *Handbook of Combinatorial Optimization*, vol. 2, ed. by D.-Z. Du, P.M. Pardalos (Kluwer, Dordrecht, 1998), pp. 331–395
72. J. Pattillo, A. Veremyev, S. Butenko, V. Boginski, On the maximum quasi-clique problem. *Discrete Appl. Math.* 161(1–2), 244–257 (2013) doi: 10.1016/j.dam.2012.07.019, <http://www.sciencedirect.com/science/article/pii/S0166218X12002843>
73. J. Pei, D. Jiang, A. Zhang, Mining cross-graph quasi-cliques in gene expression and protein interaction data, in *Proceedings of the 21st International Conference on Data Engineering*, Tokyo, 2005, pp. 353–354

74. V. Plerou, P. Gopikrishnan, B. Rosenow, L.A.N. Amaral, H.E. Stanley, Universal and nonuniversal properties of cross correlations in financial time series. *Phys. Rev. Lett.* **83**(7), 1471–1474 (1999)
75. O.A. Prokopyev, V. Boginski, W. Chaovallwongse, P.M. Pardalos, J.C. Sackellares, P.R. Carney, Network-based techniques in EEG data analysis and epileptic brain modeling, in *Data Mining in Biomedicine*, ed. by P.M. Pardalos, V. Boginski, A. Vazacopoulos (Springer, New York, 2007), pp. 559–573
76. L. Ralaivola, J.S. Swamidassa, H. Saigoa, P. Baldi, Graph kernels for chemical informatics. *Neural Netw.* **18**, 1093–1110 (2005)
77. J. Scott, T. Ideker, R.M. Karp, R. Sharan, Efficient algorithms for detecting signaling pathways in protein interaction networks. *J. Comput. Biol.* **13**, 133–144 (2006)
78. S.B. Seidman, B.L. Foster, A graph theoretic generalization of the clique concept. *J. Math. Sociol.* **6**, 139–154 (1978)
79. C. Spearman, The proof and measurement of association between two things. *Am. J. Psychol.* **15**(1), 72–101 (1904)
80. M. Steffen, A. Petti, J. Aach, Automated modelling of signal transduction networks. *BMC Bioinform.* **3**, 34 (2002)
81. P.-N. Tan, M. Steinbach, V. Kumar, *Introduction to Data Mining* (Addison-Wesley, Boston, 2006)
82. A. Veremyev, V. Boginski, Identifying large robust network clusters via new compact formulations of maximum k-club problems. *Eur. J. Obstet. Gyn. R. B.* **218**(2), 316–326 (2012)
83. N. Wale, X. Ning, G. Karypis, Trends in chemical graph data mining, in *Managing and Mining Graph Data* (Springer, New York, 2010), pp. 581–606
84. T. Washio, H. Motoda, State of the art of graph-based data mining. *SIGKDD Explor. Newsl.* **5**(1), 59–68 (2003)
85. W. Willinger, D. Alderson, J.C. Doyle, Mathematics and the internet: a source of enormous confusion and great potential. *Not. Am. Math. Soc.* **56**(5), 286–299 (2009)
86. Q. Yang, S.-H. Sze, Path matching and graph matching in biological networks. *J. Comput. Biol.* **14**(1), 56–67 (2007)
87. S.-H. Yook, Z.N. Oltvai, A.-L. Barabasi, Functional and topological characterization of protein interaction networks. *Proteomics* **4**, 928–942 (2004)
88. Z. Zeng, J. Wang, L. Zhou, G. Karypis, Coherent closed quasi-clique discovery from large dense graph databases, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, PA, 2006, pp. 797–802

Combinatorial Optimization in Transportation and Logistics Networks

Chrysafis Vogiatzis and Panos M. Pardalos

Contents

1	Introduction	674
1.1	Transportation	674
1.2	Logistics	675
1.3	Chapter Overview	676
2	Combinatorial Optimization in Traffic Assignment	676
2.1	Introduction	676
2.2	Branch-and-Bound Methods for Traffic Assignment and Infrastructure Investment	677
2.3	Equilibrium Models	679
2.4	Decomposition Methods	681
2.5	Heuristics and Recent Advancements	686
3	Combinatorial Optimization in Toll Pricing	692
3.1	Introduction	692
3.2	First-Best vs Second Best Frameworks	692
3.3	Modern Optimization Techniques	693
3.4	A Generalized Second-Best Congestion Pricing Model	695
3.5	Hybrid Genetic Algorithms for the Toll Booth and the Toll Pricing Problems	699
4	Vehicle Routing and Vehicle Fleet Management	700
4.1	Introduction	700
4.2	Vehicle Routing Problem	703
4.3	Special Cases and Recent Advancements	711
4.4	Fixed Charge Network Flows	716
5	Conclusion	717
	Recommended Reading	717

C. Vogiatzis (✉) • P.M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: chvogiat@ufl.edu; pardalos@ufl.edu

Abstract

Transportation and logistic networks have always been offering significant practical applications for optimization and operations research techniques. Especially in the last two decades, numerous success stories for large-scale, realistic networks have attracted the interest of the scientific and research society. A typical example of such a success story is the vehicle routing problem, where recent advancements have made it possible for large, complex problems to be solved to optimality. This chapter is designed so as to introduce the reader in the notions tackled by important problems in transportation and logistics engineering and the algorithms that have been devised over the years to solve them. The problems presented and studied in this contribution include the traffic assignment, the vehicle routing problem, and the toll pricing among others.

1 Introduction

1.1 Transportation

Transportation has always been an important society factor. To fully satisfy the needs and the desires of society, it has proven very important to analyze the mathematical models of transportation networks. In order to make the required decisions which help to meet the requirements of an enterprise, a city, and a country, certain optimization techniques have been proposed and explored over the years. In the following sections, the prevalent techniques are described and analyzed.

A formal definition of transportation can be traced back to Steenbrink [109]. Transportation is defined as the transfer of persons and/or goods, in a vehicle or otherwise, between geographically separate places. The collective movement of these vehicles or persons between two places or within an area is called traffic. In order for the movement to take place, the *transportation infrastructure* is used. The infrastructure can be characterized as:

- Fixed objects, describing highways, routes, railways, airports, pipelines, and so on
- Vehicles using the fixed infrastructure
- Organizational infrastructure vital to ensure the well-being and well utilization of the above objects

Let us now focus on the demand for transport. There have been extensive surveys on the models built; however, the bases of the models used in modern transport networks can be found in the works of Hamerslag [50], Overgaard [97], and Steenbrink [109], whereas readers interested in recent advancements based on the models mentioned above are directed to Hensher and Button [59] and Hoogendoorn and Bovy [61]. First of all, it is important to distinguish between the costs and benefits to the trip maker and to those of the society, which are not necessarily the same, as noted by Hamerslag [50] and Henderson and Quandt [58]. When moving from a place-node a to another b , there exist certain benefits (e.g., necessary presence at b) and costs (e.g., time consumption) with that trip which are specific

and particular to the trip maker. However, it can be easily assumed that every one using the transport network for their needs has the objective of maximizing the difference between their benefits and costs as shown in P .

$$P : \max_{m,p,b} u^{ab} - t^{mpab},$$

where

- u^{ab} are the benefits gained by moving from a to b .
- t^{mpab} are the costs incurred when moving from a to b using vehicle (mode) m by route p .

1.2 Logistics

Logistics and logistic networks are widely considered to be closely related to transportation networks. That is why notorious problems such as the traveling salesman or the vehicle routing problems are also tackled in logistic operations research.

One of the first logistic network problems tackled in literature can be tracked back to Fulkerson and Ford [37] and is the famous Hitchcock transportation problem. The statement is pretty simple, yet has immense practical applications:

Given m customers (retailers) and n production plants (factories), find the optimal allocation of shipments that minimize the total costs of transporting the commodities, satisfying the offer and demand restrictions imposed by the production plants and the customers accordingly.

This problem shows the close relation of the fields of transportation and logistics. However, logistics is much more than just optimizing the transportation of commodities and vehicle scheduling. Timetabling is also an important aspect, the problem of determining whether there exists a timetable or schedule that can explicitly satisfy a set of pairing and distributing constraints. Applications extend from the vehicle routing problem and the crew scheduling problem to the most recent ones of sports timetabling [82, 90]. The problem is defined as follows:

- There are N teams and every two teams must play against each other exactly once.
- Hence, the full season will have $N - 1$ game days.
- Every team plays each game day.
- There are also $\frac{N}{2}$ available periods and each period can be used to accommodate one game.
- A team cannot use the same period again in the whole season.

Even though the problem definition is simple and it is true that problems of size less than or equal to 8 can be solved easily by brute force, the combinatorics of the problem increase extremely fast. For that reason, algorithms for the solution of this problem behave poorly and are considered inefficient [47].

In general, logistics engineering and optimization have vast applications, including supply chain management and evacuation planning. In this chapter, the focus is primarily on the transportation network problems involved in logistics such as the vehicle routing problem which is discussed in Sect. 4.

1.3 Chapter Overview

The chapter is structured as follows. First, a small introduction is presented in order to accommodate the reader with useful information on the reasons why transportation and logistics are vital social factors. Then, the traffic assignment problem is described with special focus on the techniques that are widely used to solve the system optimum and the user equilibrium formulations. After the notions of the traffic assignment problem are established, the toll pricing problem is presented along with the recent developments in solving it for large-scale transportation networks. Next, the vehicle routing problem and the vehicle fleet management problems, which are the problems that connect the transportation and logistics frameworks, are presented. Last, conclusions are given so as the reader obtains a better understanding of the state of the art and can apply this information to advance scientific interest in transportation and logistics optimization.

2 Combinatorial Optimization in Traffic Assignment

2.1 Introduction

According to Patriksson [99], there has been an extensive interest in mathematical models and optimization methods in traffic planning for about 45 years now. Operations research techniques have been proposed over the years in order to solve real-life traffic assignment problems, a solution that will provide individual travelers, and the society in general, with a traffic system of a much better performance.

It is important to elaborate first on the notion of congestion which is fundamental to transportation analysis. As traffic volumes grow, the speed on a link of the infrastructure tends to decrease. At first, average speed decreases slowly, however with significant queuing effects tends to decrease more rapidly, as noted by Vickrey [116].

In addition to the above, it is of utmost importance to point out an obvious assumption. Kohl [66] had recognized as soon as 1841 that alternate route choice behaviors are to be encountered concerning different trip makers and their perspective of the shortest one under the current traffic conditions.

In a seminal contribution, which is respected by most transport researchers as the first attempt to state the behavioral assumptions, Wardrop [119] presented the two principles that formalize the notion of user equilibrium (UE) and the alternative

Table 1 Notation used in the branch-and-bound algorithms

Notation	Description
f_{ij}	Traffic flow on arc $(i, j) \in A$
f_a	Traffic flow on arc $a \in A$
t_{ij}	Travel time on arc $(i, j) \in A$
t_a	Travel time on arc $a \in A$
$G = (N, A)$	Transport network G , where N is the set of the nodes and A is the set of the directed arcs
d_{pq}	Demand of travel between two nodes of the graph, p and q
R_{pq}	Set of routes connecting nodes p and q
h_{pqr}	Flow on route $r \in R_{pq}$
c_{pqr}	Travel cost on route r between nodes p and q
\bar{c}_{pqr}	Marginal cost on route r between nodes p and q
π_{pq}	Cost of the shortest route between p and q at equilibrium

of minimizing the total travel costs or system optimal (SO). The two principles are cited by Wardrop:

Wardrop's first principle: "The journey times on all the routes actually used are equal, and less than those which would be experienced by a single vehicle on any unused route."

Wardrop's second principle: "The average journey time is a minimum."

These two principles had already been noted by Pigou [100] in 1920. Moreover, Wardrop's principles will be the central focus of all the algorithms and methods that will be presented in this section of the chapter.

2.2 Branch-and-Bound Methods for Traffic Assignment and Infrastructure Investment

The branch-and-bound optimization technique remains a very useful tool in the hands of optimization researchers in the fields of combinatorial optimization problems. This subsection presents the algorithms presented by Ridley [102] and Chan [21]. These techniques are widely considered to be very important since they were the first attempts to provide useful ideas on how in reality transportation and logistic networks are interconnected. The insight of using optimal flows in order to detect bottlenecked arcs in the transportation network and, therefore, try to invest in these links to provide better quality service was firstly tackled by Ridley and followed by Chan. Their work has immensely contributed in combining the economic world with known combinatorial optimization problems (Table 1).

A good description of the branch-and-bound technique can be found in Lawler and Wood [70] and in Coffman [24]. One of the first researchers to use

branch-and-bound methods for transport networks was Ridley. The formulation of Ridley [102] is the following:

$$\min_{y_{ij}} F = \sum_{ij \in A} f_{ij} t_{ij} \quad (1)$$

$$\text{s.t. } \sum_i f_{ij}^{ab} - \sum_k f_{jk}^{ab} \begin{cases} = 0 & \forall j \neq a \text{ or } b; j \in N \\ = -f^{ab} & \text{if } j=a \\ = f^{ab} & \text{if } j=b \end{cases} \quad (2)$$

$$f_{ij}^{ab} \geq 0, \quad \forall ab \in N, \quad \forall ij \in A \quad (3)$$

$$x_{ij} = \sum_{ab \in N} x_{ij}^{ab}, \quad \forall ij \in A \quad (4)$$

$$x^{ab} = x^{0ab} \quad \forall ab \in N \quad (5)$$

$$\begin{aligned} t^{ai} + t_{ij} &= t^{aj} && \text{if } x_{ij}^{ab} > 0 \\ t^{ai} + t_{ij} &> t^{aj} && \text{if } x_{ij}^{ab} = 0 \end{aligned}, \quad \text{for all } ij \in A, ab \in N \quad (6)$$

$$t_{ij} = a_{ij} - b_{ij} y_{ij} \quad \text{for all } ij \in A \quad (7)$$

$$y_{ij} = 0 \quad \text{for } ij \in \bar{L}_I \quad \text{and } y_{ij} = 0 \text{ or } 1 \quad \text{for } ij \in L_I \quad (8)$$

$$\sum_{ij \in L} y_{ij} i_{ij} \leq I^0 \quad (9)$$

$$i_{ij} = 1 \quad \text{for } ij \in L. \quad (10)$$

Ridley's purpose was to find an investment algorithm that minimizes the time each traveler has to spend while using a link of the infrastructure (denoted as L). Then Eqs. (4)–(6) refer to the network constraints. More specifically, Eq. (4) represents the conservation law in the network flows, (5) ensures the nonnegativity of the number of trips a link can accommodate, and (6) generalizes the algorithm for different sets of commodities. Moreover, Eq. (6) declares the fixed trip matrix, while (7) is the shortest-route principle as shown by Wardrop.

For Eqs. (8)–(10), the reader needs the following notations used:

- y_{ij} —boolean variable which indicates whether an investment is made on link ij or not
- L_I —the set of links where an investment can be made
- \bar{L}_I —the set of links where an investment cannot be made
- I^0 —the budget available for investments.

In the beginning, minimization of F takes no consideration for the budget constraint. Therefore, the initial total investment is equal to I^{\max} . Next, Ridley's approach attempts to minimize the objective function with a virtual budget of $I^{\max} - 1$. This bounding procedure is continued until the true budget I^0 is reached.

In the effort spent to compute the optimal network for a given budget, the method helps find interesting by-products, such as all the intermediate optimal networks for possible budgets $I^0 + 1$ up to I^{\max} . The algorithm can be described as follows:

1. Initialize $m = I^{\max}$.
2. Set $m = m - 1$.
3. Generate all A_m and compute $F^l(A_m)$. Then, compute $F_m^U = F_m^{U(0)}$.
4. For all A_m , check if $F^l(A_m) < F_m^U$. If yes, then compute $F(A_m)$. If this is also less than F_m^U then substitute F_m^U with $F(A_m)$. Otherwise, proceed to the next A_m .
5. If $m = I^0$, then stop. Otherwise return to Step 2.

However, only the bounding process is well developed, thus making this method inefficient. It has thoroughly been studied until recently, providing significant insight to other researchers. Some of them were Ochoa-Rosso [94], Ochoa-Rosso and Silva [95], and Chan [21]. The common drawback these techniques have is the same branch-and-bound feature: a complete assignment is required at every step for the calculation of the total users' costs. That makes the methods mentioned here extremely costly, and it seems prohibitive to use them for even medium sized networks.

2.3 Equilibrium Models

Before proceeding to analyzing the state of the art combinatorial optimization techniques that are used in practice, it is a good idea to consider the most common model formulations. The first mathematical model was provided by Beckmann [9] in 1956 and led to a number of publications on the field. Analytically the topics covered by this very important contribution are presented by Patriksson [99] and Hearn and Florian [38]. According to Florian and Hearn [39], the major formulations are:

1. *Deterministic models*
2. *Stochastic models*

First of all, let us consider the deterministic traffic assignment model. The notation that will be used is presented in [Table 2](#).

It follows clearly that all the flows on paths k add up to the overall demand of the particular origin–destination pair or, more formally,

$$\sum_{k \in K_i} h_k = g_i, \forall i \in I, h_k \geq 0, k \in K. \quad (11)$$

In addition to that, the flow of a specific arc α can be calculated by

$$v_\alpha = \sum_{k \in K} h_k \delta_{\alpha k}, \alpha \in A. \quad (12)$$

Table 2 Notations used for the deterministic mathematical model

Notation	Description
$G = (N, A)$	The transportation network G , where N is the set of the nodes and A is the set of the directed arcs
I	The set of origin–destination pairs in the network
K_i	The set of paths connecting the origin–destination pair i
v	The vector of vehicle flows on the network
v_α	The number of vehicles on a specific arc $\alpha \in A$
$s_\alpha(v)$	The cost of using arc α as a function of vector v
g_i	The demand of the origin–destination pair $i \in I$.
$\delta_{\alpha k}$	Binary variable equal to 1 if arc $\alpha \in A$ and 0 otherwise

Now, by considering Wardrop’s user equilibrium principle, the conclusion that

$$s_k(v^*) - v_i^* \begin{cases} = 0 & \text{if } h_k^* > 0 \\ \geq 0 & \text{if } h_k^* = 0 \end{cases} \quad (13)$$

can be reached.

After several modifications [39], the Pareto optimality condition that is most commonly used in practical applications can be reached as

$$s(v)(v - v^*) - w(g^*)(g - g^*) \geq 0. \quad (14)$$

can be obtained.

Stochastic models are more realistic approaches to the traffic assignment problem, since they take into consideration the fact that travelers systematically make false decisions based on the wrong cost perceptions they assume during their trips. According to Florian and Hearn [39], it is vital to introduce the probability pr_k that an individual will choose path $k \in K_i$ during the trip which is defined as

$$pr_k = pr_k(Z_i), \forall k \in K_i, \forall i \in I, \quad (15)$$

where Z_i represents the vector of perceived times of all the paths k that can be formed for a specific origin–destination pair. These perceived travel times are considered to be represented by a probability density function

$$z_\alpha D(s_\alpha, \theta s_\alpha). \quad (16)$$

Now, the probability that a user will choose path k as they perceive it to be the shortest can be computed as

$$pr_k = Pr[z_k = \min_{k' \in K_i} (z_{k'})], \forall k \in K_i, \forall i \in I. \quad (17)$$

Finally, it has been shown by Evans [35] that the stochastic network optimum model is summarized by the optimization model presented here:

$$\min_v \left(\sum_{i \in I} g_i E\{\min_{k \in K_i} [z_k | s^k(v)]\} \right) + \sum_{\alpha \in A} v_\alpha s_\alpha(v_\alpha) - \sum_{\alpha \in A} \int_0^{v_\alpha} s_\alpha(x) dx \quad (18)$$

subject to the nonnegativity constraints. This objective function is clearly not convex; however, by Evans [35] it has been shown that there exists one stationary point in the neighborhood of which it is strictly convex.

2.4 Decomposition Methods

One important historical reference in the evolution of algorithms for the traffic assignment problem is the Frank–Wolfe method [40]. This method originally appeared in order to solve quadratic problems. Soon though, it became evident that easily it can be suited to solve any pseudoconvex linearly constrained problem such as the traffic assignment as shown by Hearn [56].

The traffic assignment model on which the Frank–Wolfe method can be deployed is the *user equilibrium* model, shown in standard form:

$$\min T(f) = \sum_{a \in A} \int_0^{f_a} t_a(s) ds \quad (19)$$

$$\text{s.t. } \sum_{r \in R_{pq}} h_{pqr} = d_{pq}, \forall (p, q) \in C \quad (20)$$

$$h_{pqr} > 0, \forall r \in R_{pq}, \forall (p, q) \in C \quad (21)$$

$$\sum_{(p,q) \in C} \sum_{r \in R_{pq}} \delta_{pqr\alpha} h_{pqr} = f_a, \forall \alpha \in A. \quad (22)$$

The insight behind the Frank–Wolfe method can be summed up to the following:

- *Initialization:* Let $f^{(0)}$ be an initial feasible solution. Then, set the iteration counter $k = 0$ and $\varepsilon_1, \varepsilon_2$, and ε_3 to be positive parameters.
- *Linearization of the objective:* Approximation of the objective function by a first-order Taylor series
- *Lower-bound search:* Solve the linear program presented in Eqs. (3)–(6). The solution $\hat{y}^{(k)}$ yields a new lower bound (LBD).
- *Check criterion:* If $\frac{T(f^{(k)}) - LBD}{LBD} < \varepsilon_1$, then the method terminates.

Table 3 Notations used for the Frank–Wolfe method

Notation	Description
f_{ij}	Traffic flow on arc $(i, j) \in A$
f_a	Traffic flow on arc $a \in A$
t_{ij}	Travel time on arc $(i, j) \in A$
t_a	Travel time on arc $a \in A$
$G = (N, A)$	Transport network G , where N is the set of the nodes and A is the set of the directed arcs
C	Set of commodities, characterized as pairs of origins and destinations (O–D pairs)
d_{pq}	Demand of travel between two nodes of the graph, p and q
R_{pq}	Set of routes connecting nodes p and q
h_{pqr}	Flow on route $r \in R_{pq}$
c_{pqr}	Travel cost on route r between nodes p and q
\bar{c}_{pqr}	Marginal cost on route r between nodes p and q
π_{pq}	Cost of the shortest route between p and q at equilibrium

- *Solve the line search problem:* The line search problem can be described as follows:

$$\min_{l \in [0,1]} \bar{T}(l) = T(f^{(k)} + l(\hat{y}^{(k)} - f^{(k)})) \quad (23)$$

in order to find the $l^{(k)}$.

- *Next iteration or stop criterion:* For the new iteration, update the solution-flow

$$f^{(k+1)} = f^{(k)} + l^{(k)}(\hat{y}^{(k)} - f^{(k)}).$$

In the case of nonlinear network flows, the Frank–Wolfe method can be efficiently used in order to decompose the problem of traffic assignment into $|C|$ shortest-route problems, which are more easily tackled (Table 3). The costs considered during the shortest-route problems are for each arc:

$$\frac{\partial}{\partial t} T(f^{(k)}) = t_\alpha(f_\alpha^{(k)}), \quad (24)$$

which present the travel times of each arc as perceived by the traveler. Thus, it is easy to see that this method resembles a real-life iterative procedure where the decision makers of the travel routes chosen adjust their decisions based on the prevailing conditions.

The method has been successfully applied to transportation management by several researchers during the 1970s and 1980s. The most noted ones are LeBlanc et al. [72–75], Golden [45, 46], and Nguyen [91, 92]. This research has enabled the

method to reach commercial software designed and programmed to solve instances of traffic flow networks, such as the TRAFFIC by Nguyen and James [93] and EMME/2 by Babin et al. [4].

However, not everything is satisfactory as far as the efficiency of the algorithm in traffic assignment problems is concerned, a result that was early extracted by Zangwill in 1969 [122]. Zangwill proved that the convergence rate of the Frank–Wolfe method can be quite slow. Although this has rendered the method inapplicable to practical applications, still transportation researchers are developing more efficient algorithms based on the method.

That is the main reason why researchers went on to propose several more efficient decomposition methods, mostly simplicial and cyclic, for solving the traffic assignment problem. The first transportation researcher who discussed the simplicial decomposition is Murchland [86], followed by Holloway [60] and Cantor and Gerla [18] who were the pioneers for implementing a simplicial decomposition algorithm for nonlinear commodity flows. The method is common in computer network optimization as the *extremal flows method*.

In general, simplicial decomposition is based on the Caratheodory theorem, stating that any point in the convex hull of a set X can be represented as a convex combination of at most $\dim(X)+1$ points and directions. Therefore, if the set of the extreme points is denoted as $\hat{y}^{(k)}$ and as $\hat{z}^{(j)}$ the corresponding set of directions of the polyhedron X for all $k = 1, \dots, m$ and $j = 1, \dots, l$, then any point $\bar{x} \in X$ can be represented as

$$\bar{x} = \sum_{k=1}^m \bar{\lambda}^{(k)} \hat{y}^{(k)} + \sum_{j=1}^l \bar{\mu}^{(j)} \hat{z}^{(j)} \quad (25)$$

where

$$\sum_{k=1}^m \bar{\lambda}^{(k)} = 1$$

and

$$\bar{\lambda}^{(k)}, \bar{\mu}^{(j)} \geq 0, \forall k, j.$$

When applying simplicial decomposition to a problem, the same problem as in the Frank–Wolfe method is actually being solved, that is,

$$\hat{y}^{(k)} \in \arg \min_{y \in X} \nabla T(x^{(k)})^T y. \quad (26)$$

In the Frank–Wolfe method, the next step would be to find the solution to the line search problem, as described in a previous paragraph. This is where the simplicial decomposition method differs: all the extreme points and directions obtained by

Table 4 Notations used for the simplicial decomposition

Notation	Description
$G = (N, A)$	The transportation network G , where N is the set of the nodes and A is the set of the directed arcs
I	The set of origin–destination pairs in the network
K_i	The set of paths connecting the origin–destination pair i
v	The vector of vehicle flows on the network
v_α	The number of vehicles on a specific arc $\alpha \in A$
$s_\alpha(v)$	The cost of using arc α as a function of vector v
g_i	The demand of the origin–destination pair $i \in I$
$\delta_{\alpha k}$	Binary variable equal to 1 if arc $\alpha \in A$ and 0 otherwise

previous iterations of the method are stored. Then, a master problem is solved using simplex. The master problem would be:

$$\min T(\bar{x}) \text{ s.t. the conditions of Eq. (20).}$$

The solution of the master problem leads to a new iteration point $x^{(k+1)}$.

Very important publications in the scientific field of simplicial decomposition are considered the ones by Hearn et al. [51, 52, 113], where a special version of the decomposition is shown. More specifically, the maximum number of extreme points stored is specified by the user as $r \leq |A| + 1$. That way, Hearn et al. managed to prove convergence within finite number of steps if r is greater than the optimal face dimension. Therefore, the result helped store less extreme points, which in turn implies that the number of variables required decreases. Von Hohenbalken [117] had noted that when the transportation network is rather small, then even Newton-type methods can converge quickly to an optimal solution. The drawback of the restricted simplicial decomposition algorithm proposed by Hearn is that the optimal choice for parameter r is unknown a priori, since the dimension of the optimal face is also unknown.

A restricted simplicial decomposition method is presented in [39]. The notation used can be found at **Table 4**. The algorithm begins ($k := 0$) from a feasible starting point (let v_0) and sets $\Theta_x = \emptyset$, $\Theta_v = v_0$. At each iteration the following problem is solved:

$$\min \sum_{\alpha \in A} (s_\alpha(v_\alpha^k)) x_\alpha \quad (27)$$

$$\text{s.t. } \sum_{k \in K_i} x_k = g_i, \quad i \in I, \quad x_k \geq 0, \quad k \in K \quad (28)$$

$$z_\alpha = \sum_{k \in K} \delta_{\alpha k} x_k. \quad (29)$$

The solution x_k is then checked. If

$$s(u^k)(x^k - v^k) \geq 0 \quad (30)$$

then an optimal solution has been reached as shown in Eq. (14). Otherwise, proceed to update the sets Θ_z and Θ_v . The updating process is as follows:

- If $|\Theta_x^k| < q$
 $\Theta_x^{(k+1)} = \Theta_x \cup x^k$ and $\Theta_v^{(k+1)} = \Theta_v^k$.
- If $|\Theta_x^k| = q$
Replace the minimally weighted extreme point with the solution found x_k to update the set $\Theta_x^{(k+1)}$ and $\Theta_v^{(k+1)} = v^k$.

It is clear to all transportation researchers that decomposition of real-life large-scale traffic assignment problems is more than vital. As it has been discussed by the author, the Frank–Wolfe method has been proven to achieve tractability while remaining economic resource-wise. However, its behavior near the optimal solution and its convergence rate made it an unlikely candidate for practical problems. Therefore, researchers have been looking for years for algorithms of much smaller dimension (i.e., requiring less resources) and which manage to preserve the original problem's special structure. The subproblems produced are then to be solved iteratively, while the rest of the problem, including its values and parameters, are discarded.

Because of the independence between commodity flows, they have been the natural choice for approaching a decomposition. This happens since there exist efficient algorithms for solving single commodity network flows. Let us consider again now the subproblem shown in Eq. (20), where $X = \{x : Ax = b; x \geq 0\}$, and let also $x = (x_1, x_2, \dots, x_n)$ be a variable vector where $x_i \in X_i$ is the feasible region for each x_i . There are two decomposition methods for the solution of the subproblem of Eq. (20). They are:

- *Gauss–Seidel*
- *Jacobi*

The subproblems that are solved according to each of the methods mentioned are:

- *Gauss–Seidel*

$$x_i^{(k+1)} \in \arg \min_{x_i \in X_i} T(x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{i-1}^{(k)}, x_i^{(k)}, x_{i+1}^{(k)}, \dots, x_n^{(k)}). \quad (31)$$

This approach is regarded as the *method of successive replacements* and *cyclic decomposition*, because the newest information from the most recent subproblems is used in the subproblem i .

- *Jacobi*

$$x_i^{(k+1)} \in \arg \min_{x_i \in X_i} T(x_1^{(k)}, x_2^{(k)}, \dots, x_{i-1}^{(k)}, x_i^{(k)}, x_{i+1}^{(k)}, \dots, x_n^{(k)}). \quad (32)$$

In contrast to the previous technique, this one is called the *method of simultaneous replacements*.

The convergence rates of the algorithms based on those decomposition techniques, as well as their special structure for the interested reader, can be found in the studies by Bertsekas and Tsitsiklis [12].

The application of cyclic decomposition methods in traffic assignment was first encountered in Dafermos [26] and has, thereafter, been successfully applied [69]. Moreover, scientific research on cyclic decomposition offered the bases on which modern heuristics advancements were founded. These subjects are treated in the following sections.

2.5 Heuristics and Recent Advancements

As the problem sizes increase, the traffic assignment becomes computationally intractable. This leads to approaches that attempt to tackle the problem by detecting a near-optimal or even just a feasible solution. Heuristics that are famous for their capability to solve hard discrete problems have been implemented in order to solve the traffic assignment problem. These heuristics include:

- Greedy randomized adaptive search procedure (GRASP)
- Simulated annealing
- Tabu search
- Genetic algorithms

Especially genetic algorithms, coupled with a random search procedure, seem to be obtaining high quality solutions in reasonable computational times.

Based on the previous work of Merchant and Nemhauser [83, 84] and Carey [19, 20], Waller and Ziliaskopoulos presented the DUO algorithm in 2006 [120] for dynamic traffic assignment. Their combinatorial approach required the discretization of the network arcs into smaller segments chosen so as to prevent a vehicle of traversing more than one “cell” at a single time unit. Then, they divided the aforementioned cells into five subcategories:

- Ordinary
- Merging
- Diverging
- Source
- Sink

More information on the cell transmission model that the authors adopted can be found at the work of Daganzo [27, 28].

To sum up, the algorithm they devised is at first calculating all the time-dependent shortest paths from every origin to the destination and then choosing the one shortest path with the smallest expected time of arrival and assigning it to a vehicle. Then, the capacities along the links of that path are decreased. In case now there exist more vehicles to be routed, the method is repeated. Otherwise, the allocation of vehicles to paths is optimal.

The algorithm is simple and can be implemented in an easy way. It is, also, very efficient since it is taking into consideration the special structure of the problem. However, it is limited to work under the assumption that there is only one destination in the network.

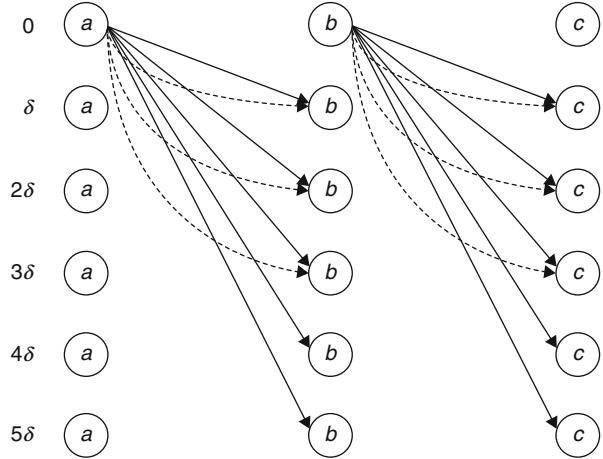
Another important advancement in the last years is the attempts to represent the problems arising in a more realistic approach. In their work, Zheng and Arulselvan [123] take into consideration the notion of the managed lanes and the dynamic and continuous nature of traffic assignment and try to implement techniques that make use of these special characteristics. According to the authors, managed lanes are lanes with restricted access to a subset of the vehicles currently using the infrastructure. So the lanes in a traffic system can be, more easily, distinguished into two major categories:

- Unmanaged (unrestricted, free) lanes
- Managed (restricted) lanes

In their work, they use the time-expanded network and the dynamic traffic assignment model and formulation presented in the work of Nahapetyan and Langpopanich [88]. In that, the planning time horizon is continuous, but is discretized into a set $\Gamma = \{0, \delta, 2\delta, \dots, C\delta\}$. Then, the time-expanded network can be modeled by copying each node in the original static network $C + 1$ times. One such time-expanded network can be found in Fig. 1.

G	The set of all group users, defined by the set (a, b, r, d)
(a, b)	The origin–destination pair
r	The desired starting time of the trip
d	The expected arriving time of the trip
$\Delta_{p,q}$	The set of all possible discrete travel times of path (p, q)
$M_{p,q}$	The capacity of the path (p, q)
$\phi()$	Penalty function for late/early departure
$\psi()$	Penalty function for late/early arrival
$r_{p,q}$	Maximum outflow rate for arc (p, q)
$h_{a,b,r,d}$	The demand for each of the groups $(a, b, r, d) \in G$
$v_t^{(a,b,r,d)}$	Vehicle flow starting at time t for group (a, b, r, d)
$w_t^{(a,b,r,d)}$	Vehicle flow arriving at time t for group (a, b, r, d)
$y_{p_t, q_{t+\tau}}^{(a,b,r,d)}$	The flow on the expanded arc $(p_t, q_{t+\tau})$ for group (a, b, r, d)
D_1, D_2, D_3	The arc-node incidence matrices
$\Gamma_{p,q}$	The set of all possible times of departure for arc (p, q)

Fig. 1 An example of a time-expanded network



Finally, the formulation of the problem based on the notation presented and the notions behind the time-expanded network, described previously, is the following:

$$\min \sum_{(a,b,r,d) \in G} \left\{ \sum_s \phi(s-r) w_s^{(a,b,r,d)} \right\} \quad (33)$$

$$+ \sum_s \psi(s-r) v_s^{(a,b,r,d)} \quad (34)$$

$$+ \sum_{(p,q) \in A} \sum_{t \in \Gamma_{p,q}} \phi_{p,q}(x_t^{p,q}) \sum_{\tau \in \Delta_{p,q}} y_{p_t, q_{t+\tau}}^{(a,b,r,d)} \quad (35)$$

$$\text{s.t. } \sum_t w_t^{(a,b,r,d)} = h_{(a,b,r,d)}, \quad \forall (a,b,r,d) \in G \quad (36)$$

$$\sum_t v_t^{(a,b,r,d)} = h_{(a,b,r,d)}, \quad \forall (a,b,r,d) \in G \quad (37)$$

$$D_1 w + D_2 y + D_3 v = 0 \quad (38)$$

$$x_t^{p,q} = \sum_{(a,b,r,d) \in G} \sum_{\alpha \leq t, \beta \geq t} \frac{\beta - t}{\beta - \alpha} y_{p_\alpha, q_\beta}^{(a,b,r,d)}, \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (39)$$

$$x_t^{p,q} \geq \sum_{\tau \in \Delta_{p,q}} \phi_{p,q}^{-1}(\tau - \delta) z_{p_t, q_{t+\tau}}, \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (40)$$

$$x_t^{p,q} \leq \sum_{\tau \in \Delta_{p,q}} \phi_{p,q}^{-1}(\tau) z_{p_t, q_{t+\tau}}, \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (41)$$

$$\sum_{(a,b,r,d) \in G} y_{p_t,q_{t+\tau}}^{(a,b,r,d)} \leq M_{p,q} z_{p_t,q_{t+\tau}}, \quad \forall \tau \in \Delta_{p,q}, t \in \Gamma_{p,q}, (p,q) \in A \quad (42)$$

$$\sum_{\tau \in \Delta_{p,q}} z_{p_t,q_{t+\tau}} = 1, \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (43)$$

$$\sum_{(a,b,r,d) \in G} \sum_{\tau \in \Delta_{p,q}} y_{p_{t-\tau},q_t}^{(a,b,r,d)} \leq r_{p,q} \delta, \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (44)$$

$$v, w, x, y \geq 0 \quad (45)$$

$$z_{p_t,q_{t+\tau}} \in \{0, 1\}, \quad \forall \tau \in \Delta_{p,q}, t \in \Gamma_{p,q}, (p,q) \in A. \quad (46)$$

Constraints (36)–(38) are ensuring that the flow in the network is balanced after the time expansion modification. Then, the weighted flows are modeled by the set of constraints in Eq. (39), while the travel times are taken care of by Eqs. (40)–(41). Last, the uniqueness of the travel time for each static arc and the capacity constraints are ensured by (42)–(43) and (44) accordingly.

The authors then proceed to introduce a formulation that can be decomposed into a master problem and a series of relaxed subproblems. This way, they can exploit a series of properties that they prove the new problems possess in order to show the effectiveness of their heuristic approach. The master problem is defined in Eqs. (47)–(49):

$$\min H(z) = \min \{c_1^T w + c_2^T v + c_3^T y | (w, v, y) \in S(z)\} \quad (47)$$

$$\text{s.t. } \sum_{\tau \in \Delta_{p,q}} z_{p_t,q_{t+\tau}} = 1 \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (48)$$

$$z_{p_t,q_{t+\tau}} + z_{p_s,q_{s+\tau}} \leq 1 \quad \forall t < s, s > \beta, (p,q) \in A, \quad (49)$$

while the relaxed subproblem is presented in Eqs. (50)–(59):

$$\tilde{H}(z^k) = \min c_1^T w + c_2^T v + c_3^T y + m^T \gamma \quad (50)$$

$$\text{s.t. } \sum_t w_t^{(a,b,r,d)} = h_{(a,b,r,d)} \quad \forall (a,b,r,d) \in G \quad (51)$$

$$\sum_t v_t^{(a,b,r,d)} = h_{(a,b,r,d)} \quad \forall (a,b,r,d) \in G \quad (52)$$

$$D_1 w + D_2 y + D_3 v = 0 \quad (53)$$

$$x_t^{p,q} = \sum_{(a,b,r,d) \in G} \sum_{\alpha \leq t, \beta \geq t} \frac{\beta - t}{\beta - \alpha} y_{p_\alpha, q_\beta}^{(a,b,r,d)} \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (54)$$

$$x_t^{p,q} \geq \sum_{\tau \in \Delta_{p,q}} \phi_{p,q}^{-1}(\tau - \delta) z_{p_t,q_{t+\tau}}^k, \quad \forall t \in \Gamma_{p,q}, (p,q) \in A \quad (55)$$

$$x_t^{p,q} \leq \sum_{\tau \in \Delta_{p,q}} \phi_{p,q}^{-1}(\tau) z_{p_t, q_t + \tau}^k, \quad \forall t \in \Gamma_{p,q}, (p, q) \in A \quad (56)$$

$$\sum_{(a,b,r,d) \in G} y_{p_t, q_t + \tau}^{(a,b,r,d)} \leq M_{p,q} z_{p_t, q_t + \tau}^k, \quad \forall \tau \in \Delta_{p,q}, t \in \Gamma_{p,q}, \\ (p, q) \in A \quad (57)$$

$$\sum_{(a,b,r,d) \in G} \sum_{\tau \in \Delta_{p,q}} y_{p_t - \tau, q_t}^{(a,b,r,d)} \leq r_{p,q} \delta, \quad \forall t \in \Gamma_{p,q}, (p, q) \in A \quad (58)$$

$$v, w, y, \gamma \geq 0 \quad (59)$$

The heuristic approach adopted needs some operations that are defined by the authors and are the following:

- *Shrinking* reduces the carrying capability of the expanded network associated with the value of z^k . It is defined by Eqs. (60)–(62):

$$z_{\hat{p}_u, \hat{q}_{u+\tau}}^{k+1} = \mathbf{1}_{[\tau=\pi-\delta]} \quad \forall \tau \in \Delta_{\hat{p}, \hat{q}} \quad (60)$$

$$z_{\hat{p}_t, \hat{q}_{t+\tau}}^{k+1} = z_{\hat{p}_t, \hat{q}_{t+\tau}}^k \quad \forall \tau \in \Delta_{\hat{p}, \hat{q}}, t \in \Gamma_{\hat{p}, \hat{q}} / \{u\} \quad (61)$$

$$z_{p_t, q_{t+\tau}}^{k+1} = z_{p_t, q_{t+\tau}}^k \quad \forall \tau \in \Delta_{p,q}, t \in \Gamma_{p,q}, (p, q) \in A / (\hat{p}, \hat{q}). \quad (62)$$

- *Expanding* increases the carrying capability of the network of z^k and is described in (63)–(65):

$$z_{\hat{p}_u, \hat{q}_{u+\tau}}^{k+1} = \mathbf{1}_{[\tau=\pi+\delta]} \quad \forall \tau \in \Delta_{\hat{p}, \hat{q}} \quad (63)$$

$$z_{\hat{p}_t, \hat{q}_{t+\tau}}^{k+1} = z_{\hat{p}_t, \hat{q}_{t+\tau}}^k \quad \forall \tau \in \Delta_{\hat{p}, \hat{q}}, t \in \Gamma_{\hat{p}, \hat{q}} / \{u\} \quad (64)$$

$$z_{p_t, q_{t+\tau}}^{k+1} = z_{p_t, q_{t+\tau}}^k \quad \forall \tau \in \Delta_{p,q}, t \in \Gamma_{p,q} / \{u\}, (p, q) \in A / (\hat{p}, \hat{q}). \quad (65)$$

After a solution has been obtained by the relaxed subproblem, a set $\Omega = \{(t, p, q)\}$ is created with the candidate time-expanded arcs. The basic operations of shrinking and expanding are then performed to detect if a better solution can be obtained.

So the heuristic can be summed up to the following. First, an initial binary solution z^0 which has the lowest carrying capacity. This happens because of the fact that all time-expanded arcs are associated with the least travel times. However, this initial solution can very well be infeasible, hence an expanding operation as described above needs to take place. When feasibility is achieved, implying that now the constraint (55) is binding, then a better solution can be searched for hence requiring the execution of a shrinking operation. According to the state of constraints (55)–(56), different actions are to be taken (Fig. 2). In order to deal with this fact, the authors decided to partition the set $\Omega = \{(t, p, q) | t \in \Gamma_{p,q}, (p, q) \in A\}$ into a number of different subsets defined below:

```

procedure DTA-UB Heuristic
1   Set  $z^0$  as defined in equation (21)
2   Solve RSP0 associated with  $z^0$ , and get the solution  $(w^0, v^0, x^0, y^0, \gamma^0)$ .
3   Set the incumbent solution,  $(\bar{w}, \bar{v}, \bar{x}, \bar{y}, \bar{\gamma}) = (w^0, v^0, x^0, y^0, z^0, \gamma^0)$ .
4   Update  $\Omega_0, \Omega_1, \Omega_2, \Omega_3$ , and  $n = |\Omega_1|$ .
5   if  $n = 0$  then
6       Terminate and return the incumbent solution.
7   end if
8   set  $c = 0$ 
9   white !(improvement ||  $c > 4$ )
10  if  $c = 1 \parallel c = 2$ 
11      Call ExpandGraph( $\Omega_c$ )
12  else
13      Call ShrinkGraph( $\Omega_c$ )
14  end if
15   $c = c + 1$ 
16  end while
17 if  $c > 4 \&&$  improvement then
18   if  $n > 0$  then
19       No solution is found
20   else
21       Terminate and return the incumbent solution
22   end if
23 else
24    $(\bar{v}, \bar{w}, \bar{x}, \bar{y}, \bar{z}, \bar{\gamma}) = (\tilde{v}, \tilde{w}, \tilde{x}, \tilde{y}, \tilde{z}, \tilde{\gamma})$ 
25   Go to Step 4.
26 end if
27 if  $c = 3 \parallel c = 2$  then
28    $(\bar{v}, \bar{w}, \bar{x}, \bar{y}, \bar{z}, \bar{\gamma}) = (\tilde{v}, \tilde{w}, \tilde{x}, \tilde{y}, \tilde{z}, \tilde{\gamma})$ 
29   Call ShrinkGraph( $\Omega_3$ )
30   if improvement then
31    $(\bar{v}, \bar{w}, \bar{x}, \bar{y}, \bar{z}, \bar{\gamma}) = (\tilde{v}, \tilde{w}, \tilde{x}, \tilde{y}, \tilde{z}, \tilde{\gamma})$ 
32   end if
33   Go to Step 4.
34 end if
35 if  $c = 1$  then
36    $(\bar{v}, \bar{w}, \bar{x}, \bar{y}, \bar{z}, \bar{\gamma}) = (\tilde{v}, \tilde{w}, \tilde{x}, \tilde{y}, \tilde{z}, \tilde{\gamma})$ 
37   Go to Step 4.
38 end if
end procedure DTA – UB Heuristic

```

Fig. 2 The heuristic introduced by Zheng and Arulselvan [123]

$$\begin{aligned}
\Omega_0 &= \{(t, p, q) | \bar{x}_t^{p,q} = \sum_{\tau \in \Delta_{p,q}} \phi_{p,q}^- 1(\tau - \delta) \bar{z}_{p_t, q_{t+\tau}}, (t, p, q) \in \Omega\} \\
\Omega_1 &= \{(t, p, q) | \bar{y}_t^{p,q} > 0, (t, p, q) \in \Omega\} \\
\Omega'_2 &= \{(t, p, q) | \bar{\mu}_t^{p,q} < 0, (t, p, q) \in \Omega\} \\
\Omega_2 &= \Omega'_2 \setminus \Omega_1 \\
\Omega_3 &= \Omega \setminus \Omega'_2,
\end{aligned}$$

where $\bar{\mu}_t^{p,q}$ is the optimal dual multiplier of constraint (56).

3 Combinatorial Optimization in Toll Pricing

3.1 Introduction

As noted in the previous section, Wardrop's principles [119] have played a very important role in shaping transportation engineering. The two principles seem and are incompatible with each other in practice. The insight to have a solution that is simultaneously system and user optimal is what led to more advanced techniques in toll pricing.

One of the main objectives of transportation engineering is to figure ways to achieve unique solutions in order to solve both problems of traffic assignment and toll pricing. The second problem is clearly solved in conjunction with data derived from the traffic assignment problem and consists of the determination of fares and tariffs that would lead the users to select a network optimal route rather than a "selfish" route.

In the last years, there has been an extensive interest in the toll pricing problem due to the recurrent congestion that appears especially in urban and metropolitan areas. As was noted by Arnott and Small [2], trying to maximize a driver's convenience may result in deteriorating the overall congestion level.

In addition to the above, in the last two decades, a significant change in federal management of transportation is noticed [121]. Instead of focusing on construction and upgrading infrastructure, the key to successful utilization is now lying with the optimal management of the existing infrastructure. Hence, it is implied that new policy makers are actually trying to improve efficiency. One way to do so is by deciding on a congestion pricing scheme that leads to a system optimal solution, through appropriate toll pricing.

Numerous practical applications of toll pricing schemes have been presented along the years. The most noted ones in literature are according to Tsekeris and Voß [111]:

- Travel demand and congestion management
- Toll revenues used to construct or improve infrastructures
- Environmental management

Especially important is congestion management. A big variety of European cities have adopted tolls as a means to control and improve congestion levels. The studies in literature usually focus on optimizing toll fares in London [48, 71, 103] and Stockholm [1, 33, 79].

3.2 First-Best vs Second Best Frameworks

The first-best principle in toll pricing is based on the cost of marginal social cost [65]. More specifically, it consists of the unconstrained marginal-cost pricing, where the cost for the traveler is equal to the marginal cost of the road use as perceived by the infrastructure manager plus the cost incurred because of the delay imposed to the rest of the network users.

The notion of a second-best solution is becoming more and more applicable as the problems become more large scale and realistic [81]. As was noted by Luenberger [78], the optimal solution for a toll pricing problem would be to price all possible links on the network. As the reader can immediately realize, this approach is highly ineffective and expensive. Therefore, recent studies in the past 15 years have focused on second-best approaches for toll pricing. In addition to that, new optimization problems have been proposed, such as the toll booth location problem, which take into consideration the minimization of the number of toll booths installed in a network.

The insights of first-best toll pricing stated by Pigou in 1920 [100] are recognized by all researchers to be [114] the best benchmark available for toll pricing. However, it is also a fact that the Pigou assumptions are unrealistic, meaning that:

- The regulator is able to place toll booths in all arcs of a network.
- All network users and the regulator have full and accurate knowledge of the traffic and infrastructure conditions at all times.

From the economic point of view, the assumptions made are strong; hence, unconstrained maximization ends up being unrealistic and therefore only used as a benchmark and often infeasible solution.

That is the main reason why second-best toll pricing policies have prevailed in literature receiving much scientific interest. For instance, as early as 1968, Lévy-Lambert [76] studied the second-best regulation problem concerning the problem where there exist two routes parallel to each other; however, one of them is untolled.

Increasingly used are some game-theoretic modelings of the toll pricing problem. Because of the large-scale nature of a typical transportation network, there have appeared noncooperative game theoretic models [111] where management is considered to come from a higher federal power. Examples of such models include but are not limited to the works of Chau and Sim [22], Correa et al. [25], Roughgarden [104], and Roughgarden and Tardos [105]. The above models consider a set of selfish players that aim to minimize their total costs while also minimizing the price of anarchy [22]. The price of anarchy is calculated by measuring the network inefficiency levels when the users are let to decide their own selfish routes compared to the ones imposed by a central management system.

3.3 Modern Optimization Techniques

Recent computational methods for toll pricing in order to control and adapt congestion levels in metropolitan areas have been studied extensively by Hearn et al. [53]. Especially in the journal article published in 2002 by Hearn et al. [54], they focus on a fixed demand toll pricing framework and benchmark two approaches. The first method is based on a toll set approximation, while the second one makes use of a cutting plane methodology ([Table 5](#)).

Then, using the principles of traffic assignment presented in [Sect. 2.3](#), which state that a feasible network flow v is tolled-user optimal if and only if

Table 5 Notations used in the toll pricing methodology presented

Notation	Description
G	The transportation network
N	The set of nodes
A	The set of arcs
p	An origin node
q	A destination node
K	The set of all $k = (p, q)$ representing o-d pairs
\bar{t}_k	The demand of a specific o-d pair k
x_k	The commodity flow of o-d pair k
v	The aggregate flow vector
$s(v)$	The cost function for the current flow
$s_\alpha(v)$	The cost function for a specific link $a \in A$

$$(s(\bar{v}) + \beta)^T (v - \bar{v}) \geq 0, \forall v \in V \quad (66)$$

The authors begin by using the very useful lemma proved by Hearn and Ramana [55]:

Lemma 1 For a given optimal system optimum solution flow v^* , let T be the set of all tolls that ensure that $v^* \in V$ is also a tolled user equilibrium optimal flow. Then, $T = W_{FD}(v^*)$ is the polyhedron given by the β variable part of the following system

$$s(v^*) + \beta \geq A^T \rho^k \quad \forall k \in K \quad (67)$$

$$(s(v^*) + \beta)^T v^* = \sum_{k \in K} \bar{t}_k E_k^T \rho^k. \quad (68)$$

Then, by imposing other constraints or choosing different objective functions an alternative tolls set can be obtained. The most common objective functions used can be summed up to the following:

- Minimize the total tolls collected (**MINSYS**)
- Minimize the number of toll booths installed
- Minimize the largest toll fare collected

The MINSYS problem as studied by Hearn and Ramana [55] is then transformed by penalizing the aggregate complementarity constraint in order to find an approximate toll vector solution. The problem becomes then

$$\min u_1 Z + u_2 ((s(v^*) + \beta)^T v^* - b^T \rho) \quad (69)$$

$$\text{s.t. } s(v^*) + \beta \geq A^T \rho^k, \quad \forall k \in K \quad (70)$$

$$(\beta, \rho, y, z) \in \hat{T}, \quad (71)$$

where u_1 and u_2 are positive scalars and Z is the objective of the MINSYS problem, that is, $\beta^T v$ and \hat{T} are the constraints that $\beta \geq 0$.

After these notes, the authors proceed to formalize their algorithmic idea. It can be described as follows:

1. Solve the system optimal problem to obtain an approximate optimal flow v^* and the corresponding cost $s(v^*)$.
2. Solve the MINSYS problem presented above by penalizing the complementarity constraint in Eq. (68). Let the violation be denoted by

$$\bar{\varepsilon} = (s(v^*) + \bar{\beta}_{\text{MINSYS}})^T v^* - b^T \bar{\rho}.$$

3. Compute the approximation toll set as $W_{\text{FD}}(\bar{v}, \bar{\varepsilon})$.
4. Define and optimize one of the toll pricing problems with the current toll set $W_{\text{FD}}(\bar{v}, \bar{\varepsilon})$.

Then, the authors proceed with the proposal of a new cutting plane algorithm to solve the MINSYS problem. They decompose the initial problem into a master linear problem and a subproblem which consists of $|K|$ shortest-path detections. The steps of the cutting plane algorithm proposed are described as follows:

1. Obtain an aggregate vector flow v^* and the corresponding costs $s(v^*)$.
2. Begin choosing an approximate toll vector T^0 and setting the iteration counter $l = 0$.
3. Solve the LP

$$\min\{\beta^T v^* | \beta \in T^l\}. \quad (72)$$

4. Compute the shortest paths as $s(v^*) + \beta_l$. Also, obtain the extreme point solution $v_l = \sum_{k \in K} x_l^k$.
5. Check the condition with a tolerance ε . If

$$(s(v^*) + \beta)^T (v_l - v^*) \geq -\varepsilon$$

then the user equilibrium condition is approximately satisfied.

6. Otherwise, the constraint is added to the set of constraints T^l to obtain the new set T^{l+1} .
7. Increase the iteration counter $l = l + 1$ and go to Step 3.

3.4 A Generalized Second-Best Congestion Pricing Model

In this seminal publication, Verhoef [114] derives a second-best tax rule and shows that many known problems and tax rules can be considered as special cases to the general solution obtained. First of all, let us introduce the notation of the model in the following table (Table 6).

First of all, the basic assumption of the toll pricing model presented is that the regulator sets tolls on arcs according to the maximization of social welfare,

Table 6 Notation used for the second-best toll pricing problem

Notation	Description
G	Transport network
N	Set of nodes of the network
J	Set of the arcs of the network
I	Set of origin–destination pairs
N_i	Number of users for each pair i
N_j	Number of users of arc j
$D_i(N_i)$	Inverse demand function for each pair i
P	Set of paths (noncyclical) of the network
N_p	Number of users for path p
P_i	Set of noncyclical paths connecting origin–destination pair i
δ_{jp}	Binary parameter equal to 1 if arc j belongs to path p and 0 otherwise
δ_j	Binary parameter equal to 1 if it is possible to place a toll booth in arc j and 0 otherwise
f_j	Fee charged at toll on arc j

that is, total benefits minus total costs. That can be stated mathematically as follows:

$$\max_{f_j \forall j: \delta_j = 1} \quad W = \sum_{i=1}^I \int_0^{N_i} D_i(x_i) dx_i - \sum_{j=1}^J N_j c_j(N_j) \quad (73)$$

$$\text{s.t.} \quad N_p \geq 0 \quad (74)$$

$$\sum_{j=1}^J \delta_{jp}(c_j + \delta_j f_j) - D_i \geq 0 \quad (75)$$

$$N_p \left(\sum_{j=1}^J \delta_{jp}(c_j + \delta_j f_j) - D_i \right) = 0, \forall p \in P_i, \forall i \in I. \quad (76)$$

This model is maximizing an objective, based on the conditions that each trip decision maker is aiming to maximize their own objectives represented by Wardrop's principles. Then, Verhoef had the insight of discarding all irrelevant paths in order to transform the Wardrop conditions involved in the model to strict equalities. Therefore, the problem can be represented by the Lagrangian:

$$\begin{aligned} \Lambda = & \sum_{i=1}^I \int_0^{\sum_{p=1}^P \delta_{ip} N_p} D_i(x_i) dx_i - \sum_{j=1}^J \sum_{i=1}^I \sum_{p=1}^P \delta_{jp} \delta_{ip} N_p c_j \left(\sum_{k=1}^I \sum_{q=1}^P \delta_{jq} \delta_{kq} N_q \right) \\ & + \sum_{i=1}^I \sum_{p=1}^P \delta_{ip} \lambda_p \left[\sum_{j=1}^J \delta_{jp} \left(c_j \left(\sum_{k=1}^I \sum_{q=1}^P \delta_{jq} \delta_{kq} N_q \right) + \delta_j f_j \right) - D_i \left(\sum_{q=1}^P \delta_{iq} N_q \right) \right] \end{aligned}$$

Then, based on this equation, the first-order conditions can be computed:

$$\begin{aligned} \frac{\partial \Lambda}{\partial N_p} &= \sum_{i=1}^I \delta_{ip} D_i - \sum_{j=1}^J \delta_{jp} \left(c_j + \sum_{k=1}^I \sum_{q=1}^P \delta_{jq} \delta_{kq} N_q c'_j \right) \\ &\quad + \sum_{k=1}^I \sum_{q=1}^P \delta_{kq} \lambda_q \left(\sum_{j=1}^J \delta_{jp} \delta_{jq} c'_j \right) - \sum_{i=1}^I \sum_{q=1}^P \delta_{ip} \delta_{iq} \lambda_q D'_i = 0, \\ \forall p : \delta_{ip} &= 1 \\ \frac{\partial \Lambda}{\partial f_j} &= \sum_{i=1}^I \sum_{p=1}^P \delta_{ip} \delta_{jp} \lambda_p = 0, \forall j : \delta_j = 1 \\ \frac{\partial \Lambda}{\partial \lambda_p} &= \sum_{j=1}^J \delta_{jp} (c_j + \delta_j f_j) - \sum_{i=1}^I \delta_{ip} D_i = 0, \forall p : \delta_{ip} = 1. \end{aligned}$$

Substituting the last of the first-order conditions in the first one, the Lagrange multipliers can be calculated. The multipliers that are nonzero are the factors that cause the second-best optimal solution found (the one that does not allow a toll to be placed on all arcs) to be always inferior to the first-best framework. It can be verified that if it was permitted to use tolls in all the arcs of the network, then the second first-order condition would yield $\lambda_p = 0$.

After these important findings, Verhoef went along to compare his interesting results to earlier research outcomes on the field. First of all, the comparison with the first-best toll pricing framework, which implies that $\delta_j = 1, \forall j$, proves that the results obtained by the author are consistent with the Pigou tax rule [100], as expected. Then, the author performs another important comparison to the standard two-route problem. As mentioned in the introduction, this problem was initially studied by Lévy-Lambert [76] in 1968, to be followed by Verhoef et al. [115] in 1996.

Both studies concluded that the optimal second-best toll for the tolled route in the problem can be calculated by:

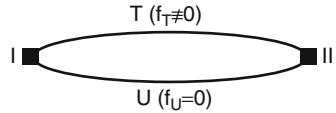
$$f_T = N_T c'_T - N_U c'_U \frac{-D'}{c'_U - D'}. \quad (77)$$

This can be shown when substituting the first-order conditions:

$$\frac{\partial \Lambda}{\partial N_T} = D - c_T - N_T c'_T + \lambda_T c'_T - (\lambda_T + \lambda_U) D' = 0 \quad (78)$$

$$\frac{\partial \Lambda}{\partial N_U} = D - c_U - N_U c'_U + \lambda_U c'_U - (\lambda_T + \lambda_U) D' = 0 \quad (79)$$

Fig. 3 The untolled two-route problem



$$\frac{\partial \Lambda}{\partial f_T} = \lambda_T = 0 \quad (80)$$

$$\frac{\partial \Lambda}{\partial \lambda_T} = c_T + f_T - D = 0 \quad (81)$$

$$\frac{\partial \Lambda}{\partial \lambda_U} = c_U - D = 0. \quad (82)$$

Solving the second-best framework presented, the author obtains that

$$\lambda_U = \frac{N_U c'_U}{c'_U - D'}, \quad (83)$$

and by substituting the first-order conditions into the last equation, the result given in Eq. (77) is found (Fig. 3).

Another example of Verhoef's framework consistency to earlier research results is the optimal parking policy for congestion management. As was noted by Glazer and Niskanen [44], it is realistic to assume that several network users have access to private parking spots; therefore, they form a subset of users that are not required to pay the parking fees. Once more, Verhoef applies the first-order conditions to obtain

$$\frac{\partial \Lambda}{\partial N_{II}} = D_{II} - c_1 - N_1 c'_1 + (\lambda_{II} + \lambda_{III}) c'_1 - \lambda_{II} D'_{II} = 0 \quad (84)$$

$$\frac{\partial \Lambda}{\partial N_{III}} = D_{III} - c_1 - N_1 c'_1 + (\lambda_{II} + \lambda_{III}) c'_1 - \lambda_{III} D'_{III} = 0 \quad (85)$$

$$\frac{\partial \Lambda}{\partial f_2} = \lambda_{II} = 0 \quad (86)$$

$$\frac{\partial \Lambda}{\partial \lambda_{II}} = c_1 + f_2 - D_{II} = 0 \quad (87)$$

$$\frac{\partial \Lambda}{\partial \lambda_{III}} = c_1 - D_{III} = 0, \quad (88)$$

where, clearly, $N_1 = N_{II} + N_{III}$. Again, solving the framework presented, the author obtains that

$$\lambda_{III} = \frac{N_1 c'_1}{c'_1 - D'_{III}}, \quad (89)$$

which, in turn, substituted in the first-order condition gives us

$$f_2 = N_1 c'_1 \frac{-D'_{III}}{c'_1 - D'_{III}}, \quad (90)$$

which is the same result that Glazer and Niskanen extracted in their work in 1992.

3.5 Hybrid Genetic Algorithms for the Toll Booth and the Toll Pricing Problems

A genetic algorithm is a metaheuristic, based on population members that provide us with high quality solutions in combinatorial optimization. In the context of the problems studied here, every member of the population is a feasible solution. In every generation (iteration) the solutions become of higher quality, since elite members of the population have a higher probability of selection, until a solution that is considered satisfactory is reached.

In literature, genetic algorithms have been successfully applied to a series of large-scale network combinatorial optimization problems. It was applied to the Open Shortest Path First problem Internet routing problem by Ericsson et al. in 2002 [34] with success and that led to Buriol et al. adapting the genetic algorithm presented with a local improvement procedure to further improve the results [15].

The above success stories gave the idea to Buriol et al. [16] to approach the toll booth problem with a hybrid genetic algorithm. The similarities between the Internet routing problem tackled and the toll booth problem predicated the very good results obtained in this research.

The feasible solutions are represented by two arrays, w and b . The first array stores the arc weights, while the latter is a binary array with the toll locations. An arc can have a positive weight if the corresponding entry in the binary array is true. Otherwise it is set to be zero. Another important note that needs to be made is that two paths are considered to be of equal cost if they have the same distance and the same number of nodes visited.

At each iteration, crossover is applied on the members selected in order to produce the next generation of population. At this point, a biased random key scheme [8] is used, and each pair of parents consists exclusively of an elite member of the population and a nonelite one. Each element of the eight array w either inherits the value of one of its parents or a mutation resets it with probability p_m . If a mutation does not take place, then the elite parent's weight value is favored, implying that the probability that this is the value inherited to the offspring is more than 1/2.

After the crossover is performed, the binary array b is adjusted as follows. If both parents i th element is equal to true, then the offspring inherits that value. If only one of the parents has an element set to true, then 50% of these positions are set to true, while all other positions are filled us false.

That way, the algorithm devised proceeds to the solution evaluation. After this step, a local improvement procedure is deployed in order to detect better quality solutions in the neighborhood of the initial solution. Last, the solution is dynamically updated in order to avoid recomputing all the shortest paths, a process that would be computationally expensive.

This work inspired an improved biased random key genetic algorithm by Buriol et al. in 2010 [17]. The problem of the traffic optimization tackled can be written as:

$$\min \Phi = \sum_{\alpha \in A} l_\alpha t_\alpha \left[1 + B_\alpha \left(\frac{l_\alpha}{c_\alpha} \right)^{p_\alpha} \right] \Bigg/ \sum_k \in K d_k \quad (91)$$

$$\text{s.t. } l_\alpha = \sum_{k \in K} x_\alpha^k, \quad \forall \alpha \in A \quad (92)$$

$$\sum_{i:(j,i) \in A} x_{(j,i)}^k - \sum_{i:(i,j) \in A} x_{(i,j)}^k = \begin{cases} -d_k & \text{if } j = d_k \\ d_k & \text{if } j = o(k), \forall j \in N, k \in K \\ 0 & \text{otherwise} \end{cases} \quad (93)$$

$$x_\alpha^k \geq 0, \quad \forall \alpha \in A, \forall k \in K. \quad (94)$$

The approach adopted is to place toll fares on K arcs of the network, such that the user equilibrium and the system optimum are simultaneously accomplished and the objective-fitness function Φ is minimized. An extension to the hybrid genetic algorithm, a random key genetic algorithm was devised in [17]. Therein, the authors depict its success in obtaining better quality results in large-scale, realistic transportation networks.

4 Vehicle Routing and Vehicle Fleet Management

4.1 Introduction

Another important problem in both transportation and logistics engineering is the vehicle routing problem, which was introduced by Dantzig and Ramser [29]. Its statement is simple, however it is a generalization of the infamous Traveling Salesman Problem, which proves the complexity and difficulties encountered when attempting to solve it.

Given a depot and a set of customers, find the least cost path/paths that a vehicle/fleet of vehicles needs to follow in order to serve each customer and return to the depot.

Fig. 4 The optimal parking policy problem

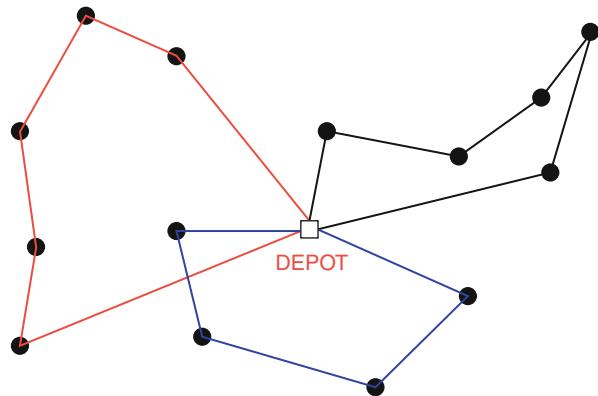
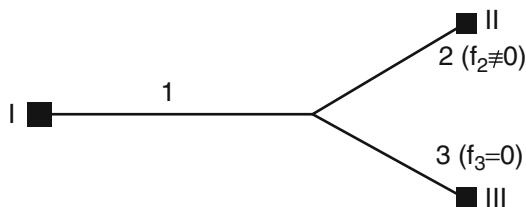


Fig. 5 A vehicle routing problem solution with a fleet of three vehicles

An example for a fleet of three and one vehicles accordingly are given in Figs. 4 and 5. It is clear that Fig. 4 (the vehicle routing problem solution with one vehicle) is essentially the Hamiltonian path that optimally solves the corresponding Traveling Salesman Problem (Fig. 6).

As Toth and Vigo [110] mention, the utilization of advanced operations research techniques in distribution and transportation systems produces very significant savings. Apart from the financial savings that are produced, even a small increase in the fleet's utilization has important environmental and social gains.

In general, the important parts of a vehicle routing or vehicle scheduling problem are the following:

1. Transportation infrastructure—graph
2. Customers—nodes
3. Vehicles

Of course, each of these elements is associated with different specificities and characteristics. The network is represented as a graph and consists of arcs-links, which connect nodes. If two nodes are connected, then the vehicle has a certain cost of traversing that link. The most commonly used objective is to minimize the costs of transportation of each of the vehicles using the infrastructure.

Each customer has different characteristics as to how they need to be serviced. Hence, customers are represented as special nodes on the graph that is studied. It is vital for the vehicle routing problem to have the knowledge of the exact node where the customer has to be serviced. Another very important detail for any customer is

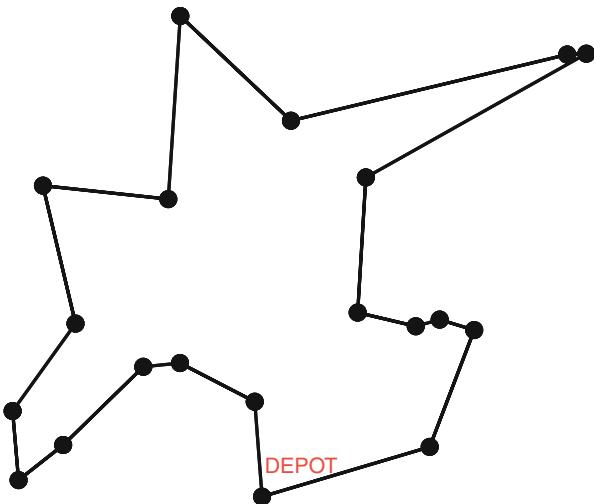


Fig. 6 A vehicle routing problem solution with a fleet of one vehicle

the demand of goods that would need to be transported. According to the model used, there exist other specifications which are necessary such as:

- Customer location
- Demand
- Time window of service
- Collection and/or delivery customer
- Service time required
- Type of vehicle suitable for the customer

Last, the vehicles that are used to satisfy the demands while meeting the objectives set can be characterized by:

- Initial point-depot
- Capacity
- Type
- Links that they can use
- Customers that they can serve

There has been a brief mention of objectives, which is a general notion and, hence, there exist many objective functions associated with different types of vehicle routing problems, often contradicting each other. Usually one of the following is encountered:

1. Minimize the overall costs
2. Minimize the number of vehicles used
3. Balancing of the vehicles' times and loads
4. Maximizing the quality of service

Table 7 Notation used for the vehicle routing problems presented

Notation	Definition
G	Transportation network
V	Set of nodes-vertices of the network
c_{ij}	Cost of traversing the edge that connects nodes i and j
K	Set of vehicles in our disposal, $k = 1, \dots, K$
C	Set of vehicle capacities
S	Set of customers, $S \subseteq V$
d_i	Demand of customer $i \in S$
x_{ijk}	Binary variable $x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ uses the arc that connects } i \text{ and } j \\ 0, & \text{otherwise} \end{cases}$
y_{ik}	Binary variable $y_{ik} = \begin{cases} 1, & \text{if vehicle } k \text{ serves customer } i \\ 0, & \text{otherwise} \end{cases}$

In the following subsection, the most widely known vehicle routing problems will be tackled, beginning with the simple, uncapacitated version and moving towards more realistic approaches.

4.2 Vehicle Routing Problem

The most commonly used versions of the vehicle routing problem are:

- Capacitated vehicle routing
- Vehicle routing with time windows

Each one of them is characterized by different aspects and details. The most commonly used formulations are presented in this subsection ([Table 7](#)).

At this point, it is interesting to study the simple version of the *capacitated* vehicle routing problem. The model most commonly used is the three-index model presented in [110].

$$\min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} \sum_{k=1}^K x_{ijk} \quad (95)$$

$$\text{subject to } \sum_{k=1}^K y_{ik} = 1, \quad \forall i \in V \setminus \{0\} \quad (96)$$

$$\sum_{k=1}^K y_{0k} = K \quad (97)$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik}, \quad \forall i \in V, k = 1, \dots, K \quad (98)$$

$$\sum_{i \in V} d_i y_{ik} \leq C \quad (99)$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ijk} \geq y_{hk}, \quad \forall S \subseteq V \setminus \{0\}, h \in S, \quad k = 1, \dots, K \quad (100)$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i \in V, k = 1, \dots, K \quad (101)$$

$$y_{ik} \in \{0, 1\}, \quad \forall i, j \in V, k = 1, \dots, K. \quad (102)$$

The similarities with the traveling salesman problem are immediately apparent. For instance, Eq. (100) can be studied. The constraint ensures that the optimal route found is complete, that is, it does not include any subtours. In Fisher and Jaikumar [36], it is replaced by

$$\sum_{i \in S} \sum_{j \in S} x_{ijk} \leq |S| - 1, \quad S \subseteq V \setminus \{0\}, |S| \geq 2, k = 1, \dots, K, \quad (103)$$

which is the subtour elimination constraint used in the TSP. The model presented is widely used because it is very easy to incorporate a nonhomogeneous fleet of vehicles with distinct capacities and costs. In addition to that, by adopting large costs in the arcs leaving the depot and changing the constraint

$$\sum_{k=1}^K y_{0k} = K$$

to

$$\sum_{k=1}^K y_{0k} \leq K,$$

then the solution found is forced to use the minimum number of vehicles possible.

Initially, it is interesting to note that the TSP has provided researchers with numerous relaxations for the vehicle routing problem, starting as early as 1971 and the 1-tree relaxation of Held and Karp [57]. This relaxation gave way to more recent attempts like the proposed branch-and-bound algorithm of Christofides [23], who focused on the k -degree center tree relaxation. This tree is a minimum cost spanning tree with a degree of k at the initial node. Immediately after that, the algorithm proceeds to choose K least cost arcs to be added, where $2K - k$ are incident while the remaining are not. This relaxation worked efficiently solving problems of up to 25 customers.

Another bound can be obtained by the Lagrangian dual function, which in reality can provide researchers with solutions of much better quality. The problem with these dualizations arise with the exponentially large number of constraints that are included (i.e., either the capacity-cuts or the subtour eliminations). Thus, the process

is as follows. Find a solution to the Lagrangian problem and at each iteration, find a violated constraint and add it to the problem. This method is applied in Fisher and Jaikumar [36] and Miller [85] and is exact.

Last, the set partitioning branch-and-bound algorithm that was proposed by Balinsky and Quandt [5] and was advanced by Hadjiconstantinou [49] is also a method that has attracted scientific interest because of its high efficiency. The model used in [5] is presented in Eqs. (104)–(107).

$$\min \sum_{j=1}^M c_j x_j \quad (104)$$

$$\text{s.t. } \sum_{j=1}^M a_{ij} x_j = 1, \quad \forall i \in V \setminus \{0\} \quad (105)$$

$$\sum_{j=1}^M x_j = K \quad (106)$$

$$x_j \in \{0, 1\}, \quad \forall j = 1, \dots, M. \quad (107)$$

This model has been applied successfully in a number of applications as observed by Descrosiers [30].

For large-scale problems which reflect more practical applications and, therefore, are very important, the use of branch-and-bound algorithms is limiting and inefficient. Mainly several heuristic approaches have been implemented with success, increasing the size of the problems solved to optimality in the last decades as was noted by Laporte et al. [68]. In their work, they present the most commonly used heuristics, such as:

- Sweep [43]
- Petal [107]
- Cluster-First, Route-Second [14]

The sweep algorithm is pretty simple and easily implemented given that each customer-vertex can be represented by its polar coordinates (ρ_i, θ_i) . First of all, the vertices need to be sorted in increasing angles from an arbitrary vertex \tilde{i} with $\theta_{\tilde{i}} = 0$. Then, these steps can be followed:

1. Choose an unassigned vehicle k .
2. Detect the unserviced customer i with the smallest θ_i and assign customers to vehicle k until the capacity is not exceeded. If there exist other customers remaining to be serviced, return to Step 1.
3. Solve the corresponding TSP for each of the vehicle routes created above.

A natural extension of the sweep algorithm was given by Ryan et al. [107]. It consists of generating different routes and, then, solve a set partitioning problem in order to select the best routes possible. The form of the set partitioning problem is given in Eqs. (108)–(110).

Table 8 Notation used for the vehicle routing problem with time windows

Notation	Definition
G	Transport network
V	Set of vehicles to be routed
N	Set of all the nodes of the network
A	Set of all the arcs of the network
c_{ij}	Cost associated with each arc $(i, j) \in A$
t_{ij}	Time required to cross arc $(i, j) \in A$ including the service time at customer $i \in N$
q_k	Capacity of vehicle $k \in V$
d_i	Demand of customer at node $i \in N$
$[a_i, b_i]$	Time window of customer $i \in N$
x_{ijk}	Binary decision variable $x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ uses arc } (i, j) \\ 0, & \text{otherwise} \end{cases}$
s_{ik}	Decision variable that represents the time vehicle k services customer i

$$\min \sum_{k \in S} d_k x_k \quad (108)$$

$$\text{s.t. } \sum_{k \in S} a_{ik} x_k, \quad \forall i = 1, \dots, n \quad (109)$$

$$x_k \in \{0, 1\}, \quad \forall k \in S, \quad (110)$$

where S is the set of routes. Also, it can be obtained for the binary variable x_k :

$$x_k = \begin{cases} 1 & \text{if route } k \text{ is in the solution} \\ 0 & \text{otherwise.} \end{cases}$$

Of course it is vital for the efficiency of the algorithm to produce and check only a sufficient subset of all the possible routes set, otherwise the method is computationally expensive. Heuristic rules for detecting such subsets were proposed by Foster and Ryan [106] and Ryan et al. [107]. In 1996, however, an advancement was proposed by Renaud et al. [101] to include configurations of multiple vehicles with embedded or intersecting routes, instead of just one vehicle route.

Now, a more realistic approach to the classical vehicle routing problem can be considered, the one with time windows. It is widely known that every customer needs not only to be serviced but also to be satisfied within certain time slots. The major difference to the original version studied until now is this: every customer vertex of the graph is assigned, in addition to the previous characteristics, a time period in which they require the service to take place. These constraints are considered soft by some researchers, that is, they can be violated with a certain penalty cost, and hard by others (Table 8).

The mathematical formulation of the vehicle routing problem with time windows is presented here:

$$\min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ijk} x_{ijk} \quad (111)$$

$$\text{s.t. } \sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \quad \forall i \in C \quad (112)$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq q, \quad \forall k \in V \quad (113)$$

$$\sum_{j \in N} x_{0jk} = 1, \quad \forall k \in V \quad (114)$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \quad \forall h \in C, \quad \forall k \in V \quad (115)$$

$$\sum_{i \in N} x_{i,n+1,k} = 1, \quad \forall k \in V \quad (116)$$

$$x_{ijk} (s_{ik} + t_{ij} - s_{jk}) \leq 0, \quad \forall i, j \in N, \quad \forall k \in V \quad (117)$$

$$a_i \leq s_{ik} \leq b_i \quad \forall i \in N, \quad \forall k \in V \quad (118)$$

$$x_{ijk} \in \{0, 1\} \quad \forall i, j \in N, \quad \forall k \in V. \quad (119)$$

The problem is defined as the attempt to service each customer in the graph exactly once, while starting and ending by a depot vertex. The difference is that now there is a time of service associated with each of the customers that is required to be met by the vehicle planned to satisfy the demand of that customer. The objective function shown above is minimizing the total travel cost, selecting the least expensive arcs in order to satisfy all the demands. The next two constraints are used to visit exactly once each customer in the graph and to ensure that the capacity of each of the vehicles is not exceeded. Equations (114)–(116) ensure that each vehicle is starting from the depot, servicing a customer and moving on to another customer or returning to the depot. The next constraint indicates when arriving at a customer at a specific time, then the next customer will be serviced at some time after that event. Last, the time window constraints are given. Once more, a limit to the number of vehicles that are planned to be used is given like in the case of Descrosiers et al. [30] using Eq. (120).

$$\sum_{k \in V} \sum_{j \in N} x_{0jk} \leq |V| \quad \forall k \in V, \quad \forall j \in N. \quad (120)$$

It becomes immediately clear that the vehicle routing problem is a generalization of the original vehicle routing problem. When the time window constraints are

dropped, the problem is relaxed to the vehicle routing problem studied earlier in this chapter. By looking closer at the mathematical formulation, only Eq. (112) is a constraint that binds a vehicle to a particular customer [62]. Hence, decomposition methods such as Lagrange or Dantzig–Wolfe seem like natural choices for algorithmic designs. One of the first research groups to apply Dantzig–Wolfe decomposition was Desrochers, Descrosiers, and Solomon in 1992 [31], considering the assignment constraints as the coupling ones, while the subproblem consisted of a shortest-path problem with resource constraints.

If the problem formulated in Eqs. (111)–(119) is considered again, then, as stated, the coupling constraint is considered to be Eq. (112). Hence, let us define the master problem consisting of Eqs. (111)–(112) and (119), so basically the problem is restricted to take into consideration the objective function, the coupling constraint of every vehicle to a specific customer, and the binary restriction on the variables. The rest of the constraints are still used in the subproblem which will be described shortly.

Now it can be assumed that P^k is the set of all the paths that vehicle k , $k \in V$ can follow. Also, let it be defined that

$$x_{ijp}^k = \begin{cases} 1, & \text{if vehicle } k \text{ is moving from node } i \text{ to node } j \text{ using path } p \\ 0, & \text{otherwise.} \end{cases}$$

Then, any solution to the master problem can be represented as convex combination of some of the paths, that is,

$$x_{ij}^k = \sum_{p \in P^k} x_{ijp}^k y_p^k \quad \forall k \in V, \quad \forall (i, j) \in A \quad (121)$$

$$\sum_{p \in P^k} y_p^k = 1 \quad \forall k \in V \quad (122)$$

$$y_p^k \geq 0 \quad \forall k \in V, \quad \forall p \in P^k. \quad (123)$$

Using the notation for x_{ijp}^k , c_p^k (the cost of a path p for vehicle k) and α_{ip}^k (number of times customer i is serviced by vehicle k) can be defined as

$$c_p^k = \sum_{(i,j) \in A} c_{ij}^k x_{ijp}^k \quad \forall k \in V, \quad \forall p \in P^k \quad (124)$$

$$\alpha_{ip}^k = \sum_{j \in N \cup \{n+1\}} x_{ijp}^k \quad \forall k \in V, \forall i \in N, \forall p \in P^k. \quad (125)$$

Using the above notations and definitions, the master problem is rewritten as

$$\min \sum_{k \in V} \sum_{p \in P^k} c_p^k u_p^k \quad (126)$$

$$\text{s.t. } \sum_{k \in V} \sum_{p \in P^k} \alpha_{ip}^k y_p^k = 1 \quad \forall i \in C \quad (127)$$

$$\sum_{p \in P^k} y_p^k = 1 \quad \forall k \in V \quad (128)$$

$$y_p^k \geq 0 \quad \forall k \in V, \forall p \in P^k. \quad (129)$$

Usually, the case studied involved one depot and a homogeneous fleet of vehicle [62]; hence, the problem can be transformed into

$$\min \sum_{p \in P} c_p y_p \quad (130)$$

$$\text{s.t. } \sum_{p \in P} \alpha_{ip} y_p = 1 \quad \forall i \in C \quad (131)$$

$$y_p \geq 0 \quad \forall p \in P, \quad (132)$$

which is derived by aggregating the constraints in Eq. (128) and by letting $y_p = \sum_{k \in V} y_p^k$. The model described by Eqs. (130)–(132) is the linear relaxation of the set partitioning formulation. Now, the problem can be restricted to only take into consideration the set of columns that have already been generated, let P' . The resulting formulation uses the decision variables y_p as a counter of the times that path $p \in P'$ has been used and in the relaxed version can take any value in the interval $[0, 1]$.

$$\min \sum_{p \in P'} c_p y_p \quad (133)$$

$$\text{s.t. } \sum_{p \in P'} \alpha_{ip} y_p = 1 \quad \forall i \in C \quad (134)$$

$$y_p \geq 0 \quad \forall p \in P', \quad (135)$$

Solving the above model, a solution $y = (y_1, y_2, \dots, y_{|P'|})$ and the corresponding dual solution $\phi = (\phi_1, \phi_2, \dots, \phi_{|C|})$ are obtained. It is important to note here that the solution obtained might not be an integer however, even in the case that it is, the procedure does not guarantee an optimal solution, but merely a feasible one.

Now, the subproblem that the column generation approach aims to solve can be brought to attention. As stated before, the subproblem decomposes in the case of the vehicle routing problem with time windows into $|V|$ identical problems. They all are a shortest-path problem with resource constraints, which reflect the time windows and the vehicle capacities [62]. The subproblem is then formulated mathematically as

$$\min \sum_{i \in N} \sum_{j \in N} \hat{c}_{ij} x_{ij} \quad (136)$$

$$\text{s.t. } \sum_{i \in C} d_i \sum_{j \in N} x_{ij} \leq q \quad (137)$$

$$\sum_{j \in N} x_{0j} = 1 \quad (138)$$

$$\sum_{i \in N} x_{ih} - \sum_{j \in N} x_{hj} = 0 \quad \forall h \in C \quad (139)$$

$$\sum_{i \in N} x_{i,n+1} = 1 \quad (140)$$

$$s_i + t_{ij} - M_{ij}(1 - x_{ij}) \leq s_j \quad \forall i, j \in N \quad (141)$$

$$\alpha_i \leq s_i \leq b_i \quad \forall i \in N \quad (142)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in N. \quad (143)$$

In the above formulation, \hat{c}_{ij} is the *modified* cost and is equal to $\hat{c}_{ij} = c_{ij} - \pi_i$. The above problem has already been proved to be NP-hard by Dror in 1994 [32]. In order to solve it, this approach needs to be adapted by including another set of decision variables

$$x_{ij}^l = \begin{cases} 1, & \text{if arc } (i, j) \text{ is the } l\text{-th arc of the shortest path} \\ 0, & \text{otherwise} \end{cases}$$

and s_i^l that signal the starting time of service for a customer at node i when l is the rank of the customer, that is, the l th customer of the vehicle service. Also, the customer number $l \in L = \{1, 2, \dots, |L|\}$ and $|L| = \left\lfloor \frac{b_{n+1}}{\min t_{ij}} \right\rfloor$. Then the above problem formulation can be rewritten as

$$\min \sum_{l \in L} \sum_{i \in N} \sum_{j \in N} \hat{c}_{ij} x_{ij}^l \quad (144)$$

$$\text{s.t. } \sum_{i \in N} \sum_{j \in N} x_{ij}^1 = 1 \quad (145)$$

$$\sum_{i \in N} \sum_{j \in N} x_{ij}^l - \sum_{i \in N} \sum_{j \in N} x_{ij}^{l-1} \leq 0 \quad \forall l \in L - \{1\} \quad (146)$$

$$\sum_{i \in C} d_i \sum_{l \in L} \sum_{j \in N} x_{ij}^l \leq q \quad (147)$$

$$\sum_{j \in N} x_{0j}^1 = 1 \quad (148)$$

$$\sum_{i \in N} x_{ih}^{l-1} - \sum_{j \in N} x_{hj}^l = 0 \quad \forall h \in C \quad \forall l \in L - \{1\} \quad (149)$$

$$\sum_{l \in L} \sum_{i \in N} x_{i,n+1}^l = 1 \quad (150)$$

$$s_i^l + t_{ij} - K(1 - x_{ij}^l) \leq s_j^l \quad \forall i, j \in N \quad \forall l \in L - \{1\} \quad (151)$$

$$\alpha_i \leq s_i^l \leq b_i \quad \forall i \in N \quad (152)$$

$$x_{ij}^l \in \{0, 1\} \quad \forall i, j \in N. \quad (153)$$

In that formulation, Eq. (145) ensures that the first arc can only be used once, the constraint in Eq. (146) guarantees that arc l cannot be used unless arc $l - 1$ is also used, while the rest of the constraints remain the same with the alterations required. This formulation has been solved in the past by the algorithm designed by Desrochers, Descrosiers, and Solomon and is described in [31]. However, it has been shown that if the distances in the problem satisfy the triangle inequality (this is actually the case in most practical applications), then the optimal solution contains *only* elementary routes. That implies that there is a possibility that the master problem lower bound obtained decrease and, hence, improvements need to be made in the algorithmic part. Such an improvement was studied by Kolen et al. [67].

4.3 Special Cases and Recent Advancements

Recently, the vehicle routing problem has escaped the fields of logistics and transportation and has advanced in other fields where machinery operation and route planning is costly. The mostly noted example is the one of robotic autonomous routing and is investigated in numerous publication during the last 5 years. The issues investigated include:

- Communication coverage planning [96]
- Robot docking [13]
- Multi-robot motion in an environment with obstacles [112] or in an environment with other moving robots [118]

This is the framework that Baskar et al. [7] adopted in their research on optimal routing that uses an Intelligent Vehicle Highway System. The assumption they make for the infrastructure is that it is an automated highway system where autonomous vehicles are fully controlled in order to reach their respective destinations.

The framework, described in Baskar et al. [6], consists of the following elements:

1. The vehicle controllers, which control the speed and steering of the vehicles by receiving orders from the platoon controllers

Table 9 Notation used in the autonomous vehicle routing by Baskar et al. [7]

O	Origin nodes
D	Destination nodes
I	Internal nodes
V	The set of all nodes, $V = O \cup I \cup D$
L	The set of links in the network
(o, d)	One origin–destination pair, $(o, d) \in O \times D$
$L_{o,d}$	The set of links that belong to a route connecting o to d
$D_{o,d}$	The demand of the pair (o, d)
C_l	The capacity of the link $l \in L$
v_l	The speed on link $l \in L$
τ_l	The travel time on link $l \in L$
L_v^{in}	The set of all links incoming to node v
L_v^{out}	The set of all links leaving node v
$x_{l,o,d}$	Decision variables denoting the flows for every pair $(o, d) \in O \times D$
T	The simulation period

2. The platoon controllers, which take care of the merges and splits of platoons and the vehicle to vehicle distances by receiving control commands from the roadside controllers
3. The roadside controllers, which are in charge of a segment of the whole network in the infrastructure
4. The higher-level controllers, which coordinate the whole network and supervise all other controllers

Using this infrastructure, the authors focus on optimal routing to each platoon that is currently in the network (Table 9). The notation that is used is given in the following network.

The authors focus on the following cases:

1. The static case with sufficient capacities
2. The static case with queues forming at the boundaries only
3. The dynamic case with queues forming at the boundaries only

In the static case, the assumption has to do with the constant nature of the demand of each (o, d) pair in the network. Hence, the model for the static case with sufficient capacities is described in Eqs. (154)–(157).

$$\min J_{\text{links}} = \sum_{(o,d) \in O \times D} \sum_{l \in L_{o,d}} x_{l,o,d} \tau_l T \quad (154)$$

$$\text{s.t. } \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d} = D_{o,d}, \quad \forall o \in O, \forall d \in D \quad (155)$$

$$\sum_{l \in L_v^{\text{in}} \cap L_{o,d}} x_{l,o,d} = \sum_{l \in L_v^{\text{out}} \cap L_{o,d}} x_{l,o,d}, \quad \forall v \in V, \forall (o, d) \in O \times D \quad (156)$$

$$\sum_{(o,d) \in I_{od,l}} x_{l,o,d} \leq C_l, \quad \forall l \in L. \quad (157)$$

The above model is simple to grasp since it involves only the preservation of the flows in the network and the capacity constraints for each of the links. The objective function described in Eq. (154) is a measure of the time that the vehicles have to stay on the network using its infrastructure while traveling.

In order to make the model a little more realistic, the authors then proceed to tackle the problem of queues being formed at the “entrance” points of the infrastructure. So now the model can be written as

$$\min J_{\text{links}} + J_{\text{queue}} = \sum_{(o,d) \in O \times D} \sum_{l \in L_{o,d}} x_{l,o,d} \tau_l T + \quad (158)$$

$$\sum_{(o,d) \in O \times D} \frac{1}{2} (D_{o,d} - F_{o,d}^{\text{out}}) T^2 \quad (159)$$

$$\text{s.t. } \sum_{l \in L_v^{\text{in}} \cap L_{o,d}} x_{l,o,d} = \sum_{l \in L_v^{\text{out}} \cap L_{o,d}} x_{l,o,d}, \quad \forall v \in V, \forall (o,d) \in O \times D \quad (160)$$

$$\sum_{(o,d) \in I_{od,l}} x_{l,o,d} \leq C_l, \quad \forall l \in L \quad (161)$$

$$\sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d} \leq D_{o,d}, \quad \forall o \in O, \forall d \in D \quad (162)$$

$$F_{o,d}^{\text{out}} = \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d}. \quad (163)$$

In order to estimate J_{queue} which is a measure of the time spent by the vehicles in the queues formed in the origin nodes, the authors note that the queue size increases with time with a rate of $D_{o,d} - F_{o,d}^{\text{out}}$. Hence, at the end of the simulation time, the total length is $(D_{o,d} - F_{o,d}^{\text{out}})T$ and the average is calculated as $\frac{1}{2}(D_{o,d} - F_{o,d}^{\text{out}})T$. That is how the term of J_{queue} is computed in the objective function above. Once more, the mathematical program is linear.

It is easy to observe that the model is yet unrealistic. In most practical applications, it is not wise to assume that the demands are constant and, thus, a dynamic modeling approach has to be adopted. In order to do so, the authors introduce a discretization of the time spent on each link, with the elementary measurement of T_s . This can be written more clearly as

$$\tau_l = \kappa_l T_s, \text{ where } \kappa_l \in \mathbb{Z}^+. \quad (164)$$

Letting $q_{o,d}(k)$ be the partial queue length of vehicles traveling from o to d at time k , that is, $t = kT_s$, and by assuming that the network is initially empty, that is,

$q_{o,d}(k) = 0$ and $x_{l,o,d}(k) = 0$ for $k \leq 0$ for each of the origin nodes o , it can be obtained that

$$\sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d}(k) \leq D_{o,d}(k) + \frac{q_{o,d}(k)}{T_s} \quad \forall d \in D, \quad (165)$$

and by definition $D_{o,d}(k) = 0$ for $k \geq K$. Now, by considering the fact that every vehicle on link l will reach the end of the link after κ_l time segments, it is obtained that

$$\begin{aligned} \sum_{l \in L_v^{\text{in}} \cap L_{o,d}} x_{l,o,d}(k - \tau_l) &= \sum_{l \in L_v^{\text{out}} \cap L_{o,d}} x_{l,o,d}(k), \\ \forall v \in I \quad \forall (o, d) \in O \times D. \end{aligned} \quad (166)$$

Also for every link, the capacity constraints need to be enforced, however taking into consideration the time period. So Eq. (161) is now transformed in the dynamic case into

$$\sum_{(o,d) \in I_{od,l}} x_{l,o,d}(k) \leq C_l, \quad \forall l \in L. \quad (167)$$

The important part of this model is the way the queues formed are described. The flow is given by a similar constraint to the one presented in Eq. (163), which however is transformed in order to accommodate the time factor into

$$F_{o,d}^{\text{out}}(k) = \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d}(k). \quad (168)$$

Therefore, the queue length is increasing linearly with the rate of $D_{o,d}(k) - F_{o,d}^{\text{out}}(k)$ for the time interval $[k T_s, (k+1) T_s]$ and the following equation for the queue length can be written:

$$q_{o,d}(k+1) = \max(0, q_{o,d}(k) + (D_{o,d}(k) - F_{o,d}^{\text{out}}(k)) T_s). \quad (169)$$

Two cases exist for the determination of the time $J_{\text{queue},o,d}(k)$ that a vehicle has to spend in the queue formed at an origin o :

- The queue length becomes zero while the interval $[k T_s, (k+1) T_s]$.
- The queue length remains positive in the same interval.

Now, the latter case can be considered. Defining the time after $k T_s$ at which the queue length becomes zero as

$$T_{o,d}(k) = \frac{q_{o,d}(k)}{F_{o,d}^{\text{out}}(k) - D_{o,d}(k)}, \quad (170)$$

J_{queue} can now be estimated as

$$J_{\text{queue},o,d}(k) = \begin{cases} \frac{1}{2}(q_{o,d}(k) + Q_{o,d}(k+1))T_s & \text{for the first case} \\ \frac{1}{2}q_{o,d}(k)T_{o,d}(k) & \text{for the second case.} \end{cases} \quad (171)$$

In general now, for the queue

$$J_{\text{queue}} = \sum_{k=0}^{K_{\text{end}}-1} \sum_{(o,d) \in O \times D} \sum_{l \in L_{o,d}} J_{\text{queue},o,d}(k) \quad (172)$$

and

$$J_{\text{links}} = \sum_{k=0}^{K_{\text{end}}-1} \sum_{(o,d) \in O \times D} \sum_{l \in L_{o,d}} x_{l,o,d}(k) \kappa_l T_s^2. \quad (173)$$

So finally the mathematical formulation becomes

$$\min (J_{\text{links}} + J_{\text{queue}}) \quad (174)$$

$$\text{s.t. } (165) - (169). \quad (175)$$

This model is for the second case a nonlinear, nonconvex, and nonsmooth problem. As such, this problem is hard to solve, and hence, the authors present an approximate solution algorithm. Either way, by transforming the above problem into a mixed integer linear program, there exist several solvers that can solve it efficiently [98]. For this transformation to take place, the authors use some binary variables with no practical meaning and a series of properties proved by Bemporad and Morari in 1999 [10] that state:

Property 1 $[f \leq 0] \iff [\delta = 1]$ is true iff

$$\begin{cases} f \leq M(1-\delta) \\ f \geq \varepsilon + (m-\varepsilon)\delta, \end{cases}$$

where ε is a small positive number.

Property 2 $y = \delta f$ is equivalent to

$$\begin{cases} y \leq M\delta \\ y \geq m\delta \\ y \leq f - m(1-\delta) \\ y \geq f - M\delta. \end{cases}$$

Then, by eliminating the term of $F_{o,d}^{\text{out}}(k)$ from Eq. (169), they obtain

$$q_{o,d}(k+1) = \max(0, q_{o,d}(k) + (D_{o,d}(k) - \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d}(k))T_s), \quad (176)$$

which still is nonlinear. However, by letting $D_{\max,o,d}$ be the maximum demand for $(o, d) \in O \times D$, $F_{\max,o,d}$ be the maximum feasible flow (i.e., $F_{\max,o,d} = \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} C_l$) and $q_{\max,o,d}$ be the maximum queue length formed from origin o to destination d and equal to $D_{\max,o,d} T_s K_{\text{end}}$, then two new parameters can be defined as

$$m_{o,d}^{\text{low}} = -F_{\max,o,d} T_s \quad (177)$$

$$m_{o,d}^{\text{upp}} = q_{\max,o,d} + D_{\max,o,d} T_s, \quad (178)$$

hence the following always stands:

$$m_{o,d}^{\text{low}} \leq q_{o,d}(k) + (D_{o,d}(k) - \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d}(k))T_s \leq m_{o,d}^{\text{upp}}. \quad (179)$$

Now, by introducing the binary variables $\delta_{o,d}(k)$ as

$$\delta_{o,d}(k) = \begin{cases} 1 & \text{iff } q_{o,d}(k) + (D_{o,d}(k) - \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d}(k))T_s \geq 0 \\ 0 & \text{otherwise.} \end{cases} \quad (180)$$

So, applying Property 1, constraint (Eq. 176) takes the form of

$$q_{o,d}(k+1) = \delta_{o,d}(k) \left(q_{o,d}(k) + D_{o,d}(k) - \sum_{l \in L_o^{\text{out}} \cap L_{o,d}} x_{l,o,d}(k) T_s \right). \quad (181)$$

Note that this constraint remains *nonlinear*; however, it can be turned into one by using Property 2 mentioned above. Finally, a series of *linear* inequalities is obtained and the problem can now be solved approximately by one of the well known mixed integer linear programming solvers. The authors tested their formulation using several solvers from the overviews presented by Atamtürk and Savelsbergh [3] and Linderoth and Ralphs [77] and obtained high quality solutions, even better at some instances than local search algorithms.

4.4 Fixed Charge Network Flows

Last, but definitely not least, the fixed charge network flow problem has attracted scientific interest recently, due to the major practical applications that it has. For

instance, the fixed charge network problem can be encountered in network design, logistics and transportation engineering, and computer networks [42].

The problem is known to be NP-hard, and is a concave minimization problem. Hence, the optimal solution can be found on one of the vertices of the feasible region. With that in mind, Murty [87] suggested a vertex ranking procedure. Unfortunately, the method was still unusable for large-scale, realistic instances. Other heuristics were proposed over time, most notable of which was the effort by Kim and Pardalos [63, 64], called Dynamic Slope Scaling Procedure. The heuristic solves a series of linear problems, where the slope of the cost function is updated using the information obtained by previous iterations. An advancement to that work can be found by Nahapetyan and Pardalos [89] and Sorokin et al. [108].

5 Conclusion

Combinatorial optimization techniques have become more refined leading to numerous success stories in operations research. Especially in transportation and logistics, the vehicle routing problem stands out as an example of the immense capabilities that these new techniques offer. As was noted by Magnanti in 1981 [80], the size of the problems that could be solved was very small, especially compared to the large-scale problems that it is possible to solve to optimality nowadays.

The modern issues that arise have to do with solving real life applications with large-scale instances. The problem is that the more realistic an approach is the more unstable its parameters are. That implies that solutions can be obtained, which by the time they are adopted and put into practice have moved from being optimal to “bad” or, even worse, infeasible. This is a major discrepancy in most approaches that needs to be dealt with and a trade-off between a better solution and a more robust one is to be decided.

Robust optimization [11] is an important aspect of combinatorial problems that needs to be addressed. This branch of optimization deals with uncertainty in the parameters of a given problem and attempts to optimize a problem with the intention of choosing a solution that remains feasible even when the worst case scenarios occur. This is the route that most researchers seem to be choosing for future work, since it is a well known fact that our world is highly stochastic and, as such, it is important to adopt solutions that will remain plausible instead of theoretical approaches that prove to be inflexible towards change and unpredictable events.

Recommended Reading

1. H. Armelius, L. Hultkranz, The politico-economic link between public transport and road pricing: an ex-ante study of the Stockholm road-pricing trial. *Transport. Policy* **13**(1), 167–172 (2006)
2. R. Arnott, K. Small, The economics of traffic congestion. *Am. Sci.* **82**(5), 446–455 (1994)
3. A. Atamtürk, M.W.P. Savelsbergh, Integer programming software systems. *Ann. Oper. Res.* **140**(1), 67–124 (2005)

4. A. Babin, M. Florian, L. James-Lefebvre, H. Spiess, EMME/2 An interactive graphic method for road and transit planning, in *RTAC Annual Conference Preprints*, vol. 1 (1981)
5. M.L. Balinski, R.E. Quandt, On an integer program for a delivery problem. *Oper. Res.* **12**(2), 300–304 (1964). JSTOR
6. L.D. Baskar, B. De Schutter, H. Hellendoorn, Hierarchical traffic control and management with intelligent vehicles, in *IEEE Intelligent Vehicles Symposium*, 2007, pp. 834–839
7. L.D. Baskar, B. De Schutter, H. Hellendoorn, Optimal routing for intelligent vehicle highway systems using mixed integer linear programming. Technical Report, Delft University, Netherlands, 2010
8. J.C. Bean, Genetic algorithms and random keys for sequencing and optimization. *ORSA J. Comput.* **6**, 154–160 (1994)
9. M.J. Beckmann, C.B. McGuire, C.B. Winsten, *Studies in the Economics of Transportation* (Yale University Press, New Haven, 1956)
10. A. Bemporad, M. Morari, Control of systems integrating logic, dynamics and constraints. *Automatica* **35**(3), 407–427 (1999)
11. A. Ben-Tal, A. Nemirovski, Robust optimization—methodology and applications. *Math. Program.* **92**(3), 453–480 (2002)
12. D.P. Bertsekas, J.N. Tsitsiklis, *Parallel and Distributed Computation* (Prentice Hall, Old Tappan, 1989)
13. D.D. Bochtis, S.G. Vougioukas, H.W. Griepentrog, A mission planner for an autonomous tractor. *Trans. ASABE* **52**(5), 1429–1440 (2009)
14. J.B. Bramel, D. Simchi-Levi, A location based heuristic for general routing problems. *Oper. Res.* **43**, 649–660 (1995)
15. L.S. Buriol, M.G.C. Resende, C.C. Ribiero, M. Thorup, A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks* **46**, 36–56 (2005)
16. L.S. Buriol, M.J. Hirsch, P.M. Pardalos, T. Querido, M.G.C. Resende, M. Ritt, A hybrid genetic algorithm for road congestion minimization, in *Proceedings of the XL Simpósio Brasileiro de Pesquisa Operacional*, 2009, pp. 2515–2526
17. L.S. Buriol, M.J. Hirsch, P.M. Pardalos, T. Querido, M.G.C. Resende, M. Ritt, A biased random-key genetic algorithm for road congestion minimization. *Optim. Lett.* **4**(4), 619–633 (2010)
18. D.G. Cantor, M. Gerla, Optimal routing in a packet-switched computer network. *IEEE Trans. Comput.* **C-23**, 1062–1069 (1974)
19. M. Carey, A constraint qualification for a dynamic traffic assignment model. *Transport. Sci.* **20**, 55–88 (1986)
20. M. Carey, Nonconvexity of the dynamic traffic assignment problem. *Transport. Res.* **26B**, 127–133 (1992)
21. Y.P. Chan, Optimal travel time reduction in a transport network: an application of network aggregation and branch-and-bound techniques. *Research Report R*, vol. 69, pp. 693–695, 1969
22. C.K. Chau, K.M. Sim, The price of anarchy for non-atomic congestion games with symmetric cost maps and elastic demands. *Oper. Res. Lett.* **31**(5), 327–334 (2003)
23. N. Christofides, A. Mingozzi, P. Toth, Exact algorithms for the vehicle routing problem based on the spanning tree and shortest path relaxations. *Math. Program.* **20**, 255–282 (1981)
24. E.G. Coffman Jr, *Scheduling in Computer and Job Shop Systems* (Wiley, New York, 1976)
25. J.R. Correa, A.S. Schulz, N.E. Stier-Moses, Selfish routing in capacitated networks. *Math. Oper. Res.* **29**(4), 961–976 (2004)
26. S. Dafermos, Convergence of a network decomposition algorithm for the traffic equilibrium model, in *Proceedings of the 8th International Symposium on Transportation and Traffic Theory*, Toronto, ed. by V.F. Hurdle et al. (University of Toronto Press, Toronto, 1983), pp. 43–156
27. C. Daganzo, The cell transmission model: a simple dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transport. Res.* **28B**(4), 269–287 (1994)

28. C. Daganzo, The cell transmission model, Part II: network traffic. *Transport. Res.* **29B**(2), 79–93 (1995)
29. G.B. Dantzig, J.H. Ramser, The truck dispatching problem. *Manag. Sci.* **6**(1), 80–91 (1959)
30. J. Desrosiers, Y. Dumas, M.M. Solomon, F. Soumis, Time constrained routing and scheduling, in *Network Routing, Handbooks in Operations Research and Management Science*, ed. by M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (North-Holland, Amsterdam, 1995), pp. 35–139
31. M. Desrochers, J. Desrosiers, M.M. Solomon, A new optimization algorithm for the vehicle routing problem with time windows. *Oper. Res.* **40**, 342–354 (1992)
32. M. Dror, Note on the complexity of the shortest path models for column generation in VRPTW. *Oper. Res.* **42**, 977–978 (1994)
33. J. Eliasson, L.-G. Mattsson, Equity effects of congestion pricing: Quantitative methodology and a case study for Stockholm. *Transport. Res. A* **40**(7), 602–620 (2006)
34. M. Ericsson, M.G.C. Resende, P.M. Pardalos, A genetic algorithm for the weight setting problem in OSPF routing. *J. Combin. Optim.* **6**(3), 299–333 (2002)
35. S.P. Evans, Derivation and analysis of some models for combining trip distribution and assignment. *Transport. Res.* **10**, 37–57 (1976)
36. M.L. Fisher, R. Jaikumar, A generalized assignment heuristic for vehicle routing. *Networks (Wiley Online Library)* **11**(2), 109–124 (1981)
37. L.R. Ford Jr, D.R. Fulkerson, Solving the transportation problem. *Manag. Sci.* **3**(1), 24–32 (1956)
38. M. Florian, D. Hearn, Network equilibrium models and algorithms, in *Handbooks in OR and MS*, vol. 8, Chap. 6, ed. by M.O. Ball et al. (North-Holland, Amsterdam, 1995), pp. 485–550
39. M. Florian, D.W. Hearn, Traffic assignment: equilibrium models, in *Pareto Optimality, Game Theory and Equilibria* (Springer, New York, 2008) (pp. 571–592)
40. M. Frank, P. Wolfe, An algorithm for quadratic programming. *Nav. Res. Logist. Q.* **3**(1–2), 95–110 (1956), Wiley
41. R.S. Garfinkel, G.L. Nemhauser, *Integer Programming*, vol. 4 (Wiley, New York, 1972)
42. J. Geunes, P.M. Pardalos, *Supply Chain Optimization*, vol. 98 (Springer, New York, 2005)
43. B.E. Gillett, L.R. Miller, A heuristic algorithm for the vehicle dispatch problem. *Oper. Res.* **22**, 240–349 (1974)
44. A. Glazer, E. Niskanen, Parking fees and congestion. *Reg. Sci. Urban Econ.* **22**, 123–132 (1992)
45. B. Golden, A problem in network interdiction. *Nav. Res. Logist. Q.* **25**(4), 711–713 (1978), Wiley
46. B.L. Golden, A minimum-cost multicommodity network flow problem concerning imports and exports. *Networks (Wiley)* **5**(4), 331–356 (1975)
47. C.P. Gomes, B. Selman, N. Crato, Heavy-tailed distributions in combinatorial search, in *Principles and Practice of Constraint Programming-CP97* (Springer, Berlin/Heidelberg 1997), pp. 121–135
48. P. Goodwin, Congestion charging in central London: lessons learned. *Plann. Theor Pract.* **5**(4), 501–505 (2004)
49. E. Hadjiconstantinou, N. Christofides, An exact algorithm for general, orthogonal, two-dimensional knapsack problems. *Eur. J. Oper. Res. (Elsevier)* **83**(1), 39–56 (1995)
50. R. Hamerslag, *Prognosemodel voor het personenvervoer in Nederland* (Technische Hogeschool Delft, Delft, 1971)
51. D.W. Hearn, S. Lawphongpanich, J.A. Ventura, Restricted simplicial decomposition: computation and extensions, in *Computation Mathematical Programming* (Springer, Berlin/Heidelberg, 1987), pp. 99–118
52. D.W. Hearn, S. Lawphongpanich, J.A. Ventura, Finiteness in restricted simplicial decomposition. *Oper. Res. Lett. (Elsevier)* **4**(3), 125–130 (1985)
53. D.W. Hearn, M.B. Yildirim, A toll pricing framework for traffic assignment problems with elastic demand. *Appl. Optim.* **63**, 135–143 (2001)

54. D.W. Hearn, M.B. Yildirim, M.V. Ramana, L.H. Bai, Computational methods for congestion toll pricing models. In *Proceedings of the Intelligent Transportation Systems*, 2001, pp. 257–262. IEEE
55. D.W. Hearn, M.V. Ramana, in *Solving Congestion Toll Pricing Models*, University of Florida, Department of Industrial & Systems Engineering
56. D.W. Hearn, J. Ribera, Convergence of the Frank–Wolfe method for certain bounded variable traffic assignment problems. *Transport. Res. B Methodol.* (Elsevier) **15**(6), 437–442 (1981)
57. M. Held, R.M. Karp, The traveling salesman problem and minimum spanning trees: Part II. *Math. Program.* **1**, 6–25 (1971)
58. J.M. Henderson, R.E. Quandt, *Microeconomic Theory: A Mathematical Approach* (McGraw-Hill, New York, 1958)
59. D.A. Hensher, K.J. Button, *Handbook of Transport Modelling*, 2nd edn. (Emerald, Bingley, 2007)
60. C.A. Holloway, An extension of the Frank–Wolfe method of feasible directions. *Math. Program.* **6**, 14–27 (1974)
61. S.P. Hoogendoom, P.H. Bovy, State-of-the-art of vehicular traffic flow modelling. In *Proceedings of the Institution of Mechanical Engineers. Part I: J. Syst. Control Eng.* **215**(4), 283–303
62. B. Kallehauge, J. Larsen, O.B. Madsen, M.M. Solomon, *Vehicle Routing Problem with Time Windows* (Springer, New York 2005), pp. 67–98
63. D. Kim, P.M. Pardalos, A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Oper. Res. Lett.* (24), 195–203 (1999)
64. D. Kim, P.M. Pardalos, Dynamic slope scaling and trust interval techniques for solving concave piecewise linear network flow problems. *Network* (35–3), 216–222 (2000)
65. F. Knight, Some fallacies in the interpretation of social cost. *Q. J. Econ.* **38**(4), 582–606 (1924)
66. J. Kohl, *Der Verkehr und die Ansiedlungen der Menschen* (Dresden/Leipzig, 1841)
67. A.W.J. Kolen, A.H.G. Rinnoy Kan, H.W.J.M. Trienekens, Vehicle routing with time windows. *Oper. Res.* **35**, 266–273 (1987)
68. G. Laporte, M. Gendreau, J.Y. Potvin, F. Semet, Classical and modern heuristics for the vehicle routing problem. *Int. Trans. Oper. Res.* (Wiley) **7**(4–5), 285–300 (2000)
69. T. Larsson, A. Migdalas, M. Patriksson, A partial linearization method for the traffic assignment problem. Report LiTH-MAT-R-89-06, Optimization, Linköping Institute of Technology, Department of Mathematics, Linköping, 1989
70. E.L. Lawler, D.E. Wood, Branch and bound methods: a survey. *Oper. Res.* **14**(4), 699–719 (1966)
71. J. Leape, The London congestion charge. *J. Econ. Perspect.* **20**(4), 157–176 (2006)
72. L.J. Leblanc, *Mathematical Programming Algorithms for Large Scale Network Equilibrium and Network Design Problems* (Northwestern University, Evanston, 11973)
73. L.J. LeBlanc, K. Farhangian, Efficient algorithms for solving elastic demand traffic assignment problems and mode split-assignment problems. *Transport. Sci.* **15**(4), 306 (1981)
74. L.J. LeBlanc, R.V. Helgason, D.E. Boyce, Improved efficiency of the Frank–Wolfe algorithm for convex network programs. *Transport. Sci.* (Institute for Operations Research and the Management Sciences) **19**(4), 445–462 (1985)
75. K. LeBlanc Edward, J. Larry, An efficient approach to solving the road network equilibrium traffic assignment problem. *Transport. Res.* **9**(5), 309–318 (1975), Elsevier
76. H. Lévy-Lambert, Tarification des services à qualité variable: application aux péages de circulation. *Econometrica* **36**(3–4), 564–574 (1968)
77. J. Linderoth, T. Ralphs, *Noncommercial Software for Mixed Integer Linear Programming*. Optimization Online, 2005
78. D.G. Luenberger, *Investment Science* (Oxford University Press, New York, 1998)
79. L.-G. Mattsson, Road pricing: consequences for traffic, congestion and location, in *Road Pricing, the Economy and the Environment*, ed. by C. Jensen-Butler, B. Sloth, M.M. Larsen, B. Madsen, O.A. Nielsen, (Springer, Berlin, 2008), pp. 29–48

80. T.L. Magnanti, Combinatorial optimization and vehicle fleet planning: perspectives and prospects. *Networks* (Wiley Online Library) **11**(2), 179–213 (1981)
81. R.S. Markovits, Second-best theory and law & economics: an introduction. *Chi.-Kent L. Rev.* **73**, 3 (1997)
82. K. McAlloon, C. Tretkoff, G. Wetzel, Sports league scheduling, in *Proceedings of Third Illog International Users Meeting*, Paris, July 1997
83. D.K. Merchant, G. Nemhauser, A model and an algorithm for the dynamic traffic assignment problem. *Transport. Sci.* **12**, 183–199 (1978)
84. D.K. Merchant, G. Nemhauser, Optimality conditions for a dynamic traffic assignment model. *Transport. Sci.* **12**, 200–207 (1978)
85. D.L. Miller, A matching based exact algorithm for capacitated vehicle routing problems. *INFORMS J. Comput.* **7**(1), 1–9 (1995)
86. J.D. Murchland, Road network traffic distribution in equilibrium. *Math. Model Soc. Sci.* **8**, 145–183 (1970)
87. K. Murty, Solving the fixed charge problem by ranking the extreme points. *Oper. Res.* **(16)**, 268–279 (1968)
88. A. Nahapetyan, S. Langpopanich, Discrete-time dynamic traffic assignment problem with periodic planning horizon: system optimum. *J. Global Optim.* **38**(1), 41–60 (2007)
89. A. Nahapetyan, P.M. Pardalos, Adaptive dynamic cost updating procedure for solving fixed charge network flow problems. *Comput. Optim. Appl.* **39**(1), 37–50 (2008)
90. G. Nemhauser, M. Trick, Scheduling a major college basketball conference, Georgia Tech., Technical Report, 1997
91. S. Nguyen, An algorithm for the traffic assignment problem. *Transport. Sci.* **8**(3), 203 (1974)
92. S. Nguyen, others, Equilibrium traffic assignment for large scale transit networks. *Eur. J. Oper. Res.* **37**(2), 176–186 (1988)
93. S. Nguyen, L. James, *TRAFFIC: An Equilibrium Traffic Assignment Program* (University of Montreal, Montreal, 1975)
94. W. Ochoa-Rosso, *Applications of Discrete S Optimization Techniques to Cq Capital Investment and Network Synthesis Problems* (1968)
95. W. Ochoa-Rosso, A. Silva, Optimum project addition in urban transportation networks via descriptive traffic assignment models. Research Report R, vol. 44, 1968
96. T. Oksanen, A. Visala, Coverage path planning algorithms for agricultural field machines. *J. Field Robot.* **26**(8), 651–668 (2009)
97. K.R. Overgaard, Urban transportation planning traffic estimation. *Traffic Q.* **21**(2) (1967)
98. P.M. Pardalos, M.G.C. Resende, *Handbook of Applied Optimization* (Oxford University Press, Oxford, 2002)
99. P. Patriksson, The traffic assignment problem: models and methods (1994)
100. A.C. Pigou, *The Economics of Welfare*, vol. 2 (Cosimo, New York, 2006)
101. J. Renaud, F.F. Boctor, G. Laporte, A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS J. Comput.* **8**, 134–143 (1996)
102. T.M. Ridley, *An Investment Policy to Reduce the Travel Time in a Transportation Network* (Operations Research Center, University of California, Berkeley, 1965)
103. M.G. Richards, Congestion harging in London, in *The Policy and the Politics* (Palgrave Macmillan, Basingstoke, 2006)
104. T. Roughgarden, *Selfish Routing and the Price of Anarchy* (MIT, Cambridge, 2005)
105. T. Roughgarden, E. Tardos, Bounding the inefficiency of equilibria in nonatomic congestion games. *Games Econ. Behav.* **47**(2), 389–403 (2002)
106. D.M. Ryan, B.A. Foster, An integer programming approach to scheduling, in *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* (1981), pp. 269–280
107. D.M. Ryan, C. Hjorring, F. Glover, Extensions of the petal method for vehicle routing. *J. Oper. Res. Soc.* **44**, 289–296 (1993)
108. A. Sorokin, V. Boginski, A. Nahapetyan, P.M. Pardalos, Computational risk management techniques for fixed charge network flow problems with uncertain arc failures. *J. Comb. Optim.* **25**(1), 99–122 (2013)

109. P.A. Steembrink, *Optimization of Transport Networks* (Wiley, 1979)
110. P. Toth, D. Vigo (eds.), *The Vehicle Routing Problem*, vol. 9 (Society for Industrial and Applied Mathematics, 1987)
111. T. Tsekeris, S. Voß, Design and evaluation of road pricing. *Netnomic* **10**(1), 5–52 (2009)
112. E.J. Van Henten, J. Hemming, B.A. Van Tuijl, J.G. Kornet, J. Bontsema, Collision-free motion planning for a cucumber picking robot. *Biosyst. Eng.* **86**(2), 135–144 (2003)
113. J.A. Ventura, D.W. Hearn, Restricted simplicial decomposition for convex constrained problems. *Math. Program.* **59**(1), 71–85 (1993)
114. T.E. Verhoef, Second-best congestion pricing in general static transportation networks with elastic demands. *Reg. Sci. Urban Econ.* **32**, 281–310 (2002)
115. T.E. Verhoef, P. Nijkamp, P. Rietveld, Second-best congestion pricing: the case of an untolled alternative. *J. Urban Econ.* **40**(3), 279–302 (1996)
116. W.S. Vickrey, Congestion theory and transport investment. *Am. Econ. Rev.* **59**(2), 251–260 (1969)
117. B. Von Hohenbalken, Simplicial decomposition in nonlinear programming algorithms. *Math. Program.* **13**, 49–68 (1977)
118. S. Vougioukas, S. Blackmore, J. Nielsen, S. Fountas, A two-stage optimal motion planner for autonomous agricultural vehicles. *Precis. Agr.* **7**, 361–377 (2006)
119. J.G. Wardrop, Some theoretical aspects of road traffic research. *Oper. Res.* **4**(4), 72–73 (1953)
120. S.T. Waller, A.K. Ziliaskopoulos, A combinatorial user optimal dynamic traffic assignment algorithm. *Ann. Oper. Res.* **144**(1), 249–261 (2006)
121. B.-W. Wie, R.L. Tobin, D. Bernstein, T.L. Friesz, A comparison of system optimum and user equilibrium dynamic traffic assignments with schedule delays. *Transport. Res. C* **3**(6), 389–411 (1995)
122. W.I. Zangwill, Convergence conditions for nonlinear programming algorithms. *Manag. Sci.* **16**(1), 1–13 (1969)
123. Q.P. Zheng, A. Arulselvan, Discrete time dynamic traffic assignment models and solution algorithm for managed lanes. *J. Global Optim.* **51**(1), 47–68 (2011)

Complexity Issues on PTAS

Jianer Chen and Ge Xia

Contents

1	Introduction	724
2	Preliminary	726
3	Syntactic Characterizations of PTAS	728
3.1	Characterization of PTAS	729
3.2	Characterization of FPTAS	730
3.3	Characterization of EPTAS	732
4	Techniques for Designing PTAS Algorithms	733
4.1	Techniques for Designing FPTAS Algorithms	733
4.2	Lipton and Tarjan's Approach of Planar Separators	735
4.3	Baker's Approach of Layerwise Decomposition	737
4.4	Bidimensionality Theory	737
5	Computational Lower Bounds for PTAS	739
5.1	Problems That Do Not Have FPTAS	739
5.2	Problems That Do Not Have EPTAS	739
5.3	Lower Bounds on the Running Time of PTAS	741
5.4	Lower Bounds on the Running Time of EPTAS	742
6	Conclusion	743
	Cross-References	743
	Recommended Reading	744

J. Chen (✉)

Department of Computer Science and Engineering, Texas A&M University, College Station,
TX, USA

e-mail: chen@cs.tamu.edu

G. Xia

Department of Computer Science, Lafayette College, Easton, PA, USA

e-mail: gexia@cs.lafayette.edu; xiag@lafayette.edu

Abstract

This chapter surveys various complexity issues of *polynomial time approximation schemes* (PTAS). Also under consideration are the refined subclasses of PTAS, including *fully polynomial time approximation schemes* (FPTAS) and *efficient polynomial time approximation schemes* (EPTAS). The key questions under consideration are as follows: what optimization problems have PTASs, FPTASs, and EPTASs, and how efficient are these algorithms?

On the positive side, there are syntactic classes for characterizing PTAS, FPTAS, and EPTAS. Expressing a problem as a member in these classes automatically gives an approximation scheme for the problem. For many graph problems on planar graphs, there are powerful graph-theory techniques for designing PTAS and EPTAS algorithms, which can be extended to some generalizations of planar graphs. On the negative side, there are convincing evidences that, for a large set of NP-hard optimization problems, efficient approximation schemes do not exist. Even for many problems that have PTASs or EPTASs, their running time has a strong lower bound.

1 Introduction

According to the NP-completeness theory [31], many optimization problems of theoretical interest and practical importance are NP-hard, thus cannot be solved optimally in polynomial time unless $P = NP$. Many approaches have been proposed to cope with the NP-hardness of such problems. The most well studied among these approaches is polynomial time approximation, which involves computing a “good” approximation of an optimal solution in polynomial time.

A notable class of NP-hard optimization problems admits *polynomial time approximation schemes* (PTAS). A PTAS is an algorithm that, for any given instance of an optimization problem and a fixed relative error bound $\epsilon > 0$, produces a solution in polynomial time that is within a ratio $1 + \epsilon$ from being optimal. A large number of NP-hard optimization problems have PTASs, including, based on the recent research, the following well-known problems: Euclidean traveling salesman problem [1, 2], general multiprocessor job scheduling [14, 37], planar Steiner tree [8], planar Steiner forest [5], geometric k-clustering [44], multiple knapsack [11], distinguishing (sub)string selection [24], maximum subforest [55], and k-consensus clustering [18]. In particular, many important graph problems have PTASs on planar graphs and some generalizations of planar graphs, such as vertex cover, independent set, feedback vertex set, longest path, cycle packing, connected vertex cover, graph metric TSP, dominating set, and connected dominating set [20].

The complexity issues of PTAS have received a lot of attention recently. The running time of a general PTAS algorithm of approximation ratio $1 + \epsilon$ can be of the form $O(n^{t(\epsilon)})$, where n is the input size and $t(\epsilon)$ is a function of ϵ . The value of $t(\epsilon)$ can be very large even for moderate values of ϵ . For example, Downey [26] calculated the time complexity of a few well-known PTAS algorithms and showed

that for a moderate error bound of $\epsilon = 20\%$, the running time of these PTASs exceed $n^{10,000}$. Downey [26] gave the following list to illustrate the point:

- Arora [1] gave a PTAS for Euclidean traveling salesman problem with running time $O(n^{3,000/\epsilon})$, which is $O(n^{15,000})$ for $\epsilon = 20\%$.
- Chekuri and Khanna [11] gave a PTAS for multiple knapsack with running time $O(n^{12(\log(1/\epsilon)/\epsilon^8)})$, which is $O(n^{9,375,000})$ for $\epsilon = 20\%$.
- Shamir and Tsur [55] gave a PTAS for maximum subforest with running time $O(n^{2^{2^{1/\epsilon}}-1})$, which is $O(n^{958,267,391})$ for $\epsilon = 20\%$.
- Chen and Miranda [14] gave a PTAS for general multiprocessor job scheduling with running time $O(n^{(3mm!)^{m/\epsilon}+1})$, where m is the number of processors, which is $> O(n^{10^{60}})$ for $\epsilon = 20\%$ and $m = 4$.
- Erlebach et al. [29] gave a PTAS for maximum independent set on geometric graphs with running time $O(n^{4/\pi(1/\epsilon^2+1)^2(1/\epsilon^2+2)^2})$, which is $O(n^{523,804})$ for $\epsilon = 20\%$.

The astronomical growth in the running time of these PTAS algorithms even for a moderate error bound makes them quite infeasible for practical applications. Observing this fact, the class PTAS is further refined in terms of their running time.

A PTAS is referred to as a *fully polynomial time approximation scheme* (FPTAS) if its running time is bounded by a polynomial in both input size and $1/\epsilon$. A notable class of NP-hard optimization problems has FPTASs, including the well-known knapsack, subset-sum, and makespan problem on a fixed number of processors [34]. However, FPTAS does not exist for strongly NP-hard optimization problems unless $P = NP$ [31].

A PTAS is referred to as an *efficient polynomial time approximation scheme* (EPTAS) if the running time of the algorithm is bounded by a polynomial of the input size whose degree is independent of ϵ , that is, in the form $f(\epsilon)n^{O(1)}$, for any function f . EPTAS algorithms are obviously superior to PTAS algorithms of running time of the form $O(n^{t(\epsilon)})$ in the efficiency of the algorithms. In fact, many PTAS algorithms developed for NP-hard optimization problems are actually EPTAS algorithms. Moreover, there are a number of well-known NP-hard optimization problems for which early developed PTAS algorithms had running time of form $O(n^{t(\epsilon)})$, but later were improved to EPTAS algorithms. For example, Arora [2] improved his earlier PTAS algorithm for Euclidean traveling salesman problem [1] to an efficient and practical EPTAS algorithm. The PTAS algorithm for general multiprocessor job scheduling by Chen and Miranda [14] was recently improved by Klaus [37] to an EPTAS algorithm. On the other hand, Cai et al. [10] showed that there are problems that have PTASs but are unlikely to have EPTASs.

This chapter surveys various complexity issues of PTAS. The key questions under consideration are as follows:

What optimization problems have PTASs, FPTASs, and EPTASs, and how efficient are these algorithms?

On the positive side, there are syntactic classes for characterizing PTAS, FPTAS, and EPTAS. Expressing a problem as a member in these classes automatically gives an approximation scheme for the problem. For many graph problems on

planar graphs, there are powerful graph-theory techniques for designing PTASs and EPTASs, which can be extended to some generalizations of planar graphs. On the negative side, there are convincing evidences that, for a large set of NP-hard optimization problems, EPTASs do not exist. For many problems that have PTAS or EPTASs, their running time has a strong lower bound.

The rest of this chapter is organized as follows. [Section 2](#) gives the necessary terms and definitions. [Section 3](#) gives syntactic characterizations of PTAS, FPTAS, and EPTAS. [Section 4](#) discusses the techniques for designing PTAS and EPTAS algorithms. [Section 5](#) presents some recent results on the computational lower bounds of PTAS and EPTAS. The chapter concludes with a summary of the key results in [Sect. 6](#).

2 Preliminary

The problems under consideration in this chapter are NP optimization problem, defined as follows:

Definition 1 An *NP optimization problem* Q is a 4-tuple (I_Q, S_Q, f_Q, opt_Q) , where

- I_Q is the set of input instances. It is recognizable in polynomial time;
- For each instance $x \in I_Q$, $S_Q(x)$ is the set of feasible solutions for x . Given x , $S_Q(x)$ is recognizable in polynomial time.
- $f_Q(x, y)$ is the objective function mapping a pair $x \in I_Q$ and $y \in S_Q(x)$ to a nonnegative integer. The function f_Q is computable in polynomial time.
- $opt_Q \in \{\max, \min\}$. Q is called a *maximization problem* if $opt_Q = \max$ and a *minimization problem* if $opt_Q = \min$.

An *optimal solution* y for an instance $x \in I_Q$ of an optimization problem Q is a feasible solution in $S_Q(x)$ such that $f_Q(x, y)$ is optimized (maximized or minimized depending on opt_Q) among all solutions in $S_Q(x)$. Denote by $opt_Q(x)$ the value of $f_Q(x, y)$ for any optimal solution y to the instance x of the problem Q .

An algorithm A is an *approximation algorithm* for an optimization problem Q if, for any input instance x in I_Q , the algorithm A returns a feasible solution y in $S_Q(x)$. The approximation algorithm A has an *approximation ratio* r if for any instance x in I_Q , the solution y constructed by the algorithm A approximates $opt_Q(x)$ by a factor of r , that is

- $opt_Q(x)/f_Q(x, y) \leq r$ if Q is a maximization problem.
- $f_Q(x, y)/opt_Q(x) \leq r$ if Q is a minimization problem.

An optimization problem Q has a *polynomial time approximation scheme* (PTAS) if there is an algorithm A_Q that takes a pair (x, ϵ) as input, where $x \in I_Q$ is an instance of Q and $\epsilon > 0$ is a real number, and returns a feasible solution $y \in S_Q(x)$ for x such that the approximation ratio of the solution y is bounded by $1 + \epsilon$, and for any fixed $\epsilon > 0$, the running time of the algorithm A_Q is bounded by a polynomial in $|x|$ (but not necessarily a polynomial in $1/\epsilon$). A PTAS algorithm A_Q is called an FPTAS algorithm if the running time of A_Q is bounded by a polynomial

in both $|x|$ and $1/\epsilon$. A PTAS algorithm A_Q is called an EPTAS algorithm if the running time of A_Q is bounded by a polynomial in $|x|$ whose degree is independent of ϵ .

Throughout this chapter, a well-known NP optimization problem planar maximum independent set (planar MIS) is used to illustrate the complexity issues of PTAS. In planar MIS, one is given a planar graph and is asked to find a maximum set of vertices in the graph that do not induce any edge. The formal definition is given as follows:

Planar MIS

- I_Q : the set of planar graphs $G = (V, E)$.
- S_Q : $S_Q(G)$ is the collection of all independent sets in G , where an *independent set* of a graph G is a subset of vertices of G in which no two vertices are adjacent.
- f_Q : $f_Q(G, D)$ is equals to the number of vertices in the independent set D in G .
- opt_Q : \max .

Planar MIS is an NP optimization problem because one can easily verify that the size of any independent set of a graph $G = (V, E)$ is bounded by $|V|$, the set of planar graphs and the independent sets of planar graphs are recognizable in polynomial time, and the number of vertices in an independent set is computable in polynomial time. It is straightforward to prove that planar MIS is NP-hard since the decision version of planar MIS is NP-complete [31].

For an instance $G \in I_Q$ of planar MIS, an optimal solution is a maximum independent set in G . An approximation algorithm A for Planar MIS will return an independent set D of G . Since planar MIS is a maximization problem, the approximation ratio of A is $opt_Q(G)/f_Q(G, D)$, where $opt_Q(G)$ is the size of a maximum independent set of G and $f_Q(G, D)$ is the size of D . A PTAS for planar MIS is an algorithm which takes an instance $G \in I_Q$ and an error bound $\epsilon > 0$ and in polynomial time returns an independent set D of G whose size is $\geq opt_Q(G)/(1 + \epsilon)$.

Many complexity issues of PTAS are closely related to the *parameterized complexity theory*. Parameterized complexity theory [27] is a approach of dealing with NP optimization problems. In contrast to the traditional computational complexity theory that measures the complexity in terms of input size, parameterized complexity theory studies the computational complexity of optimization problems in terms of *both* instance size *and* a properly selected parameter. Parameterized complexity theory has drawn considerable attention because of its applications in developing practical algorithms. More recently, parameterized complexity theory has been used to obtain both positive and negative results on the approximability of NP-hard optimization problems.

A brief review of the terminologies in parameterized complexity is given in the following. For more details, see [27].

A *parameterized problem* Q , formally interpreted as a language, is a subset of $\Omega^* \times N$, where Ω is a finite alphabet set and N is the set of nonnegative integers. Each instance of Q is a pair (x, k) , where the nonnegative integer k is called the *parameter*. The parameterized problem Q is *fixed-parameter tractable* [27] if there

is an algorithm that decides whether an input (x, k) is a yes instance of Q in time $f(k)|x|^{O(1)}$, where f is an arbitrary function. FPT is the class of all fixed-parameter tractable problems.

The inherent difficulty for efficiently solving certain problems even in the presence of small parameters has led to the common belief that certain parameterized problems are not fixed-parameter tractable. A parameterized intractability hierarchy, the W -hierarchy $\bigcup_{t \geq 0} W[t]$, has been introduced and studied, where $W[t] \subseteq W[t+1]$ for all $t \geq 0$. The class FPT of fixed-parameter tractable problems lies at the bottom of the W -hierarchy (the 0th level). The classes $W[t]$, for $t \geq 1$, capture the levels of parameterized intractability of parameterized problems. The notion of W -hardness is defined under FPT reductions for those classes. A parameterized problem Q is $W[h]$ -hard if every parameterized problem in $W[h]$ is FPT -reducible to Q .

A large number of $W[1]$ -hard parameterized problems have been identified [27] including many important NP-hard problems such as INDEPENDENT SET, CLIQUE, DOMINATING SET, and SET COVER. Now it is commonly accepted that no $W[1]$ -hard problem is fixed-parameter tractable [27], and $W[1] \neq FPT$ has become a standard hypothesis in parameterized complexity. Some of the recent results on the lower bounds of the running time of approximation schemes were obtained based on this hypothesis [10, 15].

There is an essential difference between approximation problems and parameterized problems: an approximation problem requires a solution for a given instance of the problem, while a parameterized problem only seeks a “yes/no” decision on an input. To make a meaningful comparison between approximability and parameterized complexity, a formal procedure is introduced to *parameterize* an optimization problem.

Definition 2 Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an NP optimization problem.

- If $opt_Q = \max$, then the *parameterized version* of Q is $Q_{para} = \{(x, k) \mid x \in I_Q, opt_Q(x) \geq k\}$.
- If $opt_Q = \min$, then the *parameterized version* of Q is $Q_{para} = \{(x, k) \mid x \in I_Q, opt_Q(x) \leq k\}$.

For example, the parameterized version of planar MIS is characterized by a set Q_{para} of tuples (G, k) , where G is a planar graph and k is an integer, satisfying the condition that the size of an maximum independent set of G is $opt_Q(G) \geq k$.

The above definition offers the possibility to study the relationship between the approximability and the parameterized complexity of NP optimization problems.

3 Syntactic Characterizations of PTAS

Approximability of optimization problems has been studied in terms of their syntactic descriptions. Earlier, Papadimitriou, and Yannakakis [45] introduced the syntactic classes MAXNP and MAXSNP of optimization problems.

Khanna et al. [39] showed that, under proper approximation-ratio-preserving reductions, MAXSNP captures APX, the class of NP optimization problems that can be approximated in polynomial time with constant approximation ratios. Following this important discovery, syntactic characterizations of PTAS, FPTAS, and EPTAS have been studied. This section surveys the main results along this line of research.

3.1 Characterization of PTAS

Khanna and Motwani [38] attempted to characterize PTAS by syntactic classes restricted to instances with a “planar structure.” They introduced three syntactic classes – Planar TMAX, Planar TMIN, and Planar MPSAT – to characterize PTAS. These classes are defined on disjunctive normal form (DNF) formulas, where each DNF formula is a disjunction of conjunctions (sum of products) over n Boolean variables. Both the formulas and the variables are weighted.

Definition 3 (TMAX) The class TMAX consists of NP optimization problems expressible as: given a collection C of DNF formulas whose input literals are all negative, find a maximum weight truth assignment T that satisfies each formula in C .

Definition 4 (TMIN) The class TMIN consists of NP optimization problems expressible as: given a collection C of DNF formulas whose input literals are all positive, find a minimum weight truth assignment T that satisfies each formula in C .

Definition 5 (MPSAT) The class MPSAT consists of NP optimization problems expressible as: given a collection C of DNF formulas, find a truth assignment T of weight at most W that maximizes the total weight of the satisfied formulas in C .

For any instance I in TMAX, TMIN, or MPSAT, the incidence graph of I is a bipartite graph that has a vertex for each variable and a vertex for each formula and an edge between them if the variable occurs in the formula. The class Planar TMAX, Planar TMIN, and Planar MPSAT are TMAX, TMIN, and MPSAT, respectively, restricted to instances with planar incidence graphs.

Khanna and Motwani [38] proved that:

Theorem 1 *Problems in Planar MPSAT, Planar TMAX, and Planar TMIN have PTASs.*

To illustrate the application of these syntactic classes, consider our recurring example of planar MIS. Planar MIS can be expressed by Planar TMAX as follows. Let G be a planar graph. Encode each vertex as a variable. Insert a vertex in the middle of each edge $e = (x, y)$ in G and label it by the formula $(\bar{x} \vee \bar{y})$. The resulting graph is the planar incidence graph of an instance in Planar TMAX. By Theorem 1, it follows that planar MIS has a PTAS.

3.2 Characterization of FPTAS

Since the early results on FPTAS for NP optimization problems [36, 54], researchers have been trying to characterize problems in FPTAS [3, 46, 56]. Earlier results in this research [3, 46] characterized the class FPTAS by certain polynomial time computable functions, but these results did not provide information on how to decide whether such functions exist for a given problem and how to develop an FPTAS for the problem (e.g., see [46]). A more recent approach by Woeginger [56] focused on problems that have a dynamic programming formulation. He showed that as long as such a problem satisfies certain structural conditions, the problem has FPTAS. However, Woeginger's characterization [56] only provides a necessary condition for problems with FPTASs.

Chen et al. [17] provided a stronger characterization. They showed that for NP optimization problems that are “scalable,” the notion of FPTAS is equivalent to a subclass of FPT.

Recall that a fixed-parameter tractable problem has an algorithm of running time of the form $f(k)|x|^{O(1)}$, where f is an arbitrary recursive function. The following is a subclass of FPT that requires the function $f(k)$ to be a polynomial.

Definition 6 An NP optimization problem Q is *polynomial fixed-parameter tractable* (PFPT) if its parameterized version is solvable in time $O(k^{O(1)}|x|^{O(1)})$.

Note that polynomial fixed-parameter tractability does not necessarily imply polynomial time computability because an NP optimization problem Q may have its optimal value $opt_Q(x)$ much larger than the instance size $|x|$. Cai and Chen [9] proved that every FPTAS problem is fixed-parameterized tractable. Their proof in fact shows that every FPTAS problem is in the class PFPT.

Chen et al. [17] showed that the condition PFPT actually coincides with the condition FPTAS when an NP optimization problem is “scalable” in the following sense.

Definition 7 An optimization problem $Q = (I_Q, S_Q, f_Q, opt_Q)$ is *scalable* if there are polynomial time computable functions g_1 and g_2 and a fixed polynomial q such that:

- For any instance $x \in I_Q$, and any integer $d \geq 1$, $x_d = g_1(x, d)$ is an instance of Q such that $|x_d| \leq q(|x|)$ and $|opt_Q(x_d) - opt_Q(x)/d| \leq q(|x|)$.
- For any solution y_d to the instance x_d , $y = g_2(x_d, y_d)$ is a solution to the instance x such that $|f_Q(x_d, y_d) - f_Q(x, y)/d| \leq q(|x|)$.

The property of scalability comes naturally for many NP optimization problems. Take $Q = \text{MAKESPAN}$ as an example. An instance x of MAKESPAN consists of n jobs with integral processing times t_1, t_2, \dots, t_n , respectively, and an integer m , the number of identical processors, and the problem is to schedule the jobs on the m processors so as to minimize the completion time (i.e., the *makespan*). For a given

instance $x = (t_1, t_2, \dots, t_n; m)$ of MAKESPAN and a given integer $d \geq 0$, the scaled instance is

$$x_d = (t'_1, t'_2, \dots, t'_n; m),$$

where $t'_i = \lceil t_i/d \rceil$ for $i = 1, 2, \dots, n$. So x_d is also an instance for MAKESPAN. A solution y_d for x_d is a partitioning of the jobs in x_d among the m processors: $y_d = (T'_1, \dots, T'_m)$, where T'_i is the set of jobs in x_d that are assigned to the i -th processor. The corresponding solution $y = (T_1, \dots, T_m)$ for x is the same partitioning (in terms of index) of the jobs in x among the m processors. Note that the makespan of y is equal to $\max_i \{\sum_{t_j \in T_i} t_j\}$, and the makespan of y_d is equal to $\max_i \{\sum_{t'_j \in T'_i} t'_j\}$. Thus

$$\begin{aligned} f_Q(x_d, y_d) &= \max_i \left\{ \sum_{t'_j \in T'_i} t'_j \right\} = \max_i \left\{ \sum_{t_j \in T_i} \lceil t_j/d \rceil \right\} \\ &\geq \max_i \left\{ \sum_{t_j \in T_i} t_j/d \right\} = \max_i \left\{ \sum_{t_j \in T_i} t_j \right\} / d \\ &= f_Q(x, y)/d, \end{aligned} \tag{1}$$

and

$$\begin{aligned} f_Q(x_d, y_d) &= \max_i \left\{ \sum_{t'_j \in T'_i} t'_j \right\} = \max_i \left\{ \sum_{t_j \in T_i} \lceil t_j/d \rceil \right\} \\ &\leq \max_i \left\{ \sum_{t_j \in T_i} (t_j/d + 1) \right\} \leq \max_i \left\{ \sum_{t_j \in T_i} t_j \right\} / d + n \\ &= f_Q(x, y)/d + n. \end{aligned} \tag{2}$$

It follows from Eqs. (1) and (2) that $|f_Q(x_d, y_d) - f_Q(x, y)/d| \leq n$. Similarly, it can be verified that the instances x and x_d satisfy $|opt_Q(x_d) - opt_Q(x)/d| \leq n$. The mappings from x to x_d and from y_d to y are computable in polynomial time. This verifies that MAKESPAN is scalable.

Chen et al. [17] proved that a “scalable” NP optimization problem is in PFPT if and only if it has FPTAS.

Theorem 2 ([17]) *Let Q be a scalable NP optimization problem. Then Q has an FPTAS if and only if Q is in PFPT.*

In general, the scalability property of an NP optimization problem can be checked in a straightforward manner, as shown above for MAKESPAN. To design

an FPTAS for a scalabe NP optimization problem, one can start by developing an PFPT algorithm for the parameterized version of the problem and then convert the PFPT algorithm into an FPTAS algorithm, following a simple procedure given in [17].

3.3 Characterization of EPTAS

As mentioned in Sect. 3.1, Khanna and Motwani [38] tried to capture the class PTAS by imposing a planar structure on syntactic classes TMAX, TMIN, and MPSAT. In a parallel approach to that of Khanna and Motwani [38], Chen et al. [17] tried to characterize EPTAS by imposing a planar structure on the W -hierarchy in parameterized complexity.

The W -hierarchy has been defined based on satisfiability problems on bounded-depth circuits of a special leveled form: all inputs are in level 0, all AND and OR gates are organized into alternating levels, and edges only going from a level to the next level [13]. The *depth* of a node v is d if v is at the d -th level. The *depth* of a circuit is d if its output node has depth d . A circuit is a Π_h -circuit if it has depth h and its output node is an AND gate. The *weight* of a truth assignment T is the number of variables that are TRUE. A Π_h -circuit is a Π_h^+ -circuit if all of its inputs are labeled by positive literals and is a Π_h^- -circuit if all of its inputs are labeled by negative literals. A Π_h -circuit α is *planar* if α becomes a planar graph after removing the output gate in α .

Chen et al. [17] defined the following syntactic optimization classes:

Definition 8 (Planar MIN- $W[h]$) The class Planar MIN- $W[h]$ consists of optimization problems Q such that each instance of Q can be expressed as a planar Π_h^+ -circuit α , and the problem is to look for a truth assignment of minimum weight that satisfies α .

Definition 9 (Planar MAX- $W[h]$) The class Planar MAX- $W[h]$ consists of optimization problems Q such that each instance of Q can be expressed as a planar Π_h^- -circuit α , and the problem is to look for a truth assignment of maximum weight that satisfies α .

Definition 10 (Planar $W[h]$ -SAT) The class Planar $W[h]$ -SAT consists of optimization problems Q such that each instance of Q can be expressed as a planar Π_h -circuit α , and the problem is to look for a truth assignment that satisfies the largest number of depth- $(h - 1)$ gates in the circuit α .

Note that the classes Planar MIN- $W[h]$, Planar MAX- $W[h]$, and Planar $W[h]$ -SAT are the planar versions of the problem $wcs(h)$, which is the representative problem for the h -th level $W[h]$ of the W -hierarchy in parameterized complexity theory [27].

The class Planar $W[h]$ -SAT captures the optimization problems where the objective is to construct a solution that satisfies the maximum number of constraints. The classes Planar MIN- $W[h]$ and Planar MAX- $W[h]$ capture the optimization problems where the objective is to construct an optimal (minimum or maximum) solution that satisfies all the constraints. Together, Planar MIN- $W[h]$, Planar MAX- $W[h]$, and Planar $W[h]$ -SAT capture many optimization problems on planar graphs.

Based on dynamic programming techniques similar to those in the Baker's approach [4] (discussed in Sect. 4.3), Chen et al. [17] showed that:

Theorem 3 *All optimization problems in the syntactic classes Planar MIN- $W[h]$, Planar MAX- $W[h]$, and Planar $W[h]$ -SAT have EPTAS.*

Take planar MIS as an example. An instance G of planar MIS can be converted into a planar Π_2^- -circuit α_G as follows: make each vertex v in G an input \bar{v} of α_G , replace each edge (v, w) in G by an OR gate with the two inputs \bar{v} and \bar{w} , and connect the OR gate to the unique output AND gate of the circuit α_G . It is easy to see that the circuit α_G is planar and the maximum independent sets of the graph G correspond to the maximum weight assignments that satisfy the circuit α_G . This means that planar MIS is in planar max- $W[2]$, and by Theorem 3, planar MIS has an EPTAS.

4 Techniques for Designing PTAS Algorithms

The main techniques for designing FPTAS include pseudo-polynomial time algorithms and approximation by scaling. These techniques will be demonstrated by the knapsack problem.

Many NP-hard graph problems become more tractable when the underlying graphs are planar, that is, can be drawn in the plane without edge crossing. For example, although maximum independent set on general graphs is unlikely to be approximable within $|V|^{1-\epsilon}$ for any $\epsilon > 0$ [33], the planar version of the problem, Planar MIS, has EPTAS. There are powerful graph-theory techniques for developing PTASs for a broad range of problems on planar graphs and some generalizations of planar graphs.

The two main techniques of designing PTAS algorithms for problems on planar graphs are Lipton-Tarjan's graph separators [41] and Baker's layerwise decomposition [4]. These techniques have been enhanced and generalized by bidimensionality theory [20] to obtain EPTASs for many problems on a broad family of graphs.

4.1 Techniques for Designing FPTAS Algorithms

In this subsection, major techniques for designing FPTAS algorithms are presented, which include pseudo-polynomial time algorithms and approximation by scaling.

The techniques are demonstrated with the knapsack problem, which is defined as follows:

Knapsack

I_Q : tuples $x = \langle s_1, \dots, s_n; v_1, \dots, v_n; B \rangle$, where s_i, v_j, B are integers.

S_Q : $S_Q(x)$ is all sets $S \subseteq \{1, \dots, n\}$ such that $\sum_{i \in S} s_i \leq B$.

f_Q : $f_Q(x, S) = \sum_{i \in S} v_i$.

opt_Q : \max .

In other words, the goal of the knapsack problem is to take the maximized value with a knapsack of size B , given n items of size s_i and value v_i , $1 \leq i \leq n$. For a subset $S \subseteq \{1, \dots, n\}$, $\sum_{i \in S} s_i$ is referred to as the *size* of S and $\sum_{i \in S} v_i$ the *value* of S .

There is a classic dynamic programming algorithm *Knapsack-Dyn* for solving the knapsack problem, which works as follows. Let $V = \sum_{i=1}^n v_i$. For each index i , $1 \leq i \leq n$ and for each value $v \leq V$, consider the following question:

Question $K(i, v)$

Is there a subset S of $\{1, \dots, i\}$ such that the size of S is at most B and the value of S is equal to v ?

The answer to question $K(i, v)$ is “yes” if and only if at least one of the following two cases is true:

1. There is a subset S' of $\{1, \dots, i-1\}$ such that the size of S' is at most B and the value of S' is equal to v (in this case $S = S'$).
2. There is a subset S'' of $\{1, \dots, i-1\}$ such that the size of S'' is at most $B - s_i$ and the value of S'' is equal to $v - v_i$ (in this case $S = S'' \cup i$).

Therefore, the answer to the question $K(i, v)$ can be obtained if the answers to the questions $K(i-1, v')$ for all values v' are known. Using dynamic programming, the algorithm *Knapsack-Dyn* starts with answers to $K(0, v)$ which are always “no” for $v > 0$ and “yes” for $K(0, 0)$. The answers to $K(i, v)$ for $i > 0$ then are inductively computed based on the answers to $K(i-1, v')$ for all $0 \leq v' \leq V$. The solution to the knapsack problem corresponds to the largest value v such that $K(n, v)$ is “yes.”

The *Knapsack-Dyn* algorithm runs in time $O(nV)$ where $V = \sum_{i=1}^n v_i$. Unfortunately, since the value V can be larger than any polynomial of n , *Knapsack-Dyn* is not a polynomial time algorithm in terms of the input length n .

In fact, *Knapsack-Dyn* represents a typical method for solving a class of optimization problems.

Definition 11 Let $Q = (I_Q, S_Q, f_Q, opt_Q)$ be an optimization problem. An algorithm A solving Q runs in *pseudo-polynomial time* if the running time of the algorithm A is bounded by a polynomial in $\text{length}(x)$ and $\max(x)$, where $\text{length}(x)$ is the length of the binary representation of x and $\max(x)$ is the largest number that appears in the input x . In this case, we say that the problem Q is solvable in pseudo-polynomial time.

For the knapsack problem, $V = \sum_{i=1}^n v_i \leq n \cdot \max(x)$. So *Knapsack-Dyn* is a pseudo-polynomial time algorithm for knapsack. In other words, knapsack is solvable in pseudo-polynomial time.

The *Knapsack-Dyn* algorithm can be turned into an approximation algorithm for knapsack that runs in polynomial time. This involves *scaling*: dividing each value v_i by a sufficiently large number K so that $v'_i = \lfloor v_i/K \rfloor$. In doing so, V is scaled down to $V' = V/K$. By selecting a sufficiently large $K > V/(cn^d)$ for some constant c and d , *Knapsack-Dyn* runs in time $O(cn^{d+1})$, a polynomial in n . However, one has to pay a price for “scaling”: because of the floor operation $\lfloor \cdot \rfloor$, one loses precision; an optimal solution for the scaled instance may not be an optimal solution for the original instance. Furthermore, the larger the value K , the better the running time of the *Knapsack-Dyn* algorithm, but also the more precision would be lost. By carefully selecting the value of K (e.g., $K = V/[(1 + 1/\epsilon)n^2]$), it can be shown that while still keeping the running time of the algorithm *Knapsack-Dyn* polynomial of n , one can also bound the approximation ratio to $1 + \epsilon$. This procedure is summarized in the following theorem.

Theorem 4 *For any input instance α of the knapsack problem and for any real number $\epsilon > 0$, there is an approximation algorithm that runs in time $O(n^3/\epsilon)$ and produces a solution to α with approximation ratio bounded by $1 + \epsilon$.*

4.2 Lipton and Tarjan’s Approach of Planar Separators

Lipton and Tarjan [40] in a celebrated paper proved that a planar graph G has a separator of $O(\sqrt{n})$ vertices that separates the graph into two subgraphs of roughly the same size. More precisely, they proved the following theorem.

Theorem 5 *The vertices of a planar graph $G = (V, E)$ can be partitioned into three sets A , B , and S such that*

1. *S contains no more than $\sqrt{8n}$ vertices.*
2. *Neither A nor B contains more than $2n/3$ vertices.*
3. *Vertices in A are not adjacent to vertices in B .*

Moreover, there is an $O(n)$ -time algorithm that, given a planar graph G , constructs the triple (A, B, C) as above.

Taking advantage of such a separator, Lipton and Tarjan [41] developed an approach for designing PTAS for certain problems on planar graphs. In this approach, one partitions a planar graph into three sets A , B , and S as mentioned above, then recursively approximates the solutions on the smaller subgraphs induced by A and B , and finally combine the approximated solutions to the subgraphs into an approximated solution to the original graph.

For this approach to work, the optimal solution of the problem has to be large enough (in the order of $\Theta(n)$) such that the combining process does not increase the approximation ratio. Furthermore, the trade-off between the time complexity and the approximation ratio offered by the Lipton and Tarjan’s separator approach is often unfavorable. A small or moderate approximation ratio could entail a very high time complexity, making the approach infeasible for practical applications.

Let us briefly describe how to apply Lipton and Tarjan's approach to planar MIS (for more details, see [12, 41]). Given a planar graph G of n vertices, one finds a separator S that splits G into two subgraphs A and B each containing no more than $2n/3$ vertices. Recursively split A and B until the number of vertices in each connected component is bounded by a function of ϵ (e.g., $O(1/\epsilon^2)$). Find a maximum independent set in each connected component by brute force. The union of the independent sets of the components yields an independent set of G . The details of the algorithm is given in the following.

Algorithm PlanarIndSet

Input: a planar graph $G = (V, E)$ and a constant ϵ

Output: an independent set D of G

1. If $(|V| \leq K)$ where $K = O(1/\epsilon^2)$ to be determined later, **then**
find a maximum independent set D in G by exhaustive search;
return D .
 2. Partition V into (A, B, S) as in [Theorem 5](#)
 3. Recursively call **PlanarIndSet** on the subgraph G_A induced by A to find an independent set D_A and recursively call **PlanarIndSet** on the subgraph G_B induced by B to find an independent set D_B .
 4. Return $D_A \cup D_B$.
-

The running time of the **PlanarIndSet** algorithm is dominated by the time spent on solving all connected components by brute force, which is polynomial in n for any fixed constant ϵ .

Let F be the set of vertices in the separators constructed by **PlanarIndSet**. Let $opt(G)$ be the size of the maximum independent set of G . One verifies the following:

Fact 1. $n/4 \leq opt(G) \leq |D| + |F|$.

Fact 2. $|F| \leq cn/\sqrt{K}$ for some constant c .

Thus, the approximation ratio of **PlanarIndSet** is

$$\begin{aligned}
\frac{opt(G)}{|D|} &\leq \frac{|D| + |F|}{|D|} \\
&= 1 + \frac{|F|}{|D|} \\
&\leq 1 + \frac{|F|}{opt(G) - |F|} \\
&\leq 1 + \frac{|F|}{n/4 - |F|} \\
&\leq 1 + \frac{cn/\sqrt{K}}{n/4 - cn/\sqrt{K}} \\
&= 1 + \frac{1}{\frac{\sqrt{K}}{4c} - 1}.
\end{aligned} \tag{3}$$

Set $K = (4c)^2(1+1/\epsilon)^2$ and from Eq. (3), $\frac{opt(G)}{|D|} \leq 1+\epsilon$, that is, the approximation ratio of **PlanarIndSet** is $(1 + \epsilon)$. This yields a PTAS for planar MIS (it is in fact an EPTAS). However, this algorithm is hardly practical even for moderate values of ϵ . For example, an analysis by [12] shows that the running time is $O(n \log n + 2^{5184(1+1/\epsilon)^2} n)$.

4.3 Baker's Approach of Layerwise Decomposition

A later approach by Baker [4] is based on layerwise decomposition of plane graphs (i.e., a planar graph embedded in the plane) into k -outerplanar subgraphs and dynamic programming techniques. Baker observed that the vertices of an embedded plane graph G can be arranged in a layered structure as follows:

The vertices on the outer face of G give the first layer. By peeling the first layer, i.e., removing the vertices in the first layer, one obtains a plane subgraph G' of G (which may have more than one connected components). Now the first layer of G' forms the second layer for G . By peeling the second layer of G , one obtains the third layer, and so on.

For any given $0 \leq i < k$, after the vertices in layers $i, i+k, i+2k, \dots$ are removed, the plane graph is divided into a set of subgraphs each of which has at most k layers and becomes a k -outerplanar graph. A k -outerplanar graph has a bounded treewidth, and hence many NP-hard graph problems can be solved in polynomial time on a k -outerplanar graph by dynamic programming. The final solution is constructed from the solutions to the subgraphs. After trying at most k different values of i , one can find a solution which is at least $(1 + 1/k)$ -optimal. By adjusting the value of k , the algorithm can approximate the optimal solution to within ratio $1 + \epsilon$ for any $\epsilon > 0$ in time $2^{O(1/\epsilon)} n^{O(1)}$, which yields an EPTAS. Eppstein [28] has generalized Baker's approach to work on graphs of bounded local treewidth.

Baker [4] demonstrated her approach by applying it to planar MIS. The result was an $O(8^{1/\epsilon} n/\epsilon)$ -time EPTAS for planar MIS, which is much more efficient than the PTAS obtained from Lipton and Tarjan's approach.

4.4 Bidimensionality Theory

Recently, a meta-algorithmic framework called *bidimensionality theory* has received a lot of attention. Bidimensionality theory was introduced by Demaine et al. [19] to develop subexponential-time parameterized algorithms for problems on H -minor-free graphs (graphs exclude any fixed minor). Bidimensionality theory has subsequently been used by Demaine and Hajiaghayi [20] to obtain EPTASs for bidimensional problems. Recently, Fomin et al. [30] have used it to derive linear kernels for bidimensional problems [30]. Bidimensionality seems to capture the core of those problems that become more tractable on planar graphs.

Bidimensionality theory provides powerful algorithmic techniques for a broad range of bidimensional problems on planar graphs and some generalizations of

planar graphs, including bounded genus graphs and H -minor-free graphs. A brief definition of bidimensionality is given below. For the exact definition, see [22].

Definition 12 A parameterized graph problem is *bidimensional* if (1) the parameter is $\Omega(r^2)$ on an $r \times r$ “grid-like” graph, and (2) the parameter does not increase when an edge of the graph is contracted (or deleted in some cases).

Examples of bidimensional problems include natural problems such as vertex cover, independent set, feedback vertex set, longest path, cycle packing, connected vertex cover, graph metric TSP, dominating set, and connected dominating set.

The algorithmic applications of bidimensional theory build on the deep theory of graph minor by Robertson and Seymour [47–53]. In particular, there are two key structural results that link bidimensionality with its algorithmic applications.

Theorem 6 ([21, 23]) Every H -minor-free graph of treewidth w has an $\Omega(w) \times \Omega(w)$ grid as a minor.

Theorem 7 ([21]) If the parameterized version of a graph optimization problem Q is bidimensional, then for every graph G in the associated graph family, the treewidth of G is bounded by $O\left(\sqrt{\text{opt}_Q(G)}\right)$.

Based on the above structural results and by enhancing and generalizing techniques from Lipton-Tarjan’s planar separator [41] and Baker’s layerwise decomposition [4], Demaine and Hajiaghayi proved that:

Theorem 8 ([20]) There is an $(1 + \epsilon)$ -approximation scheme with running time $h(1/\epsilon)n^{O(1)}$ for any bidimensional optimization problem that (1) is computable in $h(w)n^{O(1)}$ on a graph of treewidth w , (2) has a constant ratio approximation, and (3) satisfies the following separation properties:

- The optimal solution on disconnected components equals to the union of the optimal solution of each connected component.
- Let C be a separator, and let X be the union of some connected components in $G - C$. The optimal solution to $G - X$ and the optimal solution to $G - X - C$ differ by $O(|C|)$ in size.
- The optimal solution to G induced on X and the optimal solution to X differ by $O(|C|)$ in size.

As a corollary, it follows that:

Corollary 1 ([20]) Bidimensional problems that satisfy the properties (1)–(3) listed above have EPTASS on planar graphs and some generalizations of planar graphs.

Usually, it is easy to verify bidimensionality of a given problem. Take planar MIS as an example. In the parameterized version of the problem, one is given a planar

graph G and a parameter k and is asked if there is an independent set S of at least k vertices. Planar MIS is bidimensional:

- For an $r \times r$ “grid-like” planar graph, the size of the maximum independent set is at least $\Omega(r^2)$ because of the four-color theorem.
- Contracting an edge of G does not increase the size of the maximum independent set of G .

It is easy to verify that planar MIS satisfies the separation properties. Furthermore, planar MIS can be solved on a graph of treewidth k in time $O(2^k n)$ [7], and planar MIS has a simple constant ratio approximation from five-coloring the planar graph. It follows from Corollary 1 that planar MIS has an EPTAS. The same techniques prove that maximum independent set has an EPTAS on apex-minor-free graphs, which are generalizations of planar graphs.

5 Computational Lower Bounds for PTAS

In this section, negative aspects of the complexity of PTAS are discussed. The emphasis is placed on deriving computational lower bounds on approximation schemes for NP-hard optimization problems.

5.1 Problems That Do Not Have FPTAS

First, many NP-hard optimization problems do not have FPTAS unless $P = NP$.

Definition 13 An NP-hard optimization problem Q is said to be NP-hard in the strong sense (strongly NP-hard) if the problem remains NP-hard when all of its parameters are polynomially bounded in the input size.

Theorem 9 ([31]) *If an optimization problem Q is strongly NP-hard, then Q does not have a fully polynomial time approximation scheme unless $P = NP$.*

It is easy to see that planar MIS is strongly NP-hard because all parameters are polynomially bounded. Thus, by Theorem 9, planar MIS does not have an FPTAS unless $P = NP$.

5.2 Problems That Do Not Have EPTAS

As mentioned in the Introduction, the running time of a PTAS can be quite infeasible for practical applications. On the other hand, the complexity of EPTAS is much more manageable. So the question is

Does every problem that admits a PTAS also has an EPTAS?

To this question, the answer is “very unlikely.”

Theorem 10 ([6, 9]) *If an optimization problem Q has an EPTAS, then the parameterized version of the problem is fixed-parameter tractable.*

Proof Let $Q = (I_Q, S_Q, f_Q, \text{opt}_Q)$ be a minimization problem. Let A be an EPTAS algorithm for Q that runs in time $f(\epsilon)n^{O(1)}$ for some function f . Let $Q_{\text{para}} = \{(x, k) \mid x \in I_Q, \text{opt}_Q(x) \leq k\}$ be the parameterized version of Q . Consider the following algorithm A_{para} for Q_{para} :

Algorithm A_{para} :

On an instance (x, k) of Q_{para} , call the EPTAS algorithm A on x and $\epsilon = 1/(k+1)$. Suppose that A returns a solution y in $S_Q(x)$. If its value is $f_Q(x, y) \leq k$, then return “yes”; otherwise, return “no.”

The approximation ratio of A_{para} is $\epsilon = 1/(k+1)$. So

$$1 \leq f_Q(x, y)/\text{opt}_Q \leq 1 + \frac{1}{k+1}. \quad (4)$$

If $f_Q(x, y) \leq k$, then

$$\text{opt}_Q \leq f_Q(x, y) \leq k,$$

and hence (x, k) is a “yes” instance of Q_{para} . If $f_Q(x, y) \geq k+1$, then by Eq. (4),

$$\text{opt}_Q \geq \frac{k+1}{k+2} \cdot f_Q(x, y) \geq \frac{k+1}{k+2} \cdot (k+1) > k$$

and hence (x, k) is a “no” instance of Q_{para} . Therefore, A_{para} is an algorithm for Q_{para} with running time $f(1/(k+1))n^{O(1)}$. This means that Q_{para} is fixed-parameter tractable. A similar argument applies when Q is a maximization problem. \square

Under the widely believed hypothesis that $W[1] \neq FPT$, this suggests that if the parameterized version of an optimization problem Q is $W[1]$ -hard, then Q does not have an EPTAS.

Take Planar TMIN as an example. Planar TMIN is one of the syntactic classes introduced by Khanna and Motwani [38] to characterize PTAS. So, Planar TMIN has a PTAS. Cai et al. [10] proved that the problem is $W[1]$ -hard and hence does not have EPTAS unless $W[1]=FPT$. Similarly, Cai et al. [10] showed that Planar TMAX and Planar MPSAT are $W[1]$ -hard and hence have no EPTAS unless $W[1]=FPT$.

As another example, Hunt et al. [35] gave an $n^{O(1/\epsilon)}$ -time PTAS for maximum independent set on unit disk graphs. Marx [42] showed that the parameterized version of maximum independent set is $W[1]$ -complete for the intersection graphs of unit disks and axis-parallel unit squares in the plane, and hence maximum independent set on unit disk graphs does not have an EPTAS.

5.3 Lower Bounds on the Running Time of PTAS

Suppose that a problem Q has PTAS but does not have EPTAS from [Theorem 10](#). But this does not answer the following question:

Is there a PTAS for Q whose running time is moderately exponential in $1/\epsilon$, for example, in the form of $f(1/\epsilon)n^{o(1/\epsilon)}$?

Answering this question, Chen et al. [15] gave strong lower bounds on the running time of PTAS. Their lower bounds are based on the widely believed *Exponential Time Hypothesis* (ETH) which states that n -variable 3SAT cannot be solved in subexponential time $2^{o(n)}$. Chen et al. [15] defined a subclass of W[1]-hard, called $W_l[1]$ -hard, consisting of problems that are W[1]-hard under *linear FPT reductions*: FPT reductions where the parameters before and after the reduction are linearly related. They proved the following lower bounds for $W_l[1]$ -hard parameterized problems.

Theorem 11 ([16]) *No $W_l[1]$ -hard parameterized problem can be solved in time $f(k)n^{o(k)}$ for any function f , unless ETH fails.*

From [Theorem 11](#), they prove the following theorem about the nonexistence of PTAS for optimization problems whose parameterized versions are $W_l[1]$ -hard.

Theorem 12 ([16]) *Let Q be an optimization problem. If the parameterized version of Q is $W_l[1]$ -hard, then Q has no PTAS of running time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any function f , unless ETH fails.*

Proof Let $Q = (I_Q, S_Q, f_Q, \text{opt}_Q)$ be a maximization problem such that the parameterized version Q_{para} of Q is $W_l[1]$ -hard.

Suppose to the contrary that Q has a PTAS algorithm A_Q of running time $f(1/\epsilon)n^{o(1/\epsilon)}$ for a function f . The algorithm A_Q can be used to derive the following algorithm A_{para} for the parameterized problem Q_{para} :

Algorithm A_{para} :

On an instance (x, k) of Q_{para} , call the PTAS algorithm A_Q on x and $\epsilon = 1/(2k)$. Suppose that A_Q returns a solution y in $S_Q(x)$. If $f_Q(x, y) \geq k$, then return “yes”; otherwise, return “no.”

We verify that the algorithm A_{para} solves the parameterized problem Q_{para} . Since Q is a maximization problem, if $f_Q(x, y) \geq k$, then obviously $\text{opt}_Q(x) \geq k$. Thus, the algorithm A_{para} returns a correct decision in this case. On the other hand, suppose $f_Q(x, y) < k$. Since $f_Q(x, y)$ is an integer, $f_Q(x, y) \leq k - 1$. Since A_Q is a PTAS for Q and $\epsilon = 1/(2k)$,

$$\text{opt}_Q(x)/f_Q(x, y) \leq 1 + 1/(2k).$$

It follows that (note that $f_Q(x, y) < k$)

$$\text{opt}_Q(x) \leq f_Q(x, y) + f_Q(x, y)/(2k) \leq k - 1 + 1/2 = k - 1/2 < k.$$

Thus, in this case the algorithm A_{para} also returns a correct decision. This proves that the algorithm A_{para} solves the parameterized version Q_{para} of the problem Q . The running time of the algorithm A_{para} is dominated by that of the algorithm A_Q , which by our hypothesis is bounded by $f(1/\epsilon)n^{o(1/\epsilon)} = f(2k)n^{o(k)}$. Thus, the $W_1[1]$ -hard problem Q_{para} is solvable in time $f(2k)n^{o(k)}$, which implies ETH fails by [Theorem 11](#).

The proof is similar for the case when Q is a minimization problem. \square

To demonstrate an application for [Theorem 12](#), consider the problem distinguishing substring selection (DSSP), which is NP-complete [32] and has applications in computational biology [25]. Recently, Deng et al. [24] (see also [25]) developed a PTAS algorithm for DSSP. The running time of the algorithm is bounded by $n^{O(1/\epsilon^6)}$. Obviously, such an algorithm is not practical even for moderate values of the error bound ϵ . Chen et al. [15] showed that the problem is $W_1[1]$ -hard. It follows from [Theorem 12](#) that the optimization problem DSSP has no PTAS of running time $f(1/\epsilon)n^{o(1/\epsilon)}$ for any function f unless ETH fails. Thus essentially, no PTAS for DSSP can be practically efficient even for moderate values of the error bound ϵ .

Recall the syntactic classes Planar TMAX, Planar TMIN, and Planar MPSAT introduced by Khanna and Motwani [38] to characterize PTAS. Khanna and Motwani [38] gave $n^{O(1/\epsilon)}$ -time PTASs for these problems. As discussed in the previous subsection, Planar TMAX, Planar TMIN, and Planar MPSAT are $W[1]$ -hard and hence have no EPTAS unless $W[1]=\text{FPT}$ [10]. Marx [43] gave a stronger lower bound. He proved that if there is a $\delta > 0$ such that any of these problems admits a $2^{(1/\epsilon)^{O(1)}}n^{O((1/\epsilon)^{1-\delta})}$ -time PTAS, then ETH fails.

In the previous subsection, it is shown that maximum independent set on unit disk graphs has a PTAS [35], but not an EPTAS [42]. Marx [43] further improved the inapproximability result by proving that maximum independent set on unit disk graphs does not have a PTAS of running time $2^{(1/\epsilon)^{O(1)}}n^{O((1/\epsilon)^{1-\delta})}$ for any $\delta > 0$ unless ETH fails.

5.4 Lower Bounds on the Running Time of EPTAS

For a problem Q that has EPTAS, the typical running time of the EPTAS is in the form of $2^{O(1/\epsilon)}n^{O(1)}$. One would still be interested in the following question:

Is there an EPTAS for Q whose running time is subexponential in $1/\epsilon$, for example, in the form of $2^{O((1/\epsilon)^{1-\delta})}n^{O(1)}$ for some $\delta > 0$?

Marx [43] proved that the answer to this question is “no” for some important problems. He showed that for several planar and geometric problems, including planar MIS and planar graph metric TSP, if there is a $\delta > 0$ such that any of these problems admits a $2^{O((1/\epsilon)^{1-\delta})}n^{O(1)}$ time PTAS, then ETH fails. This implies that the current $2^{O(1/\epsilon)}n^{O(1)}$ -time EPTASs for planar MIS and for planar graph metric TSP are essentially optimal.

6 Conclusion

In this chapter, various complexity issues concerning PTAS and its refined subclasses FPTAS and EPTAS are surveyed. On the positive side, there are syntactic characterizations and graph-theory techniques that can be used to obtain approximation schemes for optimization problems. On the negative side, parameterized intractability provides strong evidences that certain optimization problems do not have efficient approximation schemes. Many of these results point to a strong connection between approximability and parameterized complexity. The main results are summarized as follows:

- The syntactic classes Planar TMAX, Planar TMIN, and Planar MPSAT characterize PTAS [38].
- The syntactic class PFPT restricted to “scalable” problems characterizes FPTAS [17].
- The syntactic classes Planar MIN-W[h], Planar MAX-W[h], and Planar W[h]-SAT characterize EPTAS [17].
- Graph-theory techniques including Lipton and Tarjan’s planar separator [41] and Baker’s layerwise decomposition [4] lead to PTASs for problems on planar graphs.
- Bidimensionality theory provides general techniques for designing EPTAS for bidimensional problems on planar graphs and some generalizations of planar graphs [20].
- Strongly NP-hard problems do not have FPTASs unless P=NP [31].
- W[1]-hard problems do not have EPTASs unless W[1]=FPT [10].
- W₁[1]-hard problems do not have PTASs of running time $f(1/\epsilon)m^{o(1/\epsilon)}$ unless ETH fails [15].
- Certain optimization problems do not have EPTASs whose running time is $2^{O((1/\epsilon)^{1-\delta})}n^{O(1)}$ for any $\delta > 0$ unless ETH fails [43].

Many of these results can be illustrated by the example of planar MIS. Khanna and Motwani [38] showed that planar MIS is in the syntactic class Planar TMAX and hence has a PTAS. Chen et al. [17] showed that planar MIS is in the syntactic class Planar MAX-W[2] and hence has an EPTAS. Lipton and Tarjan’s separator approach [41] yields an EPTAS for planar MIS of running time $O(n \log n + 2^{5184(1+1/\epsilon)^2}n)$, which is quite inefficient. A better trade-off between the running time and the approximation ratio can be achieved by Baker’s layerwise decomposition approach [4], which gives an EPTAS for planar MIS of running time $O(8^{1/\epsilon}n/\epsilon)$. The various EPTAS algorithms for planar MIS obtained under these different approaches all point to the observation that many graph problems became more tractable on planar graphs, which was captured by the bidimensionality theory [20].

On the negative side, planar MIS is strongly NP-hard and hence has no FPTAS unless P = NP. Marx [43] showed that planar MIS has no EPTAS of running time $2^{O((1/\epsilon)^{1-\delta})}n^{O(1)}$ for any $\delta > 0$ unless ETH fails. This means that the $O(8^{1/\epsilon}n/\epsilon)$ -time EPTAS for planar MIS from Baker’s layerwise decomposition approach [4] is essentially optimal.

Cross-References

- Advances in Scheduling Problems
- Bin Packing Approximation Algorithms: Survey and Classification
- Network Optimization

Recommended Reading

1. S. Arora, Polynomial time approximation schemes for euclidean tsp and other geometric problems, in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, 1996), pp. 2–12
2. S. Arora, Nearly linear time approximation schemes for Euclidean TSP and other geometric problems, in *Proceedings of the 38th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, 1997), pp. 554–563
3. G. Ausiello, A. Marchetti-Spaccamela, M. Protasi, Toward a unified approach for the classification of NP-complete optimization problems. *Theor. Comput. Sci.* **12**, 83–96 (1980)
4. B.S. Baker, Approximation algorithms for NP-complete problems on planar graphs. *J. ACM* **41**, 153–180 (1994)
5. M. Batani, M. Hajaghayi, D. Marx, Approximation schemes for steiner forest on planar graphs and graphs of bounded treewidth, in *Proceedings of the 42nd ACM Symposium on Theory of Computing* (ACM, New York, 2010), pp. 211–220
6. C. Bazgan, Schémas d'approximation et complexité paramétrée. Rapport de stage de DEA d'Informatique à Orsay, 1995
7. H.L. Bodlaender, A partial k -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.* **209**, 1–45 (1998)
8. G. Borradaile, C. Kenyon-Mathieu, P. Klein, A polynomial-time approximation scheme for steiner tree in planar graphs, in *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2007), pp. 1285–1294
9. L. Cai, J. Chen, Fixed parameter tractability and approximability of NP-hard optimization problems. *J. Comput. Syst. Sci.* **54**, 465–474 (1997)
10. L. Cai, M. Fellows, D. Juedes, F. Rosamond, The complexity of polynomial-time approximation. *Theory Comput. Syst.* **41**, 459–477 (2007)
11. C. Chekuri, S. Khanna, A PTAS for the multiple knapsack problem, in *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2000), pp. 213–222
12. J. Chen, Computational optimization. Manuscript. Available at <http://faculty.cs.tamu.edu/chen/notes/opt.pdf>
13. J. Chen, Characterizing parallel hierarchies by reducibilities. *Inf. Process. Lett.* **39**, 303–307 (1991)
14. J. Chen, A. Miranda, A polynomial time approximation scheme for general multiprocessor job scheduling. *SIAM J. Comput.* **31**, 1–17 (2002)
15. J. Chen, X. Huang, I.A. Kanj, G. Xia, Linear FPT reductions and computational lower bounds. *J. Comput. Syst. Sci.* **72**(8), 1346–1367 (2006)
16. J. Chen, X. Huang, I.A. Kanj, G. Xia, Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.* **72**, 1346–1367 (2006)
17. J. Chen, X. Huang, I.A. Kanj, G. Xia, Polynomial time approximation schemes and parameterized complexity. *Discret. Appl. Math.* **155**, 180–193 (2007)
18. T. Coleman, A. Wirth, A polynomial time approximation scheme for k -consensus clustering, in *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2010), pp. 729–740

19. E.D. Demaine, F.V. Fomin, M. Hajiaghayi, D.M. Thilikos, Subexponential parameterized algorithms on bounded-genus graphs and H-minor-free graphs. *J. ACM* **52**, 866–893 (2005)
20. E.D. Demaine, M. Hajiaghayi, Bidimensionality: new connections between FPT algorithms and PTASs, in *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2005), pp. 590–601
21. E.D. Demaine, M. Hajiaghayi, Graphs excluding a fixed minor have grids as large as treewidth, with combinatorial and algorithmic applications through bidimensionality, in *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2005), pp. 682–689
22. E.D. Demaine, M. Hajiaghayi, The bidimensionality theory and its algorithmic applications. *Comput. J.* **51**(3), 292–302 (2008)
23. E.D. Demaine, F.V. Fomin, M.T. Hajiaghayi, D.M. Thilikos, Bidimensional parameters and local treewidth. *SIAM J. Discret. Math.* **18**, 501–511 (2005)
24. X. Deng, G. Li, Z. Li, B. Ma, L. Wang, A PTAS for distinguishing (sub)string selection, in *Proceedings of the 29th International Colloquium on Automata, Languages, and Programming*. Volume 2380 of LNCS (Springer, Berlin, 2002), pp. 740–751
25. X. Deng, G. Li, Z. Li, B. Ma, L. Wang, Genetic design of drugs without side-effects. *SIAM J. Comput.* **32**, 1073–1090 (2003)
26. R. Downey, Parameterized complexity for the skeptic, in *Proceedings of the 18th IEEE Annual Conference on Computational Complexity* (IEEE Computer Society, Los Alamitos, 2003), pp. 147–169
27. R. Downey, M. Fellows, *Parameterized Complexity* (Springer, New York, 1999)
28. D. Eppstein, Diameter and treewidth in minor-closed graph families. *Algorithmica* **27**, 275–291 (1999)
29. T. Erlebach, K. Jansen, E. Seidel, Polynomial-time approximation schemes for geometric graphs, in *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2001), pp. 671–679
30. F.V. Fomin, D. Lokshtanov, S. Saurabh, D.M. Thilikos, Bidimensionality and kernels, in *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2010), pp. 503–510
31. M. Garey, D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, New York, 1979)
32. J. Gramm, J. Guo, R. Niedermeier, On exact and approximation algorithms for distinguishing substring selection, in *Proceedings of the 4th International Symposium on Fundamentals of Computation Theory*. Volume 2751 of LNCS (Springer, Berlin, 2003), pp. 195–209
33. J. Hästad, Clique is hard to approximate within $n^{1-\epsilon}$, in *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, Los Alamitos, CA, USA, 1996, pp. 627–636
34. D.S. Hochbaum (ed.), *Approximation Algorithms for NP-Hard Problems* (PWS Publishing Co., Boston, 1997)
35. H.B. Hunt, III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, R.E. Stearns, NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms* **26**, 238–274 (1998)
36. O. Ibarra, C. Kim, Fast approximation algorithms for the Knapsack and Sum of Subset problems. *J. ACM* **22**, 463–468 (1975)
37. K. Jansen, Parameterized approximation scheme for the multiple knapsack problem, in *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms* (ACM, New York, 2009), pp. 665–674
38. S. Khanna, R. Motwani, Towards a syntactic characterization of PTAS, in *In Proceedings of the 28th ACM Symposium on Theory of Computing* (ACM, New York, 1996), pp. 329–337
39. S. Khanna, R. Motwani, M. Sudan, U. Vazirani, On syntactic versus computational views of approximability. *SIAM J. Comput.* **28**, 164–191 (1999)
40. R.J. Lipton, R.E. Tarjan, A separator theorem for planar graphs. *SIAM J. Appl. Math.* **36**(2), 177–189 (1979)

41. R.J. Lipton, R.E. Tarjan, Applications of a planar separator theorem. *SIAM J. Comput.* **9**(3), 615–627 (1980)
42. D. Marx, Efficient approximation schemes for geometric problems? in *Proceedings of 13th Annual European Symposium on Algorithms*. Volume 3669 of LNCS (2005), pp. 448–459
43. D. Marx, On the optimality of planar and geometric approximation schemes, in *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science* (Springer, Berlin/Heidelberg/New York, 2007), pp. 338–348
44. R. Ostrovsky, Y. Rabani, Polynomial time approximation schemes for geometric k -clustering, in *Proceedings of the 41st Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, 2000), pp. 349–358
45. C. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.* **43**, 425–440 (1991)
46. A. Paz, S. Moran, Nondeterministic polynomial optimization problems and their approximations. *Theor. Comput. Sci.* **15**, 251–277 (1981)
47. N. Robertson, P.D. Seymour, Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms* **7**(3), 309–322 (1986)
48. N. Robertson, P.D. Seymour, Graph minors. V. Excluding a planar graph. *J. Comb. Theory B* **41**, 92–114 (1986)
49. N. Robertson, P.D. Seymour, Excluding a graph with one crossing, in *Graph Structure Theory'91*, 1991, pp. 669–676
50. N. Robertson, P. Seymour, R. Thomas, Quickly excluding a planar graph. *J. Comb. Theory B* **62**, 323–348 (1994)
51. N. Robertson, P.D. Seymour, Graph minors. XII: distance on a surface. *J. Comb. Theory B* **64**, 240–272 (1995)
52. N. Robertson, P.D. Seymour, Graph minors. XVI. Excluding a non-planar graph. *J. Comb. Theory B* **89**, 43–76 (2003)
53. N. Robertson, P.D. Seymour, Graph minors. XX. Wagner's conjecture. *J. Comb. Theory B* **92**, 325–357 (2004)
54. S. Sahni, Algorithms for scheduling independent tasks. *J. ACM* **23**, 116–127 (1976)
55. R. Shamir, D. Tsur, The maximum subforest problem: approximation and exact algorithms, in *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 394–399 (Association for Computing Machinery, New York, 1998)
56. G. Woeginger, When does a dynamic programming formulation guarantee the existence of an FPTAS, in *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (Association for Computing Machinery, New York, 2001), pp. 820–829

Computing Distances Between Evolutionary Trees

Bhaskar DasGupta, Xin He, Tao Jiang, Ming Li, John Tromp,
Lusheng Wang and Louxin Zhang

Contents

1	Introduction	748
2	The nni and Subtree-Transfer Distances	750
2.1	The Case of Unweighted Trees	750
2.2	The Case of Weighted Trees	751
3	Computing the nni Distance	752
3.1	Unweighted Trees: Computing nni Distance Exactly	753
3.2	Unweighted Trees: Computing nni Distance Approximately	759
3.3	Weighted Trees: Generalizing the nni Distance	761

B. DasGupta (✉)

University of Illinois, Chicago, IL, USA

e-mail: dasgupta@cs.uic.edu

X. He

Computer Science and Engineering, University at Buffalo, The State University of New York,
Buffalo, NY, USA

e-mail: xinhe@cs.buffalo.edu

T. Jiang

Department of Computer Science and Engineering, University of California, Riverside, CA, USA
e-mail: jiang@cs.ucr.edu

M. Li

Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada
e-mail: mli@uwaterloo.ca

J. Tromp

CWI, Amsterdam, The Netherlands

e-mail: tromp@cwi.nl

L. Wang

Department of Computer Science, City University of Hong Kong, Hong Kong

e-mail: lwang@cs.cityu.edu.hk

L. Zhang

Department of Mathematics, National University of Singapore, Singapore

e-mail: matzlx@nus.edu.sg

4	Computing the Subtree-Transfer Distance.....	762
4.1	The NP-Hardness.....	763
4.2	An Approximation Algorithm of Ratio 3.....	764
5	Linear-Cost Subtree-Transfer Distance on Weighted Phylogenies.....	766
5.1	An NP-Hardness Result.....	766
5.2	An Approximation Algorithm.....	767
6	The Rotation Distance.....	773
6.1	Rotation and Its Equivalences.....	773
6.2	Upper and Lower Bounds for the Rotation Distance.....	775
6.3	Approximating the Rotation and Diagonal-Flip Distances.....	776
6.4	Miscellaneous Remarks.....	778
7	Open Questions.....	779
8	Conclusion.....	779
Cross-References.....		779
Recommended Reading.....		779

Abstract

In this chapter, we survey some results on some transformation-based distances for evolutionary trees. The authors will focus on the nearest-neighbor distance and a closely related distance called the subtree-transfer distance used in dealing with evolutionary histories involving events like recombinations or gene conversions; some variants of these distances will also be discussed.

1 Introduction

Comparing objects to find their *similarities* or, equivalently, *dissimilarities* is a fundamental issue in many fields including pattern recognition, image analysis, drug design, the study of thermodynamic costs of computing, and cognitive science. Various models have been introduced to measure the degree of similarity or dissimilarity in the literature. In the latter case the degree of dissimilarity is also often referred to as the *distance*. While some distances are straightforward to compute, e.g., the Hamming distance for binary strings and the Euclidean distance for geometric objects, some others are formulated as combinatorial optimization problems and thus pose nontrivial challenging algorithmic problems, sometimes even uncomputable, such as the universal information distance between two objects [3].

Distances based on the notion of *economic transformation* usually fall in the latter category. In a nutshell, a transform-based distance model assumes a set of transformation *operations* or *moves*, each associated with a fixed *cost*, which can be applied on the objects in the domain studied. The set of transformation operations should be *complete* in the sense that any object can be transformed into any other object by performing a sequence of such operations. The distance between two objects is then defined as the minimum cost of any sequence of operations transforming one object into the other.¹ The best known transform-based

¹Usually the operations are reversible, so the authors do not have to specify the direction of a transformation.

distances are perhaps the *edit* distances for strings [39], labeled trees [43, 47], and graphs [48] using operations insertion, deletion, and replacement. The edit distances have applications in many fields including computational molecular biology and text processing and have been studied extensively in both the literature and practical settings. For example, the UNIX command *diff* is essentially based on string edit distance. String edit distance is also a particularly suitable model for biological molecular sequence comparison because the edit operations often represent the most common form of evolutionary events.

In this chapter, the authors survey some results on some transformation-based distances for *evolutionary trees* (also called *phylogenies*). Such a tree is an *unordered* tree; it has uniquely labeled leaves and unlabeled interior nodes, can be *unrooted* or *rooted* if the evolutionary origin is known, can be *unweighted* or *weighted* if the evolutionary length on each edge is known, and usually has internal nodes of degree 3. Reconstructing the correct evolutionary tree for a set of species is one of the fundamental yet difficult problems in evolutionary genetics. Over the past few decades, many approaches for reconstructing evolutionary trees have been developed, including (not exhaustively) parsimony [11, 13, 38], compatibility [31], distance [14, 37], and maximum likelihood [2, 11, 12]. The outcomes of these methods usually depend on the data and the amount of computational resources applied. As a result, in practice they often lead to different trees on the same set of species [26]. It is thus of interest to compare evolutionary trees produced by different methods or by the same method on different data. Several distance models for evolutionary trees have been proposed in the literature. Among them, the best known is perhaps the *nearest-neighbor interchange* (nni) distance introduced independently in [34, 36]. The authors will focus on nni and a closely related distance called the *subtree-transfer* distance introduced in [17, 18] for dealing with evolutionary histories involving events like *recombinations* or *gene conversions*. Some variants of these distances will also be discussed. Since computing each such distance is NP-hard, our main interest is in the design of efficient approximation algorithms with guaranteed performance ratios.

The rest of the chapter is organized as follows. The authors first formally define the nni and subtree-transfer distances as well as a variant of subtree-transfer distance, called the *linear-cost subtree-transfer distance*, in Sect. 2. It is also demonstrated that the nni distance coincides with the linear-cost subtree-transfer distance on unweighted evolutionary trees. Section 3 presents results concerning the nni distance on both weighted and unweighted evolutionary trees. In particular, the authors give some tight upper and lower bounds on the nni distance; prove that computing the nni distance is NP-hard, which was a long-standing open problem; and give some logarithmic ratio approximation algorithms. Section 4 is concerned with the subtree-transfer distance on unweighted evolutionary trees. The main results include the NP-hardness of computing the subtree-transfer distance and an approximation algorithm with ratio 3. In Sect. 5, the authors consider the linear-cost subtree-transfer distance on weighted evolutionary trees and present a ratio 2 approximation algorithm. In Sect. 6, the authors discuss a variant of the nni distance for rooted, ordered trees, called the *rotation distance*, and present a nontrivial approximation algorithm. Some open problems are listed in Sect. 7.

The authors assume the reader has the basic knowledge of algorithms and computational complexity (such as NP and P). Consult, e.g., [15] otherwise. Unless otherwise mentioned, all the trees in this chapter are *degree-3* trees with *unique labels on leaves*. An edge of a tree is *external* if it is incident on a leaf; otherwise it is *internal*.

2 The nni and Subtree-Transfer Distances

In this section, the authors first define the nni, subtree-transfer, and linear-cost subtree-transfer distances for unweighted trees. Then the authors extend the nni and linear-cost subtree-transfer distances to weighted trees.

2.1 The Case of Unweighted Trees

An *nni* operation swaps two subtrees that are separated by an internal edge (u, v) , as shown in [Fig. 1](#).

The nni operation is said to *operate* on this internal edge. The nni distance, $D_{\text{nni}}(T_1, T_2)$, between two trees T_1 and T_2 is defined as the minimum number of nni operations required to transform one tree into the other. Although the distance has been studied extensively in the literature [4, 6, 9, 23, 24, 27–29, 32, 34, 36, 41, 46], the computational complexity of computing it has puzzled the research community for nearly 25 years until about a decade ago [7].

An nni operation can also be viewed as moving a subtree past a neighboring internal node. A more general operation is to transfer a subtree from one place to another arbitrary place. [Figure 2](#) shows such a *subtree-transfer* operation.

The subtree-transfer distance, $D_{\text{st}}(T_1, T_2)$, between two trees T_1 and T_2 is the minimum number of subtrees needed to be moved to transform T_1 into T_2 [7, 8, 17–19].

It is sometimes appropriate in practice to discriminate among subtree-transfer operations as they occur with different frequencies. In this case, each subtree-transfer operation can be charged a cost equal to the distance (the number of nodes passed) that the subtree has moved in the current tree. The *linear-cost* subtree-transfer distance, $D_{\text{lcost}}(T_1, T_2)$, between two trees T_1 and T_2 is then the minimum total cost required to transform T_1 into T_2 by subtree-transfer operations [7, 8]. Clearly, both subtree-transfer and linear-cost subtree-transfer models can also be used as alternative measures for comparing evolutionary trees generated by different tree reconstruction methods.

It is easy to demonstrate that the linear-cost subtree-transfer and nni distances in fact coincide. As mentioned before, an nni move is just a restricted subtree transfer where a subtree is only moved across a single node. (In [Fig. 1](#), the first exchange can alternatively be seen as moving node v together with subtree C past node u towards subtree A , or vice versa.) On the other hand, a subtree transfer over a distance d can always be simulated by a series of d nni moves. Hence, the linear-cost

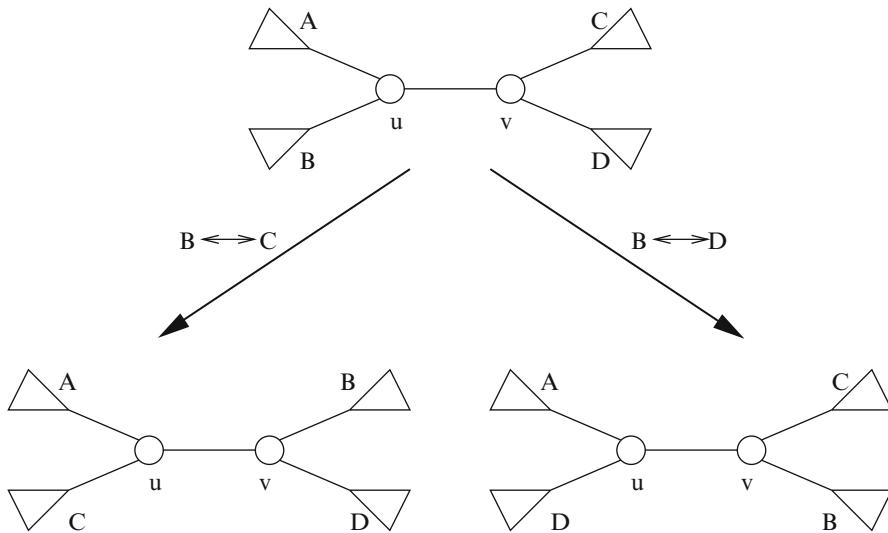


Fig. 1 The two possible nni operations on an internal edge (u, v): exchange $B \leftrightarrow C$ or $B \leftrightarrow D$

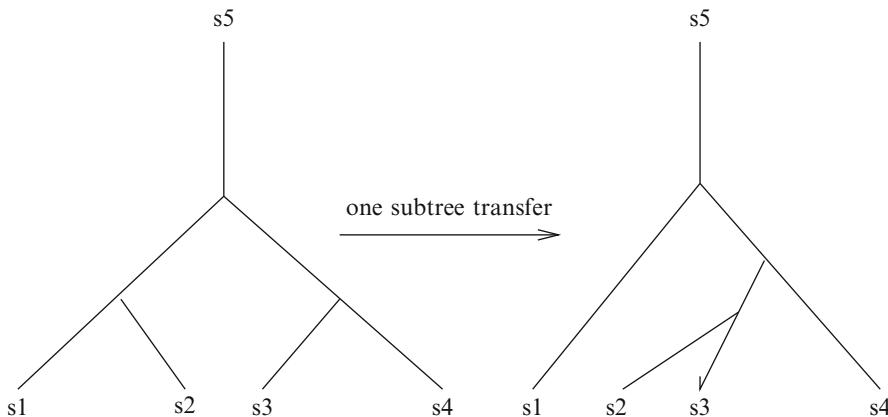


Fig. 2 An example of subtree transfer

subtree-transfer distance is in fact *identical* to the nni distance. However, it will soon become clear that the two models are different on weighted trees.

2.2 The Case of Weighted Trees

An evolutionary may also have weights on its edges, where an edge weight (more popularly known as *branch length* in genetics) could represent the evolutionary

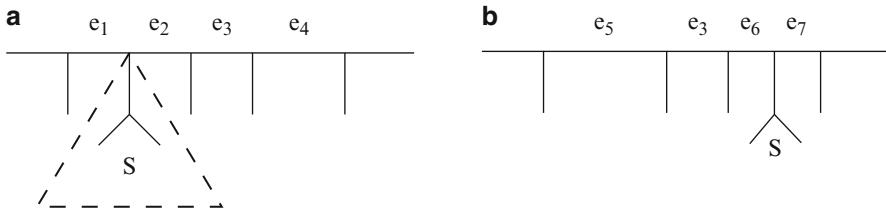


Fig. 3 Subtree transfer on weighted phylogenies. Tree (b) is obtained from tree (a) with one subtree transfer

distance along the edge. Many evolutionary tree reconstruction methods, including the distance and maximum likelihood methods, actually produce weighted evolutionary trees. Comparison of weighted evolutionary trees was studied in [26]. The distance measure adopted is based on the difference in the partitions of the leaves induced by the edges in both trees and has the drawback of being somewhat insensitive to the tree topologies (J. Felsenstein, 1996, personal communication). Both the linear-cost subtree-transfer and nni models can be naturally extended to weighted trees. The extension for nni is straightforward: An nni is simply charged a cost equal to the weight of the edge it operates on. In the case of linear-cost subtree transfer, although the idea is immediate, i.e., a moving subtree should be charged for the weighted distance it travels, the formal definition needs some care and is given below.

Consider (unrooted) trees in which each edge e has a weight $w(e) \geq 0$. To ensure feasibility of transforming a tree into another, it is required that the total weight of all edges to equal one. A subtree transfer is defined as follows. Select a subtree S of T at a given node u and select an edge $e \notin S$. Split the edge e into two edges e_1 and e_2 with weights $w(e_1)$ and $w(e_2)$ ($w(e_1), w(e_2) \geq 0$, $w(e_1) + w(e_2) = w(e)$), and move S to the common endpoint of e_1 and e_2 . Finally, merge the two remaining edges e' and e'' adjacent to u into one edge with weight $w(e') + w(e'')$. The cost of this subtree transfer is the total weight of all the edges over which S is moved. Figure 3 gives an example. The edge weights of the given tree are normalized so that their total sum is 1. The subtree S is transferred to split the edge e_4 to e_6 and e_7 such that $w(e_6), w(e_7) \geq 0$ and $w(e_6) + w(e_7) = w(e_4)$; finally, the two edges e_1 and e_2 are merged to e_5 such that $w(e_5) = w(e_1) + w(e_2)$. The cost of transferring S is $w(e_2) + w(e_3) + w(e_6)$.

Note that for weighted trees, the linear-cost subtree-transfer model is more general than the nni model in the sense that one can slide a subtree along an edge with subtree transfers. Such an operation is not realizable with nni moves.

3 Computing the nni Distance

In this section, the authors discuss the complexity of computing the nni distance between labeled or unlabeled trees, either exactly or approximately. The authors first discuss the case of unweighted trees and then consider the more general case of weighted trees.

Fig. 4 A linear tree with k leaves



3.1 Unweighted Trees: Computing nni Distance Exactly

The *nni* distance was introduced independently in [34, 36]. The complexity of computing the *nni* distance has been open for 25 years (since [36]). The problem is surprisingly subtle given the history of many erroneous results, disproved conjectures, and a faulty NP-completeness proof [4, 23, 24, 27, 28, 32, 46].²

Culik II and Wood [6] (improved later by [32]) proved that $n \log n + O(n)$ *nni* moves are sufficient to transform a tree of n leaves to any other tree with the same set of leaves. Sleator et al. [41] proved an $\Omega(n \log n)$ lower bound for most pairs of trees. A restricted version of the *nni* operation, known as the tree rotation operation (discussed in Sect. 6), was considered in [40], and a trivial approximation algorithm with approximation ratio of 2 was given. But given an individual pair of two trees (labeled or unlabeled), computing the *nni* distance between them (for either labeled or unlabeled trees) has been a long-standing open question until about a decade ago when this problem was settled (for both labeled and unlabeled trees) in [7, 8].

Theorem 1 *Computing the *nni* distance (between two labeled or unlabeled trees) is NP-hard.*

The authors provide a rough sketch of the proof of Theorem 1 for labeled trees (which is the more difficult case). The proof is by a reduction from Exact Cover by 3-Sets (X3C), which is known to be NP-complete [15], to our problem. Recall that, given an instance $S = \{s_1, \dots, s_m\}$, where $m = 3q$, and C_1, \dots, C_n , where $C_i = \{s_{i_1}, s_{i_2}, s_{i_3}\}$, the X3C problem is to find disjoint sets C_{i_1}, \dots, C_{i_q} such that $\bigcup_{j=1}^q C_{i_j} = S$. One will construct two trees T_1 and T_2 with unique leaf labels, such that transforming from T_1 into T_2 requires at most N (to be specified later) *nni* moves iff an exact cover of S exists.

Here is an outline of our reduction. One can perform sorting with *nni* moves and thus view *nni* as a special sorting problem. A sequence $x_1 \dots x_k$ can be represented as a linear tree as in Fig. 4. For convenience, such a linear tree will be simply called a sequence of length k . Sorting such a sequence means to transform it by *nni* operations to a linear tree whose leaves are in ascending order.

To construct the first tree T_1 , for each $s_i \in S$, a sequence S_i of leaves is created that takes a *large* number of *nni* moves to sort. One will make sure that S_i and S_j are *very different* permutations for each pair $i \neq j$, in the sense that one cannot hope to have the sequence S_i sorted *for free* while sorting the sequence S_j by *nni* moves and vice versa. Then for each set $C_i = \{s_{i_1}, s_{i_2}, s_{i_3}\}$, three sequences with

²In [27], the author reduced the partition problem to *nni* by constructing a tree of i nodes for a number i , in an attempt to prove the NP-hardness of computing *nni* distance between unlabeled trees.

the same permutations as the sequences $S_{i_1}, S_{i_2}, S_{i_3}$, respectively, are created but with distinct labels. Such n groups of sequences for C_1, \dots, C_n , each consisting of three sequences, will be placed *far away* from each other and from the m sequences S_1, \dots, S_m in tree T_1 . Tree T_2 has the same structure as T_1 except that all sequences are *sorted*.

Here is the connection between exactly covering S and transforming T_1 into T_2 by nni moves. To transform T_1 into T_2 , all one needs is to sort the sequences defined above. If there is an exact cover C_{i_1}, \dots, C_{i_q} of S , one can partition the m sequences S_1, \dots, S_m into $\frac{m}{3} = q$ groups, according to the cover. For each C_j ($j = i_1, \dots, i_q$) in the cover, one sends the corresponding group of sequences $S_{j_1}, S_{j_2}, S_{j_3}$ to their counterparts, merges the three pairs of sequences with identical permutations, sorts the three permutations, and then splits the pairs and transports the three sorted versions of $S_{j_1}, S_{j_2}, S_{j_3}$ back to their original locations in the tree. Thus, instead of sorting six sequences separately, one does three merges, three sortings, three splits, and a round trip transportation of three sequences. Our construction will guarantee that the latter is significantly cheaper. If there is no exact cover of S , then either some sequence S_i will be sorted separately or one will have to send at least $q + 1$ groups of sequences back and forth. The construction guarantees that both cases will cost significantly more than the previous case.

More details are now given. Apparently many difficult questions have to be answered: How can one find these m sequences S_1, \dots, S_m that are *hard* to sort by nni moves? How does one make sure that sorting one such sequence will never help to sort others? How can one ensure that it is most beneficial to bring the sequences $S_{j_1}, S_{j_2}, S_{j_3}$ to their counterparts defined for C_j to get sorted and not the other way?

One begins with the construction of the sequences S_1, \dots, S_m . Recall that each such sequence is actually a linear tree, as in Fig. 4. Intuitively, it would be a good idea to take a long and difficult-to-sort sequence and break it into m pieces of equal length. But this simple idea does not work for two reasons. First, such a sequence probably cannot be found in polynomial time. Second, even one finds such a sequence, because the upper bound in [6, 32] and the lower bound in [41] (see [32]) do not match, these pieces may still help each other in sorting possibly by merging, sorting together, and then splitting. The following lemma states that there exist two sequences of constant size that are hard to sort and do not help each other in sorting. One will build our m sequences using these two sequences.

Lemma 1 *For any positive constant $\epsilon > 0$, there exist infinitely many k for which there are a constant c and two sequences x and y of length k such that (i) each of them takes at least $(c - \epsilon)k \log k$ nni moves to sort, (ii) each of them takes at most $ck \log k$ nni moves to sort, and (iii) it takes at least $(1 - \epsilon)c(2k) \log(2k)$ nni moves to sort both of them together, i.e., the sequence xy .*

Proof Note that for any c, k, x, y , statements (ii) and (iii) imply statement (i). So it suffices to prove the existence of a constant c and an infinite number of k 's that satisfy conditions (ii) and (iii).

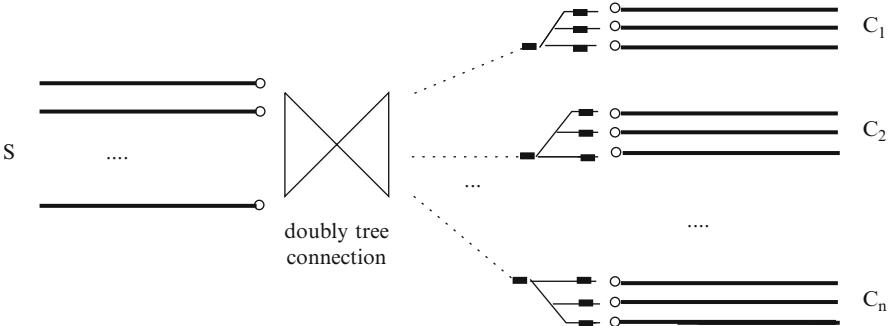


Fig. 5 Structure of tree T_1

From the results in [6, 32, 41], it is known that for each k , there exists a sequence of k leaves such that sorting the sequence takes at most $k \log k + O(k)$ nni moves and at least $\frac{1}{4}k \log k - O(k)$ nni moves. Let us define c_k , for any k , as the maximum number of nni steps to sort any sequence of length k , divided by $k \log k$. Since $\frac{1}{4} - o(1) \leq c_k \leq 1 + o(1)$, there must be infinitely many k satisfying $c_{2k} \geq c_k - \frac{\epsilon}{2}$. Taking x and y to be the two halves of a hardest sequence of length $2k$, for large enough such k , and taking $c = c_k$, one can see that conditions (ii) and (iii) are satisfied. \square

Let $\epsilon = 1/2$, k a sufficiently large integer satisfying Lemma 1 and c, x, y the corresponding constant and sequences. Next one uses x and y , each of length k , to construct m long sequences S_1, \dots, S_m . Choose m distinct binary sequences in $\{0, 1\}^{\lceil \log m \rceil}$. Replace each letter 0 with the sequence x^{m^3} and each letter 1 with the sequence y^{m^3} . Give each occurrence of x and y unique labels. Insert in front of every x and y block a *delimiter* sequence of length k^2 with unique labels. This results in sequences S_1, \dots, S_m , all with distinct labels. One can show that these sequences have the desired properties concerning sorting. The m sequences will have specific orientations in the tree; let us refer to one end as *head* and the other end as *tail*.

One is now ready to do the reduction. From sets $S = \{s_1, \dots, s_m\}$ and C_1, C_2, \dots, C_n , one constructs the two trees T_1 and T_2 as follows. For each element s_i , T_1 has a sequence S_i as defined above. For each set $C_i = \{s_{i_1}, s_{i_2}, s_{i_3}\}$, one creates three sequences $S_{i,i_1}, S_{i,i_2}, S_{i,i_3}$, with the same permutations as $S_{i_1}, S_{i_2}, S_{i_3}$, respectively, but with different and unique labels (one is not allowed to repeat labels).

Figure 5 outlines the structure of tree T_1 . Here a thick solid line represents a sequence S_i or $S_{i,j}$ with the circled end as head; a dotted line represents a toll sequence of m^2 uniquely labeled leaves; a small black rectangle represents a one-way circuit as illustrated in Fig. 6(i).

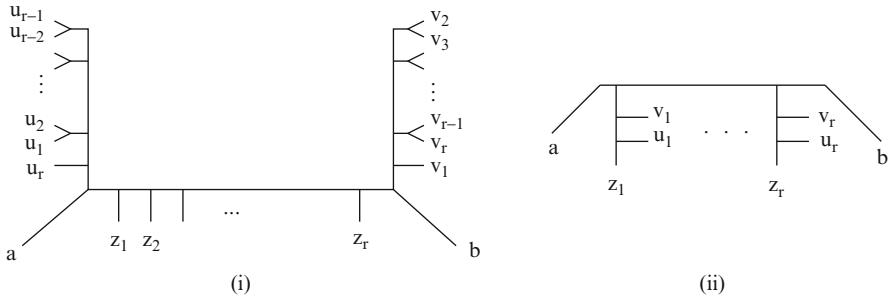


Fig. 6 One-way circuit

The heads of m sequences at the left of Fig. 5 are connected by two full binary trees connected root to root of depth $\log m + \log n$ to the n toll sequences, each leading to the *entrance* of a one-way circuit. The *exit* of each such one-way circuit is connected to the entrances of three one-way circuits leading finally to the three sequences corresponding to some set C_i .

A one-way circuit is designed for the purpose of giving free rides to subtrees moving first from a to b and then later from b to a while imposing a large extra cost for subtrees first moving from b to a and then from a to b . One will choose r so large (i.e., $r = m^4$) that it is not worthwhile to move any sequence $S_{i,j}$, corresponding to some C_i , to the left through the one-way circuits to sort and then move it back to its original location in T_1 . This can be seen as follows. The counterpart of the one-way circuit in T_2 is as shown in Fig. 6(ii).

In any optimal transformation of circuit (i) to (ii), the u 's are paired up with the z 's first, and then the v 's are paired with the u - z pairs. This requires u_r and v_1 to move up and out of the way. The pairing of the u 's essentially provides a shortcut for u_r to reach z_r in half as many steps and similarly for v_1 .

In the following, *sorting* a sequence S_i or $S_{i,j}$ means to have each of its x/y blocks sorted and then the whole sequence *flipped*. The tree T_2 has the same structure as T_1 except that:

- All sequences S_i and $S_{i,j}$ are sorted.
- Each circuit in Fig. 6(i) is changed to (ii).

Let M be the cost for sorting a sequence $S_{i,j}$ optimally (M can be computed easily). The following lemma completes the reduction and thus the proof of Theorem 1.

Lemma 2 *The set S has no exact cover iff $D_{\text{nni}}(T_1, T_2) \geq N + m^2/2$, where $N = q(\log m + \log n) + qm^2 + 28nm^4 - 28n + O(q) + 3nM + (k^2 + 6k)m^3 \log m + O(1)$.*

The authors provide an informal sketch of the proof of Lemma 2; the reader is referred to [7, 8] for more formal proofs. Assume that there is an exact cover for S . First, it is shown that the one-way circuit in Fig. 6 behaves as was claimed. This can be seen as follows. The counterpart of the one-way circuit in T_2 is as shown in Fig. 6(ii). Consider any optimal transformation of circuit (i) to (ii). A precise

breakdown of the cost is as follows: $(r - 3)/2$ steps to move u_r up, then $\frac{r-1}{2}$ times six steps to move each u pair down between the proper z 's and pair them up, and one final step to pair u_r . The exact same number of steps is needed for the symmetric pairing of v 's. Hence, assuming r is odd, in total one needs $2(\frac{r-3}{2} + 6\frac{r-1}{2} + 1) = 7r - 7$ nni moves. Note that a subtree situated at a can initially pair up with u_r in two steps and move together with it, spending three more steps to pop off just before u_r pairs with z_r , to end up at b . It can later spend another 5 steps to move together with v_1 ending up back at a . Going first from b to a and then back to b could only be done *for free* by pairing with v_1 first and with u_r later, since these are the only leaves to move away from b and a respectively, in an optimal transformation. But for v_1 to reach a with minimum cost requires collapsing all the v 's which imposes an extra cost on pairing u 's with z 's later. The least penalty for moving from b to a back to b is thus for v_1 not to take the shortcut which costs an extra $\frac{r}{2}$ steps.

In the following, *sorting* a sequence S_i or $S_{i,j}$ means to have each of its x/y blocks sorted and then the whole sequence *flipped*. In order to transform T_1 into T_2 , one needs to sort the sequences S_i and $S_{i,j}$ and convert each one-way circuit to the structure shown in Fig. 6(ii). If the set S has an exact cover C_{i_1}, \dots, C_{i_q} , one can do the transformation efficiently as follows. For each C_j , $j = i_1, \dots, i_q$, in the cover, one sends the three sequences $S_{j_1}, S_{j_2}, S_{j_3}$ to their counterparts $S_{j,j_1}, S_{j,j_2}, S_{j,j_3}$, merges each pair and sorts them together, and then moves the sorted $S_{j_1}, S_{j_2}, S_{j_3}$ sequences back. During this process, one also gets each one-way circuit involved into the correct shape. One then sorts the other sequences $S_{i,j}$ and get their leading one-way circuits into the correct shape.

The total cost N for this process is calculated as follows. Recall that one sends precisely q groups of sequences to the right.

1. The overhead for these q groups to cross the tree connection network: $q(\log m + \log n) + O(1)$ nni moves.
2. The cost of crossing the q toll sequences of length m^2 before the first batch of one-way circuits: qm^2 nni moves.
3. Converting each one-way circuit to the structure in Fig. 6(ii) costs $7r - 7$ nni's. Since one selects $r = m^4$ and there are in all $4n$ one-way circuits, the total cost is $28nm^4 - 28n$.
4. Moving a group of sequences across a one-way circuit and back costs $O(1)$ extra nni moves, for each of the q groups. The total cost is therefore $O(q)$.
5. Let M be the cost for sorting a sequence $S_{i,j}$ optimally. M can be computed easily, given optimal ways to sort an x block and a y block. Observe that the k^2 delimiter sequences inserted in front of each x/y block prevent the *folding* of any sequence $S_{i,j}$ in an optimal sorting procedure, i.e., it will not be beneficial for two blocks on the same sequence to be merged and sorted together because it costs at most $ck \log k$ nni moves to sort a block and k^2 nni moves to bring a block across a delimiter sequence. Similarly, shrinking any sequence $S_{i,j}$ does not help either. So totally one needs $3nM$ nni moves to sort the $3n$ sequences defined for C_1, \dots, C_n .
6. The extra cost of merging each sequence S_i with its counterpart $S_{j,i}$ while sorting the latter and splitting it out when the sorting is done. The process is as follows.

One sorts $S_{j,i}$ block by block from head to tail. Before processing each block, one first merges this block with the corresponding block of S_i . After sorting this pair of blocks, one splits out the sorted block of S_i and moves down to the next block of S_i , passing a delimiter path of length k^2 . So the extra cost to sorting $S_{j,i}$ is $(k^2 + 6k)m^3 \log m$. Observe that the above process automatically reverses $S_{j,i}$ and S_i .

Conversely, suppose that S has no exact cover. Then to transform T_1 into T_2 , either one has to send $q + 1$ groups to the right crossing the one-way circuits or some sequence S_i is sorted separately from $S_{j,i}$'s or some sequence S_i is sorted together with a *wrong* sequence $S_{j,h}$, where $h \neq i$. In the first case, the cost will be increased by m^2 nni moves, which is the cost of moving an extra group past a delimiter sequence of length m^2 . In the last case, at least one segment of m^3 x 's is sorted together with a segment of y 's. By Lemma 1 and the choice $\epsilon = 0.5$, this is not much better than sorting the two segments separately and costs at least $0.5 cm^3 k \log k - m^3 k$ more nni moves than sorting one such segment, which is larger than m^2 for sufficiently large k and m . The second case introduces an extra cost of $(0.5 cm^3 k \log k \log m) - m^3 k \log m - m^2$ by Lemma 1, which is again larger than m^2 for sufficiently large k and m .

Notice that in the above definition of N , the bounds in items 2,3,5,6 are all optimal. The bounds in items 1 and 4 are the worst case overheads and may not be optimal. But these two items only account for $O(m(\log m + \log n))$ nni moves, which is not sufficient to compensate for the extra cost m^2 given above. This completes the sketch of proof of Lemma 2.

In practice, however, the trees to be compared usually have small nni distances between them, and it is of interest to devise efficient algorithms for computing the optimal nni sequence when the nni distance is small, say d . An $n^{O(d)}$ algorithm for this problem is trivial. With careful inspection, one can derive an algorithm that runs in $O(n^{O(1)} \cdot d^{O(d^2)})$ time. It turns out that by using the results in [32, 41], one could improve this asymptotically to $O(n^2 \log n + n \cdot 2^{23d/2})$ time. To be precise, the following result was proved in [7, 8].

Theorem 2 Suppose that the nni distance between T_1 and T_2 is at most d . Then, an optimal sequence of nni operations transforming T_1 into T_2 can be computed in $O(n^2 \log n + n \cdot 2^{23d/2})$ time.

A sketch of proof of Theorem 2 is as follows. Let T_1 and T_2 be the two trees being compared. An edge $e_1 \in T_1$ is *good* if there is another edge $e_2 \in T_2$ such that e_1 and e_2 partition the leaf labels of T_1 and T_2 identically; e_1 is *bad* otherwise. It is easy to see that T_1 contains at least 1 and at most d bad edges. Moreover, assume that these bad edges form t connected components B_1, \dots, B_t ($1 \leq t \leq d$). As observed in [32], for an optimal nni transformation, sometimes one or more nni operations are needed across a good internal edge of T_1 . Consider the set of at most $d - 1$ good edges in T_1 across which at least one nni operation is performed in an optimal nni sequence. This set of good edges forms at most $d - 1$ connected components in T_1 . Consider any one such connected component S . Since good edges in T_1 and T_2

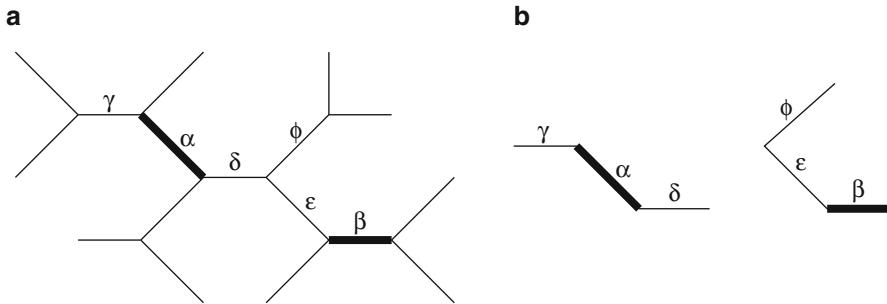


Fig. 7 Illustration of how algorithm NNI-d works ($d = 6, k_1 = k_2 = 3, t = 2$)

partition the trees in similar manner, it is very easy to see that there must be at least one connected component B_i sharing a vertex with S .

Using these observations, one can devise the algorithm shown in the next page. Figure 7 illustrates how the algorithm works. Figure 7a shows two bad edges α, β in T_1 (shown by thick lines) forming two connected components ($t = 2$). Figure 7b shows one choice of two subtrees containing k_1 and k_2 edges and including the edges α and β , respectively. For each subtree, algorithm NNI-d computes all possible nni sequences such that at most 3 nni are performed across edges of each subtree.

How fast does the algorithm run? There are at most $\binom{d+t-1}{d} < 2^{5d/2}$ choices for the integers k_1, \dots, k_t (using the fact that $\binom{n}{j} \leq (2.8n/j)^j$). Note that any subtree of k edges including a fixed edge can be represented by a rooted binary tree on $k+2$ nodes (the root corresponding to the middle of the fixed edge); hence, there are at most $C_{k+2} = \frac{1}{k+3} \binom{2k+4}{k+2} \leq 2^{2k}$ such trees. It follows that the total number of choices for the subtrees A_1, \dots, A_t (for any particular value of k_1, \dots, k_t) is at most $2^{\sum_{i=1}^t ((2k_i+1))} \leq 2^{3d}$. For each tree A_i , the number of sequences of k_i nni operations to consider is at most $3^{k_i-1} 2^{4k_i} < 2^{6k_i}$ by Lemma 1 of [32]. Combining everything, the number of trees one has to examine is at most $2^{5d/2} \cdot 2^{3d} \cdot 2^{6d} < 2^{23d/2}$. The set of all good edges of T_1 can be found in $O(n^2 \log n)$ time, and this time bound is also sufficient to find the connected components of good edges. Using the adjacency-list representation of trees, updating a tree during a single nni operation can easily be done in $O(1)$ time, and whether two trees are isomorphic can be easily checked in $O(n)$ time. Hence, this algorithm finds an optimal nni sequence in $O(n^2 \log n + n \cdot 2^{23d/2})$ time.

3.2 Unweighted Trees: Computing nni Distance Approximately

Since computing the nni distance is NP-hard, the next obvious question is the following: *can one get a good approximation of the distance?* The following result appeared in [32].

```

For every choice of integers  $k_1, \dots, k_t \geq 1$ ,  $\sum_{i=1}^t k_i \leq d$  do
  For every choice of subtrees  $A_1, \dots, A_t$  of  $T_1$  such that
     $A_i$  has at most  $k_i$  edges and contains the component  $B_i$  do
      Examine all sequence of nni transformations across edges
      of all  $A_i$ 's such that no more than  $k_i$  nni operations
      are performed across the edges of  $A_i$ 
    Among all sequences examined, select the one of shortest length that
    transforms  $T_1$  to  $T_2$ 

```

Algorithm NNI-d for the case when nni distance is bounded

Theorem 3 *The nni distance can be polynomial time approximated within a factor of $\log n + O(1)$.*

Proof Given two trees, T_1 and T_2 , one first identifies the bad edges in T_1 with respect to T_2 . These edges induce a subgraph of T_1 consisting of one or more components, each of which is a subtree of T_1 . Each bad-edge component links up the same set of neighboring shared-edge components in T_1 and T_2 , but it does so in different ways.

The algorithm transforms T_1 into T_2 by transforming each non-shared-edge component separately. Consider a component consisting of k non-shared edges in T_1 . This links up $k + 3$ shared-edge components, which one can consider as leaves for the purpose of linking them up differently. So one wants to transform C_0 into C_1 , where C_i is the $(k + 3)$ -tree corresponding to the component in T_i . By the *compression-method* of [6], the distance between C_0 and C_1 is at most $4(k + 3) \log(k + 3) + (4 - \log 3)(k + 3) - 12$. On the other hand, it is clear that any transformation from T_1 into T_2 must use at least one nni operation on every non-shared edge.

The approximation factor of this algorithm is at most

$$\frac{\sum 4(k + 3) \log(k + 3) + (4 - \log 3)(k + 3) - 12}{\sum k} \leq \frac{4n \log n + O(n)}{n - 3},$$

since $\sum k$ is at most the number of internal edges, which is $n - 3$. □

As is apparent from the previous two sections, the question of the computability of the nni distance measure, which will be denoted by d , has generated a lot of interest. Of course, a brute-force method can be employed which searches all (or a significant fraction of) trees in *exponential time and space* ([32] implemented a C program that uses $O(n)$ space to find the distance of any tree to a given one using a brute-force approach and could run it for trees up to size 11). In an attempt to improve efficiency, Waterman and Smith in [46] propose another distance measure, *closest partition*, which they conjecture is actually equal to d . The closest partition distance $c(T, S)$ for trees sharing a partition is defined recursively as the sum of the

two distances between the corresponding smaller parts resulting from splitting each tree into two. For trees T and S not sharing a partition, it is defined as $k + c(R, S)$, where k is the minimum number of nni operations required to transform a tree T into a tree R that shares a partition with tree S . Note that the nondeterminism in choosing R makes this measure somewhat ill-defined. They base their conjecture on what [9] aptly calls a *decomposability property* (DP) of nni. Informally, DP says that if two trees can each be split at some internal edge into identical subsets of leaves, then an optimal transformation of one into the other can be found in which no nni operation affects that internal edge. This claim appears in [46] as Theorem 4. Its proof however appeals to their Theorem 3, which was shown invalid in [24] with a 6-node counterexample. Consequently, [24] concludes that the status of Theorem 4 is unresolved and observes that Theorem 5 of [46] is a single step version of the Waterman and Smith's conjecture that $c = d$. This conjecture was shown to fail in [24] and [4] in a weak sense (for some choices that c allows) and shortly thereafter in [23] in a strong sense (for all choices in defining c). These papers also point out that computation of c appears to require exponential time as well, since there is no obvious bound on k in the definition of c . The work in [28] shows a logarithmic gap between measures c and d . Their example is a pair of trees, each on $n = 2^k$ nodes equidistant from the central internal edge. In one tree, the leaves can be drawn in normal order, while in the other, the leaves can be drawn in bit-reverse order (e.g., 0,4,2,6,1,5,2,7). For this pair of trees, one can show $d = \Theta(n)$, whereas $c = \Theta(n \log n)$ (in the weak sense at least). Finally, the following result was proved in [32], serving as a counterexample to all three theorems 3, 4, and 5 of [46].

Lemma 3 *There are trees T_1, T_2 sharing a partition which is not shared by any intermediate tree on a shortest path from T_1 to T_2 .*

3.3 Weighted Trees: Generalizing the nni Distance

In this section the authors discuss how to generalize the nni distance between two trees T_1 and T_2 when both T_1 and T_2 are weighted. The cost of an nni operation is now the weight of the edge across which two subtrees are swapped. As mentioned before, many phylogeny reconstruction methods produce weighted phylogenies. Hence, the weighted nni distance problem is also very important in computational molecular biology. NP-hardness of the (unweighted) nni distance problem (in Sect. 3.1) implies the NP-hardness of the weighted nni distance problem also.

The authors in [7, 8] present a polynomial-time algorithm with logarithmic approximation ratio for computing the nni distance on weighted phylogenies, generalizing the logarithmic ratio approximation algorithm in [32] (discussed in Sect. 3.2). The approximation for the weighted case is considerably more complex. Note that nni operations can be performed only across internal edges. For feasibility of weighted nni transformation between two given weighted trees T_1 and T_2 , one requires in this section that the following conditions are satisfied: (1) For each leaf

label a , the weight of the edge in T_1 incident on a is the same as the weight of the edge in T_2 incident on a ; (2) the multisets of weights of internal edges of T_1 and T_2 are the same.

Theorem 4 *Let T_1 and T_2 be two weighted phylogenies, each with n leaves. Then, $D_{\text{nni}}(T_1, T_2)$ can be approximated to within a factor of $6 + 6 \log n$ in $O(n^2 \log n)$ time.*

Note that the approximation ratio does not depend on the weights. Intuitively, the idea of the algorithm is as follows. One first identifies *bad* components in the tree that need a lot of nni moves in transformation process. Then, for each bad component, one puts things in correct order by first converting them into balanced shapes. But notice that one cannot afford to perform nni operations many times on heavy edges. Furthermore, not only the leaf nodes need to be moved to the right places, so do the weighted edges. The main contribution of our algorithm is the careful coordination of the transformations so that at most $O(\log n)$ nni operations are performed on each heavy edge.

4 Computing the Subtree-Transfer Distance

When *recombination* of DNA sequences occurs in an evolution, two sequences meet and generate a new sequence, consisting of genetic material taken left of the recombination point from the first sequence and right of the point from the second sequence [17, 18, 25]. From a phylogenetic viewpoint, before the recombination, the ancestral material on the present sequence was located on two sequences, one having all the material to the left of the recombination point and another having all the material to the right of the breaking point. The recombination makes the two evolutionary trees describing neighboring regions differ. However, two neighbor trees cannot be arbitrarily different; one must be obtainable from the other by a *subtree-transfer operation*. When more than one recombination occurs, one can describe an evolutionary history using a list of evolutionary trees; each corresponds to some region of the sequences, and each can be obtained by several subtree-transfer operations from its predecessor. The computation of subtree-transfer distance is useful in reconstructing such a list of trees based on parsimony [17, 18].

In this section, the authors show that computing the subtree-transfer distance between two evolutionary trees is NP-hard and give an approximation algorithm with performance ratio 3. Before one proves the results, it is again convenient to reformulate the problem. Let T_1 and T_2 be two evolutionary trees on set S . An *agreement forest* of T_1 and T_2 is any forest which can be obtained from both T_1 and T_2 by cutting k edges (in each tree) for some k and applying forced contractions in each resulting component trees. Define the size of a forest as the number of components it contains. Then the *maximum agreement forest* (MAF) problem is

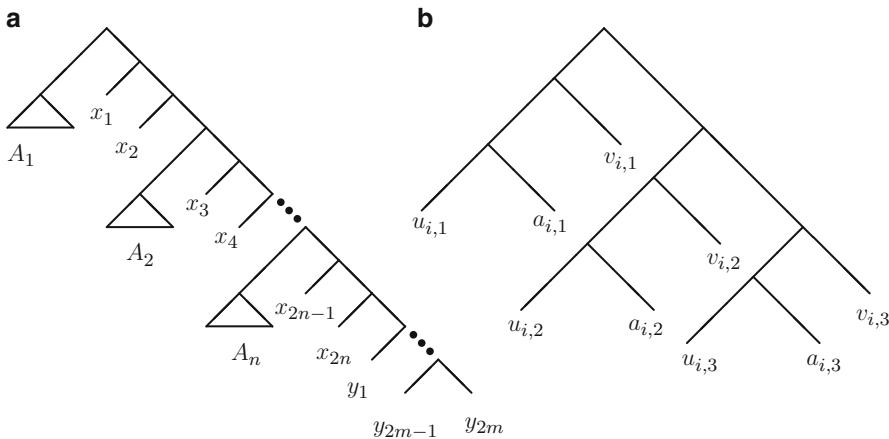


Fig. 8 (a) The tree T_1 . (b) The subtree A_i

to find an agreement forest with the *smallest* size. The following lemma shows that MAF is really equivalent to computing the subtree-transfer distance.

Lemma 4 *The size of an MAF of T_1 and T_2 is one more than their subtree-transfer distance.*

The lemma can be proven by a simple induction on the number of leaves. Intuitively, the lemma says that the transfer operations can be broken down into two stages: First one cuts off the subtrees to be transferred from the rest in T_1 (not worrying where to put them), and then one assembles them appropriately to obtain T_2 . This separation will simplify the proofs.

4.1 The NP-Hardness

Theorem 5 *It is NP-hard to compute the subtree-transfer distance between two binary trees.*

Proof (Sketch) The reduction is from Exact Cover by 3-Sets. Let $S = \{s_1, s_2, \dots, s_m\}$ be a set and C_1, \dots, C_n be an instance of this problem. Assume $m = 3q$.

The tree T_1 is formed by inserting n subtrees A_1, \dots, A_n into a chain containing $2n + 2m$ leaves $x_1, \dots, x_{2n}, y_1, \dots, y_{2m}$ uniformly (see Fig. 8a). Each A_i corresponds to $C_i = \{c_{i,1}, c_{i,2}, c_{i,3}\}$ and has 9 leaves, as shown in Fig. 8b. Suppose that $c_{j,j'}, c_{k,k'}$, and $c_{l,l'}$ are the three occurrences of an $s_i \in S$ in C . Then in T_2 , one has a subtree B_i , as shown in Fig. 9a. For each C_i , one also has a subtree D_i in T_2 , as shown in Fig. 9b. The subtrees are arranged as a linear chain, as shown in Fig. 9c.

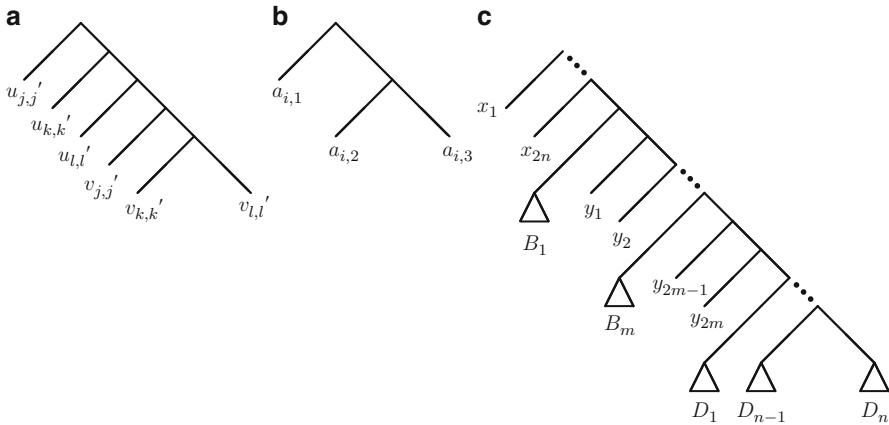


Fig. 9 (a) The subtree B_i . (b) The subtree D_i . (c) The tree T_2

Note that each adjacent pair of subtrees A_i and A_{i+1} in T_1 is separated by a chain of length 2 which also appears in T_2 . Thus, to form an MAF of T_1 and T_2 , our best strategy is clearly to cut off A_1, A_2, \dots, A_n in T_1 and similarly cut off B_1, B_2, \dots, B_m in T_2 . This then forces us to cut off D_1, D_2, \dots, D_n in T_2 . Now in each A_i , either one can cut off the leaves $u_{i,1}, v_{i,1}, u_{i,2}, v_{i,2}, u_{i,3}, v_{i,3}$ to form a subtree containing three leaves $a_{i,1}, a_{i,2}, a_{i,3}$ (yielding $6 + 1 = 7$ components totally) or one can cut off $a_{i,1}, a_{i,2}$, and $a_{i,3}$. In the second case, one will be forced to also cut links between the three subtrees containing leaves $\{u_{i,1}, v_{i,1}\}$, $\{u_{i,2}, v_{i,2}\}$, and $\{u_{i,3}, v_{i,3}\}$, respectively, as the B_i 's are already separated. Hence, in this case the best one can hope for is $3 + 3 = 6$ components (if one can keep all three 2-leaf subtrees in the agreement forest).

It can be shown that C has an exact cover of S if and only if T_1 and T_2 have an agreement forest of size $1 + 6q + 7(n - q) = 7n - q + 1$. \square

4.2 An Approximation Algorithm of Ratio 3

Our basic idea is to deal with a pair of sibling leaves a, b in the first tree T_1 at a time. If the pair a and b are siblings in the second tree T_2 , one replaces this pair with a new leaf labeled by (a, b) in both trees. Otherwise, one will cut T_2 until a and b become siblings or separated. Eventually both trees will be cut into the same forest. Five cases need be considered. Figure 10 illustrates the first four cases. The last case [case (v)] is that a and b are also siblings in T_2 .

The approximation algorithm is given in Fig. 11. The variable N records the number of components (or the number of cuts plus 1).

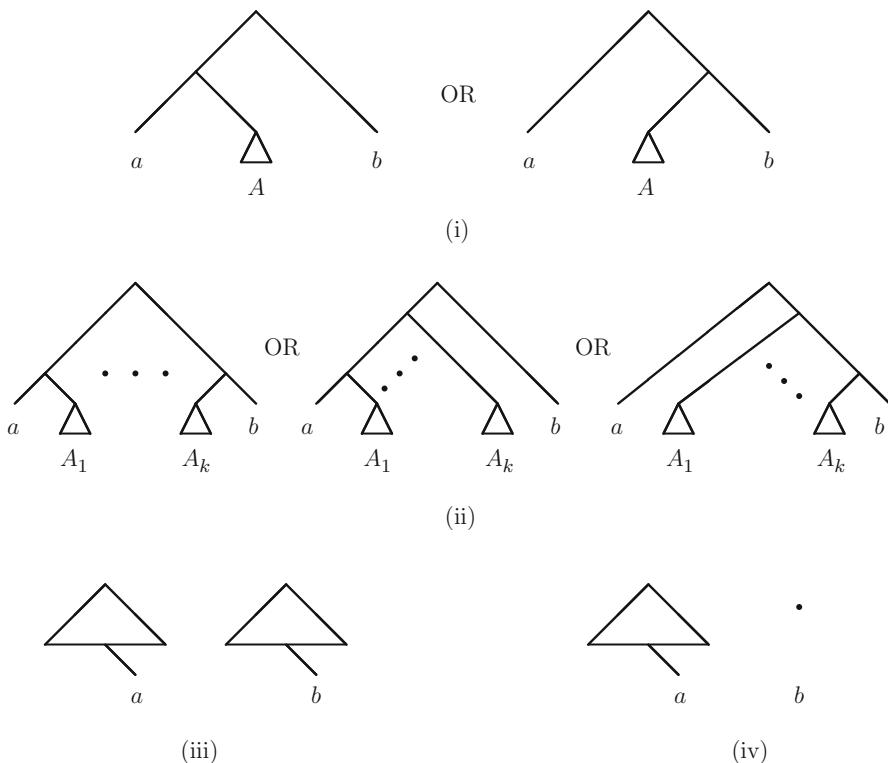


Fig. 10 The first four cases of a and b in T_2

Input: T_1 and T_2 .

0. $N := 1$;
 1. For a pair of sibling leaves a, b in T_1 , consider how they appear in T_2 and cut the trees:
 - Case (i):* Cut off the middle subtree A in T_2 ; $N := N + 1$;
 - Case (ii):* Cut off a and b in both T_1 and T_2 ; $N := N + 2$;
 - Case (iii):* Cut off a and b in both T_1 and T_2 ; $N := N + 2$;
 - Case (iv):* Cut off b in T_1 ;
 - Case (v):* Replace this pair with a new leaf labeled (a, b) in both T_1 and T_2
 2. If some component in the forest for T_1 has size larger than 1, repeat Step 1.
- Output: The forest and N .

Fig. 11 The approximation algorithm of ratio 3

Theorem 6 *The approximation ratio of the algorithm in Fig. 11 is 3, i.e., it always produces an agreement forest of size at most three times the size of an MAF for T_1 and T_2 .*

The NP-hardness proof can be easily strengthened to work for MAX-SNP-hardness. Thus there is no hope for a polynomial-time approximation scheme for this problem. Moreover, the small distance exact algorithm described in Sect. 3 for nni also works here.

5 Linear-Cost Subtree-Transfer Distance on Weighted Phylogenies

In this section the authors investigate the linear-cost subtree-transfer model on weighted phylogenies.

5.1 An NP-Hardness Result

It is open whether the linear-cost subtree-transfer problem is NP-hard for weighted phylogenies. However, one can show that the problem is NP-hard for weighted trees with nonuniquely labeled leaves.

Theorem 7 *Let T_1 and T_2 be two weighted trees with (not necessarily uniquely) labeled leaves. Then, computing $D_{st}(T_1, T_2)$ is NP-hard.*

Proof Our proof is by a reduction from the following *Exact Cover by 3-Sets* (X3C) problem that was used in the proof of Theorem 1. For the convenience of the reader, we state the X3C problem again below:

Instance: $S = \{s_1, \dots, s_m\}$, where $m = 3q$, and C_1, \dots, C_n , where $C_i = \{s_{i_1}, s_{i_2}, s_{i_3}\} \subseteq S$.

Question: Are there q disjoint sets C_{i_1}, \dots, C_{i_q} such that $\bigcup_{j=1}^q C_{i_j} = S$?

Given an instance of the X3C problem, one will construct two trees T_1 and T_2 with leaf labels (not necessarily unique), as shown in Fig. 12, such that transforming from T_1 into T_2 requires subtree transfers of total cost exactly 1 iff an exact cover of S exists.

T_1 has n long arms, $\alpha_1, \dots, \alpha_n$. T_2 has $n - q$ long arms, $\beta_1, \dots, \beta_{n-q}$, and m short arms, $\gamma_1, \dots, \gamma_m$. Each long (resp. short) arm consists of an edge of weight $\frac{1}{n}$ (resp. $\frac{1}{3n}$), with three leaves (resp. one leaf) labeled by the same label x ($x \notin S$), connected to it as shown in Fig. 12. For notational convenience, let e_{α_i} (resp. $e_{\beta_i}, e_{\gamma_i}$) denote the edge of nonzero weight in the long arm α_i (resp. in the long arm β_i , in the

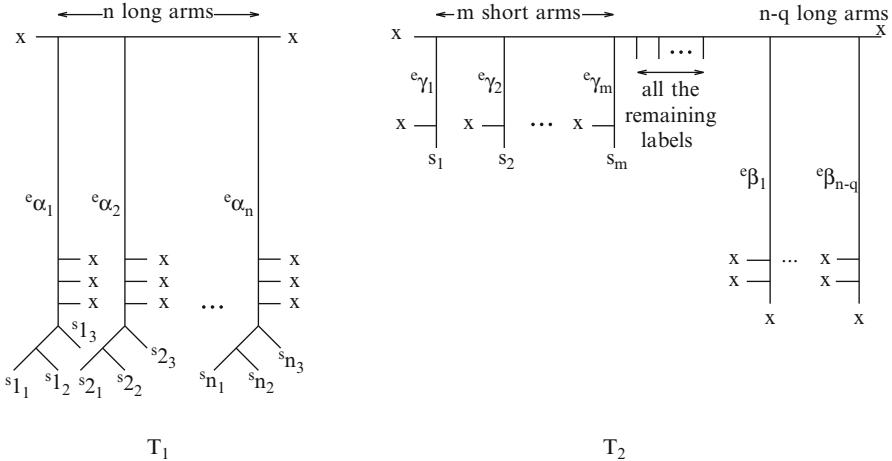


Fig. 12 Trees T_1 and T_2 used in the proof of Theorem 7. The leaf labels are shown beside the corresponding leaves. The notations for some of the internal edges are shown beside the corresponding edges. The edge weights are as follows: $w(e_{\alpha_1}) = w(e_{\alpha_2}) = \dots = w(e_{\alpha_n}) = w(e_{\beta_1}) = w(e_{\beta_2}) = \dots = w(e_{\beta_{n-q}}) = \frac{1}{n}$, $w(e_{\gamma_1}) = w(e_{\gamma_2}) = \dots = w(e_{\gamma_m}) = \frac{1}{3n}$, and all other edges have zero weights

short arm γ_i). In T_1 , at the bottom of the i th long arm α_i , one attaches a subtree t_i consisting of three leaves, as shown in Fig. 12, labeled by the three elements s_{i1}, s_{i2} , and s_{i3} of C_i . At the bottom of each long arm of T_2 , there are no additional subtrees attached. The labeling of the remaining leaves of T_2 is as follows:

- At the bottom of the i th short arm γ_i , one attaches a leaf labeled by s_i .
- The remaining $3n - m$ leaf labels (each leaf label is an element of S) are associated (in any order) with the $3n - m$ leaves in the middle of T_2 between the long and the short arms.

Note that the trees T_1 and T_2 are not uniquely labeled. The following claim proves the correctness of the NP-hardness reduction:

$$D_{st}(T_1, T_2) = 1 \text{ iff there is a solution of the X3C problem.}$$

A proof of the above claim can be found in [8]. □

5.2 An Approximation Algorithm

In this section, the authors present an approximation algorithm for computing the linear-cost subtree-transfer distance on weighted trees. First, the authors introduce some notations and a lower bound on the subtree-transfer distance which will be useful in subsequent proofs. For any tree T , let $E(T)$ (resp. $V(T)$) denote the edge set (resp. node set) of T and $L(T)$ denote the set of leaf nodes of T . An external

edge of T incident on a leaf node a is denoted by $e_T(a)$. Let $E_{\text{int}}(T)$ and $E_{\text{ext}}(T)$ denote the set of internal and external edges of T , respectively. For a subset $E' \subseteq E(T)$, define $w(E') = \sum_{e \in E'} w(e)$. Define $W_{\text{int}}(T) = w(E_{\text{int}}(T))$ and $W_{\text{ext}}(T) = w(E_{\text{ext}}(T))$. Partition $E_{\text{ext}}(T_1)$ into three subsets as follows:

$$\begin{aligned} E_{\text{ext}, T_1 > T_2}(T_1) &= \{e_{T_1}(a) \mid w(e_{T_1}(a)) > w(e_{T_2}(a))\} \\ E_{\text{ext}, T_1 = T_2}(T_1) &= \{e_{T_1}(a) \mid w(e_{T_1}(a)) = w(e_{T_2}(a))\} \\ E_{\text{ext}, T_1 < T_2}(T_1) &= \{e_{T_1}(a) \mid w(e_{T_1}(a)) < w(e_{T_2}(a))\} \\ W_{\text{ext}, T_1 > T_2}(T_1) &= \sum_{e_{T_1}(a) \in E_{\text{ext}, T_1 > T_2}(T_1)} w(e_{T_1}(a)) - w(e_{T_2}(a)). \end{aligned}$$

Similarly, $E_{\text{ext}}(T_2)$ can be partitioned into $E_{\text{ext}, T_1 > T_2}(T_2)$, $E_{\text{ext}, T_1 = T_2}(T_2)$, and $E_{\text{ext}, T_1 < T_2}(T_2)$. $W_{\text{ext}, T_1 < T_2}(T_2)$ is defined analogously. The following lemma is easy to prove.

Lemma 5 $W_{\text{int}}(T_1) + W_{\text{ext}, T_1 > T_2}(T_1) = W_{\text{int}}(T_2) + W_{\text{ext}, T_1 < T_2}(T_2)$.

Proof Since $w(E_{\text{ext}, T_1 > T_2}(T_1)) = w(E_{\text{ext}, T_1 > T_2}(T_2)) + W_{\text{ext}, T_1 > T_2}(T_1)$, one has

$$W_{\text{int}}(T_1) + w(E_{\text{ext}, T_1 = T_2}(T_1)) + w(E_{\text{ext}, T_1 < T_2}(T_1)) + w(E_{\text{ext}, T_1 > T_2}(T_2)) + W_{\text{ext}, T_1 > T_2}(T_1) = w(T_1) = 1.$$

Similarly, one has

$$W_{\text{int}}(T_2) + w(E_{\text{ext}, T_1 = T_2}(T_2)) + w(E_{\text{ext}, T_1 > T_2}(T_2)) + w(E_{\text{ext}, T_1 < T_2}(T_1)) + W_{\text{ext}, T_1 < T_2}(T_2) = w(T_2) = 1.$$

Since $w(E_{\text{ext}, T_1 = T_2}(T_1)) = w(E_{\text{ext}, T_1 = T_2}(T_2))$, the lemma follows from the above two equations. \square

One next defines the notion of good edge pairs as follows:

Definition 1 Let $e_1 \in E_{\text{int}}(T_1)$ and $e_2 \in E_{\text{int}}(T_2)$. Let T'_1 and T''_1 be the two subtrees of T_1 partitioned by e_1 . Let T'_2 and T''_2 be the two subtrees of T_2 partitioned by e_2 . e_1 and e_2 are called a *good pair* of T_1 and T_2 iff the following two conditions hold:

1. $L(T'_1) = L(T'_2)$ and $L(T''_1) = L(T''_2)$.
2. One of the following two conditions holds:
 - a. $w(E(T'_1)) \leq w(E(T'_2)) < w(E(T'_1)) + w(e_1)$.
 - b. $w(E(T'_2)) \leq w(E(T'_1)) < w(E(T'_2)) + w(e_2)$.

Lemma 6 If T_1 and T_2 share no good edge pairs, then:

1. $D_{\text{st}}(T_1, T_2) \geq W_{\text{int}}(T_1) + W_{\text{ext}, T_1 > T_2}(T_1)$.
2. $D_{\text{st}}(T_1, T_2) \geq W_{\text{int}}(T_2) + W_{\text{ext}, T_1 < T_2}(T_2)$.

Proof The authors only prove (1). The proof of (2) follows from (1) and Lemma 5. For each edge $e \in E(T_1)$, one determines the minimum portion of e over which some subtrees of T_1 must be transferred in order to transform T_1 to T_2 . First, consider an edge $e_1 \in E_{\text{int}}(T_1)$. By the assumption of the lemma, there is no edge e_2 in T_2 such that e_1 and e_2 are a good pair. There are two cases:

Case 1. The partition of $L(T_1)$ induced by e_1 is different from the partition of $L(T_2)$ induced by any edge in T_2 . Then, in order to transform T_1 to T_2 , some leaf nodes of T_1 must be transferred across the entire length of e_1 .

Case 2. The partition of $L(T_1)$ induced by e_1 is the same as the partition of $L(T_2)$ induced by an edge e_2 in T_2 . Let T'_1 and T''_1 be the two subtrees of T_1 partitioned by e_1 . Let T'_2 and T''_2 be the two subtrees of T_2 partitioned by e_2 , where $L(T'_1) = L(T'_2)$ and $L(T''_1) = L(T''_2)$.

Case 2.1. $w(E(T'_2)) \geq w(E(T'_1)) + w(e_1)$. In this case, in order to transform T'_1 to T'_2 , some subtree in T'_1 must be transferred across entire length of e_1 .

Case 2.2. $w(E(T'_1)) \geq w(E(T'_2)) + w(e_2)$. This implies $w(E(T''_1)) + w(e_1) \leq w(E(T''_2))$. In order to transform T''_1 to T''_2 , some subtree in T''_1 must be transferred across the entire length of e_1 .

In either case, some subtree of T_1 must be transferred across the entire length of e_1 with cost $w(e_1)$.

Next consider an edge $e_{T_1}(a) \in E_{\text{ext}, T_1 > T_2}(T_1)$. In order to transform $e_{T_1}(a)$ to $e_{T_2}(a)$, a subtree of T_1 must be transferred across a portion of $e_{T_1}(a)$ of length $w(e_{T_1}(a)) - w(e_{T_2}(a))$. Thus,

$$D_{\text{st}}(T_1, T_2) \geq \sum_{e \in E_{\text{int}}(T_1)} w(e) + \sum_{e \in E_{\text{ext}, T_1 > T_2}(T_1)} [w(e_{T_1}(a)) - w(e_{T_2}(a))]$$

$$= W_{\text{int}}(T_1) + W_{\text{ext}, T_1 > T_2}(T_1).$$

□

Remark 1 Assume that the given trees T_1 and T_2 are not uniquely labeled (i.e., a label may appear in more than one leaf). Extend the definition of good pair of edges by treating $L(T)$ as the multiset of leaves for a tree T and considering the condition $L(T'_1) = L(T'_2)$ or $L(T''_1) = L(T''_2)$ to hold if the corresponding multisets are identical. Assume that all the leaves are incident on *zero-weight* edges (i.e., $W_{\text{int}}(T_1) = W_{\text{int}}(T_2)$ and $W_{\text{ext}}(T_1) = W_{\text{ext}}(T_2) = 0$) and that T_1 and T_2 share no good pair of edges. Then, in a manner very similar to the proof of Lemma 6, one can show that $D_{\text{st}}(T_1, T_2) \geq W_{\text{int}}(T_1)$ (and that some subtree is transferred over every internal edge of T_1 to transform T_1 to T_2).

One says that nodes connected by 0-weight edges are equivalent and calls the resulting equivalence classes *super-nodes*. Let e_1, \dots, e_k be all positive weight edges incident to a super-node o . With 0 cost, one can reconnect the edges e_1, \dots, e_k by any subtree, consisting of only 0 weight edges. In particular, the following observation will be useful in our subsequent descriptions.

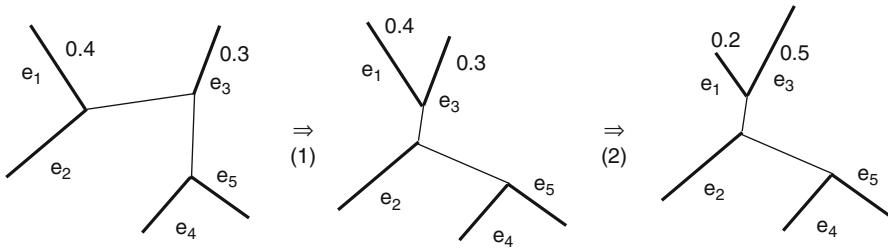


Fig. 13 The operation $\text{move}(e_1, 0.2, e_3)$. (1) e_2, e_4 , and e_5 are assembled into a tree S ; (2) S is moved along e_1 by a length of 0.2

Observation. Let o be a super-node of T . Let e_1, \dots, e_k be all positive weight edges incident on o . Pick any e_i and e_j . One can assemble $\{e_1, \dots, e_k\} - \{e_i, e_j\}$ into a single subtree S with 0 cost, and then transfer S along e_i by a distance $d \leq w(e_i)$. The effect of this operation is that the edges e_1, \dots, e_k are still incident on a super-node and a portion of e_i of length d is *moved* into e_j . The total cost of this operation is d . One denotes this operation by $\text{move}(e_i, d, e_j)$. This operation can be implemented in $O(k)$ time using the adjacency-list representation of the tree (where the weight of the edge is also stored in the adjacency list).

Figure 13 shows an example of this operation. In the figure, the thin lines denote 0 weight edges, and heavy lines denote positive weight edges.

A tree T is called a *superstar* if all of its internal edges have 0 weight. In other words, all external edges of a superstar T are incident to a single super-node.

In the rest of this section, the authors prove the following theorem.

Theorem 8 For any two weighted phylogenies T_1 and T_2 , $D_{\text{st}}(T_1, T_2)$ can be approximated to within a factor of 2 in $O(n^2 \log n)$ time.

First, the authors describe the algorithm DST which approximates $D_{\text{st}}(T_1, T_2)$ to within a factor of 2 for the special case when T_1 and T_2 do not have any good edge pairs. Then the authors will show how to apply the algorithm to the general case.

The algorithm transforms T_1 into a superstar T'_1 (by moving the weight of internal edges into external edges). Similarly, the algorithm transforms T_2 into a superstar T'_2 . The transformations are chosen to make T'_1 coincide with T'_2 . To transform T_1 to T_2 , one first transforms T_1 to $T'_1 (= T'_2)$ and then transforms this to T_2 . Let T'_1 (resp. T'_2) denote the tree during the transformation of T_1 (resp. T_2). T'_1 (resp. T'_2) is initialized to be T_1 (resp. T_2).

Algorithm DST

Step 0. Initialize $T'_1 = T_1$ and $T'_2 = T_2$.

Step 1. While T'_1 is not a superstar yet and there is an external edge $e_{T'_1}(a) = (a, u)$ in T'_1 such that $w(e_{T'_1}(a)) < w(e_{T'_2}(a))$, do:

- Let e_1 be any positive weight internal edge of T'_1 incident on the super-node containing u . Let $d = \min\{w(e_1), [w(e_{T'_2}(a)) - w(e_{T'_1}(a))]\}$.
- Perform the operation $\text{move}(e_1, d, e_{T'_1}(a))$ in T'_1 . (Note: after this move operation, either the entire length of e_1 is moved into $e_{T'_1}(a)$ or $w(e_{T'_1}(a)) = w(e_{T'_2}(a))$).

(Note: after the loop terminates, either T'_1 is a superstar or $w(e_{T'_1}(a)) \geq w(e_{T'_2}(a))$ for all leaf nodes a . Also one performs subtree transfer only on internal edges of T_1 .)

Step 2. Similar to Step 1, with the roles of T'_1 and T'_2 swapped.

Step 3. One transforms T'_1 and T'_2 into two superstars such that $w(e_{T'_1}(a)) = w(e_{T'_2}(a))$ for all leaf nodes a . There are two possible cases as follows:

Case 3.1. $w(e_{T'_1}(a)) = w(e_{T'_2}(a))$ for all leaf nodes a . Perform the following loop to transform both T'_1 and T'_2 into superstars. During the execution of the loop, one maintains the condition $w(e_{T'_1}(a)) = w(e_{T'_2}(a))$ for all leaf nodes a (this condition implies that T'_1 is a superstar iff T'_2 is a superstar).

Repeat

Pick any edge $e_{T'_1}(a) = (a, u_1)$ in T'_1 . Suppose that the corresponding edge $e_{T'_2}(a)$ in T'_2 is (a, u_2) . Let e_1 be any positive weight internal edge of T'_1 incident on the super-node containing u_1 . Let e_2 be any positive weight internal edge of T'_2 incident on the super-node containing u_2 . Let $d = \min\{w(e_1), w(e_2)\}$. In T'_1 , perform the operation $\text{move}(e_1, d, e_{T'_1}(a))$. In T'_2 , perform the operation $\text{move}(e_2, d, e_{T'_2}(a))$. (After this, one has moved the entire length of either e_1 or e_2 into external edges.)

Until both T'_1 and T'_2 are superstars.

(Note: during this step, one performs subtree transfer only on internal edges of T_1 and T_2 .)

Case 3.2. There exists a leaf node a such that $w(e_{T'_1}(a)) \neq w(e_{T'_2}(a))$. This can happen only if both T'_1 and T'_2 are superstars already. One needs to make $w(e_{T'_1}(a)) = w(e_{T'_2}(a))$ for all leaf nodes a . This is done as follows. Partition $L(T'_1)$ into three subsets A , B , and C as follows: A (resp. B , C) is the set of leaf nodes a (resp. b , c) such that $w(e_{T'_1}(a)) = w(e_{T'_2}(a))$ (resp. $w(e_{T'_1}(b)) < w(e_{T'_2}(b))$, $w(e_{T'_1}(c)) > w(e_{T'_2}(c))$).

Repeat

Pick any edge $e_{T'_1}(b)$ with $b \in B$ and $e_{T'_1}(c)$ with $c \in C$. Let $d = \min\{[w(e_{T'_1}(c)) - w(e_{T'_2}(c))], [w(e_{T'_2}(b)) - w(e_{T'_1}(b))]\}$. In T'_1 , perform $\text{move}(e_{T'_1}(c), d, e_{T'_1}(b))$. Then:

- If $d = w(e_{T'_2}(b)) - w(e_{T'_1}(b))$, remove b from B and put b into A .
- If $d = w(e_{T'_1}(c)) - w(e_{T'_2}(c))$, remove c from C and put c into A .
- If $d = w(e_{T'_1}(c)) - w(e_{T'_2}(c)) = w(e_{T'_2}(b)) - w(e_{T'_1}(b))$, remove b from B ; remove c from C ; put both b and c into A .

Until $B = C = \emptyset$.

Step 4. Now both T'_1 and T'_2 are superstars and $w(e_{T'_1}(a)) = w(e_{T'_2}(a))$ for all leaf nodes a . One adjusts the topology of the super-nodes of T'_1 and T'_2 so that T'_1 and T'_2 are identical.

Lemma 7 Assume that T_1 and T_2 do not share any good edge pairs. Then, algorithm DST approximates $D_{st}(T_1, T_2)$ to within a factor of 2 in $O(n^2)$ time.

Proof One analyzes the cost and running time of each step of the algorithm. One uses the adjacency-list representation of a tree. Steps 0 and 4 incur no costs and can easily be implemented in $O(n)$ time. During Steps 1, 2, and 3.1, one only transfers subtrees across internal edges of T_1 and T_2 . Over any portion of such an edge e , at most one subtree-transfer operation occurs. So the total cost of these steps is bounded above by $W_{int}(T_1) + W_{int}(T_2)$. Moreover, it is easy to see that at most $O(n)$ moves are performed during Steps 1, 2, and 3.1, and since each move operation can be implemented in $O(n)$ time, the total time for all these steps is at most $O(n^2)$.

Next, consider Step 3.2. Before the repeat loop is entered, for any $c \in C$, one has:

- $w(e_{T'_1}(c)) = w(e_{T_1}(c))$. (This is because no additional weight is moved to the edge $e_{T'_1}(c)$ during Steps 1 and 2.)
- $w(e_{T'_2}(c)) \geq w(e_{T_2}(c))$.

During Step 3.2, one only transfers subtrees across the edges $e_{T'_1}(c)$ for $c \in C$. Fix such an edge. Note that any portion of $e_{T'_1}(c)$ is traversed at most once during Step 3.2. Once the length of $e_{T'_1}(c)$ is reduced to $w(e_{T'_2}(c))$, c is removed from C . So the portion of $e_{T'_1}(c)$ traversed during Step 3.2 is $w(e_{T'_1}(c)) - w(e_{T'_2}(c)) = w(e_{T_1}(c)) - w(e_{T'_2}(c)) \leq w(e_{T_1}(c)) - w(e_{T_2}(c))$. So the total cost of Step 3.2 is at most $\sum_{c \in C} [w(e_{T'_1}(c)) - w(e_{T'_2}(c))] \leq \sum_{c \in C} [w(e_{T_1}(c)) - w(e_{T_2}(c))] \leq W_{ext,T_1>T_2}(T_1)$. Also, one performs at most $O(n)$ move operations during Step 3.2, and hence, this step can also be implemented in $O(n^2)$ time.

Thus the total cost of the algorithm is bounded above by $W_{int}(T_1) + W_{int}(T_2) + W_{ext,T_1>T_2}(T_1)$, which is at most $2D_{st}(T_1, T_2)$ by Lemma 6. \square

Next, one shows how to apply algorithm DST to achieve an approximation ratio of 2 when T_1 and T_2 may share some good edge pairs. One concentrates on the algorithm and omits implementation details. Let K be the number of good edge pairs in T_1 and T_2 . Our algorithm is by induction on K . If $K = 0$, algorithm DST works by Lemma 7. Suppose $K > 0$. Let $e_1 = (u_1, v_1) \in E(T_1)$ and $e_2 = (u_2, v_2) \in E(T_2)$ be a good pair. Let T'_1 and T''_1 be the two subtrees of T_1 partitioned by e_1 . Let T'_2 and T''_2 be the two subtrees of T_2 partitioned by e_2 , where $L(T'_1) = L(T'_2)$ and $L(T''_1) = L(T''_2)$.

Assume $w(E(T'_1)) \leq w(E(T'_2)) < w(E(T'_1)) + w(e_1)$. (The other case can be handled in a similar way.) Add a new edge (u_1, x) to T'_1 and assign $w((u_1, x)) = w(E(T'_2)) - w(E(T'_1))$. Add a new edge (x, v_1) to T''_1 and assign $w((x, v_1)) = w(e_1) - w((u_1, x))$. Add a new edge (u_2, x) to T'_2 and assign $w((u_2, x)) = 0$. Add a new edge (x, v_2) to T''_2 and assign $w((x, v_2)) = w(e_2)$ (see Fig. 14.) Note that the weights of all new edges are nonnegative.

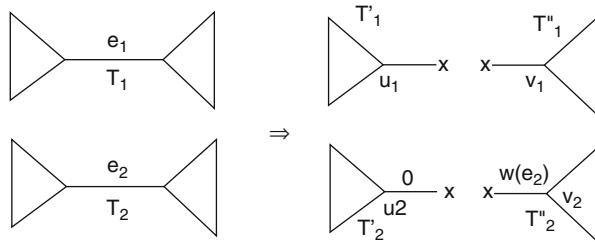


Fig. 14 Cut each of T_1 and T_2 into two smaller trees

Now one has $L(T'_1) = L(T'_2)$ and $w(T'_1) = w(T'_2)$. One can normalize the weights of T'_1 and T'_2 such that their sum is 1. By induction hypothesis, one can transform T'_1 to T'_2 with cost at most $2D_{st}(T'_1, T'_2)$. Similarly, one can transform T''_1 to T''_2 with cost at most $2D_{st}(T''_1, T''_2)$. Combining the two transfer sequences, one can transform T_1 to T_2 with cost at most $2D_{st}(T_1, T_2)$. The complete algorithm takes $O(n^2 \log n)$ time. This completes the proof of [Theorem 8](#).

6 The Rotation Distance

6.1 Rotation and Its Equivalences

A *rotation* is an operation that changes one rooted binary tree into another with the same size. [Figure 15](#) shows the general rotation rule. Note that the rotation is an invertible operation. If a tree T is changed into T' by a rotation, then T' can be changed back into T by another rotation. In a rooted binary tree of size n , there are $n - 1$ possible rotations, each corresponding a non-root node.

A *symmetric order* traversal of a rooted tree visits all of the nodes exactly once. The order can be described recursively as follows: For a node in the tree, traverse its left subtree (if there is one), visit the node itself, and then traverse its right subtree (if there is). A rotation maintains the symmetric order of the nodes.

The rotation on binary trees can be formulated with respect to different systems of combinatorial objects and their transformations. The diagonal-flip operation in triangulations is perhaps more intuitive and so supplies more insight.

Consider the standard convex $(n + 2)$ -gon. One chooses an edge of the polygon as a distinguished edge, called *root edge*, and labels its ends as 0 and $n + 1$. One also labels the other n vertices from 1 to n counterclockwise. Any triangulation of the $(n + 2)$ -gon has n triangles and $n - 1$ diagonals. From a triangulation of the $(n + 2)$ -gon, one derives a binary tree of size n by assigning a node for each triangle and connecting two nodes if the corresponding triangles sharing a common diagonal. The root of the tree corresponds to the triangle containing the root edge. It is not difficult to see that the i th node of the binary tree in symmetric order corresponds

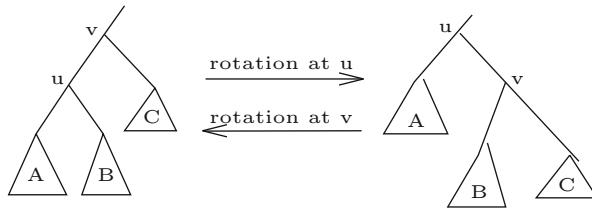


Fig. 15 The definition of rotation

Fig. 16 A triangulation and its corresponding n -node rooted binary tree

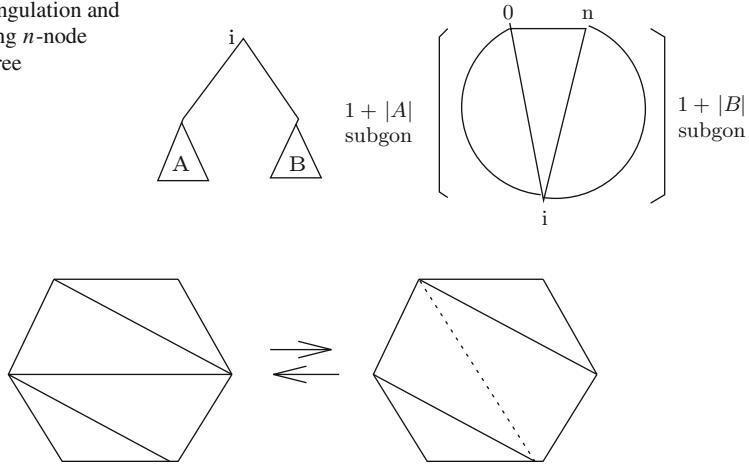


Fig. 17 A diagonal flip in a triangulation of the hexagon

to the triangle with vertices i , j , and k such that $j < i < k$. In this way, one obtains a 1–1 correspondence between n -node binary trees and triangulations of the $(n+2)$ -gon as illustrated in Fig. 16.

A *diagonal flip* is an operation that transforms one triangulation of a convex polygon into another, as shown in Fig. 17. A diagonal inside the polygon is removed, creating a quadrilateral. Then the opposite diagonal of this quadrilateral is inserted in place of the one removed, restoring a triangulation of the polygon. It is not difficult to see that diagonal flips in a triangulation correspond one-to-one to rotations in the corresponding binary tree.

Given a triangulation π of a polygon, one defines the *internal degree* of a vertex v as the number of diagonals adjacent to v , denoted by $\text{id}(v)$. Now let us see how $\text{id}(v)$ is reflected in the corresponding binary tree. In a rooted binary tree T , the *left* (*resp.* *right*) *path* is a maximal sequence of nodes that form a path starting at the root all of whose edges go in left (*resp.* right) direction. For a node $v \in T$, the left and right subtree rooted at v are denoted by LT_v and RT_v , respectively. Recall that all non-leaf nodes are *internal nodes* in a tree. The following result is of interest itself and has not appeared in literature to the best of the authors knowledge.

Theorem 9 ([33]) Suppose that the $(n + 2)$ -gon P is oriented by labeling its vertices from 0 to $n + 1$ and $(0, n + 1)$ is the root edge. Let π be a triangulation of P and T be the corresponding n -node rooted binary tree. Then:

1. The internal degree $\text{id}(0)$ of vertex 0 in P equals the number of internal nodes on the left path of T .
2. The internal degree $\text{id}(n + 1)$ of vertex $n + 1$ in P equals the number of internal nodes on the right path of T .
3. The internal degree $\text{id}(i)$ of vertex $i \in P$ ($0 < i < n + 1$) equals the number of subtrees at node i , which is at most 2, plus the number of internal nodes on the right path of the left subtree LT_i and the number of internal nodes on the left path of the right subtree RT_i .

Other interesting relationship between a triangulation of a convex polygon and its corresponding rooted binary tree can be found in a nice survey article [1].

6.2 Upper and Lower Bounds for the Rotation Distance

Any rooted binary tree of size n can be converted into any other with the same size by performing an appropriate sequence of rotations. Therefore, one can define the *rotation distance* between two trees as the minimum number of rotations required to convert one tree into the other. Let $\text{rt}(T_1, T_2)$ denote the rotation distance between two trees T_1 and T_2 . Define

$$\text{rt}(n) = \max\{\text{rt}(T_1, T_2) \mid |T_1| = |T_2| = n\}.$$

Similarly, one can define the *diagonal-flip* distance between two triangulations of the n -gon and denote the maximum distance between any pair of such triangulations by $\text{fd}(n)$. Obviously, $\text{rd}(n) = \text{fd}(n + 2)$.

Culik and Wood showed that $\text{rd}(n) \leq 2n - 2$ [6]. Sleator, Tarjan, and Thurston improved this bound to $2n - 6$ and showed that the bound is tight for all sufficiently large n using hyperbolic geometry.

Theorem 10 ([40]) $\text{rd}(n) = \text{fd}(n + 2) \leq 2n - 6$ for all $n > 10$. Furthermore, the equality holds for all sufficiently large n .

The exact values of $\text{rd}(n)$ for $n \leq 16$ are listed below [40]:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\text{rd}(n)$	0	1	2	4	5	7	9	11	12	15	16	18	20	22	24	26

However, little is known about the lower bound for the rotation distance $\text{rd}(T_1, T_2)$ of two given trees T_1 and T_2 . The following two theorems are the only known lower bounds, which are presented in terms of the diagonal-flip distance for simplicity. The first one is a variant of Lemma 3 in [40].

Theorem 11 Let π_1 and π_2 be two triangulations of the n -gon. If π_1 and π_2 have k different diagonals, then $\text{fd}(\pi_1, \pi_2) \geq k$.

Let π_1 and π_2 be two triangulations of the n -gon. Consider a sequence Π of diagonal flips that transforms π_1 into π_2 . A diagonal flip $(ab, cd) \in \Pi$ is *auxiliary* if $cd \notin \pi_2$. One also says that the flip (ab, cd) *touches* the vertices a, b, c, d . Let $A(\Pi)$ denote the set of all auxiliary diagonal flips in S . Let $|\Pi|$ denote the number of flips in Π . Then

$$|\Pi| \geq |A(\Pi)| + n - 3. \quad (1)$$

Finally, a triangle of a triangulation is said to be *internal* if it contains three diagonals of the triangulation.

Theorem 12 ([33]) Let π_1 and π_2 be two triangulations of a convex polygon and let v be a vertex of the polygon. Suppose that the following conditions are satisfied:

- (a) v is an end of at least two diagonals in π_2 .
- (b) v is not a vertex of any internal triangles in π_1 or π_2 .
- (c) v is not connected by a π_2 -diagonal to any vertices of internal triangles in π_2 .
- (d) Flipping any π_1 -diagonal adjacent to v does not create a π_2 -diagonal.

Then, there is at least one auxiliary diagonal touching v in any sequence Π of diagonal flips that converts π_1 into π_2 .

As shown in the next subsection, [Theorem 12](#) is useful for estimating the diagonal-flip distance between two triangulations.

6.3 Approximating the Rotation and Diagonal-Flip Distances

Since the rotation and diagonal-flip distance are equivalent, the authors just state results in terms of the diagonal-flip distance.

Suppose that we are given two triangulations π_1 and π_2 with k different diagonals. Since every different π_1 -diagonal has to be flipped, any diagonal-flip transformation from one to another contains at least k flips. On the other hand, by [Theorem 10](#), $2k$ flips are enough to transform π_1 into π_2 . This implies an approximation with ratio 2.

Theorem 13 The diagonal-flip distance can be approximated with ratio 2 in polynomial time.

However, it is very hard to develop a polynomial approximation algorithm with constant ratio <2 for the distance. In what follows, the authors prove a slightly better approximation.

Theorem 14 ([33]) There is a polynomial approximation algorithm that, on the input of two triangulations π_1 and π_2 of the n -gon, outputs a diagonal-flip

Table 1 Transformation algorithm

Input: Two triangulations π_1 and π_2 ;

Do until the following “if” conditions fails

if there are isolated diagonals **then**

pick such an edge e ;

let e' be the unique diagonal that intersects e ;

if $e' \in \pi_1$ then $\pi_1 := \pi_1 + e - e'$ else $\pi_2 := \pi_2 + e - e'$;

Enddo

Let the resulting polygon triangulations have k cells $P_i (i \leq k)$, and let $\pi_j|_{P_i}$ denote the restriction of π_j on P_i for $j = 1, 2$ and $i \leq k$; assume P_i has n_i vertices.

For each cell P_i

pick a node v ;

transform $\pi_1|_{P_i}$ into the unique triangulation π all of whose

diagonals have one end at v using at most n_i steps;

transform π into $\pi_2|_{P_i}$ reversely.

Endfor

transformation of length at most $\left(2 - \frac{2}{4(d-1)(d+6)+1}\right) \text{fd}(\pi_1, \pi_2)$, where d is the maximum number of diagonals adjacent to a vertex in one of the given triangulations.

Proof Let e be a diagonal in π_1 or π_2 . The diagonal e is said to be *isolated* if there is only one diagonal (in the other triangulation) crossing e . Suppose that we are given two triangulations of the n -gon. They may have some diagonals in common in general. All the common diagonals divide the rest of diagonals into disjoint subclasses. Each disjoint subclass together with common diagonals around it is called a *cell*. A desired algorithm is presented in **Table 1**. Obviously, the algorithm takes polynomial time. One analyzes its approximation ratio as follows.

Without loss of generality, one assumes that π_1 and π_2 do not have common diagonals. Suppose that the do loop runs m_1 times for isolated diagonals. Then, after the do loop, π_1 and π_2 have been transformed into triangulations π'_1 and π'_2 which have m diagonals in common. Without loss of generality, one may assume that different diagonals in π'_1 and π'_2 form two triangulations of a convex $(n-m)$ -gon. Note that $\text{fd}(\pi_1, \pi_2) = m + \text{fd}(\pi'_1, \pi'_2)$.

A vertex $v \in P$ is *pure* with respect to π'_i , if it is only an end of π'_i -diagonals. Let V_1 and V_2 denote the sets of pure vertices with respect to π'_1 and π'_2 , respectively. One first proves that

$$\text{fd}(\pi'_1, \pi'_2) \geq (n-m) - 3 + |V_1|/4.$$

Consider a shortest sequence S of diagonal flips that transforms π'_1 into π'_2 . Since there are no isolated edges in both π'_1 and π'_2 , each vertex in V_1 is an end of at least two π_1 -diagonals. By [Theorem 12](#), for each node in V_1 , there is at least one auxiliary flip touching it. Since each auxiliary flip can touch at most 4 vertices, there are at most $|V_1|/4$ auxiliary flips in S . Hence, by Inequality ([Eq. 1](#)), $\text{fd}(\pi'_1, \pi'_2) \geq (n - m) - 3 + |V_1|/4$.

Similarly, by considering pure vertices with respect to π_2 , one is able to prove that $\text{fd}(\pi'_1, \pi'_2) \geq (n - m) - 3 + |V_2|/4$. Combining these two bounds together, one obtains that

$$\text{fd}(\pi_1, \pi_2) \geq (n - m) - 3 + \frac{|V_1| + |V_2|}{8}. \quad (2)$$

On the other hand, one can prove that there are at least $\frac{n-m}{d-1} - 3|V_1| - (d+2)|V_2|$ vertices satisfying the conditions in [Theorem 12](#). Thus, by Inequality ([Eq. 1](#)), $\text{fd}(\pi'_1, \pi'_2) \geq (n - m) - 3 + \frac{n-m}{4(d-1)} - \frac{3|V_1|}{4} - \frac{(d+2)|V_2|}{4}$. Similarly, one has $\text{fd}(\pi'_1, \pi'_2) \geq (n - m) - 3 + \frac{n-m}{4(d-1)} - \frac{3|V_2|}{4} - \frac{(d+2)|V_1|}{4}$. Combining these two inequalities together, one has

$$\text{fd}(\pi'_1, \pi'_2) \geq (n - m) - 3 + \frac{n - m}{4(d - 1)} - \frac{(d + 5)(|V_1| + |V_2|)}{8}. \quad (3)$$

By (2) and (3), $\text{fd}(\pi_1, \pi_2) = m + \text{fd}(\pi'_1, \pi'_2) \geq m + (n - m) \left(1 + \frac{1}{4(d-1)(d+6)}\right) - 3$.

The algorithm transforms π'_1 to π'_2 using at most $M = 2(n - m) + m = 2n - m$ flips, which is less than $\left(2 - \frac{2}{4(d-1)(d+6)+1}\right) \text{fd}(\pi_1, \pi_2)$. This finishes the proof. \square

Furthermore, [33] also presented a polynomial approximation algorithm with ratio 1.97 for the diagonal-flip transformation between two triangulations without internal triangle. Such a triangulation corresponds to a rooted binary tree without degree-3 internal nodes.

6.4 Miscellaneous Remarks

The diagonal-flip operation was early studied by Wagner [44] in the context of arbitrary triangulated planar graphs and by Dewdney [10] in the case of graphs of genus one. They showed that any such graph can be transformed to any other by diagonal flips. However, they did not try to accurately estimate how many flips are necessary. After [40] was published, the rotation and diagonal-flip operations have been studied in several aspects. Pallo [35] proposed a heuristic search algorithm to compute the rotation distance between two given trees. Hurtado et al. [22] studied diagonal flips in arbitrary polygons. Guibas and Hershberger [16] abstracted polygon morphing as a sequence of rotations on weighted binary trees, and Hershberger and Suri [20] proved that a weighted rooted binary tree can be converted into any other in at most $O(n \log n)$ rotations.

7 Open Questions

In this section, the authors list some open questions concerning the nni and subtree-transfer distances:

1. Give a constant ratio approximation algorithm for the nni distance on unweighted evolutionary trees or prove that the ratio $O(\log n)$ is the best possible.
2. Is the linear-cost subtree-transfer distance NP-hard to compute on weighted evolutionary trees?
3. Can one improve the approximation ratios for subtree-transfer distance on unweighted or weighted evolutionary trees?
4. Are there simple approximation algorithms for the rotation distance with non-trivial ratios?

8 Conclusion

In this chapter, we have surveyed a variety of results on some transformation based distances for evolutionary trees. Admitted, given the vast majority of topics in computational biology, these discussed topics constitute only a part of them. However, we hope that this survey article will inspire the readers for further study and research of these and other related topics.

Cross-References

► [Combinatorial Optimization Algorithms](#)

Recommended Reading

Papers on computational molecular biology appear in many different books, journals, and conferences. Below the authors list some sources which could serve as excellent starting points for various problems that arise in computational biology:

Books: References [5, 21, 30, 39, 42, 45]

Journals: Bioinformatics, Journal of Computational Biology, Bulletin of Mathematical Biology, Journal of Theoretical Biology, etc.

Conferences: Annual Symposium on Combinatorial Pattern Matching (CPM), Pacific Symposium on Biocomputing (PSB), Annual International Conference on Computational Molecular Biology (RECOMB), Annual Conference on Intelligent Systems in Molecular Biology (ISMB), etc.

1. D. Aldous, Triangulating the circle, at random. *Am. Math. Mon.* **89**, 223–234 (1994)
2. D. Barry, J.A. Hartigan, Statistical analysis of hominoid molecular evolution. *Stat. Sci.* **2**, 191–210 (1987)
3. C.H. Bennett, P. Gács, M. Li, P. Vitányi, W. Zurek, Information distance. *IEEE Trans. Inform. Theory* **44**, 1407–1423 (1998)
4. R.P. Boland, E.K. Brown, W.H.E. Day, Approximating minimum-length-sequence metrics: a cautionary note. *Math. Soc. Sci.* **4**, 261–270 (1983)
5. J. Collado-Vides, B. Magasanik, T.F. Smith (eds.), *Integrative Approaches to Molecular Biology* (MIT, Cambridge, 1996)
6. K. Culik II, D. Wood, A note on some tree similarity measures. *Inform. Proc. Lett.* **15**, 39–42 (1982)
7. B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, L. Zhang, On distances between phylogenetic trees, in *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1997, New Orleans, LA, USA, pp. 427–436
8. B. DasGupta, X. He, T. Jiang, M. Li, J. Tromp, On the linear-cost subtree-transfer distance. *Algorithmica* **25**(2), 176–195 (1999)
9. W.H.E. Day, Properties of the nearest neighbor interchange metric for trees of small size. *J. Theor. Biol.* **101**, 275–288 (1983)
10. A.K. Dewdney, Wagner's theorem for torus graphs. *Discr. Math.*, **4**, 139–149 (1973)
11. A.W.F. Edwards, L.L. Cavalli-Sforza, The reconstruction of evolution. *Ann. Hum. Genet.* **27**, 105 (1964) (also in *Heredity* 18, 553)
12. J. Felsenstein, Evolutionary trees for DNA sequences: a maximum likelihood approach. *J. Mol. Evol.* **17**, 368–376 (1981)
13. W.M. Fitch, Toward defining the course of evolution: minimum change for a specified tree topology. *Syst. Zool.* **20**, 406–416 (1971)
14. W.M. Fitch, E. Margoliash, Construction of phylogenetic trees. *Science* **155**, 279–284 (1967)
15. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman, New York, NY, 1979)
16. L. Guibas, J. Hershberger, Morphing simple polygons, in *Proceeding of the ACM 10th Annual Symposium of Computer Geometry*, 1994, Stony Brook, NY, USA, pp. 267–276
17. J. Hein, Reconstructing evolution of sequences subject to recombination using parsimony. *Math. Biosci.* **98**, 185–200 (1990)
18. J. Hein, A heuristic method to reconstruct the history of sequences subject to recombination. *J. Mol. Evol.* **36**, 396–405 (1993)
19. J. Hein, T. Jiang, L. Wang, K. Zhang, On the complexity of comparing evolutionary trees. *Discr. Appl. Math.* **71**, 153–169 (1996)
20. J. Hershberger, S. Suri, Morphing binary trees, in *Proceeding of the ACM-SIAM 6th Annual Symposium of Discrete Algorithms*, 1995, San Francisco, CA, USA, pp. 396–404
21. L. Hunter (ed.), *Artificial Intelligence in Molecular Biology* (MIT, Cambridge, 1993)
22. F. Hurtado, M. Noy, J. Urrutia, Flipping edges in triangulations, in *Proceedings of the ACM 12th Annual Symposium of Computer Geometry*, 1996, pp. 214–223
23. J.P. Jarvis, J.K. Luedeman, D.R. Shier, Counterexamples in measuring the distance between binary trees. *Math. Soc. Sci.* **4**, 271–274 (1983)
24. J.P. Jarvis, J.K. Luedeman, D.R. Shier, Comments on computing the similarity of binary trees. *J. Theor. Biol.* **100**, 427–433 (1983)
25. J. Kececioglu, D. Gusfield, Reconstructing a history of recombinations from a set of sequences, in *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, USA, 1994
26. M. Kuhner, J. Felsenstein, A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol. Biol. Evol.* **11**(3), 459–468 (1994)
27. M. Krivánek, Computing the nearest neighbor interchange metric for unlabeled binary trees is NP-complete. *J. Classification* **3**, 55–60 (1986)

28. V. King, T. Warnow, On measuring the nni distance between two evolutionary trees, in *DIMACS Mini Workshop on Combinatorial Structures in Molecular Biology* (Rutgers University, New Brunswick, NJ, 1994)
29. S. Khuller, Open problems: 10. SIGACT News **24**(4), 46 (1994)
30. J. Meidanis, J.C. Setubal, *Introduction to Computational Molecular Biology* (PWS Publishing Company, Boston, 1997)
31. W.J. Le Quesne, The uniquely evolved character concept and its cladistic application. *Syst. Zool.* **23**, 513–517 (1974)
32. M. Li, J. Tromp, L.X. Zhang, On the nearest neighbor interchange distance between evolutionary trees. *J. Theor. Biol.* **182**, 463–467 (1996)
33. M. Li, L. Zhang, Better approximation of diagonal-flip transformation and rotation transformation, *Lecture Notes in Computer Science*, vol. 1449 (Springer, Berlin, Heidelberg, 1998), pp. 85–94
34. G.W. Moore, M. Goodman, J. Barnabas, An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets. *J. Theor. Biol.* **38**, 423–457 (1973)
35. J. Pallo, On rotation distance in the lattice of binary trees. *Inform. Proc. Lett.* **25**, 369–373 (1987)
36. D.F. Robinson, Comparison of labeled trees with valency three. *J. Combin. Theory Ser. B* **11**, 105–119 (1971)
37. N. Saitou, M. Nei, The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* **4**, 406–425 (1987)
38. D. Sankoff, Minimal mutation trees of sequences. *SIAM J. Appl. Math.* **28**, 35–42 (1975)
39. D. Sankoff, J. Kruskal (eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* (Addison Wesley, Reading, 1983)
40. D. Sleator, R. Tarjan, W. Thurston, Rotation distance, triangulations, and hyperbolic geometry. *J. Am. Math. Soc.* **1**, 647–681 (1988)
41. D. Sleator, R. Tarjan, W. Thurston, Short encodings of evolving structures. *SIAM J. Discrete Math.* **5**, 428–450 (1992)
42. G.A. Stephens, *String Searching Algorithms* (World Scientific Publishers, Singapore, 1994)
43. K.C. Tai, The tree-to-tree correction problem. *J. ACM* **26**, 422–433 (1979)
44. K. Wagner, Bemerkungen zum vierfarbenproblem. *J. Deutschen Math. Verin.* **46**, 26–32 (1936)
45. M.S. Waterman, *Introduction to Computational Biology: Maps, Sequences and Genomes* (Chapman & Hall, London, 1995)
46. M.S. Waterman, T.F. Smith, On the similarity of dendograms. *J. Theor. Biol.* **73**, 789–800 (1978)
47. K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.* **18**, 1245–1262 (1989)
48. K. Zhang, J. Wang, D. Sasha, On the editing distance between undirected acyclic graphs. *Int. J. Found. Comput. Sci.* **7**(13), 43–58 (1996)

Connected Dominating Set in Wireless Networks

Hongjie Du, Ling Ding, Weili Wu, Donghyun Kim, Panos M. Pardalos
and James Willson

Contents

1	Introduction	784
2	General CDS Construction	788
2.1	Unit Disk Graphs	788
2.2	Node-Weighted Unit Disk Graph	797
2.3	Unit Ball Graph	801
3	Connected Dominating Set Under Constraints	805
3.1	Network Model	808
3.2	Construction of CDS Under Constraints	809
4	Strongly Connected Domination Set	816
5	Weakly Connected Domination Set	819
6	Fault-Tolerant Virtual Backbone in Wireless Network	820
6.1	Preliminaries	821

H. Du (✉) • W. Wu

Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA
e-mail: hongjiedu@utdallas.edu; hongjiedu@hotmail.com; weiliwu@utdallas.edu; wxw020100@utdallas.edu

L. Ding

Institute of Technology, University of Washington Tacoma, Tacoma, WA, USA
e-mail: lingding@u.washington.edu

D. Kim

Department of Mathematics and Computer Science, North Carolina Central University, Durham, NC, USA
e-mail: donghyun.kim@nccu.edu

P.M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: pardalos@ufl.edu

J. Willson

Department of Computer Science, The University of New Mexico, Albuquerque, NM, USA
e-mail: jwillson@cs.unm.edu

6.2 Approximations for Computing (k, m) -CDS for $k \leq 3$ and $m \geq 1$	822
6.3 Approximations for General (k, m) -CDS.....	826
7 Conclusion.....	829
Recommended Reading.....	829

Abstract

In a graph $G = (V, E)$, a subset C of vertices is called a Connected Dominating Set if every vertex is either in C or adjacent to a vertex in C , and in addition the subgraph induced by C is connected. Given a graph, finding the minimum Connected Dominating Set is a classical combinatorial optimization problem, existing in literature for a long time. Due to wide applications of the minimum Connected Dominating Set in wireless networks, Connected Dominating Sets attract many recent research efforts. In this chapter, those developments are surveyed.

1 Introduction

A few years ago, there was a news at *AP NewsBreak* “Airspeed systems failed on US planes” which contains the following statements:

On at least a dozen recent flights by U.S. jetliners, malfunctioning equipment made it impossible for pilots to know how fast they were flying, federal investigators have discovered. A similar breakdown is believed to have played a role in the Air France crash into the Atlantic that killed all 228 people aboard in June.

While a car's speedometer uses tire rotation to calculate speed, an airplane relies on sensors known as Pitot tubes to measure changing air pressure.

Like the fatal Air France flight, the newly discovered Northwest incidents and the two other malfunctions under investigation all involved planes with sensors made by the European electronics giant Thales Corp. The Air France crash called into question the reliability of the sensors and touched off a rush to replace them.

So, the 228 people were killed by failure of sensors which measure the spread of airplane. This surprising news made sensor an important issue in our life. Nowadays, the sensor is everywhere existing in our daily life. It is usually a small and not so expensive device with three functions, sensing, computation, and communication.

In this chapter, their communication will be studied. There are two types of communication: wired and wireless. This chapter will study wireless communication. There are many types of wireless network, such as wireless sensor network, wireless ad hoc network, and wireless mesh network.

“Wireless network” is a generic term to refer to a kind of network with a number of computing devices which can form an instant network using wireless communication links. Recently, several types of wireless networks including ad hoc network and sensor network have attracted much attention. The networks can be quickly deployed without any preexisting infrastructure and thus have a broad range of applications such as battlefield surveillance, disaster rescue, environmental monitoring, conferences, concert, exact farming, traffic control, and health applications [1, 26]. In the rest of this chapter, the infrastructureless wireless networks will be denoted as *wireless networks*.

The wireless network is widely used in battlefield, environmental monitoring, agriculture, health-care industry, and biological system and traffic control. The society is even under the benefits of wireless network in our daily life, such as WIFI in the public area and mobile phone.

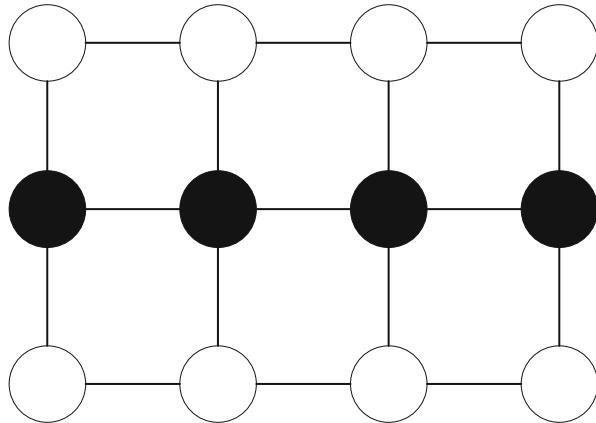
The sensors in the network are incorporated with integrated circuits to provide networking capability. The wireless sensor network essentially is an ad hoc wireless network which is composed of many sensors, also possibly many types of sensors. Usually, there is no central administration in the network, and the network is hardware-infrastructureless. In the wireless sensor network, each sensor is not only a mobile host but also a router. In other words, the sensors are able to forward the received data packages according to routing protocols.

Typically, each node (computing device) in a wireless network has a limited energy source such as a battery. The energy E consumed for transmitting a b bit message is roughly $E = b \cdot c \cdot d^\alpha$, where c is some constant and α is a value between 2 and 5. The equation implies that E increases superlinearly as the transmission distance d grows. As a result, the most dominating communication pattern in wireless network is short-distance multi-hop communication. However, multi-hop routing in a battery-operated wireless network is very challenging due to the dynamic of the network topology and the unique characteristics of wireless communication such as signal interference.

Although the wireless network is so different from the wired network and protocols in the wired network cannot be applied in the wireless network, some technologies of wired networks are still simulated by wireless networks in their developments. One of them is backbone [35], which can be used for reduction of communication overhead and increasing reliability. It is widely believed that Ephremides et al. first introduced an idea of constructing backbone-like structure in conventional wired networks such as the Internet, so-called virtual backbone (black nodes in Fig. 1), to wireless network [34]. In their scheme, a subset of nodes is selected such that (1) every node in a given network is either in the subset or adjacent to a node in the subset and (2) a subgraph induced by the subset is connected. Then, each node is required to communicate with another through the connected subset. This strategy has several apparent benefits for wireless network. Most of all, only the nodes in the structure are involved in routing. This narrows routing search space, reduces the amount of routing-related control messages, and helps a routing protocol to converge faster [71].

Clearly, the advantage of the structure can be magnified as the size of the subset becomes smaller. This motivated many people to investigate for computing smaller virtual backbones in wireless networks. Guha and Kuller modeled the problem of computing a smallest virtual backbone as the minimum Connected Dominating Set (CDS) problem [43]. However, the minimum CDS problem is a very well-known NP-Hard problem [42]. Thus, they introduced approximation algorithms for minimum CDS. Since then, extensive researches have been conducted on the problem of computing quality virtual backbone in wireless network.

Dominating Sets and Connected Dominating Sets are traditional research subjects in graph theory [69].

Fig. 1 Virtual backbone

Definition 1 (Dominating Set (DS)) Given a graph $G = (V, E)$, $V' \subseteq V$ is a DS of $G = (V, E)$ only if $\forall(u, v) \in E$, either $u \in V'$ or $v \in V'$ is true.

Definition 2 (Connected Dominating Set (CDS)) $C \subseteq V$ is a CDS of G if (1) C is a DS and (2) a graph induced by C is connected.

In graph theory, the minimum size of (Connected) Dominating Set in G is called the (*connected*) domination number of G . Determination of the domination number and the connected domination number in various graphs have attracted many research efforts [10, 16] since computing the dominating number and the connected dominating number in general graph is NP-hard [42].

Since Connected Dominating Set plays an important role of virtual backbone in wireless networks, computing the minimum Connected Dominating Set becomes a hot research topic in both theoretical computer science and computer network due to its impact in wireless networks.

In general graphs, several works have been studied in recent research literature. In [43], Guha and Khuller proposed two polynomial-time algorithms to construct a CDS in a general undirected graph G . These algorithms are greedy and centralized. The first one has a performance ratio of $2(H(\Delta) + 1)$ where H is a harmonic function. The idea of this algorithm is to build a spanning tree T rooted at the node with a maximum degree and grow T until all nodes are added to T . The non-leaf nodes in T form a CDS. In particular, all nodes in a given network are marked white initially. The greedy function that the algorithm uses to add nodes into T is the number of the white neighbors of each node or a pair of nodes. The one with the largest such number is marked in black, and its neighbors are marked in gray. These nodes (black and gray nodes) are then added to T . The algorithm stops when no white node exists in G . The second algorithm is an improvement of the first one. This algorithm consists of two phases. The first phase is to construct a Dominating Set and the second phase is to connect the Dominating Set using a Steiner tree algorithm. With such improvement, the second algorithm has a performance factor

of $H(\Delta) + 2$. These algorithms later were studied and implemented by Das et al. [7, 20, 72].

For the localized algorithms, Wu and Li [82] proposed a simple algorithm that can quickly determine a CDS based on the connectivity information within the 2-hop neighborhood. This approach uses a marking process. In particular, each node is marked true if it has two unconnected neighbors. All the marked nodes form a CDS. The authors also introduced some dominant pruning rules to reduce the size of the CDS. In [76], Wan et al. showed that the performance ratio of [82] is within a factor of $O(n)$, where n is the number of nodes in a network.

CDS algorithms can be divided into two types: one is 2-stage and the other one is 1-stage. The 2-stage algorithms can also be divided into two categories. The main idea of the first category is to construct a Dominating Set(DS) and add more nodes to connect DS into CDS. As a result, a CDS is constructed [12, 43, 61, 84]. The other category is to construct a redundant CDS first, then remove some nodes to get a smaller CDS [82]. The main idea of 1-stage algorithms aims to select a CDS directly, skipping the step of selection of a DS or a redundant CDS [43, 67].

Guha and Khuller [43] gave a two-stage greedy approximation within a factor of $3 + \ln \delta$ from optimal for the minimum Connected Dominating Set where δ is the maximum node degree of input graph. They also showed that this is almost the best possible, i.e., there is no polynomial-time approximation for the minimum Connected Dominating Set within a factor of $\rho \ln \delta$ from optimal for any $0 < \rho < 1$ unless $NP \subseteq DTIME(n^{O(\log \log n)})$. This lower bound result is obtained by a transformation from the result of Feige [37] on the set cover problem. Ruan et al. [67] improved upper bound from $3 + \ln \delta$ to $2 + \ln \delta$. Du et al. [29] showed a tight upper bound that the minimum Connected Dominating Set in general graph can be polynomial-time approximated within a factor of $\alpha(1 + \ln \delta)$ from optimal for any $0 < \alpha < 1$.

While virtual backbone in wireless networks becomes a very important topic, the computer science people pay their attention more and more on computing approximation solution for it. Because graph is usually used to represent the network model, minimizing the size of Dominating Set and Connected Dominating Set is much more important in the research of virtual backbone in wireless networks. Constructing DS and CDS is a real-world problem, people in combinatorics group are still finding the better solution for them. This area has many applications and some of them need other requirements, such as message optimal [3], distributed construction [76], strongly connected [28], weakly connected [11, 30], bounded diameter [51], minimum routing cost [23, 25], fault-tolerant [88], r -hop dominating [56], and shortest path condition [81, 82]. Those requirements give more potential research topics about Connected Dominating Set in unit disk graph in the future. In this article, those developments in the direction to approximation solution will be recalled, especially in graphs which are mathematical models of wireless networks. In Sect. 2, the general CDS construction will be introduced. CDS under constraints will also be recalled in Sect. 3. Strongly and Weakly Connected Dominating Set are shown in Sects. 4 and 5. Section 6 will introduce the fault-tolerant CDS construction in wireless networks.

2 General CDS Construction

2.1 Unit Disk Graphs

Usually, a graph is used to represent the network model of wireless networks. The simplest mathematical model is unit disk graph(UDG).

Definition 3 (Unit disk graph (UDG)) $G = (V, E)$ is a UDG if $\forall u, v \in V$; there is a bidirectional edge between u and v if and only if $Eucdist(u, v) \leq 1$.

A *unit disk graph* is associated with a set of unit disks in the Euclidean plane. Each node represents the center of a unit disk. An edge exists between two nodes u and v if and only if the distance between u and v is at most one. Huson and Sen [48] presented the first paper to use UDG to abstract homogeneous (i.e., with the equivalent capabilities) wireless network. More detailed information on the graph theoretical characteristics of UDG can be found in [17].

When researchers study wireless networks, they assume all sensors have the same communication range, and two sensors are able to communicate each other if and only if their distance is within their communication range, which is a disk with certain radius. Therefore, it is isomorphic to a *unit disk graph*.

Computing the minimum Connected Dominating Set in *unit disk graph* has been proved as NP-hard [17]. Therefore, researchers usually design the approximation algorithms for minimum Connected Dominating Set (MCDS) problems. Fortunately, computing approximation solutions for MCDS in UDG is a little bit easier than in general graphs. The first polynomial-time constant approximation is obtained by Wan et al. [76]. The most important part of their work is the frame of their approximation solution. The detail of their method will be introduced in Sect. 2.1.1. Their work is followed later by almost all approximation solutions [9, 38, 39, 58, 61, 77] for the minimum Connected Dominating Set in unit disk graph with good performance which is provable in theory and comparable in computational experiments.

2.1.1 Maximal Independent Set Method

It is a popular way to construct a CDS by constructing a DS first, then connecting the DS into a CDS. For construction of a Dominating Set, the best-known easy way is to build a *maximal independent set (MIS)*. Therefore, the approximation ratio of minimum Connected Dominating Set problem would be determined by two facts. One is the ratio compared *maximal independent set* with *Connected Dominating Set*, and the other is how many nodes are required to connect MIS into a CDS. Clearly, an MIS of a graph is a DS of the graph. Computing a minimum DS is an NP-hard problem [49], and thus computing a minimum IS is NP-hard. An MIS of a graph G can be easily computed using the following graph coloring scheme: (1) initially, color all node white; (2) color one white node black, and its neighbors blue; and (3) repeat Step 2 until there is no white node. As a result, this coloring-based MIS computation strategy is frequently used to compute a DS.

Alzoubi et al. [2, 76] made a great improvement by proposing two 2-phase distributed algorithms. They first construct a spanning tree, then labeled each node in the tree either as a dominator or a dominatee. The algorithms obtained a constant performance ratio which is 8 in UDG. Alzoubi et al. [3] noticed that it is difficult to maintain their previous algorithms of constructing the CDS. Therefore, they designed a localized distributed 2-phase algorithm which can be maintained in general graphs easily. The new algorithm described as follows: first, generate an MIS in a distributed fashion without building a tree or selecting a leader. The node becomes a dominator if it has the smallest ID within its one-hop neighbors. After there are no more white nodes, the dominators will identify a path to connect all the dominators. This algorithm got the performance ratio of 192, and no network connectivity information is utilized in this algorithm. Another localized algorithm is proposed by Li et al. in [58], they reduced the performance ratio from 192 to 172. The localized algorithm in [58] only has one phase which is a node marks itself as a dominator if it can cover the most white nodes compared to its two-hop neighbors.

Definition 4 (Independent set (IS)) An IS I of $G = (V, E)$ is a subset of V such that $\forall u, v \in I, (u, v) \notin E$.

Definition 5 (Maximal independent set (MIS)) An IS I of $G = (V, E)$ is an MIS if $\forall v \in (V \setminus I), I \cup \{v\}$ is not an IS anymore.

Wan's et al. [76] frame also consists of two stages. They construct a Dominating Set in the first stage and connect the Dominating Set into a Connected Dominating Set in the second stage. Since using the maximal independent set method can build an approximation solution, estimating the size of a maximal independent set would be a very important part of analysis of this type of approximation. They consider a minimum Connected Dominating Set C^* . They construct a disk with center u and radius one, called *dominating disk* of u for each node $u \in C^*$. Then the set of dominating disks in C^* will contain all nodes in the graph because C^* is a Dominating Set. By the definition of MIS, all nodes in a maximal independent set are away from each other with the distance larger than one.

Order nodes in $C^* = \{a_1, a_2, \dots, a_{opt}\}$ in such way that every $C_j^* = \{a_1, \dots, a_j\}$ for $1 \leq j \leq opt$ induces a connected subgraph.

First, consider the dominating disk of u , how many independent nodes it can contain? Suppose a_1, \dots, a_k are independent nodes in the dominating disk of u as shown in Fig. 2. Then $\angle a_1ua_2 > 60^\circ, \angle a_2ua_3 > 60^\circ, \dots, \angle a_kua_1 > 60^\circ$. Therefore, $k \leq 5$.

Next, they consider the dominating disk of a_j for $j \geq 2$. How many independent nodes can be contained in the dominating disk of a_j , but not in the dominating disk of a_i for any $i = 1, 2, \dots, j-1$? Suppose b_1, b_2, \dots, b_k are those independent nodes. Since C_j^* induces a connected subgraph, there must exist $i, 1 \leq i \leq j-1$, such that a_i lies in the Dominating Set of a_j . Since b_1, b_2, \dots, b_k do not lie in the dominating disk of a_i . $a_i, b_1, b_2, \dots, b_k$ are independent nodes lying in the dominating disk of a_j . Therefore, $k \leq 4$. This concludes that the total number of independent nodes lying in the union of

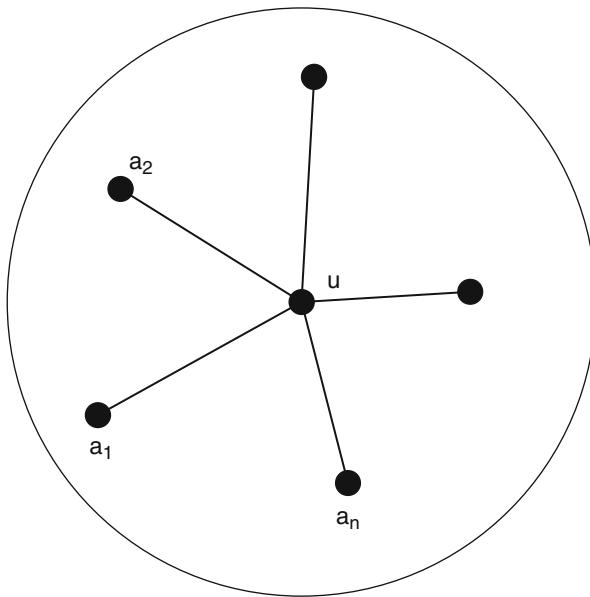


Fig. 2 Independent nodes in a dominating disk of u

dominating disk of nodes in C^* is at most $1 + 4 \text{ opt}$ where $\text{opt} = |C^*|$. This is a key lemma for Wan et al. [76] to show the constant performance ratio of their approximation.

Lemma 1 (Wan et al. [76]) Any maximal independent set has size at most $1 + 4 \text{ opt}$ where opt is the size of the minimum Connected Dominating Set.

2.1.2 Current Best Result For MIS Method

What is the possible best upper bound for the size of a maximal independent set? They have the following conjecture:

Open Problem 1 (Conjecture) In a unit disk graph, any maximal independent set has size $|mis| \leq 2 + 3 \cdot \text{opt}$ where opt is the size of the minimum Connected Dominating Set.

Several methods have established to attack this conjecture. The first method is based on new results on disk packing. Wu et al. [84] showed that:

1. Two dominating disks of u and v with distance at most one can contain at most eight independent nodes.
2. For any UDG, there exists a minimum spanning tree such that every vertex has degree at most five.
3. For every minimum spanning tree T with at least three vertices, it has a non-leaf vertex adjacent to at most one non-leaf vertex.

Based on these facts, they showed that the size of a maximal independent set is at most $3.8 \cdot opt + 1.2$. By the results of [9, 76], the performance ratio can be reduced to 7.8 from 8. Wan et al. [77] showed that for every connected UDG G with at least two nodes, if the dominating disk of u_0 contains u_1, u_2, u_3 , then the union of dominating disks of u_0, u_1, u_2, u_3 can contain at most 15 independent nodes and with this fact, further showed that $|mis| \leq 1 + 3\frac{2}{3} \cdot opt$. In another direction, Wan et al. [77] presented an example to show that $|mis| = 3 \cdot opt + 2$ is reachable.

A simple area argument was proposed by Funke et al. in [38]. They assumed that all nodes have an equal transmission range which is 2, and two nodes are adjacent if the two disks centered at the nodes have intersection. That means the inter distance of these two nodes is at most two. They use this definition instead of the commonly one to simplify their calculations. Then two adjacent nodes have at least $\frac{9}{2} \arccos \frac{1}{3} - \sqrt{2}$ area in common. Thus, there are at most

$$(opt - 1)(\pi 1.5^2 - \frac{9}{2} \arccos \frac{1}{3} + \sqrt{2}) + \pi 1.5^2$$

area in a minimum Connected Dominating Set. For every node y in a *maximal independent set*, draw a disk with center y and radius 0.5. All such disks are disjoint and lie in the adjacent areas of nodes in the minimum Connected Dominating Set. Therefore,

$$|mis| \leq \frac{(opt - 1) \left(\pi 1.5^2 - \frac{9}{2} \arccos \frac{1}{3} + \sqrt{2} \right) + \pi 1.5^2}{0.25\pi} \leq 3.748 \cdot opt + 9$$

Funke et al. [38] claimed a much better bound $|mis| \leq 3.453 \cdot opt + 8.291$ which can reduce the ratio to 6.91. However, the proof involves an unproved fact about disk packing. Therefore, this bound is not recognized as proved.

With a combinatorial method, Gao et al. [41] showed that $|mis| \leq 3.453 \cdot opt + 4.839$ by the comparison of area for MCDS and area for smallest Voronoi cell.

Definition 6 Let S be a set of n sites in Euclidean space. For each site p_i of S , the Voronoi cell $V(p_i)$ of p_i is the set of points that are closer to p_i than to other sites of S , say,

$$V(p_i) = \bigcap_{1 \leq j \leq n, j \neq i} \{p : |p - p_i| \leq |p - p_j|\}.$$

The Voronoi diagram $V(S)$ is the space partition induced by Voronoi cells.

Very recently, Li et al. [57] employed a geometry argument and push this bound further to $|mis| \leq 3.4306 \cdot opt + 4.8185$. They also tight the approximation ratio

of the algorithm in [77] to at most 6.075. They will introduce their main results in detail as follows:

Theorem 1 (Li et al. in [57]) *Let α and γ_c be the independence number and connected domination number of a connected UDG G . Then, $\alpha \leq 3.4306 \gamma_c + 4.8185$.*

They prove the above theorem by an integrated area and length argument. Let U be a minimum CDS of G , and define

$$\Omega = \cup_{u \in U} \text{disk}_{1.5}(u).$$

I is a maximum independent set of G . They used I to construct the Voronoi diagram. For each $o \in I$, its Voronoi cell is represented by V or (o) and the *truncated Voronoi cell* of o is represented by the set V or $(o) \cap \Omega$. It is easy to see that the total area of truncated Voronoi cells of all nodes in I is $|\Omega|$. They partition I into two subsets I_1 and I_2 defined by

$$\begin{aligned} I_1 &= \{o \in I : \text{disk}_{1/\sqrt{3}}(o) \subset \Omega\} \\ I_2 &= I \setminus I_1. \end{aligned}$$

α_1 and α_2 are the size of I_1 and I_2 , respectively. **Lemma 2** provides a lower bound on each truncated Voronoi cell.

Lemma 2 (Li et al. in [57]) *For each o in I_1 (respectively, I_2), the area of its truncated Voronoi cell is at least $\sqrt{3}/2$ (respectively, σ).*

Lemma 3 (Li et al. in [57]) *The length of $\vartheta\Omega'$ is at most $2(1 - 1/\sqrt{3})\alpha_2$, where $\vartheta\Omega'$ is the topological boundary of a set Ω' , and $\Omega' = \cup_{v \in U} \text{disk}_{1.5-1/\sqrt{3}}(v)$.*

By **Lemma 2**, they can get

$$|\Omega| \geq \frac{\sqrt{3}}{2}\alpha_1 + \sigma\alpha_2 = \frac{\sqrt{3}}{2}\alpha - \left(\frac{\sqrt{3}}{2} - \sigma\right)\alpha_2$$

which implies

$$\alpha \leq \frac{|\Omega|}{\frac{\sqrt{3}}{2}} + \left(1 - \frac{\sigma}{\frac{\sqrt{3}}{2}}\right)\alpha_2. \quad (1)$$

It is easy to prove by induction on γ_c that

$$|\Omega| \leq \frac{9}{2} \left((\gamma_c - 1) \left(\arcsin \frac{1}{3} + \frac{\sqrt{8}}{9} \right) + \frac{\pi}{2} \right), \quad (2)$$

and the length of $\vartheta \Omega'$ is at most

$$\begin{aligned} & 2 \left(3 - \frac{2}{\sqrt{3}} \right) \left((\gamma_c - 1) \arcsin \frac{1}{3 - \frac{2}{\sqrt{3}}} + \frac{\pi}{2} \right). \\ \alpha_2 & \leq \frac{2 \left(3 - \frac{2}{\sqrt{3}} \right) \left((\gamma_c - 1) \arcsin \frac{1}{3 - \frac{2}{\sqrt{3}}} + \frac{\pi}{2} \right)}{2 \left(1 - \frac{1}{\sqrt{3}} \right)} \\ & = \frac{\sqrt{3} + 7}{2} \left(\left((\gamma_c - 1) \arcsin \frac{1}{3 - \frac{2}{\sqrt{3}}} + \frac{\pi}{2} \right) \right). \end{aligned} \quad (3)$$

The three inequalities 1, 2, and 3 imply altogether that α is at most

$$\begin{aligned} & (\gamma_c - 1) \left(\sqrt{27} \left(\arcsin \frac{1}{3} - \frac{\sqrt{8}}{9} \right) + \frac{\left(1 - \frac{\sigma}{\frac{\sqrt{3}}{2}} \right) (\sqrt{3} + 7)}{2} \arcsin \frac{1}{3 - \frac{2}{\sqrt{3}}} \right) \\ & + \frac{\pi}{2} \left(\sqrt{27} + \left(1 - \frac{\sigma}{\frac{\sqrt{3}}{2}} \right) \frac{\sqrt{3} + 7}{2} \right) \\ & \approx 3.4305176\gamma_c + 4.8184688. \end{aligned}$$

Thus, Theorem 1 is proved.

2.1.3 Steiner Tree Method

The progress on the second stage is going quite fast when the research on Open Problem 1 is still quite active. A naive way to connect a Dominating Set into a connect Dominating Set is to construct a minimum spanning tree. This construction can be also combined with the construction of maximal independent set. Defer from the previous method, they choose a uncolored node adjacent to a gray node instead of making an arbitrary choice starting from the second iteration. Then there is only one gray node in each “edge” in a minimum spanning tree for black nodes. Thus, they need only $|mis| - 1$ gray nodes to connect all black nodes into a connect Dominating Set. This will form a Connected Dominating Set with size $2|mis| - 1 \leq 6.8612 \cdot opt + 8.637$.

- A popular strategy to compute a CDS of a graph with Steiner tree method is:
1. Compute an MIS I of the graph, which is also a DS.
 2. Add more nodes in $V \setminus I$ to I using a Steiner or spanning algorithm so that the I can be connected.

This strategy is initially introduced by Guha and Kuller [43], and nowadays it is the most common approach to find an approximate solution for the minimum CDS problem.

Of course, it is a better idea to employ the Steiner tree to do minimize the size of Connected Dominating Set.

Definition 7 (Steiner tree) A *Steiner tree* for a given subset of nodes, called *terminals*, in a graph is a tree interconnecting all terminals such that every leaf is a terminal.

Definition 8 (Steiner node) Every node other than terminals in the Steiner tree is called a *Steiner node*.

Clearly, they prefer to minimize the number of Steiner nodes instead of the total edge length for interconnecting Dominating Set. It is the objective in classic Steiner tree problems. This means that they need to study the following problem:

Definition 9 (Steiner tree with minimum number of Steiner nodes (ST-MSN)) Given a unit disk graph G and a Dominating Set D of nodes, compute a Steiner tree for D with the minimum number of Steiner nodes.

Although the ST-MSN problem in *unit disk graph* has not been studied very much, its geometric version in the Euclidean plane has been studied extensively [12, 27, 60]. However, some results cannot be extended to UDG. For example, two points with distance 2 can be connected with a Steiner point in the Euclidean plane. But, two nodes with distance 2 may not be able to be connected by a Steiner node because such a node may not exist. Fortunately, some results in the Euclidean plane can be extended to unit disk graph. By performing such an extension, Min et al. [61] obtained a polynomial-time 3-approximation for the ST-MSN problem which can be extended from the Euclidean plane to UDG with a quite different proof, which becomes a fundamental part in their approximation algorithm. They found that the size of optimal solution for the ST-MSN cannot exceed the size of the minimum Connected Dominating Set that is because the latter can also interconnect the maximal independent set. Therefore, at most $3opt$ Steiner nodes are used in the second step. Hence, they got the approximation solution for the minimum Connected Dominating Set which has size at most $6.8 \cdot opt$ based on the number bound for $|mis|$.

For the classical network Steiner tree problem, Li et al. [58] presented a greedy algorithm with performance ratio of $4.8 + \ln 5$; their algorithm can also be implemented as a distributed algorithm. Their algorithm also has two phases:

1. Find the MIS.
2. Use Steiner tree algorithm to connect the MIS.

The Steiner tree algorithm takes the property which is that there are at most five independent nodes adjacent to one node.

2.1.4 Current Best Result For Steiner Tree Method

Current best approximation is constructed by Robins et al. [66]. They used greedy strategy to present a new polynomial-time heuristic with an approximation ratio approaching $1 + \frac{\ln 3}{2} \approx 1.55$. They reduced result from the previously best-known approximation algorithm of [45] with performance ratio ≈ 1.59 . The same as Li et al. [58], they also use greedy algorithm to achieve the best performance for interconnecting Dominating Set. Actually, it is normal to use greedy algorithms to design this problem such as [39, 58, 77]. Li et al. [58] found that the greedy approximation in [77] has the performance ratio 6.075.

Almost all previous heuristics algorithms which have good approximation ratio choose appropriate full components and then prune them in the interest of keeping them for the overall solution. However, if they may find a better component later, they cannot abandon the previous component which they already selected. It is because two components disagree if they share at least two terminals.

In [66], Robins et al. proposed an algorithm called Loss-Contracting Algorithm ([Algorithm 1](#)). The algorithm will prune the full components and keep them as little as possible so that (a) a chosen full component may still participate into the overall solution but (b) not many other full components would be rejected. For example, if they delete $Loss(K)$, i.e., replace a full component K by $C[K]$, then (a) it will not cost anything to add a full component K in the overall solution and (b) they decrease the gain of full components which disagree with K by a small value such as the value is less than in the Berman-Ramayer algorithm [6] for large k and much smaller than in [50] for any k .

The algorithm first constructed a minimum spanning tree of G_S called $MST(G_S)$ then iteratively modified the terminal-spanning tree T which is originally $MST(G_S)$ by incorporating into T loss-contracted full components greedily chosen from G . The cost of the approximate solution lies between $mst = mst(G_S)$ and opt_k . The intuition behind the gain-over-loss objective ratio is as follows [66]:

1. If they accept a component K , then it increases the gap between mst and the cost of the approximation by gaining a K . Thus, the gain of K is our clear profit.
2. If K does not belong to OPT_k , then after accepting K they would no longer be able to reach Opt_k because they would need to pay for the connection of incorrectly chosen Steiner points. Therefore, the value of $loss(K)$, which is the connection cost of Steiner points of K to terminals, is an upper bound on the increase in the cost gap between opt_k and the best achievable solution after accepting K . Thus, $loss(K)$ is an estimate of our connection expense.
3. Maximizing the ratio of gain over loss is equivalent to maximizing the profit per unit expense.

Theorem 2 (Robins et al. [66]) *For any instance of the Steiner tree problem, the cost approx of the Steiner tree produced by algorithm k-LCA is at most*

$$Approx \leq loss_k \cdot \ln\left(1 + \frac{mst - opt_k}{loss_k}\right) + opt_k.$$

Algorithm 1 Loss-Contracting Algorithm ($k - LCA$) for Steiner Trees in Graphs (Robins et al. [66])

0: **Input:** A complete graph $G = (V, E, \text{cost})$ with *edge costs* satisfying the triangle inequality, a set of terminals $S \subset V$, and an integer $k \leq |S|$
Output: A k -restricted Steiner tree in G connecting all the terminals in S

- 1: $T = MST(G_S)$
- 2: $H = G_S$
- 3: **Repeat forever**
- 4: Find a k -restricted full component K with the maximum $r = gain_T(K)/loss(K)$
- 5: **If** $r \leq 0$ **then exit repeat**
- 6: $H = H \cup K$
- 7: $T = MST(T \cup C[K])$
- 8: **Output** the tree $MST(H)$

2.1.5 PTAS for MCDS in UDG

They might want to know whether there exists a lower bound for the performance ratio of polynomial-time approximation for the minimum Connected Dominating Set in unit disk graph. The truth is there is no such thing. Actually, Cheng et al. [14] already showed that there exists a PTAS (polynomial-time approximation scheme) for MCDS problem.

Definition 10 (PTAS) For any $\varepsilon > 0$, there exists a polynomial-time approximation within a factor of $1 + \varepsilon$ from optimal.

However, the running time of such an approximation is $O(n^{1/\varepsilon^2})$. It cannot be implemented to use in the real world. Therefore, they are still making efforts to look for approximations with a smaller performance ratio and a low-degree polynomial for running time.

Cheng et al. [14] designed a $(1 + 1/s)$ -approximation for the minimum Connected Dominating Set in unit disk graph. The running time is $n^{O((s \log s)^2)}$. There is an evidence [14] to show that currently existing implemented approximations have a large space for improvement. In this section, they will introduce their main results.

An algorithm \mathbb{A} is a PTAS for a minimization problem with optimal cost OPT if the following is true:

Given an instance I of the problem and a small positive error parameter ε :

1. The algorithm outputs a solution with cost at most $(1 + \varepsilon)OPT$.
2. When ε is fixed, the running time is bounded by a polynomial in the size of the instance I .

If there exists a PTAS for an optimization problem, the problem's instance can be approximated to any required degree.

To design a PTAS for MCDS in UDG, they quote the result from [13] which is:

Lemma 4 (Cheng et al. [13]) *There exists a polynomial-time approximation for the MCDS in unit disk graph, with performance ratio 8.*

The general picture of constructing the PTAS for the MCDS in UDG [14] is as follows:

1. Divide a space and make it contain all vertices of the input unit disk graph, into a grid of small cells. For each small cell, take the points h distance away from the boundary (the central area of the cell).
2. Optimally compute a minimum union of Connected Dominating Sets in each cell for connected components of the central area of the cell. The key lemma is to prove that the union of all such minimum unions is no more than the MCDS for the whole graph.
3. For vertices not in central areas, just use the part of an 8-approximation lying in boundary areas (within distance $h + 1$ away from the boundary, with some overlap with the central areas) to dominate them. This part, together with the above union, forms a Connected Dominating Set for the whole input unit disk graph.
4. Using the shifting argument (i.e., shift the grid around to get partitions at different coordinates) to make sure there are at least half good partitions having good approximations overall.

Corollary 1 (Cheng et al. [14]) *There is a $(1 + 1/s)$ -approximation for an MCDS in connected unit disk graphs, running in time $n^{O((s \log s)^2)}$.*

The proof of this corollary can be found in [14].

2.2 Node-Weighted Unit Disk Graph

In real world, different sensors do not have the same value. Therefore, when wireless sensor networks contain many types of sensors with the same transmission range but different costs, the mathematical model would be changed to the unit disk graph with node weight. For virtual backbone, they would prefer to have Connected Dominating Set with minimum total weight rather than Connected Dominating Set with minimum cardinality.

If they want to use the same construction method as unit disk graph, they should construct the minimum-weight Dominating Set first. However, they could not continue using the maximal independent set method, because, there is no efficient way for constructing a minimum-weight MIS unless $NP = P$ [42]. Actually, how to construct a constant-approximation algorithm for minimum-weight Dominating Set in UDG is still an open problem.

In 2006, Ambühl et al. [5] presented a 72-approximation algorithm for this problem by using the partition technique. Gao et al. [40] introduced a new technique, called *double partition* in 2008. They proposed an algorithm with performance ratio of $6 + \varepsilon$. A year later, Dai et al. [19] reduced the ratio to $5 + \varepsilon$. Zou et al. [90] further improved the performance ratio to $(4 + \varepsilon)$ in the same year.

For connecting the DS into minimum-weight Connected Dominating Set in unit disk graph, Ambühl et al. [5] spent $17opt$ to connect their previous solution and got an 89-approximation for minimum-weight Connected Dominating Set where opt is the object function value of an optimal solution which is the minimum total weight of Connected Dominating Set. Gao et al. [40] decreased the cost for interconnection to $4opt$. Zou et al. [92] further reduced the cost to $2.5\rho \cdot opt \approx 3.875opt$, where ρ is the best-known performance ratio for polynomial-time approximation of classical Steiner minimum tree problem in graphs. This result can be used to connect a Dominating Set by adding nodes of weight at most 2.5ρ times the weight of an optimal Connected Dominating Set, yielding the approximation ratio of 8.875 for MWCDS. Later Zou [90] and Erlebach et al. [36] both obtained $4 + \varepsilon$ -approximation for minimum-weight Dominating Set, and this result will maintain a 7.875-approximation for MWCDS, which improves the previously best approximation ratio of 8.875.

In [90], they used dynamic programming algorithm for the *min-weight chromatic disk cover* to propose a $(4 + \varepsilon)$ -approximation algorithm for **MWDS**. For a given set \mathbb{D} of unit disks, they assign positive weight to each disk by a function c and a set P of nodes in the plane. They colored each disk $D \in \mathbb{D}$ either red or blue. Assume there are $m + 1 \geq 2$ distinct horizontal lines $y = y_i$ for $0 \leq i \leq m$ from the top to the bottom, where m is a nonnegative integer constant. Therefore, the $m + 1$ lines separate the plane into $m + 2$ horizontal strips. A node $p \in P$ is chromatically covered by a red/blue disk $D \in \mathbb{D}$ if $p \in D$ and the center of D is above/below the strip which contains p . A *chromatic cover* of P is a subset $D \in \mathbb{D}$ which each node in P is chromatically covered by some disk in \mathbb{D}' . The *min-weight chromatic disk cover* (**MWCDC**) problem is to find a min-weight chromatic cover $D \in \mathbb{D}$ of P . They proved that there is a polynomial $(4\rho + \varepsilon)$ -approximation algorithm for the MWDS for any fixed ε by proposing a ρ -approximation algorithm for the **MWCDC**.

In [36], Erlebach et al. proposed a $(4 + \varepsilon)$ -approximation algorithm for **MWDS** in unit disk graph. Their algorithm is based on several ideas of previous constant-factor approximation algorithms for the problem [5, 47]. They first partitioned the plane into $K \times K$ areas where K is an arbitrary constant. They considered the following subproblem for each of these areas:

Find a minimum-weight set of disks that dominate all disks that have a center in the area.

The union of feasible solutions for each subproblem produces a Dominating Set for the original problem. For the shifting technique presented in [47], the loss in the approximation factor is only $(1 + O(1)/K)$. Therefore, the (best) combination of the solutions is a $(\rho + O(1)/K)$ -approximation for the original problem if the solution for every subproblem is a ρ -approximation by using the shifting technique.

Thus, they can set K such that the obtained solution is a $(\rho + \varepsilon)$ -approximation for any constant ε . They presented a 4-approximation algorithm for the subproblem. By using the shifting technique and setting K appropriately, the algorithm forms a $(4 + \varepsilon)$ -approximation algorithm.

2.2.1 2.5ρ Approximation Algorithm in [92]

The main idea of this algorithm is based on the classical Steiner tree problem. They first constructed an edge-weighted graph G'' with the same node and edge set as the original graph G . Then, they defined the weight of every edge in G'' as the half of the sum of its endpoints' weights in G . Finally, they used ρ -approximation algorithm to obtain a Steiner tree of G'' . From the Steiner tree of G'' , they can get a Steiner tree of G with approximation ratio $2.5\rho \approx 3.875$. In [Corollary 2](#), they obtained a $8.875 + \varepsilon$ algorithm for minimum-weight Connected Dominating Set in unit disk graph.

1. Preliminaries

In this section, they introduce some useful definitions and denotations in [\[92\]](#). For a node-weighted (or edge-weighted) graph G with weight function w and a subgraph H of G , they use $w(H)$ to represent the weight sum of all nodes (or edges) in H . For any two nodes u and v of G , $dist_G(u, v)$ is the weight of the shortest path between u and v , which is calculated as $\min \sum_{v_k \in P} w(v_k)$ among all the possible paths between u and v which v_k is the internal node on every possible path p . Given an edge-weighted graph G and a node subset S , they use $SMT(G, S)$ and $MST(G)$ to represent the ρ -approximation algorithm for Steiner minimum tree (SMT) and the algorithm for finding minimum spanning tree (MST).

2. 2.5ρ Approximation Algorithm (Zou et al. [\[92\]](#))

The idea of this algorithm is to convert the node-weighted Steiner tree problem to the classical Steiner tree problem. There are three steps:

- Construct an edge-weighted graph G'' from G as follows by initializing G'' with the same node set and edge set as G and an edge weight function w' . For every edge $e = (u, v)$ in G'' , let the edge weight $w'(u, v) = \frac{1}{2}(w(u) + w(v))$.
- Compute a Steiner tree T of G'' on S through the ρ -approximation algorithm.
- View T as the node-weighted Steiner tree of G on S and output it.

The pseudo-code of this algorithm is presented in [Algorithm 2](#).

Algorithm 2 $NWST(G = (V, E, w, S))$ (Zou et al. [\[92\]](#))

- Initialize an edge-weighted graph $G' = (V', E', w', S')$ by setting $V' = V, S' = S$ and $E' = E$
 - for** each edge (v_i, v_j) in graph G' **do**
 - Assign the weight of this edge $w'(v_i, v_j) = (w(v_i) + w(v_j))/2$
 - end for**
 - $T = SMT(G', S)$
 - Output T
-

Since G is a unit disk graph, for any terminal set S , there always exists a minimum Steiner tree with maximum degree no more than 5 [61]. The following lemma proves the relationship between the weight of the optimal SMT in graph G' and the weight of the optimal NWST they want.

Lemma 5 (Zou et al. [92]) Denote $T_{OPT_{SM}}$ as the optimal Steiner minimum tree for G'' on the set S and $T_{OPT_{NWS}}$ as the optimal node-weighted Steiner tree of G on the same terminal set S , respectively. Then $w'(T_{OPT_{SM}}) \leq 2.5w(T_{OPT_{NWS}})$ when G is a unit disk graph.

Proof (Zou et al. [92]) Consider $T_{OPT_{NWS}}$ as a Steiner tree on S of G'' . For convenience, denote T as $T_{OPT_{NWS}}$, V as $V(T_{OPT_{NWS}})$, E as $E(T_{OPT_{NWS}})$, and $d_T(u)$ as the degree of node u in tree T ; they have

$$\begin{aligned} w'(T_{OPT_{SM}}) &\leq w'(T_{OPT_{NWS}}) = \sum_{e=(u,v) \in E} \left(\frac{1}{2}(w(u) + w(v)) \right) \\ &= \sum_{u \in V} \frac{d_T(u)}{2} w(u) \leq \frac{5}{2} w(T). \end{aligned}$$

Hence, the lemma holds.

If they consider the Steiner tree T of G'' as a subgraph of G , they can get the following lemma:

Lemma 6 (Zou et al. [92]) For any Steiner tree T of G'' on terminal set S , if they view it as a subgraph G , then T is also a Steiner tree of G on set S and $w(T) \leq w'(T)$.

Proof (Zou et al. [92]) Since for any Steiner tree, the degree of each Steiner node is not less than 2 and all weight of nodes in terminal set is 0, the lemma holds.

Theorem 3 (Zou et al. [92]) Algorithm NWST is a 2.5ρ -approximation for node-weighted Steiner tree problem in unit disk graph.

Proof (Zou et al. [92]) Let T be the output of NWST. Denote $T_{OPT_{SM}}$ and $T_{OPT_{NWS}}$ as Lemma 5. By Lemmas 5 and 6, they have

$$w(T) \leq w'(T) \leq \rho w'(T_{OPT_{SM}}) \leq 2.5\rho w(T_{OPT_{NWS}}).$$

Since the node-weighted Steiner tree can be used in the MWCDS problem to interconnect all nodes of the CDS, they can obtain the following corollary:

Corollary 2 (Zou et al. [92]) *Using node-weighted Steiner tree to interconnect all nodes of the CDS produces $(9.875 + \varepsilon)$ -approximation for minimum-weight Connected Dominating Set in unit disk graph.*

Proof (Zou et al. [92]) For any node-weighted graph G and a terminal set S , denote T_{OPT} and $T_{OPT_{CDS}}$ be the optimal Steiner tree of G on S and optimal CDS of G , respectively. Since the induced graph $G[S \cup T_{OPT_{CDS}}]$ is connected, this graph contains a Steiner tree of G on S . Thus, $w(T_{OPT}) \leq w(T_{OPT_{CDS}})$. By Huang et al. [47], they can get a Dominating Set C of G with $w(C) \leq (6 + \varepsilon)w(T_{OPT_{CDS}})$. Then, using algorithm NWST for C , they can obtain a Steiner tree T with $w(T) \leq 2.5\rho w(T_{OPT})$. Clearly, $V(T)$ is a Connected Dominating Set of G and

$$\begin{aligned} w(V(T)) &= w(C) + w(T) \leq (6 + \varepsilon)w(T_{OPT_{CDS}}) + 2.5\rho w(T_{OPT}) \\ &\leq (9.875 + \varepsilon)w(T_{OPT_{CDS}}). \end{aligned}$$

2.2.2 PTAS for MCDS in Node-Weighted UDG

The same as the MCDS in UDG, they have the following question:

Open Problem 2 *Does the minimum-weight Dominating Set have a PTAS in unit disk graphs?*

Nieberg et al. [64] developed a technique, called the *local neighborhood-based scheme technique*. A PTAS for minimum Dominating Set in polynomial growth bounded graphs is constructed. Because the unit disk graph is also a polynomial growth-bounded graph, Wang et al. [80] tried to use the local neighborhood-based scheme technique to find a PTAS in node-weighted UDG. They are successful in some special case. However, it is still open in general.

Because the running time for all approximations constructed with partition techniques is very high, it is hard to implement for application in the real world. Therefore, the following open problem receives more attention:

Open Problem 3 *Can they construct a constant approximation for the minimum-weight (Connected) Dominating Set in unit disk graphs without using partition?*

2.3 Unit Ball Graph

In most cases, people studied the MCDS problem in homogeneous wireless networks. They also assumed that all nodes in the networks are on a two-dimensional space and used unit disk graph (UDG) to abstract the networks. However, sometimes, this assumption is very far from reality. Underwater sensor networks (USNs) is a good example. USN can be used for ocean sampling networks, undersea explorations, distributed tactical surveillance, disaster prevention, assisted navigation, ocean environmental monitoring, etc. In many applications of USNs, underwater sensor nodes are floating around water instead of staying on the bottom

of the ocean. In this case, the height of nodes becomes considerably important. Therefore, when identical sensors are deployed into a field which is not flat, the mathematical model would be the unit ball graph.

Definition 11 (Unit ball graph(UBG)) A graph is called a *unit ball graph* if all its nodes lie in the three-dimensional Euclidean space, and an edge (u, v) exists if and only if the distance between u and v is at most one.

In other words, if they let each node u associate a ball with center u and diameter one, called a *unit ball*, then edge (u, v) exists if and only if ball u and ball v intersect each other. Since MCDS problem in UDG is NP-hard, and UDG is a special case of UBG, MCDS problem in UBG is also NP-hard.

There are also two stages for constructing a Connected Dominating Set in unit ball graph. The same as in UDG, they construct a maximal independent set first. Therefore, they need to find an upper bound for the size of maximal independent set. To do this, they have to study the following sphere packing problems.

Call the ball at center u with radius one the *dominating ball* of node u . The following is the first packing problem that they would like to study:

How many independent nodes can lie in the dominating ball of a node?

This problem has a close relationship to Gregory-Newton Problem, a well-known packing problem as follows:

How many unit balls can touch a unit ball such that they do not touch each other?

Gregory-Newton Problem was proposed in 1694 and conjectured as 12. Hoppe in 1874 (see [86]) proved that the answer is 12. Actually, an icosahedron has 12 vertices and the distance from the center to a vertex is smaller than the edge length. Therefore, they have

Corollary 3 ([46]) *The maximum degree of a minimum spanning tree T in a UBG G is at most 12.*

Lemma 7 ([46]) *For any vertex u in a UBG G , the neighborhood $N_G(u)$ contains at most 12 independent vertices.*

Hence, they can simply obtain $|mis| \leq 11 \cdot opt + 1$ in unit ball graph by the argument similar to the proof for $|mis| \leq 4 \cdot opt + 1$ in unit disk graph. The result can be improved by study the following question:

Open Problem 4 *How many independent nodes can lie in the union of dominating balls of two nodes with distance at most one?*

There are only a few work that has been done for studying the approximation algorithms for minimum Connected Dominating Set problem in unit ball graph. Hansen and Schmutz studied the expected size of a CDS in a random UBG [44]. In 2007, Butenko et al. [8] presented a distributed algorithm for MCDS in UBG

and proved that the performance ratio of their algorithm is 22 which means there are at most 22 independent nodes that can lie in the union of dominating balls of two nodes with distance at most one. Zou et al. [91] introduced a centralized heuristic algorithm whose performance ratio is $13 + \ln 10 \approx 15.302$. Zhong et al. [89] proposed a distributed algorithm and claimed its approximation ratio to be 16. However, when Kim et al. [53] did the same work later, they found that Zhong et al.'s algorithm is flawed and the performance analysis is incorrect. Kim et al. [53] studied how to construct quality CDS in UBG in distributed environments and showed the performance ratio is 14.937 which is the current best result.

For the second stage of connecting a maximal independent set into a Connected Dominating Set, the minimum spanning tree [8] and greedy algorithms [93] can also be used. They are all simple extensions from the Euclidean plane to the three-dimensional Euclidean space.

2.3.1 Currently Best Result

Kim et al. proposed the current best result in 2010. The contributions they made can be summarized as follows:

1. They first figure out how many independent nodes can be contained in the union of two adjacent unit balls.

Lemma 8 (Kim et al. [53]) *The number of independent nodes in $(B_1 \cup B_2) \setminus (B_1 \cap B_2)$ is at most 20.*

Lemma 9 (Kim et al. [53]) *The number of independent nodes in the union of two adjacent unit balls is at most 22.*

By combining their new result with some existing one, they reduced the upper bound for the size of a maximal independent set in a UBG from $11OPT_{CDS} + 1$ in [8] to $10.917 OPT_{CDS} + 1.083$, where OPT_{CDS} is the size of an optimal CDS of the UBG.

Theorem 4 (Kim et al. [53]) *Let M be an MIS of a UBG G . Then $|M| \leq 10.917 OPT_{CDS} + 1.083$, where OPT_{CDS} is the number of vertices in an optimal CDS of G .*

2. Different from others, they first presented a centralized algorithm C-CDS-UBG. The reason they introduced C-CDS-UBG first is that it is both easier to show its performance ratio than for the distributed algorithm and simpler to understand its behavior. C-CDS-UBG has two stages:
 - (a) Compute an MIS.
 - (b) Connect MIS into CDS by using greedy strategy.
- They showed that the performance ratio of C-CDS-UBG is 14.937.
- Here are some notations in [Algorithm 3](#) :
- $N(u) = \{v | v \in V(G) \setminus \{u\}\}$.
 - $N[x] = N(x) \cup \{x\}$.

Algorithm 3 $C - CDS - UBG(G(V, E))$ (Kim et al. [53])

```

1: Set  $M_0 = \emptyset$ ,  $W = \emptyset$ ,  $V' = V$ 
2: Pick a root  $r \in V'$  with a maximum degree
3: Set  $M_0 = r$ ,  $W = N(r)$ , and  $V' = V' \setminus (r \cup N(r))$ .
4: while  $V' \neq \emptyset$  do
5:   Pick  $x \in N(W)$  such that  $|N(x) \cap V'|$  is maximized.
6:   Set  $M_0 = M_0 \cup x$ ,  $W = W \cup (N(x) \cap V')$ , and  $V' = V' \setminus (x \cup N(x))$ .
7: end while
8: Set  $C = r$  and  $M = M_0 - r$ .
9: while  $M \neq \emptyset$  do
10:   Pick a node  $v \in N(C)$  such that  $|M_{v,C}| = \max |M_{x,C}| \mid x \in N(C)$ 
11:   Set  $C = C \cup v \cup M_{v,C}$  and  $M = M \setminus M_{v,C}$ 
12: end while
13: return  $C$ 

```

- For a node set C , $N(C) = (\cup_{x \in C} N(x)) \setminus C$.
- $M_{v,C}$ is a set of MIS nodes adjacent with v and not in C .

Lemma 10 (Kim et al. [53]) After Algorithm 3 is finished, $|C \setminus M_0| \leq 4.02|OPT_{CDS}|$, where OPT_{CDS} is an optimal CDS of G .

Theorem 5 (Kim et al. [53]) The size of the node set C generated by Algorithm 3 is no more than $14.937|OPT_{CDS}| + 1.083$, where OPT_{CDS} is an optimal CDS of G .

Proof (Kim et al. [53]) By Theorem 4, $|M_0| \leq 10.917|OPT_{CDS}| + 1.083$, and by Lemma 10, Algorithm 3 adds at most $4.02|OPT_{CDS}|$ to connect the nodes in M_0 . Therefore,

$$\begin{aligned}
|C| &= \sum_{i=1}^t \sum_{x \in M_0 \cap S_i} w(x) + |M_0| \leq 4.02t + 10.917|OPT_{CDS}| + 1.083 \\
&\leq 14.937|OPT_{CDS}| + 1.083.
\end{aligned}$$

3. They introduce a distributed algorithm D-CDS-UBG which is referred from C-CDS-UBG. Therefore, the performance ratio is also 14.937.

2.3.2 PTAS in UBG

Not every technique used in the Euclidean plane can be used in the three-dimensional Euclidean space. For example, the technique for constructing PTAS for MCDS in UDG [14] does not work in three-dimensional Euclidean space, because Cheng et al.'s [14] proof depends on a geometrical property which holds in the plane but is not true in the space. Zhang et al. [87] discovered a new technique to overcome this difficulty and showed that in any Euclidean space, the minimum Connected Dominating Set in unit ball graph has PTAS. They introduced a new

Algorithm 4 PTAS for UBG(Zhang et al. [87])

Input: The geometric representation of a connected unit ball graph G and a positive real number $\varepsilon < 1$.

Output: A Connected Dominating Set D of G .

- 1: Let $m = \lceil 300\rho/\varepsilon \rceil$ where ρ is the performance ratio of a constant approximation for MCDS in UBG.
- 2: Use the ρ -approximation algorithm to compute a Connected Dominating Set D_0 of G . For each $a \in 0, 1, \dots, m - 1$, denote by $D_0(a)$ the set of vertices of D_0 lying in the boundary region of $P(a)$. Choose a^* with the minimum $|D_0(a)|$.
- 3: For each cell e of $P(a^*)$, denote by G_e the subgraph of G induced by the vertices in the central region C_e . Compute a minimum subset D_e of vertices in e , such that

$$\text{for each component } H \text{ of } G_e, G[D_e] \text{ has a connected component dominating } H. \quad (4)$$

- 4: Let $D = D_0(a^*) \cup \bigcup_{e \in P(a^*)} D_e$.
-

construction, which gave not only a PTAS for the minimum Connected Dominating Set in unit ball graph but also improved running time of PTAS for unit disk graph. In fact, their method can be used to compute CDS for any n-dimensional unit ball graph. In this section, they will introduce their main results.

Lemma 11 (Zhang et al. [87]) *The output D of the Algorithm 4 is a CDS of G*

Lemma 12 (Zhang et al. [87]) *The Algorithm 4 runs in time $n^{O(1/\varepsilon^2)}$*

Theorem 6 (Zhang et al. [87]) *The Algorithm 4 is a $(1 + \varepsilon)$ -approximation for CDS in UBG.*

The proof of Algorithm 4 is in [87].

3 Connected Dominating Set Under Constraints

Wireless network is always a hot topic in research community because of its wide military and civil use. Different from wired networks, there is no physical infrastructure in wireless networks. On-demand routing and table-driven protocols cannot be used in wireless networks [75]. Hence, the idea of virtual backbone [15] in wireless network is well acknowledged and widely studied. Virtual backbone can save searching time for routing and reduce routing table size.

A *Connected Dominating Set* (CDS) is regarded as a good option acting as a virtual backbone. CDS can also have applications in routing, broadcasting, coverage, etc.

CDS's construction will have a close relation to the performance of wireless networks. If the size of CDS is too large, then the cost of CDS-based broadcasting will increase since more nodes will be involved to help forward in the networks. Then more redundancy and interferences will be generated in the network too. Hence, a wireless network prefer small CDS. However, broadcasting is not the only operation in wireless network. Routing also happens very frequently in the network. If they reduce CDS size too much, routing cost will increase. In Fig. 3, nodes 4, 5, and 6 construct a minimum CDS. The hop distance between 1 and 3 is 2 and the shortest path between 1 and 3 is $p_{shortest} = \{1 - 2 - 3\}$. However, the routing path between 1 and 3 going through the CDS becomes $P^S = \{1 - 4 - 5 - 6 - 3\}$ of hop distance 4, twice of that of shortest path. Thus, when they construct a CDS, routing operation should also be taken into consideration and the size of the CDS is not the only criteria used to evaluate the performance of a CDS.

In [62], Mohammed et al. proposed the concept of *diameter* to evaluate the performance of a CDS. *Diameter* is defined as the length of the longest-shortest path of the graph, where shortest path between any pair of nodes is the path having the smallest hop count among all paths between the pair of nodes. Later, Kim et al. proposed a definition of Average Backbone Path Length (ABPL) which is used to evaluate the average routing path length in a graph in [51]. If the *diameter* and ABPL of the subgraph induced by a CDS is small, then the CDS is regarded as an efficient CDS. However, both [62] and [51] cannot provide guaranteed routing cost. In [23], MOC-CDS was proposed,

How to choose a CDS will determine backbone-based routing protocols' performance in wireless networks. If the size of the CDS is too large, it is difficult to maintain it and searching time and routing table size cannot be reduced significantly. If the size of the CDS is too small, some characteristics in original networks may not be preserved. For instance, in routing protocols, the length of routing path is an important factor. The smaller the length is, the fewer nodes will be involved in routing process. The benefit is that delivery delay, energy cost, and interference will be reduced since fewer nodes will participate in forwarding packets. In [62], Mohammed et al. also pointed out that in wireless networks, the probability of message transmission failure often increases when a packet is sent through a longer path. Therefore, when designing a routing protocol, they should take path length into consideration, which can be viewed as routing cost. The shorter the routing path is, the less the routing cost will be. That is why shortest path is applied to many routing protocols. However, most current CDS-based routing protocols only focus on how to reduce the size of CDS while sacrificing shortest path properties in original networks. For example, in Fig. 3a, the shortest path between A and C is $\{1, 2, 3\}$ with length of 2. However, through the chosen minimum CDS, the routing path between A and C will become $\{1, 4, 5, 6, 3\}$, with length twice as the original shortest path.

To achieve efficient CDS-based routing and broadcasting at the same time, in [22], the author studied a CDS problem – *Shortest Path Connected Dominating*

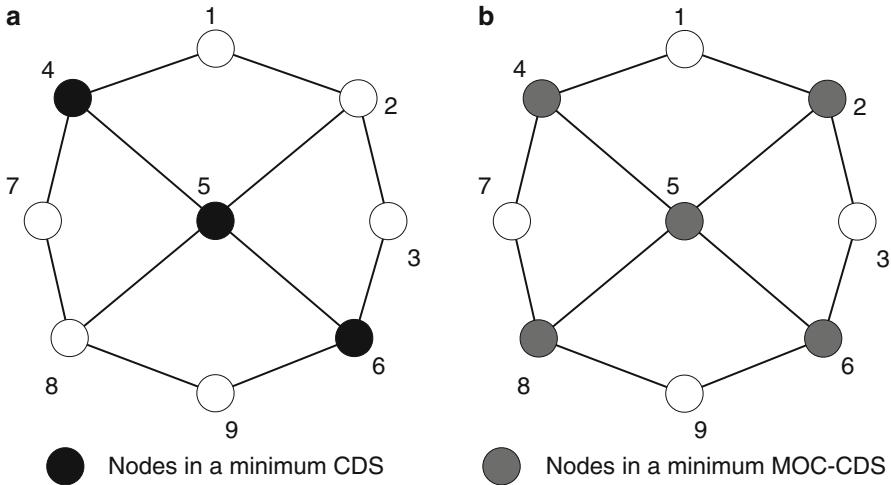


Fig. 3 Illustration of regular CDS and MOC-CDS. (a) A minimum regular CDS. (b) A minimum MOC-CDS

Set (SPCDS), which is a minimum node set including all intermediate nodes in every shortest path between any two nodes in the network. Due to this constraint, the length of path between any two nodes will not be increased even if the path is constructed through SPCDS. In [23], they study a special CDS problem – *Minimum rOuting Cost Connected Dominating Set* (MOC-CDS). First of all MOC-CDS is also a CDS. In addition, MOC-CDS has a constraint that between any pair of nodes, there exists at least one shortest path in the network, all of whose intermediate nodes must be included in MOC-CDS. Thus, packages can be delivered through MOC-CDS with the same routing cost as that in the original network. For example, in Fig. 3b, according to the constraints of MOC-CDS, node (2, 4, 5, 6, 8) are selected forming a minimum MOC-CDS. Hence, the shortest path between 1 and 3 will still be {1, 2, 3} with same length of 2 as that in the original network. However, the size of CDS will increase a lot when they put shortest constraint on CDS. To solve this problem, in [24], the shortest constraint is relaxed. The problem named as α *Minimum rOuting Cost Connected Dominating Set* (α -MOC-CDS) is studied. Besides all properties of regular CDS, α -MOC-CDS also has a special constraint – for any pair of nodes, there is at least one path all intermediate nodes on which belong to α -MOC-CDS, and the number of the intermediate nodes is smaller than α times of that on the shortest path in the original network. α -MOC-CDS is also studied in [32]. To save the energy in the unnecessary directions, directional antennas are used in [25] and MOC-DVB is studied.

In Sect. 3.1, they will recall the network models first. The details of the construction of the CDS under constraints will be introduced in Sect. 3.2.

3.1 Network Model

The network without directional antennas is modeled as a general graph when obstacles [65] exist or transmission radius are different in one network. If no obstacles exist and transmission ranges are same in one network, the network is modeled as a unit disk graph. When the directional antennas are used, the network is modeled as a directional graph where obstacles exist and transmission ranges are same.

3.1.1 General Graph

Two nodes can communicate when they satisfy two requirement.

The first one is transmission range constraint. In wireless networks, nodes may be from different providers. In one network, nodes may have different transmission ranges. In general graph model, two nodes can communicate with each other when they are in each other's transmission ranges. Two nodes cannot communicate when one is in the other one's transmission range but the other one is not in the "one's" transmission range.

The second one is non-obstacle constraint. Communications in wireless network are based on radio wave transmissions which are easily prevented by obstacles (like buildings, mountains). If two nodes satisfy the first constraint and there is no obstacle between them, then the two nodes can communicate. Otherwise, they cannot communicate even though they are physically close to each other enough.

In sum, two nodes can communicate when and only when they are close to each other and no obstacles exist between them. A general graph is denoted as $G = (V, E)$. V is a node set in the network. For each pair of nodes in the network, if there is a directed link between them in the network, then there exists an edge in E .

3.1.2 Unit Disk Graph

In unit disk graph (UDG) networks, it is assumed that all nodes in one network have the same transmission range and no obstacles are in the network. Any two nodes can communicate when and only when they satisfy the first constraint mentioned in Sect. 3.1.1. A UDG is denoted as $G = (V, E)$ where V is a node set and E is an edge set.

3.1.3 Directed Graph

In directed graph network, directional antennas are used. And you can choose the directions to forward messages through switching on antennas. In this kind of network, it is assumed that all nodes have the same transmission range and obstacles are there. There are two kinds of reception models – directional reception and omnidirectional reception. In their work, omnidirectional reception is used. Uniform and nonoverlapping directional antennas are used.

Directional antennas will divide one node's transmission range into uniform sectors. In Fig. 4, four uniform directional antennas are deployed on each node.

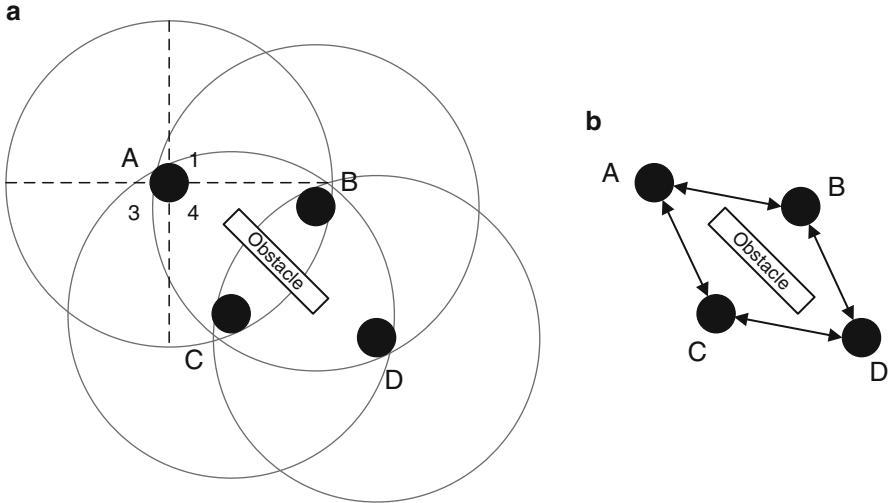


Fig. 4 Directed graph

Hence, each node's transmission range is divided to four sectors. Sector i of node u is denoted as u^i . In Fig. 4, the first sector of A is denoted as A^1 . Nodes B and C are in the fourth direction of A (A^4). If A^4 is switched off, then B and C cannot receive any messages from A , even though they are in A 's transmission range.

Thus, in directed graph network, if one node v wants to receive messages from node u , they must first satisfy two requirement in Sect. 3.1.1, and they also need to satisfy that u 's sector where v is switched on. Compared to the prior two models with node set and edge set, a directed graph has a direction set (sector set). Thus, one directed graph is denoted as $G = (V, E, S)$, where V is a node set, E is an edge set, and S is a sector set. In Fig. 4, $V = \{A, B, C, D\}$, $E = \{A^4 \rightarrow B, A^4 \rightarrow C, B^4 \rightarrow D, B^2 \rightarrow A, C^2 \rightarrow A, C^4 \rightarrow D, D^2 \rightarrow B, D^2 \rightarrow C\}$, $S = \{A^1, A^2, A^3, A^4, B^1, B^2, B^3, B^4, C^1, C^2, C^3, C^4, D^1, D^2, D^3, D^4\}$.

The three models are used in the following algorithms in Sect. 3.2. In every model, the length of a path is equal to the hop count of the path. And the distance of two nodes is the hop distance which is equal to the hop count on the shortest path between the two nodes.

3.2 Construction of CDS Under Constraints

To construct a CDS with guaranteed routing cost, many researches are done on this topic. In [82], Wu et al. proposed an algorithm to construct a CDS. The first step in [82] is to find a CDS including all intermediate nodes of all shortest path. Actually, this CDS in the first step can solve in polynomial time, which is studied in [22].

3.2.1 An Exact Algorithm to Construct SPCDS

SPCDS is a CDS. SPCDS also has another constraint that for each pair of nodes, the intermediate nodes on all shortest path should be in the SPCDS. To solve SPCDS, an equivalent problem 2PCDS is proposed in [22]. 2PCDS is a CDS firstly. 2PCDS also requires that all intermediate nodes on shortest path of distance 2. It is proved in [22] that SPCDS is equivalent to 2PCDS.

Lemma 13 *SPCDS and 2PCDS is equivalent to each other.*

Proof The first part is to prove that an SPCDS is a 2PCDS. In SPCDS, all shortest paths of any length will be considered. Thus, SPCDS also includes the shortest path of length 2. Thus, an SPCDS is also a 2PCDS.

The second part is to prove that a 2PCDS is also an SPCDS. Suppose there is a 2PCDS. For any pair of nodes u, v of distance bigger than or equal to 2, suppose that there is a shortest path between u and v denoted as $p_{u,v} = \{u, w_1, w_2, \dots, w_k, v\}$. From $p_{u,v}$, they know that the distance between u and w_2 should be of distance 2, then they can find a node w'_1 from the 2PCDS to replace w_1 . The distance between w'_1 and w_3 is 2, then they can find a node w'_2 to replace w_2 . Finally, they can find a replacement $p'_{u,v} = \{u, w'_1, w'_2, \dots, w'_k, v\}$, where all intermediate nodes are in 2PCDS. Thus, a 2PCDS is also a SPCDS. \square

The algorithm proposed in [22] is based on 2hop local information. The idea of algorithm in [22] is that for one node, if there exists a pair of nodes of distance 2, then the node will be added to 2PCDS. The check time is $O(\delta^2) \times O(n) = O(\delta^2 n)$, where δ is the maximum node degree in the network and n is the number of nodes in the network. Finally, a SPCDS is also constructed optimally in polynomial time.

3.2.2 MOC-CDS Problem

The size of SPCDS is relatively large, so in [23], MOC-CDS is proposed. Different from SPCDS, MOC-CDS does not require all nodes on the shortest path should be selected. MOC-CDS requires that for any pair of nodes, there exists at least one shortest path on all intermediate nodes that are in MOC-CDS. Thus, the size of MOC-CDS will be reduced greatly compared to that of SPCDS. Similar to SPCDS, another problem 2hop-CDS is proposed in [23] too. Referring to the proof of equivalence in [22], the equivalence between 2hop-CDS and MOC-CDS is proved in [23]. It is proved that MOC-CDS or 2hop-CDS is NP-hard in a general graph. The proof of NP-hard is based on the reduction of 2hop-CDS from set cover [37] problem.

Definition 12 (Set cover) Given a collection \mathcal{M} of subsets of a finite set X such that $\cup_{B \in \mathcal{M}} B = X$, find a minimum subcollection $\mathcal{B} \subseteq \mathcal{M}$ such that $\cup_{B \in \mathcal{B}} B = X$.

Decision version of 2hop-CDS: Given a positive integer k and a graph G , determine whether there is a 2hop-CDS of size at most k in G .

Decision version of set cover: Given a collection \mathcal{M} of subsets of a finite set X and a positive number h , determine whether there is a set cover \mathcal{M}' of at most h subsets in \mathcal{M} .

Theorem 7 *A minimum 2hop-CDS is NP-hard in a general graph.*

Proof Create a node a_B , $\forall B \in \mathcal{M}$. Create a node b_x , $\forall x \in X$. Two additional nodes will be created c and d . c and d are connected to all a_B 's, $\forall B \in \mathcal{M}$. Meanwhile d is also connected to all b_x , $\forall x \in X$. Edge between b_x and a_B represents that x is in B .

It is claimed that \mathcal{M} has a set cover \mathcal{B} of size at most k when and only when G has a 2hop-CDS of size at most $k + 1$. There are two parts of the proof.

The first one is to prove that a 2hop-CDS of size $k + 1$ can be constructed when $|\mathcal{B}| \leq k$. If there is a set cover \mathcal{B} of size at most k , then the corresponding nodes $\{d\} \cup \{a_B | B \in \mathcal{B}\}$ in G construct a 2hop-CDS of size at most $k + 1$.

c can be dominated by nodes in $\{a_B | B \in \mathcal{B}\}$. $\forall z \in \{b_x | x \in X\} \cup \{u_B | B \in \mathcal{B}\}$, z can be dominated by d . The node subset satisfies 2hop-CDS's characteristic of dominating.

Since d is connected to any other node in the node set, the node set is connected.

For any two nodes $e, f \in G$ of distance 2, $\{e, d, f\}$ must be a shortest path. For any two nodes c and $e \in G$ of distance 2, it is true that $e \in \{b_x | x \in X\} \cup \{d\}$. There should be a path $\{e, a_B, c\} \in G$ – a shortest path between e and c .

Therefore, the construction of the node set should be a 2hop-CDS of size at most $k + 1$.

The second part is to prove that if there is a 2hop-CDS of size at most $k + 1$, set cover has a solution of size at most k . The distance between c and any node e in $\{b_x | x \in X\}$ is 2. Any shortest path between c and e must pass a node a . Thus, the set cover is $\mathcal{B} = \{B | a_B \in \text{2hop-CDS and } B \in \mathcal{M}\}$. For any two different element $U, V \in \mathcal{M}$ of distance 2 in G , each shortest path must pass one node corresponding to element in \mathcal{M} . Thus, $|\mathcal{M}| \leq k$.

In summary, a minimum 2hop-CDS is NP-hard. □

Based on the above proof, it is trivial to conclude that a minimum MOC-CDS is NP-hard as well. Set cover does not have a solution of performance ratio of $\rho ln \gamma$, where γ is the maximum cardinality of element in \mathcal{M} and ρ is a nonnegative number smaller than 1, unless $NP \subseteq DTIME(n^{O(\log \log n)})$. Thus, there cannot be a solution to MOC-CDS with performance ratio of $\rho ln \delta$, where δ is the maximum node degree.

To construct a MOC-CDS, the distributed algorithm named FlagContest is proposed. For a given graph $G = (V, E)$, each node v is assigned a unique id, denoted as $id(v)$. id aims to solve the situation that more than one node can be selected at one time. The solution is to select the one having highest id . At the very beginning, each node will collect the pairs of neighbor nodes of distance 2. Each node having nonempty pair set will compute the times the node is acting as an intermediate node by counting the number of pairs in its pair set. Send the calculated

times to its neighbors. When one node receives times, it will choose one with the biggest times and send a flag to the selected one. If one node collect flags from all its neighbors, turn to black and send out its pair set 2 hops away. Nodes receiving the pair set will remove those pairs in the received pair sets from their own pair sets. Update the pair sets and recalculate the times acting as an intermediate node. The algorithm will stop when all nodes' pair sets are empty. The black nodes colored during the process will construct a 2hop-CDS.

The algorithm proposed in [23] is proved correct. Meanwhile, the performance ratio is proved – FlagContest produces approximation solution with performance ratio $H(\binom{\delta}{2})$, where H is the harmonic function and δ is the maximum node degree of input graph.

3.2.3 CDS Under Relaxed Constraints

The above CDS under shortest path constraint does achieve efficient routing; however, from their simulation results, the size of CDS will increase a lot compared to other CDS size. To reduce CDS size, [24] relaxed the shortest path constraint and proposed α Minimum rOuting Cost Connected Dominating Set (α -MOC-CDS). In α -MOC-CDS, the shortest path constraint is relaxed to that for any two nodes, there exists at least one path, and the number of intermediate nodes on this path is smaller than α times that of on the shortest path. Meanwhile, all intermediate nodes on this path must be in α -MOC-CDS. An equivalent problem α -2hop-CDS is proposed in [24] and the equivalence is proved too. Actually, MOC-CDS is a special case of α -MOC-CDS, when $\alpha = 1$. In [24], Ling et al. proposed a distributed localized algorithm to construct α -MOC-CDS which is introduced as follows.

The algorithm is based on the $\alpha + 1$ local topology information. The topology of the local information on each node v is denoted as $G(v) = (V_v, E_v)$. A weighted graph $G_w(v) = (V_w(v), E_w(v), W_w(v))$ can be induced by $G(v)$, where $V_w(v) = V(v)$, $E_w(v) = E(v)$, and $W_w(v)$ record the weight between any two nodes. For any two nodes, if there is an edge between them in $E(v)$, then the weight is 0. Otherwise, it is ∞ . For each node v , it will record a pair set which stores the pairs where distance of the two nodes in each pair is 2 in $G(v)$ and $G_w(v)$, there is two edges between v and the two nodes, and the total weight of the two edges should be smaller than α . Send out the pair set to its neighbors in $G(v)$. If one node receives all pair sets from its neighbors, it will order them based on their id and cardinality of the pair set. One node has higher rank when its cardinality is bigger. When two nodes have the same cardinality, the node that has higher *id* will have higher rank. After ordering all neighbors, one node will send flags to those nodes whose pair set has no intersection with any pair set of any node having higher rank. If one node receives all flags from its neighbors, it will turn black, send out its own pair set two hops away, and update its weighted graph. If node v turns black, then for any pair of nodes of distance 2 in $G_w(v)$ and there is a path in $G_w(v)$ of weight w , then edge between the two nodes in the pair should be added to $G_w(v)$ of weight $1 + w$. When one node receives the pair set, it will remove the pairs in the received pair set from its own pair set. The algorithm will stop when all pair sets are empty.

From the simulation results in [24], the size of CDS will reduce greatly, while the routing cost will not increase a lot when $\alpha = 3$. Thus, there is a tradeoff between CDS size and routing cost.

3.2.4 Du's Relaxation

In [32], Du et al. proposed a CDS with guaranteed routing cost named as GOC-MCDS. GOC-MCDS is also a α -MOC-CDS. In [24], Ling et al. propose a distributed algorithm; however, no performance ratio is given. In [32], one centralized and one distributed algorithms are proposed with performance ratio.

In the following two parts, the two algorithms will be recalled in details.

1. *GOC-MCDS-C* The centralized algorithm follows the steps of regular MCDS algorithms. The algorithm has two stages. In the first stage, a maximum independent set (MIS) is constructed. For any pair of MIS nodes of distance smaller than 4, find a shortest path and add all intermediate nodes on the path to the CDS. Finally, the selected MIS combined with the added nodes will construct a GOC-MCDS. The constructed GOC-MCDS will guarantee that for any pair of nodes, the length of the routing path through GOC-MCDS will be smaller five times that of the shortest path between the two nodes. And the size of the GOC-MCDS will be smaller than $|C| + |I| \leq 121 * |I| \leq 443\frac{2}{3} * opt_{GOC-MCDS} + 201\frac{2}{3}$, where $|C|$ is the number of added nodes in the second stage, $|I|$ is the size of the MIS in the first stage, and $opt_{GOC-MCDS}$ is the size of the optimal solution to GOC-CDS.
2. *GOC-MCDS-D* The distributed algorithm also has two stages. At the very beginning, the color of all nodes is white. During the algorithm, some nodes will keep white, some will change to black, and some will change to gray. All the black ones will construct a GOC-MCDS. Meanwhile, each node will be assigned a unique id . The algorithm will be introduced as follows.

The first stage is to find an MIS too. Each white node will send its own id to its neighbors. When one node receives all id 's from its neighbors and its own id is smaller than all id 's it receives, then the node will change to black and send the message “black” to its neighbors. Once a white node receives the “black,” turn to gray. When the algorithm stops, there are only black or gray nodes in the graph.

In the second stage, more nodes are added combining with black nodes in the first stage to form a GOC-MCDS. Every black node s will send its own $id(s)$ to its neighbors. When one node t receives $id(s)$, t will put its own $id(t)$ together with $id(s)$. t will send the pair $(id(s), id(t))$ to all its neighbors. For each node w with $id(w)$, when w receives a pair $(id(s), id(t))$ having $id(s) < id(t)$, w will send the tuple $(id(t), id(w), id(s))$ to the neighbor with $id(s)$. When w receives $(id(s), id(t))$ and $(id(s^*), id(t^*))$ and $id(s) < id(s^*)$, w will send a tuple $(id(s^*), id(t^*), id(w), id(s), id(t))$ to the neighbor with $id(t)$. When w receives $(id(s), id(t))$ and $id(s^*)$ and $id(s) < id(s^*)$, w will send a tuple $(id(s^*), id(w), id(t), id(s))$ to the neighbor with $id(t)$. Otherwise, w will send tuple $(id(s), id(t), id(w), id(s^*))$ to the neighbor with $id(s^*)$. When the node t receives a tuple $(id(s^*), id(t^*), id(w), id(s), id(t))$ or

$(id(s^*), id(w), id(t), id(s))$, t will send the tuple to its neighbors with $id(s)$. Each black node of $id(s)$ will have a tuple set \mathcal{M} storing tuples $(id(w_1), id(t), id(s))$, $(id(w_2), id(w_1), id(t), id(s))$, and $(id(w_3), id(w_2), id(w_1), id(t), id(s))$. Perform the policy of “computation on \mathcal{M} ” on each node. When a node $id(x)$ receives a tuple $(\dots, id(x-1), id(x), \dots)$, x will turn to black and pass the tuple to the node $id(x-1)$.

Computation on \mathcal{M} : Pick up $(id(h), \dots, id(2), id(1)) \in \mathcal{M}$ and send the tuple to the node of $id(2)$. Delete all tuple from \mathcal{M} starting with $id(h)$.

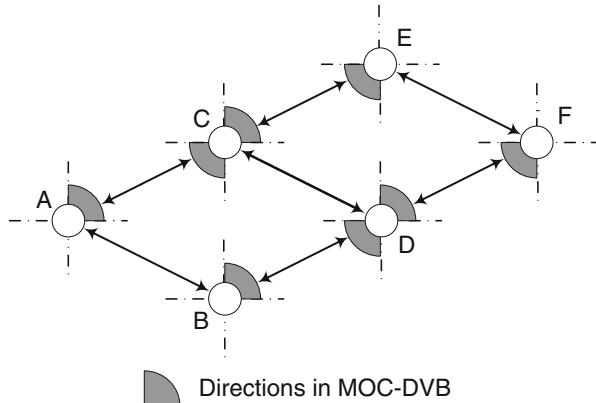
This distributed algorithm has the same performance ratio as the centralized one.

3.2.5 MOC-DVB

In wireless networks, saving energy is always the critical topic. In previous literature, directional antennas are used to help save energy in many operations, like broadcasting [21]. Thus, Ling et al. apply directional antennas to the wireless networks. The wireless networks with directional antennas are modeled as directed graphs. In such a wireless, *Minimum rOuting Cost Directional VB* (MOC-DVB) is studied in [25]. Fewer sectors selection will help save energy and reduce interference. MOC-DVB requires that if the source node initiates a message in omni-direction, then the message should be delivered by a shortest path on all intermediate directions that are in MOC-DVB. A MOC-DVB itself may be disconnected. For example, in Fig. 5, the gray sectors construct a minimum MOC-DVB. However, the MOC-DVB cannot induce a connected graph. Even though a connected subgraph of MOC-DVB, any messages can be successfully routed or broadcasted. If A has a message with destination F , the directional path should be $\{A \rightarrow B^1 \rightarrow D^1 \rightarrow F\}$ which means that A initiate a message in omnidirection, B use omni-reception and forward it to D through the first direction, D receive the message from B then forward it through the first sector of D , and lastly, D forward the message to F through the first direction of D . Meanwhile, it can go back from F to A through path $\{F \rightarrow E^3 \rightarrow C^3 \rightarrow A\}$. Originally, the directed graph is strongly connected.

In [25], Ling et al. prove that MOC-DVB is NP-hard. A distributed and greedy algorithm is proposed in [25]. For each direction u^i , a direction id is assigned, denoted as $d(u^i)$. For each direction w^i , it will compute a pair set which stores that the pair (u, v) of distance 2, u 's message can be delivered to v through direction w^i . Each direction will send out its times acting as an intermediate direction to its neighbors. When one node receives the times from all its neighbors, it will select the direction with the maximum times and send out a flag to the direction. When one direction collects flags from all the directions where it is, the direction will be added to MOC-DVB. The selected direction's pair set will be sent out 2 hops away. When one direction receives the pair set, it will remove the pair in the received pair set from its own pair set. The algorithm stops when all pair sets are empty.

Fig. 5 Comparison between CDS and MOC-DVB



3.2.6 PTAS for MCDS with Routing Cost Constraint [31]

Du et al. in [31] showed that $\alpha MOC - CDS$ has a PTAS for $\alpha \geq 5$. However, the problem is still open for $1 \leq \alpha < 5$. Actually, the main difficulty for $1 \leq \alpha < 5$ is how to connect a Dominating Set into a feasible solution for $\alpha MOC - CDS$. So far, there is no a good method to do that without increasing too much number of nodes. They will show some of their main results.

First, they use a square $[0, q] \times [0, q]$ to cover the whole graph $G = (V, E)$, then construct the other square called $P(0)$. $P(0)$ is divided into p^2 cells. Each cell is an $a \times a$ square called e . The side length for each cell a is equal to $2(\alpha + 2)k$, where $p = 1 + \lceil q/a \rceil$ and k is a positive integer. Note that all cells are disjoint, so they should ensure each cell only has left and lower boundary. The union of all cells covers the square $[0, q] \times [0, q]$.

For each cell e , construct two squares with the same center as e . The side length are $(a + 4) \times (a + 4)$ and $(a + 2\alpha + 4) \times (a + 2\alpha + 4)$. They use e^c to represent the *central area* which is the closed area bounded by the first square and e^b to represent the *boundary area* which is the area between the second square (exclude boundary) and cell e (include boundary). e^{cb} is the open area bounded by the second square which is $e^c \cup e^b$.

Now, for each cell e , they study the following problem:

- [Du et al. in [31]] LOCAL(e): Find the minimum subset D of nodes in $V \cap e^{cb}$ such that
1. D dominates all nodes in $V \cap e^c$.
 2. $m_D(u, v) \leq \alpha$ for any two nodes $u, v \in V \cap e^c$ with $m(u, v) = 1$ and $\{u, v\} \cap e \neq \emptyset$.

Let D_e denote the minimum solution for the LOCAL(e) problem. Define $D(0) = \cup_{e \in P(0)} D_e$ where $e \in P(0)$ means that e is over all cells in partition $P(0)$.

Lemma 14 (Du et al. in [31]) D_0 is a feasible solution of $\alpha MOC - CDS$ and D_0 can be computed in time $n^{O(\alpha^4)}$ where $n = |V|$.

Lemma 15 (Du et al. in [31]) $|D(0)| + |D(1)| + \dots + |D(k-1)| \leq (k+8)|D^*|$.

Algorithm 5 Algorithm PTAS(Du et al. in [31])

- 1: Compute $D(0), D(1), \dots, D(k - 1)$;
 - 2: Choose $i^*, 0 \leq i^* \leq k - 1$ such that
 - 3: $|D(i^*)| = \min(|D(0)|, |D(1)|, \dots, |D(k - 1)|)$;
 - 4: Output $D(i^*)$.
-

Theorem 8 (Du et al. in [31]) *Algorithm PTAS (Algorithm 5) produces an approximation solution for α MOC-CDS with size*

$$|D(i^*)| \leq (1 + \varepsilon)|D^*|$$

and runs in time $n^{O(1/\varepsilon^4)}$.

4 Strongly Connected Domination Set

In many scenarios, the communications in multi-hop wireless networks are asymmetric in real world. For example, nodes may have different transmission due to several reasons such as energy concern and cost. Therefore, it is possible to consider that node u can communicate with node v , but v cannot communicate with u . In such case, they need to model this kind of network to a directed graph $G = (V, E)$. In directed graph, they usually consider a strongly Connected Dominating Set (SCDS) as a virtual backbone. The problem minimum SCDS seeks an SCDS of the smallest cardinality in a specified digraph. Given a directed graph $G = (V, E)$, S is strongly connected if for every pair u and $v \in S$, there exists a directed path from u to v in the directed graph induced by S , i.e., $G(S)$. Formally, the SCDS problem can be defined as follows:

Definition 13 (Strongly Connected Domination Set) Given a directed disk graph $G = (V, E)$, find a subset $S \subset V$ with minimum size, such that the subgraph induced by S , called $G(S)$, is strongly connected and S forms a Dominating Set in G .

The SCDS problem is easy to be proved as an NP-hard problem because the MCDS problem in UDG is NP-hard and UDG is a special case of DG. Du et al. [28] consider the SCDS in directed graphs. Differ from undirected graph, an MIS is not necessary a DS in a directed graph. Therefore, they cannot find an MIS and connect it into a SCDS. Instead, they have to find a DS in the first stage, then connect it to a SCDS. Based on this approach, they present two constant-approximation algorithms for computing a minimum SCDS in DG if the transmission range is bounded which are Connected Dominating Set using the Breath First Search tree

(CDS-BFS) and Connected Dominating Set using the minimum nodes Steiner tree (CDS-MNS) algorithms. The main differences of these two algorithm are the second stage for connecting the obtained Dominating Set to an SCDS. They assume that G is a strongly connected graph in order to guarantee that the graph G has a solution for the SCDS problem.

In 2009, Li et al. [55] proposed another approximation algorithm for SCDS. In this section, they will introduce some of their main results.

They used $G = (V, E)$ to represent a digraph. A node in G is said to be a *internal* node if its out-degree is not zero and an *sink* node otherwise. $I(G)$ is the set of internal nodes in G . If a subgraph's vertex set is exactly V , they said that the subgraph is spanning. They call a subgraph of G an *arborescence* rooted at a node $s \in V$ if the in-degree of s is zero, and the in-degree of any other node is exactly one in this subgraph. An s -arborescence is also an arborescence rooted at s . For each node $v \in V$, $\delta^+(v)/\delta^-(v)$ is the out/in-degree of v in G , and $N^+(v)/N^-(v)$ is the set of out/in-neighbors of v in G . For any subset U of V , they denote by $G[U]$ the subgraph of G induced by U . G^R is the reverse of G which is a digraph obtained from G by reversing the direction of every arc. In other words, $G^R = (V, E^R)$ with

$$E^R = \{(u, v) | (v, u) \in E\}.$$

Lemma 16 (Li et al. [55]) *Let $G = (V, E)$ be a strongly connected digraph and s be an arbitrary node in V . Suppose that T_1 is a spanning s -arborescence in G , and T_2 is a spanning s -arborescence in G^R . Then $I(T_1) \cup I(T_2)$ is an SCDS of G .*

Theorem 9 (Li et al. [55]) *Suppose that \mathbb{A} is a ρ -approximation for SAFIN. Then Algorithm $SCDS(\mathbb{A})$ produces an SCDS of size at most $2\rho \cdot opt - 2\rho + 1$, where opt is the size of a minimum SCDS.*

By Lemma 16, they introduce the Spanning Arborescence with Fewest Internal Nodes (SAFIN) problem:

[Spanning Arborescence with Fewest Internal Nodes (SAFIN) problem] Given a digraph $G = (V, E)$ and a source node $s \in V$, compute a spanning s -arborescence T with minimum $|I(T) \setminus \{s\}|$.

It is easy to argue that SAFIN is at least as hard as minimum CDS. Therefore, they use SAFIN to find the minimum SCDS. Let \mathbb{A} be an arbitrary polynomial-time approximation for SAFIN. The Algorithm 6 described a polynomial-time approximation algorithm $SCDS(\mathbb{A})$ for minimum SCDS with the parametric \mathbb{A} as a subroutine. To reduce the running time, they selected a small subset S of candidates for the source node in the beginning. Specifically, let u be the node u which minimize satisfying that $\min(\delta^-(v), \delta^+(v))$ over all nodes $v \in V$. They said that S consists of u and all its out-neighbors if $\delta^-(u) > \delta^+(u)$. Otherwise, S consists of u and all

Algorithm 6 *SCDS(\mathbb{A})* (Li et al. [55])

```

1:  $u \leftarrow \arg \min_{v \in V} \min(\delta^-(v), \delta^+(v));$ 
2: If  $\delta^-(u) \leq \delta^+(u)$  then
3:    $S \leftarrow u \cup N^-(u);$ 
4: else
5:    $S \leftarrow u \cup N^+(u);$ 
6:  $U \leftarrow V;$ 
7: for each  $s \in S$ ,
8:    $T_1 \leftarrow$  spanning  $s$ -arborescence in  $G$  output by  $\mathbb{A}$ ;
9:    $T_2 \leftarrow$  spanning  $s$ -arborescence in  $G^R$  output by  $\mathbb{A}$ ;
10:  if  $|I(T_1) \cup I(T_2)| < |U|$  then
11:     $U \leftarrow I(T_1) \cup I(T_2);$ 
12: output  $U$ 

```

Algorithm 7 (\mathbb{A})

```

1:  $B \leftarrow s;$ 
2: while  $h(B) > 0$ ,
3:   construct  $\tau(B)$ ;
4:   find a cheapest  $T \in \tau(B)$ ;
5:    $B \leftarrow B \cup I(T);$ 
6: output a spanning  $s$ -arborescence of  $G\langle B \rangle$ 

```

its in-neighbors. Clearly, there must be at least one node in the selected S in every SCDS.

Next, they will show their $(1.5H(n - 1) - 0.5)$ -approximation algorithm \mathbb{A} ([Algorithm 7](#)) for SAFIN where H is the harmonic function. For such \mathbb{A} , the algorithm $SCDS\mathbb{A}$ is a $(3H(n-1)-1)$ -approximation algorithm for minimum SCDS by [Theorem 9](#).

They introduce some notations and terminologies for algorithm \mathbb{A} ([Algorithm 7](#)). They assigned each node a unique ID. And s is the source node. $G\langle B \rangle$ is the spanning subgraph of G whose arc set consists of all arcs of G leaving from the nodes in B for any $B \subset V$ containing s . If a strongly connected component of $G\langle B \rangle$ neither contains the source s nor has an incoming arc, it is an orphan with respect to B . The node with the smallest ID in this component is its head for each orphan component. The number of heads (equivalently, the number of orphans) with respect to B is denoted as $h(B)$. Clearly, a spanning s -arborescence is in $G\langle B \rangle$ if and only if $h(B) = 0$. They use $\tau(B)$ to denote the set of candidates. The construction of $\tau(B)$ is described can be found in [55].

Theorem 10 (Li et al. [55]) *The approximation ratio of the [Algorithm 7](#) for SAFIN is at most $1.5H(n - 1) - 0.5$.*

5 Weakly Connected Domination Set

In ad hoc networks, topology changes very often, and most of the changes are localized which means it is in a small area of the network. However, they do not want this kind of change effect the whole networks. Therefore, they need a new network structure such that the local changes will not be seen by the entire network which is *cluster*. Like the virtual backbone in wireless networks, they would like to control less nodes. In *cluster*, such node is called *clusterhead*. One method used in *cluster* is based on domination in graphs which they already introduced in the previous section. Of course, they wish to find the smallest number of *clusterhead* which is a minimum Dominating Set. So they can simplify the network structure as much as possible. Unfortunately, it is a NP-complete problem to find a minimum size Dominating Set in a general graph. In order to solve this problem, they can ensure that the clusterheads are adjacent to each other or at least reasonably close to each other. Hence, they can try to use CDS. However, minimum CDS problem in general graph is also NP-complete. Although a CDS provides an obvious routing, the connectivity requirement causes a very large number of *clusterhead*. Based on this, a Weakly Connected Dominating Set is proposed.

Definition 14 (Weakly Connected Domination Set (WCDS)) The subgraph weakly induced by W is $S_w = (N[W], E \cap (W \times N[W]))$, where $N[W]$ is a set of one-hop neighbor of W and W . The set W is a *Weakly Connected Dominating Set* (WCDS) if S_w is connected and W is a DS.

According to the definition of WCDS, the size of WCDS is less or equal to the size of CDS. Thus, they feel that WCDS is a better method for clustering. An approximation algorithm with ratio $O(\ln \Delta)$ was proposed by Chen [11], where Δ is the maximum degree of the input network. Alzoubi et al. [4] present two distributed algorithms with the ratio 5 and 122.5 in 2003. Dubhashi [33] gave another approximation algorithm with the same ratio as Chen's in 2003. Du et al. [30] proved that:

Theorem 11 (Du et al. [30]) *The least upper bound for the ratio of sizes between MIS and the minimum WCDS for UDG is five.*

To prove **Theorem 11**, they first proved that five is an upper bound. They considered an MIS M and a minimum WCDS D . For any vertex u , the disk with radius one and center u is called the *dominating disk* of u . Suppose $u \in D$ and the dominating disk of u contains n elements which is M, a_1, \dots, a_n , and order them in clockwise. Connect ua_1, ua_2, \dots, ua_n (Fig. 2). Because a_1, a_2, \dots, a_n are independent set, they have $|a_1a_2| > 1, \dots, |a_na_1| > 1$. Hence, they can get $\angle a_1ua_2 > \frac{\pi}{3}, \dots, \angle a_nua_1 > \frac{\pi}{3}$. Therefore, $n \leq 5$. So five is an upper bound is proven.

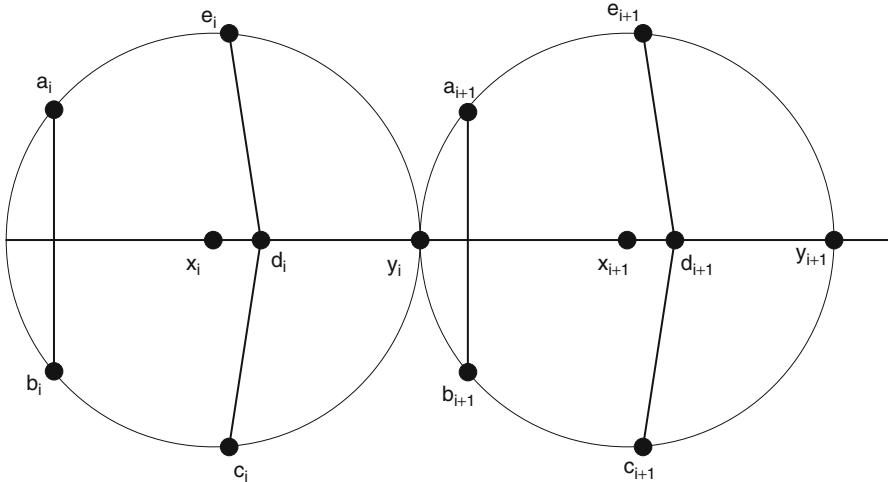


Fig. 6 An example for $|M| = 5|D|$

They constructed an example to show that five is the least upper bound. They showed that $|M| = 5|D|$ where M is an MIS and D is a minimum WCDS. They drawn the example as follows:

1. Construct a sequence of points $x_1, y_1, x_2, y_2, \dots, x_n, y_n$ on a line ℓ such that $|x_1y_1| = |y_1x_2| = |x_2y_2| = \dots = |x_ny_n| = 1$, as shown in (Fig. 6).
2. Construct four points a_i, b_i, c_i, e_i on the boundary of the dominating disk of each x_i such that segment $a_i b_i$ is perpendicular to line ℓ and $\angle e_i x_i a_i = \angle a_i x_i b_i = \angle b_i x_i c_i = \frac{\pi}{3} + \varepsilon$ where $\varepsilon > 0$ is a small constant.
3. Construct the fifth point d_i in dominating disk of x_i such that d_i is on line ℓ and at the middle between segments $a_i b_i$ and $a_{i+1} b_{i+1}$.

They study the UDG G with vertex set $\{a_i, b_i, c_i, d_i, e_i, x_i, y_i \mid i = 1, 2, \dots, n\}$. By calculating the distance of every pair of MIS, all the distance are larger than 1. Therefore, 5 is the least upper bound.

6 Fault-Tolerant Virtual Backbone in Wireless Network

Due to many factors such as mobility of nodes and limitation of energy supply, the topology of a wireless network is expected to be dynamic. However, most of the existing virtual backbone computation algorithms/protocols assume a static network topology. Naturally, even a slight topology change can make current virtual backbone structure obsolete. In such case, the backbone structure should be completely reconstructed. As a result, in a highly dynamic wireless network, those algorithms could work very inefficiently since they may need to reconstruct

virtual backbone very frequently. Motivated by this issue, the problem of computing fault-tolerant virtual backbone is introduced [18].

In graph theory, a graph is said to be *disconnected* only if there is a pair of nodes having no path between them in the graph. A graph is k -*connected* if the graph is still connected after removing any $k - 1$ nodes in the graph. They say a node dominates a set of nodes, if the node is adjacent to all of the nodes in the set in the graph. Similarly, a subset of the nodes in a graph dominates the other nodes outside the subset only if each of the nodes outside the subset has at least one neighbor in the subset. In addition, they say the subset m -dominates the nodes outside the subset only if each of the nodes outside the subset has at least m neighbors in the subset. Note that the CDS computation algorithms by Guha and Kuller and many conventional CDS computation algorithms are targeted to produce 1-connected 1-Dominating Set.

Let $G = (V, E)$ denote a connected graph representing a wireless network and C be a CDS of G , which can serve as a virtual backbone of the wireless network. In [79], the additional requirements for C to be a fault-tolerant virtual backbone is formally defined as follow:

- k -connectivity: C has to be k -connected so that the virtual backbone can survive after $k - 1$ backbone nodes fail.
- m -domination: each node $x \in (V \setminus C)$ has to adjacent to at least m nodes in C so that x can be connected even after $m - 1$ adjacent backbone nodes fail.

Here, k and m determine the level of fault tolerance. By enforcing the two new requirements on C , any pair of nodes can communicate with each other via the virtual backbone even after any $\min\{k, m\}$ nodes fails. The rest of this chapter is devoted to the survey of the recent advances in computing fault-tolerant CDS in wireless network.

6.1 Preliminaries

They first introduce several notations and definitions to simplify further discussion. $G = (V, E)$ is a connected graph. $V = V(G)$ and $E = E(G)$ are the set of vertices and edges in G , respectively. For a pair of nodes $u, v \in V(G)$, $Eucdist(u, v)$ is the Euclidean distance between them. Let $N(u)$ be the set of nodes adjacent to u , and $N[u] = \{u\} \cup N(u)$. There are two common graph models to abstract wireless network, unit disk graph (UDG) and disk graph (DG).

Unlike UDG, DG is used to abstract nonhomogeneous wireless network, in which nodes may have different maximum transmission range. As a result, DG has directional edges. Clearly, UDG is a special case of DG, and thus, every NP-hard problem in UDG is still NP-hard in DG. In addition, every algorithm working in DG also works in UDG.

Definition 15 (Disk Graphs(DG)) The nodes in V are located in the two-dimensional Euclidean plane, and each node $v_i \in V$ has a transmission range

$r_i \in [r_{\min}, r_{\max}]$. A directed edge $(v_i, v_j) \in E$ if and only if $d(v_i, v_j) \in E$, where $d(v_i, v_j)$ denotes the Euclidean distance between v_i and v_j . Such graphs are called disk graphs (DG).

An edge (v_i, v_j) is bidirectional if both (v_j, v_i) and (v_i, v_j) are in E , i.e., $d(v_i, v_j) \leq \min r_i, r_j$. Usually, they only study the CDS problem in disk graphs where all the edges in the network are bidirectional, called disk graphs with, bidirectional links (DGB). In this case, G is undirected.

Definition 16 (Disk graph with bidirectional links (DGB)) A DGB of a graph is a DG of the graph without all of its directional links.

Definition 17 (m -Dominating Set (m -DS)) $D \subseteq V$ is an m -Dominating Set (m -DS) of G if $\forall u \in (V \setminus D)$, u is adjacent to at least m nodes in D .

Definition 18 (k vertex connectivity) A graph G is k vertex connected if G is still connected after any $k - 1$ nodes are removed from G . In the rest of this section, they will use “ k vertex connected” and “ k -connected,” interchangeably.

Definition 19 (k -Connected m -Dominating Set ((k, m)-CDS)) C is a k -Connected m -Dominating Set, (k, m) -CDS, if (1) a graph induced by C is k -connected and (2) C is a m -DS. For simplicity, they will call (k, k) -CDS by k -CDS.

Definition 20 (Cut-vertex) A vertex $v \in V$ is a cut-vertex if and only if the induced graph by $V \setminus \{v\}$ is disconnected.

Definition 21 (Block) A subset $B \subseteq V$ is a block if and only if B is a maximal connected subgraph of $G = (V, E)$ without any cut-vertex.

Definition 22 (Leaf block) A block of G is called a leaf block if it contains only one cut-vertex.

6.2 Approximations for Computing (k, m) -CDS for $k \leq 3$ and $m \geq 1$

The minimum CDS problem, which is a problem of computing minimum (1,1)-CDS, of a graph, is a well-known NP-hard problem. Therefore, its generalized version, the problem of computing minimum (k, m) -CDS is also NP-hard. As a result, many researches have been conducted to find an approximation algorithm for this problem. In this section, they introduce approximation algorithms for computing (2, 1)-CDS, (1, m)-CDS, (2, m)-CDS, and (3, m)-CDS for any $m \geq 1$.

6.2.1 A Constant Approximation for (2, 1)-CDS

The CDS augmentation algorithm (CDSA) in [79] is the first approximation algorithm to compute multi-connected CDS. Given a 2-connected UDG G , CDSA first utilizes an existing approximation algorithm to compute 1-CDS of G , denoted by $C_{1,1}$. Once this is done, the algorithm augments $C_{1,1}$ to increase its connectivity and generate a (2,1)-CDS, denoted by $C_{2,1}$ as follows:

1. Let $L = \emptyset$ and $C = C_{1,1}$. Find every block in $C_{1,1}$ and put all of them in L .
2. Find a leaf block $B \in L$ and find the shortest path P from $v \in B$ to $u \in C_{1,1} \setminus B$ such that v and u are not cut-vertices. Add all nodes in P to C . If there is more than one block in L , repeat this step.

Figure 7 illustrates an example showing how this algorithm works. It is always true that there is at least one block in any 1-CDS of any graph with more than one vertex. Starting from a block of $C_{1,1}$ (a maximal 2-connected subgraph of $C_{1,1}$), the algorithm finds a shortest path P from a node u in the block to another node v in $C_{1,1}$, which is outside the block such that both of u and v are not cut-vertices in $C_{1,1}$. As a result, the union of the block, P , and some additional nodes from the original 1-CDS form a larger 2-connected subgraph C , which can be considered as a larger leaf block. By repeating such procedure until the subgraph includes all nodes in $C_{1,1}$, the subgraph, which is originally a 2-connected subgraph of $C_{1,1}$, grows into a (2, 1)-CDS.

Using geometry, the authors proved that each path P includes at most 8 new nodes. Also, each time P is added and a 2-connected subgraph is augmented, at least one node in the original $C_{1,1}$ joins to the augmented 2-connected subgraph. As a result, they have

$$2CDS = 1CDS + 8 \times 1CDS,$$

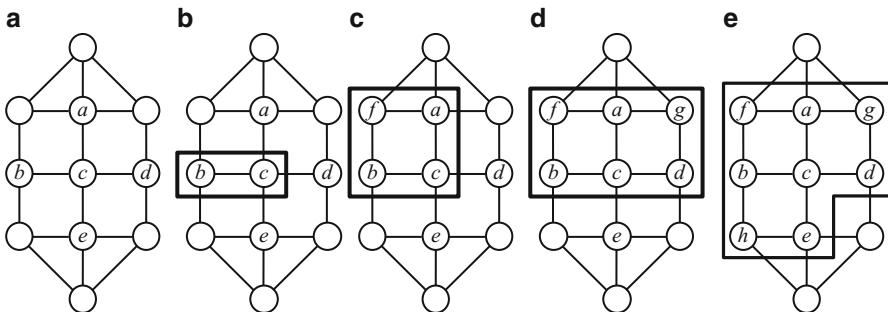


Fig. 7 These figures illustrate how a (2,1)-CDS is computed. In figure (a), the node set $\{a, b, c, d, e\}$ forms a 1-CDS, $C_{1,1}$. In figure (b), the node set $\{b, c\}$ is a leaf block of $C_{1,1}$. In figure (c), two blocks $\{b, c\}$ and $\{a, c\}$ sharing a cut-vertex c of $C_{1,1}$ are merged into a larger 2-connected subgraph $\{a, b, c, f\}$ by adding a path $\{b, f, a\}$. In figure (d), by adding a new path from node a in the 2-connected subgraph to another node d in $C_{1,1}$ outside the subgraph, they have a larger 2-connected subgraph, $\{a, b, c, d, f, g\}$. It is important that both a and d are not cut-vertices. In figure (e), one more node h is added to the subgraph, and finally the augmented subgraph contains $C_{1,1}$. Therefore, this subgraph is a (2,1)-CDS

where $2CDS$ is an output of CDSA, $1CDS$ is the original 1-CDS, and 8 is the maximum length of intermediate nodes in each path P . Based on the existence of a 6.91-approximation algorithm for the minimum CDS problem, the authors originally showed that the approximation ratio of the proposed algorithm is 62.19. Recently, a 6.075-approximation algorithm for the minimum CDS problem has been introduced [57]. By relying on this new result, it can be easily shown that the approximation ratio of CDSA is 54.675.

6.2.2 An Approximation for $(1, m)$ -CDS

In this section, they introduce an approximation algorithm to compute $(1, m)$ -CDS in [70, 74]. Roughly speaking, this method relies on an existing approximation algorithm for the minimum CDS problem. The algorithm consists of following three steps:

1. Compute a $C_{1,1}$ of $G = (V, E)$ using an existing approximation algorithm, which computes an MIS M_0 of G first and connects the nodes in M_0 by adding some nodes B in $V \setminus M_0$. Then, $C_{1,1} = M_0 \cup B$.
2. Set $C = C_{1,1}$. Repeat following from $i = 1$ to $m - 1$: compute an MIS M_i of $(V \setminus (B \cup M_0 \cup \dots \cup M_{i-1}))$ and add it to C .
3. Return C .

The correctness of this algorithm can be proven as follow. In the first step, a $C_{1,1}$ is computed. Then, each node in $V \setminus C_{1,1}$ must have at least one neighbor in $C_{1,1}$ since $C_{1,1}$ includes M_0 , which is an MIS of G and thus is also a DS of G . Similarly, after M_1 is computed, each node in $V \setminus (C_{1,1} \cup M_1)$ must have at least one neighbor in M_1 . This implies that each node in $V \setminus (C_{1,1} \cup M_1)$ must have at least two neighbors in $(C_{1,1} \cup M_1)$, at least one in $C_{1,1}$, and at least another in M_1 . Therefore, $C = C_{1,1} \cup M_1$ is a $(1,2)$ -CDS of G . As a result, the output $C = B \cup M_0 \cup M_1 \cup \dots \cup M_{m-1}$ of the algorithm is a $(1, m)$ -CDS, which is denoted by $C_{1,m}$. Now, they describe the approximation ratio analysis of this algorithm.

Lemma 17 ([70]) *Given $G = (V, E)$, $|M| \leq \max\{\frac{5}{m}, 1\}|D_m^*|$, where M is an MIS of G and D_m^* is a minimum m -DS of G .*

Theorem 12 ([70]) *The approximation ratio of the strategy in this section for the minimum $(1, m)$ -CDS problem is $(5 + \frac{5}{m})$ for $m \leq 5$ and 7 for $m > 5$.*

Proof Let $S_i = M_i \cap D_m^*$. In the algorithm, each $M_i \setminus S_i$ is an IS and $(M_i \setminus S_i) \cap D_m^* = \emptyset$. From Lemma 17, they have $|M_i \setminus S_i| \leq \frac{5}{m}|D_m^* \setminus S_i|$, and

$$\begin{aligned} |M_0 \cup \dots \cup M_{m-1}| &= \sum_{i=0}^{m-1} |S_i| + \sum_{i=0}^{m-1} |M_i \setminus S_i| \leq \sum_{i=0}^{m-1} |S_i| + \sum_{i=0}^{m-1} \frac{5}{m} |D_m^* \setminus S_i| \\ &= \sum_{i=0}^{m-1} |S_i| + \frac{5}{m} \sum_{i=0}^{m-1} (|D_m^*| - |S_i|) = (1 - \frac{5}{m}) \sum_{i=0}^{m-1} |S_i| + 5|D_m^*|. \end{aligned}$$

From this inequality, they can observe $|M_0 \cup \dots \cup M_{m-1}| \leq 5|D_m^*|$ for $m \leq 5$ and $|M_0 \cup \dots \cup M_{m-1}| \leq 6|D_m^*|$ for $m > 5$ since $\sum_{i=1}^m |S_i| \leq |D_m^*|$. In addition, from Lemma 17, they have

$$|B| \leq |M_0| \leq \max \left\{ \frac{5}{m}, 1 \right\} |D_m^*|,$$

since $|B| \leq |M_0|$ is clearly true if a coloring is used to find M_0 . Therefore, they can conclude that the size of $|C|$, which is an output of the algorithm, is bounded by $(5 + \frac{5}{m})|D_m^*|$ for $m \leq 5$ and $7|D_m^*|$ for $m > 5$, and this theorem is proved. \square

6.2.3 An Approximation for $(2, m)$ -CDS

In Sects. 6.2.1 and 6.2.2, they introduced two approximation algorithms, one for computing $(2, 1)$ -CDS and another for computing $(1, m)$ -CDS. In [70], an approximation for computing $(2, m)$ -CDS can be obtained by combining the two strategies as below:

1. Given a UDG G , compute a $C_{1,m}$ of G using an existing approximation algorithm for computing $(1, m)$ -CDS. Let $C = C_{1,m}$.
2. Apply a variation of Wang et al.'s strategy for $(2, 1)$ -CDS in Sect. 6.2.1 on C such that only a path P with length at most 3 will be identified and used to merge a leaf block with another block in $C_{1,m}$.
3. Return C .

It is easy to see that an output of this algorithm C is a $(2, m)$ -CDS. Now, they describe the approximation ratio analysis of this algorithm.

Theorem 13 ([70]) *The approximation ratio of the strategy for the minimum $(2, m)$ -CDS problem is $(5 + \frac{25}{m})$ for $2 \leq m \leq 5$ and 11 for $m > 5$.*

Proof Let $C_{1,m}^*$ and $C_{2,m}^*$ be a minimum $(1, m)$ -CDS of G and a minimum $(2, m)$ -CDS of G , respectively. Clearly, $|C_{1,m}^*| \leq |C_{2,m}^*|$. Largely, this proof depends on the following facts:

1. $C_{2,m}$, a $(2, m)$ -CDS of G , can be obtained from $C_{1,m}$, a $(1, m)$ -CDS, by adding at most $2|C_{1,m}|$ nodes from $V \setminus C_{1,m}$. This is possible based on following two facts:
 - Suppose they have an MIS M_0 of $G = (V, E)$ and a CDS $C = M_0 \cup B$ of G . Let D be a 2-connected subgraph of G which is a union of a subset of C and some other nodes from $V \setminus C$. Then, there has to be a path with length at most 3 from a node u in $D \cap M_0$ to another node v in $(C \setminus D) \cap M_0$ such that both of u and v are not cut-vertices of $C_{1,m}$. Otherwise, there has to be a node in V which is not dominated by any node in M_0 , which makes M_0 not maximal.
 - The number of blocks in $C_{1,m}$ is to at most $|C_{1,m}|$. Furthermore, each time they add a path with length at most 3 from u to v , at least one block will be merged into the 2-connected subgraph which is being augmented.
2. Due to its construction, $|C_{1,1}| = |M_0| + |B|$.

3. $|B| \leq |M_0|$ if coloring is for M_0 .
4. $|M_0| \leq \max\{\frac{5}{m}, 1\}|C_{1,1}^*|$ (from Lemma 17).

From the second and third facts, they have $|C_{1,1}| \leq 2|M_0|$, and by combining this inequality with the first and last ones, they can conclude

$$|C_{2,m}| \leq |C_{1,m}| + 2|C_{1,1}| \leq |C_{1,m}| + 4|M_0| \leq |C_{1,m}| + 4 \max\{\frac{5}{m}, 1\}|C_{1,1}^*|.$$

Finally, from Theorem 12, they have $|C_{2,m}| \leq (5 + \frac{25}{m})|C_{2,m}^*|$ for $2 \leq m \leq 5$ and $|C_{2,m}| \leq 11|C_{2,m}^*|$ for $m > 5$, and this theorem is proved. \square

6.2.4 An Approximation for $(3, m)$ -CDS

In [52], the authors introduced an approximation for computing $(3, m)$ -CDS. The core part of this work is about on how to approximate the minimum $(3, 3)$ -CDS problem. After proposing a constant-approximation algorithm $(3,3)$ -CDS, the authors also showed how to approximate $(3, m)$ -CDS for any $m \geq 1$. Now, they briefly explain how the algorithm works.

In the algorithm, each node u in a $(2,3)$ -CDS, $C_{2,3}$ is either (1) a *good-point* if $C_{2,3} \setminus \{u\}$ is still 2-connected or (2) a *bad-point* if $C_{2,3} \setminus \{u\}$ is 1-connected. Clearly, if all nodes in $C_{2,3}$ are good-points, then $C_{2,3}$ is a $(3, 3)$ -CDS. Note that some of two bad-points $u, v \in C_{2,3}$ can make $C_{2,3} \setminus \{u, v\}$ completely disconnected. Denote such node pair by a *separator* of $C_{2,3}$. A separator $\{u, v\}$ of $C_{2,3}$ can be easily identified using min-max flow algorithm and takes polynomial time.

Now, they introduce the concept of a *block graph* of a graph $G = (V, E)$: remind that a block is a maximal 2-connected subgraph, and a vertex $u \in V$ is a cut-vertex if $G \setminus \{u\}$ is disconnected. Then, a block graph G' is constructed as follows:

1. For each block B of G , there is a corresponding node v_B in G' .
2. For each cut-vertex u in G , there is a node u in G' .
3. At last, for each pair of v_B (representing a block in G) and u (representing a cut-vertex in G) in G' , establish an edge between them only if $u \in B$ in G , where B is a block represented by v_B in G' .

Suppose T_B is a block tree of $C_{2,3} \setminus \{u\}$ rooted at v . Then, v and a cut-vertex w of T_B forms a separator of $C_{2,3}$. Note that w is also a bad-point of $C_{2,3}$. Then, the algorithm adds a path P with a bounded length for w such that w becomes a good-point of $C_{2,3} \cup \{P\}$. Clearly, the number of paths that must be added to make all of the bad-points in $C_{2,3}$ to be good-points is bounded by the size of $C_{2,3}$ multiplied by the number of bad-points. In [52], the authors showed the approximation ratio of this approach is $\frac{520}{3}$ for $(3,3)$ -CDS.

6.3 Approximations for General (k, m) -CDS

So far, several approximation algorithms have been proposed for the minimum (k, m) -CDS problem [59, 74, 83, 85]. Recently, Kim et al. proved that each of them is flawed in a sense that either it does not produce a (k, m) -CDS or its

approximation ratio analysis is not correct [52]. As a result, it is still an open problem to approximate the minimum (k, m) -CDS problem for $k > 3$ and $m \geq 1$.

6.3.1 Centralized Approximation Algorithms for (k, m) -CDS

CDSAN [74] is an approximation algorithm for computing k -CDS. To calculate a k -CDS, CDSAN first computes a $(1, k)$ -CDS and gradually increases its connectivity using a strategy similar to Wang et al.'s in Sect. 6.2.1. In detail, for some l such that $1 < l < k$ and $C_{l,k}$ (note that $C_{1,k}$ can be computed using the algorithm in Sect. 6.2.2), CDSAN repeats (1) identifying a $(l + 1)$ -connected subgraph of $C_{l,k}$ (e.g., leaf block) and (2) adding a path, from a node of $C_{l,k}$ in the subgraph to another node of $C_{l,k}$ outside the subgraph, to the subgraph so that the $(l + 1)$ -connected subgraph can be augmented into a larger $(l + 1)$ -connected subgraph. This repeating is terminated once a $(l + 1)$ -connected subgraph contains $C_{l,k}$. By performing such augmentation for $l = 2, \dots, k - 1$, CDSAN generates a k -connected subgraph containing $C_{1,k}$.

CDSAN works well for computing $C_{2,k}$ since there is always a block (maximal 2-connected subgraph) in any $C_{1,k}$ if $|C_{1,k}| \geq 2$. However, it is possible that there is no 3-connected subgraph in some $C_{2,k}$ of G even though there exists a $(3, k)$ -CDS of G [52]. As a result, CDSAN will fail in some problem instances since it cannot find any $(l + 1)$ -connected subgraph in a (l, k) -CDS.

ICGA [83] is a centralized approximation algorithm to compute (k, m) -CDS in UDG. In the algorithm, a $(1, m)$ -CDS is computed and is evolved into a (k, m) -CDS using following lemma.

Lemma 18 *Given a k -connected subgraph H of $G = (V, E)$ and a connected subgraph F of G such that $H \cap F = \emptyset$ and F does k -dominate H (each node in F has k neighbors in H), the graph induced by $V(H) \cup V(F)$ is $(k + 1)$ -connected.*

Roughly speaking, ICGA first computes a $(1, m)$ -CDS, $C_{1,m}$. Then, it identifies (1) a 2-connected subgraph of $C_{1,m}$ and (2) a subset F which does k -dominate the subgraph. By the lemma, the union of the subgraph and F should be a larger 3-connected subgraph. If the lemma is true, by repeating the augmentation procedure, they may eventually have a (k, m) -CDS. However, the lemma is not always true since there is a 3-connected graph with no such H and F [52].

CSAA [59] is another centralized algorithm for (k, m) -CDS. In detail, given a $C = C_{1,m}$, CSAA first finds a set of i -cut $Sp_i = \{v_{x_1}, \dots, v_{x_i}\}$ such that $C \setminus Sp_i$ is disconnected, and adds more nodes to C to remove the i -cuts. However, it is likely to happen that C is a kind of complete graph, G_{k-1} , while they are looking for a (k, k) -CDS. Then, Sp_i will be an empty set, but C is not k -connected, and therefore this algorithm does not work any further.

6.3.2 Distributed Approximation Algorithms (k, m) -CDS

DDA [59, 85] is a distributed approximation algorithm to compute (k, m) -CDS in UDG and consists of following three steps:

1. Compute a $(1, 1)$ -CDS, $C_{1,1}$ using an existing distributed approximation [54].
2. Augment $C_{1,1}$ to a $(1, m)$ -CDS, $C_{1,m}$ by exploiting the strategy in Sect. 6.2.2.

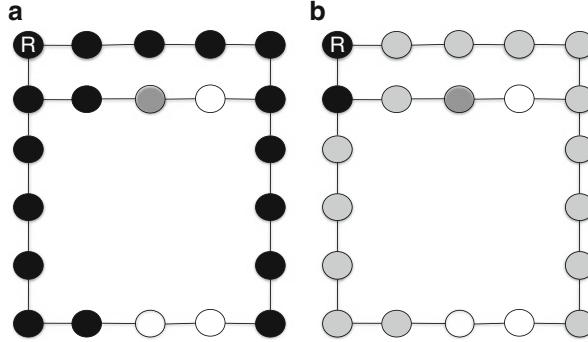


Fig. 8 These figures illustrate an example where DDA does not work correctly. Suppose they want to find a (2,1)-CDS. In figure (a), the set of *black nodes* forms a (1,1)-CDS. In figure (b), the root finds a 2-connected subgraph with its local information (the set of *black nodes*). Now, among the *gray nodes* (which is a member of the (1,1)-CDS other than the *black nodes*), there is no *gray node* which can have two vertex independent paths with length at most three from itself to a member of the 2-connected subgraph (one of the *black node*)

3. Augment $C_{1,m}$ into a (k,m) -CDS, $C_{k,m}$. This can be done by selecting a node using the well-known Expansion Lemma in Graph Theory (below) one by one.

Lemma 19 (Expansion Lemma) *Given a k -connected subgraph $H \subset V$ and $v \notin H$ such that v has at least k neighbors in H , $H' = H \cup \{v\}$ is a k -connected subgraph of $G = (V, E)$.*

Now, they describe an brief idea of Step 3. At the beginning of Step 3, DDA first selects a random root node r from $C_{1,m}$ and try to identify a 2-connected subgraph H of $C_{1,m}$ which consists of r and some or all of its two-hop neighbors. Then, it finds another node $v \in (C_{1,m} \setminus H)$ such that there is k vertex independent paths from v to H with length at most 3. The algorithm claims if they can find such paths the union of H , $\{v\}$, and the nodes on the paths will be a k -connected subgraph which includes H . As they can see in Fig. 8, there are some cases where DDA fails [52], which is also admitted by the authors of [59].

LDA [83] is a localized approximation algorithm for computing (k,m) -CDS in UDG and consists of following steps. Initially, the algorithm assumes that all nodes are colored white:

1. Compute a $C_{1,m}$ using an idea used in DDA and color all nodes in $C_{1,m}$ black.
2. $C_{1,1}$ in $C_{1,m}$ can be considered as a tree rooted at some node. For each parent and child pair $u, v \in C_{1,1} \subseteq C_{1,m}$, color its common neighbors black so that they can share at least k common black neighbors.
3. For each (black) node in $C_{1,1}$, it forms a local k -connected graph, which includes all the neighboring black nodes from Steps 1 and 2.

At the end of the steps, the union of all black nodes is a (k,m) -CDS. The main idea here is that if every parent and child pair in $C_{1,1}$ shares k black neighbors, then

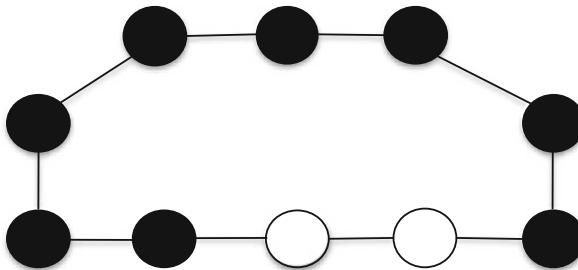


Fig. 9 An example where LDA fails. Suppose they are computing a (2,1)-CDS and the set of *black nodes* above is $C_{(1,1)}$. During the negotiation of each pair of neighboring *black node* u, v , no new *black node* will be introduced. At the end, the union of *black nodes* is still 1-connected. They can have a feasible solution, a (2,1)-CDS, by selecting all nodes

the union of them can be k -connected. However, there are some graph instances G (see Fig. 9), in which some node $u \in G_{1,1}$ of G cannot induce a local k -connected graph, and the union of the all black nodes is not k -connected, but G itself contains a feasible solution, a (k, m) -CDS, and therefore LDA fails [52].

Generally, finding a k -connected subgraph where $k \geq 2$ needs a global knowledge. Therefore, they believe that it will be very interesting and challenging question to design a localized distributed algorithm for a (k, m) -CDS where $k \geq 2$.

7 Conclusion

Virtual backbone can help reduce the searching space for the routing problem from the whole networks to the virtual backbones so that the routing path searching time will be reduced a lot. Due to the advantages of the virtual backbone, it is widely used in broadcasting and routing in wireless networks. Connected Dominating Set is a good choice for virtual backbone. In this book chapter, the previous literatures on Connected Dominating Set are recalled.

Recommended Reading

1. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks. *IEEE Commun. Mag.* **40**, 102–114 (2002)
2. K.M. Alzoubi, P.-J. Wan, O. Frieder, Distributed heuristics for connected dominating sets in wireless ad hoc networks. *IEEE ComSoc/KICS J. Commun. Netw.* **4**(1), 22–29 (2002)
3. K.M. Alzoubi, P.-J. Wan, O. Frieder, Message-optimal connected dominating sets in mobile ad hoc networks, in *Proceedings of the 3rd ACM International Symposium on Mobile ad hoc Networking and Computing*, Lausanne, Switzerland, 2002
4. K.M. Alzoubi, P.-J. Wan, O. Frieder, Weakly-connected dominating sets and sparse spanners in wireless ad hoc networks, in *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems* 96, Washington, DC, 2003

5. C. Ambühl, T. Erlebach, M. Mihalak, M. Nunkesser, Constant-factor approximation for minimum-weight (connected) dominating sets in unit disk graphs, in *Proceedings of the 9th Workshop on Approximation Algorithms for Combinatorial Optimization Problems*. Lecture Notes in Computer Science, vol. 4110 (Springer, Berlin/New York, 2006), pp. 3–14
6. P. Berman, V. Ramaiyer, Improved approximations for the Steiner tree problem. *J. Algorithms* **17**, 381–408 (1994)
7. V. Bharghavan, B. Das, Routing in ad hoc networks using minimum connected dominating sets, in *International Conference on Communication*, Montréal, 1997
8. S. Butenko, O. Ursencko, On minimum connected dominating set problem in unit-ball (Preprint Submitted to Elvier Science, 2007)
9. M. Cadei, M.X. Cheng, X. Cheng, D.-Z. Du, Connected domination in ad hoc wireless networks, in *Proceedings of the Sixth International Conference on Computer Science and Informatics (CS&I'2002)*, Durham, NC, 2002
10. M.-S. Chang, Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM J. Comput.* **27**, 1671–1694 (1998)
11. Y.P. Chen, A.L. Liestman, Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks, in *MobiHoc '02: Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing* (ACM, New York, 2002), pp. 165–172
12. D. Chen, D.-Z. Du, X. Hu, G.-H. Lin, L. Wang, G. Xue, Approximations for Steiner trees with minimum number of Steiner points. *J. Glob. Optim.* **18**, 17–33 (2000)
13. X. Cheng, D.-Z. Du, Virtual backbone-based routing in ad hoc wireless networks. Technical report 02-002, Department of Computer Science and Engineering, University of Minnesota, 2002
14. X. Cheng, X. Huang, D. Li, W. Wu, D.-Z. Du, A polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks. *Networks* **42**, 202–208 (2003)
15. X. Cheng, M. Ding, D.H. Du, X. Jia, Virtual backbone construction in multihop ad hoc wireless networks: research articles. *Wirel. Commun. Mobile Comput.* **6**(2), 183–190 (2006)
16. W.E. Clark, L.A. Dunning, Tight upper bounds for the domination numbers of graphs with given order and minimum degree. *Electron. J. Comb.* **4**(1), R26 (1997). 25p
17. B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs. *Discret. Math.* **86**, 165–177 (1990)
18. F. Dai, J. Wu, On constructing k -connected k -dominating set in wireless network, in *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, Denver, 2005
19. D. Dai, C. Yu, A $(5 + \epsilon)$ -approximation algorithm for minimum weighted dominating set in unit disk graph. *Theor. Comput. Sci.* **410**, 756–765 (2009)
20. B. Das, R. Sivakumar, V. Bharghavan, Routing in ad hoc networks using a spine, in *Proceedings of the International Conference on Computers and Communication Networks*, Las Vegas, NV, 1997
21. L. Ding, Y. Shao, M. Li, On reducing broadcast transmission cost and redundancy in ad hoc wireless networks using directional antennas, in *WCNC* (IEEE, Piscataway, 2008)
22. L. Ding, X. Gao, W. Wu, W. Lee, X. Zhu, D.-Z. Du, An exact algorithm for minimum cds with shortest path constraint in wireless networks. *Optim. Lett.* **5**(2), 297–306 (2010)
23. L. Ding, X. Gao, W. Wu, W. Lee, X. Zhu, D.-Z. Du, Distributed construction of connected dominating sets with minimum routing cost in wireless networks, *Proceedings of IEEE ICDCS*, Genova, 2010
24. L. Ding, W. Wu, W.J.K. Willson, H. Du, W. Lee, D.-Z. Du, Efficient algorithms for topology control problem with routing cost constraints in wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **22**(10), 1601–1609 (2010)
25. L. Ding, W. Wu, J.K. Willson, H. Du, W. Lee, Construction of directional virtual backbones with minimum routing cost in wireless networks, in *IEEE INFOCOM*, Shanghai, 2011
26. C.R. Dow, P.J. Lin, S.C. Chen, J.H. Lin, S.F. Hwang, A study of recent research trends and experimental guidelines in mobile ad hoc networks, in *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*, Taipei, 2005, pp. 72–77

27. D.-Z. Du, L. Wang, B. Xu, The Euclidean Bottleneck Steiner tree and Steiner tree with minimum number of Steiner points, in *COCOON*, Guilin, 2001, pp. 509–518
28. D.-Z. Du, M.T. Thai, Y. Li, D. Liu, S. Zhu, Strongly connected dominating sets in wireless sensor networks with unidirectional links, in *Proceedings of APWEB*, Harbin, 2006
29. D.-Z. Du, R.L. Graham, P.M. Pardalos, P.-J. Wan, W. Wu, W. Zhao, Analysis of greedy approximations with nonsubmodular potential functions, in *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, San Francisco, 2008, pp. 167–175
30. H. Du, W. Wu, S. Shan, D. Kim, W. Lee, Constructing weakly connected dominating set for secure clustering in distributed sensor network. *J. Comb. Optim.* **23**(2), 301–307 (2012)
31. H. Du, Q. Ye, J. Zhong, Y. Wang, W. Lee, H. Park, PTAS for minimum connected dominating set with routing cost constraint in wireless sensor networks, in *COCOA*, vol. 1, Kailua-Kona, HI, 2010, pp. 252–259
32. H. Du, Q. Ye, W. Wu, W. Lee, D. Li, D.-Z. Du, S. Howard, Constant approximation for virtual backbone construction with guarantee routing cost in wireless sensor networks, in *IEEE INFOCOM*, Shanghai, 2011
33. D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, A. Srinivasan, Fast distributed algorithms for (weakly) connected dominating sets and linear-size skeletons, in *SODA '03: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, 2003), pp. 717–724
34. A. Ephremides, J. Wieselthier, D. Baker, A design concept for reliable mobile radio networks with frequency hopping signaling. *Proc. IEEE* **75**(1), 56–73 (1987)
35. H. Eriksson, MBone: the multicast backbone. *Commun. ACM* **37**, 54–60 (1994)
36. T. Erlebach, M. Mihalak, A (4+ ϵ)-approximation for the minimum-weight dominating set problem in unit disk graphs, in *WAOA*, ed. by E. Bampis, K. Jansen, vol. 5893, 2010, pp. 135–146. http://dx.doi.org/10.1007/978-3-642-12450-1_13
37. U. Feige, A threshold of $\ln n$ for approximating set-cover, in *Proceedings of the 28th ACM Symposium on Theory of Computing*, Philadelphia, PA, 1996, pp. 314–318
38. S. Funke, A. Kesselman, U. Meyer, A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Trans. Sens. Netw.* **2**, 444–453 (2006)
39. S. Funke, A. Kesselman, F. Kuhn, Z. Lotker, M. Segal, Improved approximation algorithms for connected sensor cover. *Wirel. Netw.* **13**, 153–164 (2007)
40. X. Gao, Y. Huang, Z. Zhang, W. Wu, $(6 + \epsilon)$ -approximation for minimum weight dominating set in unit disk graphs, in *COCOON*, Dalian, 2008, pp. 551–557
41. X. Gao, Y. Wang, X. Li, W. Wu, Analysis on theoretical bonds for approximating dominating set problems. *Discret. Math. Algorithms Appl.* **1**(1), 71–84 (2009)
42. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, San Francisco, 1979)
43. S. Guha, S. Khuller, Approximation algorithms for connected dominating sets. *Algorithmica* **20**, 374–387 (1998)
44. J.C. Hansen, E. Schmutz, Comparison of two CDS algorithms on random unit ball graphs, in *Proceedings of the Seventh Workshop Algorithm Engineering and Experiments (ALENEX 05)* (SIAM, Philadelphia, 2005)
45. S. Hougaard, H.J. Proommel, A 1.598 approximation algorithm for the Steiner problem in graphs, in *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, 1999, pp. 448–453
46. H. Huang, A.W. Richa, M. Segal, Approximation algorithms for the mobile piercing set problem with applications to clustering in Ad-Hoc networks. *ACM Springer Mobile Netw. Appl.* **9**(2), 151–161 (2004)
47. Y. Huang, X. Gao, Z. Zhang, W. Wu, A better constant-factor approximation for weighted dominating set in unit disk graph. *J. Comb. Optim.* **18**, 179–194 (2008)
48. M.L. Huson, A. Sen, Broadcast scheduling algorithms for radio networks, in *IEEE Military Communications Conference (MILCOM '95)*, San Diego, vol. 2, 1995, pp. 647–651
49. D.S. Johnson, Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.* **9**, 256–278 (1974)

50. M. Karpinski, A. Zelikovsky, New approximation algorithms for the Steiner tree problem. *J. Comb. Optim.* **1**, 47–65 (1997)
51. D. Kim, Y. Wu, Y. Li, F. Zou, D.-Z. Du, Constructing minimum connected dominating sets with bounded diameters in wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **20**(2), 147–157 (2009)
52. D. Kim, W. Wang, X. Li, Z. Zhang, W. Wu, A new constant factor approximation for computing 3-connected m-dominating sets in homogeneous wireless networks, in *Proceedings of the 29th IEEE Conference on Computer Communications (IEEE INFOCOM)*, San Diego, 2010
53. D. Kim, Z. Zhang, X. Li, W. Wang, W. Wu, D.-Z. Du, A better approximation algorithm for computing connected dominating sets in unit ball graphs. *IEEE Trans. Mobile Comput.* **9**(8), 1108–1118 (2010)
54. Y. Li, S. Zhu, M.T. Thai, D.-Z. Du, Localized construction of connected dominating set in wireless networks, in *NSF International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor and Peer-to-Peer Networks (TAWN04)*, Chicago, 2004
55. D. Li, H. Du, P.-J. Wan, X. Gao, Z. Zhang, W. Wu, Construction of strongly connected dominating sets in asymmetric multihop wireless networks. *Theor. Comput. Sci.* **410**(8–10), 661–669 (2009)
56. D. Li, L. Liu, H. Yang, Minimum connected r -hop k -dominating set in wireless networks. *Discret. Math. Algorithms Appl.* **1**(1), 45–57 (2009)
57. M. Li, P.-J. Wan, F.F. Yao, Tighter approximation bounds for minimum CDS in wireless ad hoc networks, in *ISAAC'2009*, Honolulu, 2009
58. Y.S. Li, M.T. Thai, F. Wang, C.-W. Yi, P.-J. Wan, D.-Z. Du, On greedy construction of connected dominating sets in wireless networks. *Wiley J. Wirel. Commun. Mobile Comput.* **5**, 927–932 (2005)
59. Y. Li, Y. Wu, C. Ai, R. Beyah, On the construction of k -connected m -dominating Sets in wireless networks. *J. Comb. Optim.* **23**, 118–139 (2010)
60. G.-H. Lin, G.L. Xue, Steiner tree problem with minimum number of Steiner points and bounded edge-length. *Inf. Process. Lett.* **69**, 53–57 (1999)
61. M. Min, X. Huang, C.-H. Huang, W. Wu, H. Du, X. Jia, Improving construction for connected dominating set with Steiner tree in wireless sensor networks. *J. Glob. Optim.* **35**, 111–119 (2006)
62. K. Mohammed, L. Gewali, V. Muthukumar, Generating quality dominating sets for sensor network, in *Proceedings of IEEE ICCIMA*, Las Vegas, 2005
63. S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, Seattle. MobiCom '99 (ACM, New York, 1999), pp. 151–162
64. T. Nieberg, J. Hurink, W. Kern, Approximation schemes for wireless networks. *ACM Trans. Algorithms* **4**, 1–17 (2008)
65. F. Reichenbach, R. Salomon, D. Timmermann, Distributed obstacle localization in large wireless sensor networks, in *Proceedings of ACM IWCNC*, Vancouver, 2006
66. G. Robin, A. Zelikovsky, Improved Steiner trees approximation in graphs, in *SIAM-ACM Symposium on Discrete Algorithms (SODA)*, San Francisco, 2000, pp. 770–779
67. L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, K.-I. Ko, A greedy approximation for minimum connected dominating set. *Theor. Comput. Sci.* **329**, 325–330 (2004)
68. A. Salhieh, J. Weinmann, M. Kochha, L. Schwiebert, Power efficient topologies for wireless sensor networks, in *ICPP'2001*, Valencia, 2001, pp. 156–163
69. E. Sampathkumar, H.B. Walikar, The connected domination number of a graph. *J. Math. Phys. Sci.* **13**, 607–613 (1979)
70. W. Shang, F. Yao, P. Wan, X. Hu, On minimum m -connected k -dominating set problem in unit disc graphs. *J. Comb. Optim.* **16**, 99–106 (2007)
71. P. Sinha, R. Sivakumar, V. Bharghavan, Enhancing ad hoc routing with dynamic virtual infrastructures, in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3 (IEEE, Piscataway, 2001), pp. 1763–1772

72. R. Sivakumar, B. Das, V. Bharghavan, An improved spine-based infrastructure for routing in ad hoc networks, in *IEEE Symposium on Computer and Communications*, Athens, 1998
73. I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks, in *Proceedings of the IEEE Hawaii International Conference on System Sciences* (IEEE Computer Society, Los Alamitos, 2001)
74. M.T. Thai, N. Zhang, R. Tiwari, X. Xu, On approximation algorithms of k -connected m -dominating sets in disk graphs. *Theor. Comput. Sci.* **358**, 49–59 (2007)
75. Y.-P. Tsai, T.-L. Hsu, R.-S. Liu, Y.-K. Ho, A backbone routing protocol based on the connected dominating set in ad hoc networks. *World Congr. Comput. Sci. Inf. Eng.* **1**, 14–18 (2009)
76. P.-J. Wan, K.M. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks. *Mobile Netw. Appl.* **9**(2), 141–149 (2004). ACM/Springer
77. P.-J. Wan, L. Wang, F. Yao, Two-phased approximation algorithms for minimum CDS in wireless ad hoc networks, in *IEEE ICDCS*, Beijing, 2008, pp. 337–344
78. Y. Wang, X.Y. Li, Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Netw. Appl.* **11**(2), 161–175 (2006)
79. F. Wang, M.T. Thai, D.-Z. Du, 2-connected virtual backbone in wireless network. *IEEE Trans. Wirel. Commun.* **8**(3), 1230–1237 (2007). Accepted with revisions
80. Z. Wang, W. Wang, J. Kim, B.M. Thuraisingham, W. Wu, PTAS for the minimum weighted dominating set in growth bounded graphs. *J. Glob. Optim.* **54**(3), 641–648 (2012)
81. J. Willson, X. Gao, Z. Qu, Y. Zhu, Y. Li, W. Wu, Efficient Distributed Algorithms for Topology Control Problem with Shortest Path Constraints. *Discret. Math. Algorithms Appl.* **1**, 437–461 (2009)
82. J. Wu, H. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in *Proceedings of the 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Seattle, 1999, pp. 7–14
83. Y. Wu, Y. Li, Construction algorithms for k -connected m -dominating sets in wireless sensor networks, in *9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2008)* (ACM, New York, 2008)
84. W. Wu, H. Du, X. Jia, Y. Li, S.C.-H. Huang, Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theor. Comput. Sci.* **352**(1–3), 1–7 (2006)
85. Y. Wu, F. Wang, M.T. Thai, Y. Li, Constructing k -connected m -dominating sets in wireless sensor networks, in *Proceedings of 2007 Military Communications Conference (MILCOM07)*, Orlando, 2007
86. C. Zong, *Sphere Packings* (Springer, New York, 1999)
87. Z. Zhang, X. Gao, W. Wu, D.-Z. Du, A PTAS for minimum connected dominating set in 3-dimensional wireless sensor networks. *J. Glob. Optim.* **45**(3), 451–458 (2008). published online, Dec. 2008
88. Z. Zhang, X. Gao, W. Wu, Algorithms for connected set cover problem and fault-tolerant connected set cover problem. *Theor. Comput. Sci.* **410**, 812–817 (2009)
89. X. Zhong, J. Wang, N. Hu, Connected dominating set in 3-dimensional space for ad hoc network, in *Proceedings of the IEEE Wireless Communication and Networking Conference*, Hong Kong, 2007
90. F. Zou, Y. Wang, X. Li, X. Xu, H. Du, P. Wan, W. Wu, New approximations for minimum-weighted dominating set and minimum-weighted connected dominating set on unit disk graphs. *Theor. Comput. Sci.* **412**(3), 198–208 (2009)
91. F. Zou, X. Li, D. Kim, W. Wu, Construction of minimum connected dominating set in 3-dimensional wireless network, in *Proceedings of Third International Conference Wireless Algorithms, Systems and Applications (WASA 2008)*, Dallas, 2008
92. F. Zou, X. Li, D. Kim, W. Wu, Two constant approximation algorithms for node-weighted Steiner Tree in unit disk graphs, in *COCOA*, 2008, pp. 278–285. http://dx.doi.org/10.1007/978-3-540-85097-7_26
93. F. Zou, X. Li, D. Kim, W. Wu, Construction of minimum connected dominating set in 3-dimensional wireless network, in *Wireless Algorithms, Systems, and Applications*, vol. 5258 (Springer, Berlin, 2008), pp. 134–140

Connections Between Continuous and Discrete Extremum Problems, Generalized Systems, and Variational Inequalities*

Carla Antoni, Franco Giannessi and Fabio Tardella

Contents

1	Introduction	836
2	Equivalence Through Relaxation–Penalization	838
2.1	General Results	838
2.2	Equivalence Between Mixed-Integer and Real Optimization	842
2.3	On the Estimate of Penalization Parameters	846
2.4	On the Equivalence of Lagrangian Duality and Penalization	850
3	Generalized Concavity and Extreme Point Minimizers	852
3.1	Global Minimizers at Extreme Points	852
3.2	The Case of Nonconvex Domains	854
3.3	The Case of Polyhedral Domains	856
3.4	Pseudo-Boolean Optimization and Optimization with Box Constraints	863
3.5	Convex–Concave Problems	867
3.6	Max Clique and Optimization Over Simplices and Polymatroids	868
4	Piecewise Concavity and Applications	872
4.1	Piecewise Concavity	872
4.2	Minimax Problems	874
4.3	Location Problems	875

*Section 1 is due to Giannessi; Sects. 2, 5, and 6 are due to Antoni; and Sects. 3 and 4 are due to Tardella.

C. Antoni
Naval Academy, Livorno, Italy
e-mail: carla.antoni5@gmail.com

F. Giannessi
University of Pisa, Pisa, Italy
e-mail: fgiannessi3@gmail.com

F. Tardella
Sapienza University of Rome, Rome, Italy
e-mail: fabio.tardella@uniroma1.it

5 Generalized Systems.....	879
5.1 Introduction.....	879
5.2 General Results.....	879
5.3 The Case of a Discrete Domain.....	886
5.4 Discrete Vector Optimization and Variational Inequalities.....	888
6 Conclusion.....	891
Cross-References.....	895
Recommended Reading.....	895

Abstract

This writing aims at surveying what has been done to analyze some connections between continuous and discrete extremum problems and related models, like generalized systems and variational inequalities.

Keywords

Continuous Optimization • Discrete Optimization • Generalized Convexity • Relaxation • Penalization • Variational Inequalities

AMS Classification: 65K05; 65K10; 65K15; 90C05; 90C11; 90C30;

1 Introduction

Problems defined in a discrete space are, perhaps, the oldest mathematical problems. The birth and great development of infinitesimal analysis threw discrete problems somewhat into the shade. After the Second World War, due to the development of automatic computation and its wide applications, discrete problems have had a new life. In all these strong movements, one thing has remained almost stable: the fields of continuous and discrete problems have had a small intersection; this is true, in particular, for constrained extremum problems. There is a deep convincement that the two fields, which could be called “Continuous Structures of Mathematics” and “Discrete Structures of Mathematics,” should be considered explicitly and connected more and more.

The systematic study of such connections should by no way imply that one field can be reduced to the other; on the contrary, the more “spectacles” (of Galilean memory) one has to look at reality, the more insight one can achieve. The analysis of connections is meant in this light. In the last four decades of the last century, in the field of constrained extremum problems, most of the efforts have been devoted to transform discrete problems into continuous ones. Even though this activity has turned out to be useful, it has sometimes led to the wrong convincement that a continuous format be the final destination of every problem. Like every human activity, mathematical research is influenced by a sort of inertia: if a problem has been classified, since the beginning, as a continuous one, almost surely, it will be treated by the methods of continuous mathematics, independently of its deep nature.

For instance, the analysis of elastoplastic behavior of structures and materials has always been considered a continuous problem; a part of such an analysis leads to search for the minimum of a nonconvex function, which has several local, but not global, minima; this implies that, in the study of stationary points, one meets—without being ready—a combinatorial problem, which often leads the computation to quicksands.

This writing aims at surveying what has been done to analyze some connections between continuous and discrete extremum problems and related models, like generalized systems and variational inequalities. As concerns extremum problems, a first trace is contained in [64] and followed in [49]. Independently of [64], and in a slightly more general context, a first analysis of such connections has been faced in [38]. The idea expressed in [64] has been picked up in [36], where, however, the mathematical tools are independent of those in [64], inasmuch as the “ad hoc” analysis of linear algebraic problems is no longer valid for nonlinear ones; indeed, in [36], the equivalence is established between a nonlinear constrained extremum problem in a Euclidean space and a problem that is both a relaxation and a penalization of it; the equivalence between a continuous and a discrete problem is obtained as a corollary of it. The results of [36] stimulated some further research in the field; see, e.g., [2–5, 26, 29, 34, 35, 38, 44, 55–57, 61, 65].

The great development of the theory of constrained extrema and, more recently, of that of variational inequalities, has led to search for models that embody both theories. A possible answer is offered by the so-called generalized systems. Also such theories have received different and, sometimes, almost disjoint developments depending on the kind of space, where the problems have been defined. These kinds of problems will also be considered here.

Given a set X , a function $f : X \rightarrow \mathbb{R}$ and a subset S of X consider the problem:

$$\min f(x) \quad \text{s.t. } x \in S \tag{1}$$

Problem (1) is usually called a *combinatorial optimization problem* when S is finite and a *discrete optimization problem* when the points of S are isolated in some topology, i.e., every point of S has a neighborhood which does not contain other points of S . Obviously, all combinatorial optimization problems are also discrete optimization problems, but the converse is not true. A simple example is the problem of minimizing a function on the set of integer points contained in an unbounded polyhedron.

Even though some discrete and combinatorial optimization problems have been studied since ancient times, the increase of their importance and their development have been very fast only in the last few decades thanks to the possibility of practically solving them with modern computers and because of the several applications that they have found in many fields.

In most cases, discrete and combinatorial optimization problems can be formulated as linear or nonlinear integer programming problems and are solved by means of methods that exploit in a crucial way the finiteness or discreteness of the feasible region.

The linear programming formulation of most discrete and combinatorial optimization problems is typically based on the Fundamental Theorem of Linear Programming and on polyhedral theory. This subject, which is very well illustrated in the recent books [69] by Schrijver, is not covered here.

In the case where the feasible set S is defined by a family of equations and inequalities in \mathbb{R}^n and at least one of the constraints or objective functions is nonlinear, problem (1) is called a *nonlinear programming problem* or a *nonlinear program*. Problems of this kind have been studied for at least four centuries with tools that exploit the differential, geometric, and topological properties of the functions and sets involved.

Although the methods employed in integer and nonlinear programming are quite different in general, they can be complementary in several cases. In fact, it is often possible to restrict the search for a global solution of a nonlinear program to a finite set of points. On the other hand, most discrete optimization problems can be reformulated equivalently as nonlinear programs. Clearly, the restriction to a finite set of points or the nonlinear reformulation does not always lead to practical solution methods. Nevertheless, these approaches provide useful tools for many important classes of problems, as will be shown in the sequel.

This chapter is not meant to give an exhaustive survey of all connections between discrete optimization and nonlinear programming. The purpose here is to present the main tools that have been employed in the literature to transform a discrete problem into an *equivalent* nonlinear one and vice versa. This chapter does not cover, e.g., the many important continuous relaxations of discrete problems that lead to lower or upper bounds on the original problem without being equivalent to it.

2 Equivalence Through Relaxation–Penalization

2.1 General Results

This section contains general results about the equivalence of a constrained extremum problem with another one obtained from the former by enlarging the feasible region and penalizing the objective function. Several known and new results on the equivalence between mixed-integer and (continuous) nonlinear programming problems are derived as special cases of such general results.

More formally, one wishes to describe conditions that guarantee the equivalence of the problems

$$\min f(x) \quad \text{s.t. } x \in W, \tag{2}$$

and

$$\min[f(x) + \varphi(x; \epsilon)] \quad \text{s.t. } x \in X, \tag{3}$$

where $W \subseteq X \subset \mathbb{R}^n$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, and, for every $\epsilon \in \mathbb{R}_+$, $\varphi(\cdot; \epsilon) : \mathbb{R}^n \rightarrow \mathbb{R}$ is a suitable penalty function.

A recent general result in this direction is the following theorem, originally stated by Lucidi and Rinaldi with an unnecessary compactness assumption here omitted.

Theorem 1 (Lucidi and Rinaldi [56]) *Let $W \subseteq X \subset \mathbb{R}^n$, let $\|\cdot\|$ be any norm in \mathbb{R}^n , and make the following assumptions:*

(A1) *The function f is bounded on X and there exist an open set $\mathcal{A} \supset W$ and real numbers α , $L > 0$ such that, for any $x, y \in \mathcal{A}$,*

$$|f(x) - f(y)| \leq L \|x - y\|^\alpha; \quad (4)$$

The function $\varphi(\cdot; \epsilon)$ satisfies the following conditions:

(A2) *for every $x, y \in W$, and $\epsilon \in \mathbb{R}_+$,*

$$\varphi(x; \epsilon) = \varphi(y; \epsilon). \quad (5)$$

(A3) *There exists a value $\hat{\epsilon}$ such that, for every $z \in W$, there is a neighborhood $S(z)$ of z such that, for every $x \in S(z) \cap (X \setminus W)$ and $\epsilon \leq \hat{\epsilon}$,*

$$\varphi(x; \epsilon) - \varphi(z; \epsilon) \geq \hat{L} \|x - z\|^\alpha, \quad (6)$$

where $\hat{L} > L$. Furthermore, set $S := \cup_{z \in W} S(z)$ and suppose there is a point $\bar{x} \in X \setminus S$ such that

$$\lim_{\epsilon \rightarrow 0} \varphi(\bar{x}; \epsilon) - \varphi(z; \epsilon) = +\infty, \quad (7)$$

$$\varphi(x; \epsilon) \geq \varphi(\bar{x}; \epsilon), \quad \forall x \in X \setminus S. \quad (8)$$

Then, there exists a value $\tilde{\epsilon} > 0$ such that, for all $\epsilon < \tilde{\epsilon}$, (2) and (3) have the same minimum points.

Proof First, observe that a minimum point of (3) is also a minimum point of (2). Indeed, for all $\epsilon > 0$, if x^* is a minimum point of (3), then

$$f(x^*) + \varphi(x^*; \epsilon) \leq f(x) + \varphi(x; \epsilon), \quad \forall x \in X.$$

Since $W \subseteq X$, it follows that

$$f(x^*) + \varphi(x^*; \epsilon) \leq f(x) + \varphi(x; \epsilon), \quad \forall x \in W.$$

If $x^* \in W$, then Assumption (A2) ensures that

$$f(x^*) \leq f(z), \quad \forall z \in W,$$

which shows that x^* is a minimum point of (2). Now one can prove that there exists a value $\tilde{\epsilon}$ such that, for all $\epsilon > \tilde{\epsilon}$, every minimum point of (3) belongs to W . Let \bar{x} and S be, respectively, the point and the open set defined in Assumption (A3). Hence, by (7), there exists a value $\bar{\epsilon}$ such that, for $\epsilon > \bar{\epsilon}$, the following inequality holds:

$$\varphi(\bar{x}; \epsilon) - \varphi(z; \epsilon) > \sup_{x \in W} f(x) - \inf_{x \in (X \setminus S)} f(x). \quad (9)$$

Then, put

$$\tilde{\epsilon} := \max\{\bar{\epsilon}, \hat{\epsilon}\}, \quad (10)$$

where $\hat{\epsilon}$ is defined as in (A3). Ab absurdo, suppose there exists a $\epsilon > \tilde{\epsilon}$ such that x^* is a minimum point not in W . Consider two different cases:

Case 1: $x^* \in S$.

Without any loss of generality, consider $S \subset \mathcal{A}$. In this case, there exists $z \in W$ such that $x^* \in S(z)$. Using the definition of $\hat{\epsilon}$, Assumptions (A1) and (A3), one obtains

$$\begin{aligned} f(z) - f(x^*) &\leq L \|z - x^*\|^\alpha < \hat{L} \|z - x^*\|^\alpha \\ &\leq \varphi(x^*; \epsilon) - \varphi(z; \epsilon). \end{aligned}$$

Thus the contradiction

$$f(z) + \varphi(z; \epsilon) < f(x^*) + \varphi(x^*; \epsilon).$$

Case 2: $x^* \in X \setminus S$. In this case

$$f(x^*) + \varphi(x^*; \epsilon) \geq \inf_{x \in X \setminus S} f(x) + \varphi(x^*; \epsilon). \quad (11)$$

For all $z \in W$, one has $f(z) - \sup_{x \in W} f(x) \leq 0$; then

$$f(x^*) + \varphi(x^*; \epsilon) \geq f(z) - \sup_{x \in W} f(x) + \inf_{x \in X \setminus S} f(x) + \varphi(x^*; \epsilon). \quad (12)$$

By using (8) and Assumption (A3), one obtains

$$f(x^*) + \varphi(x^*; \epsilon) \geq f(z) - \sup_{x \in W} f(x) + \inf_{x \in X \setminus S} f(x) + \varphi(\bar{x}; \epsilon). \quad (13)$$

Adding and subtracting $\varphi(z; \mu)$ lead to

$$\begin{aligned} f(x^*) + \varphi(x^*; \epsilon) &\geq f(z) + \varphi(z; \epsilon) + \varphi(\bar{x}; \epsilon) - \varphi(z; \epsilon) \\ &\quad - \sup_{x \in W} f(x) + \inf_{x \in X \setminus S} f(x). \end{aligned}$$

Finally, recalling the definition of $\tilde{\epsilon}$ and exploiting (9), one obtains the contradiction

$$f(x^*) + \varphi(x^*; \epsilon) > f(z) + \varphi(z; \epsilon).$$

Now, one proves that, for all $\epsilon > \tilde{\epsilon}$, every minimum point of (2) is also a minimum point of (3). Ab absurdo, suppose that there is an $\epsilon > \tilde{\epsilon}$ such that

$$f(x^*) + \varphi(x^*; \epsilon) < f(z) + \varphi(z; \epsilon),$$

where z is a minimum point of (2) and x^* is a minimum point of (3). Recalling the first part of the proof, one obtains that, for every $\epsilon < \tilde{\epsilon}$, the point x^* is also a minimum point of (2); hence, using Assumption (A2), it follows that

$$f(x^*) < f(z),$$

which contradicts the fact that z is a minimum point of (2). \square

The previous theorem generalizes a similar result in [36] described below. In this case, the problem considered is

$$\min f(x) \text{ s.t. } x \in R \cap Z, \quad (14)$$

where Z and R are subsets of \mathbb{R}^n and $f : \mathbb{R}^n \rightarrow \mathbb{R}$; the result states conditions under which (14) is equivalent to

$$\min[f(x) + \mu\psi(x)] \text{ s.t. } x \in R \cap X, \quad (15)$$

where $X \supseteq Z$, $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$ is a suitable penalty function and μ is a real parameter.

Theorem 2 (Giannessi and Niccolucci [36]) *Let $R \subseteq \mathbb{R}^n$ be closed, let $Z \subseteq X$ be compact, and let the following hypotheses hold:*

(H₁) *$f : X \rightarrow \mathbb{R}$ is bounded, and there exist an open set $\mathcal{A} \supset Z$ and real numbers α , $L > 0$, such that, for every $x, y \in \mathcal{A}$, f fulfills the following Hölder condition:*

$$|f(x) - f(y)| \leq L \|x - y\|^\alpha.$$

(H₂) *Let $\psi : \mathbb{R}^n \rightarrow \mathbb{R}$ be such that:*

- (i) *ψ is continuous on X .*
- (ii) *$\psi(x) = 0$, $\forall x \in Z$, $\psi(x) > 0$, $\forall x \in X \setminus Z$.*
- (iii) *$\forall z \in Z$, there exists a neighborhood $S(z)$ of z , and a real $\epsilon(z) > 0$ such that*

$$\psi(x) \geq \epsilon(z) \|x - z\|^\alpha, \quad \forall x \in S(z) \cap (X \setminus Z).$$

Then, a real μ_0 exists such that, for every $\mu > \mu_0$, (14) and (15) are equivalent.

Proof Apply [Theorem 1](#): now the sets W and X are, respectively, equal to $R \cap Z$ and $R \cap X$. Furthermore, the penalty function φ of (3) is given by $\varphi(x; \epsilon) = \psi(x)/\epsilon$. Then $(H_2)(ii)$ for ψ implies $(A2)$ for φ . Also, consider any \hat{L} with $\hat{L} > L$. Since Z is compact, there is a finite cover S of Z , $S = \cup_{i=1}^k S(z_i)$, where the neighborhoods $S(z_i)$ are those of $(H_2)(iii)$. Setting $\hat{\epsilon} := \min\{\frac{\epsilon(z_i)}{\hat{L}}, i = 1, \dots, k\}$, for all $\epsilon \leq \hat{\epsilon}$ and $x \in (S(z_i) \cap X) \setminus Z$, it holds that

$$\varphi(x; \epsilon) \geq \hat{L} \|x - z_i\|.$$

Since $(R \cap X) \setminus S$ is compact, the continuous function ψ has a minimum point \bar{x} on $(R \cap X) \setminus S$; thanks to $(H_2)(ii)$, $\psi(\bar{x}) > 0$, and then (7) and (8) hold. So thanks to [Theorem 1](#), there exists $\bar{\epsilon}$ such that, for $\epsilon < \bar{\epsilon}$, (14) and

$$\min[f(x) + \psi(x)/\epsilon] \text{ s.t. } x \in R \cap X$$

have the same minimum points. The thesis follows choosing $\mu_0 = 1/\bar{\epsilon}$ and $\mu = 1/\epsilon$. \square

2.2 Equivalence Between Mixed-Integer and Real Optimization

A special case of (14), which embraces most combinatorial problems, is that where

$$Z = \{0, 1\}^n, \quad R = \{x \in \mathbb{R}^n : g(x) \geq 0\},$$

with $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$. Thus, problem (14) becomes

$$\min f(x) \text{ s.t. } x \in \{x \in \mathbb{R}^n : g(x) \geq 0\} \cap \{0, 1\}^n. \quad (16)$$

In this case, the set $X = X_Q := \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, i = 1, \dots, n\}$ is the natural relaxation of Z . In [36] the function ψ defined by

$$\psi(x) = \sum_{i=1}^n x_i(1 - x_i), \quad (17)$$

is introduced as a penalty function. In this way, (15) becomes

$$\min \left[f(x) + \mu \sum_{i=1}^n x_i(1 - x_i) \right] \text{ s.t. } x \in R \cap X_Q. \quad (18)$$

From [Theorem 2](#) one then derives the following result.

Theorem 3 ([36]) Let f be a Lipschitz function on X_Q . Then, there exists $\mu_0 \in \mathbb{R}$, such that, $\forall \mu > \mu_0$, problems (16) and (18) have the same set of global minimum points. Furthermore, if $f \in C^2(X_Q)$, then there exists $\mu_1 \in \mathbb{R}$ such that, $\forall \mu > \mu_1$, $f + \mu\psi$ is a strictly concave function.

Proof One only needs to prove that ψ of (17) satisfies Assumption (H_2) of Theorem 2 with $\alpha = 1$. Note that $(H_2)(i)$ and $(H_2)(ii)$ are trivially verified. For $\bar{\rho}$ in $]0, 1, [$ one can prove that ψ fulfills $(H_2)(iii)$, with $S(z) = \{x \in \mathbb{R}^n : \|x - z\| \leq \bar{\rho}\}$, $\epsilon(z) = 1 - \bar{\rho}$. Indeed, for $x \in S(z)$, put $\rho := \|x - z\|$ and $u := (x - z)/\rho$. Then

$$\psi(x) = \sum_{j=1}^n (\rho u_j + z_j)(1 - z_j - \rho u_j). \quad (19)$$

As $z + \rho u = x \in X_Q$, if $u_j > 0$, then $z_j = 0$; but if $u_j < 0$, then $z_j = 1$. From (19) it follows

$$\psi(x) = \sum_{u_j > 0} \rho u_j (1 - \rho u_j) + \sum_{u_j < 0} (1 + \rho u_j)(-\rho u_j) = \sum_{j=1}^n \rho |u_j| (1 - \rho |u_j|). \quad (20)$$

By construction $\|u\|^2 = 1$ and $\sum_j |u_j| \geq \|u\| = 1$; then, from (20), it follows that

$$\psi(x) = \rho \sum_{j=1}^n |u_j| - \rho^2 \geq \rho(1 - \bar{\rho}) = \epsilon(z) \|x - z\|.$$

Thus Assumption (H_2) of Theorem 2 is satisfied and the proof of equivalence of (16) and (18) follows. To prove strict concavity of $f + \mu\psi$, let $H(x)$ and $\hat{H}(x)$ denote the Hessian matrices of f and of $f + \mu\psi$, respectively. Then $\hat{H}(x) = H(x) - 2\mu I_n$. So $v(x)$ is an eigenvalue of $\hat{H}(x)$ if and only if $v(x) + 2\mu(x)$ is an eigenvalue of $H(x)$. The continuity of H implies that of the eigenvalues $\lambda_1, \dots, \lambda_n$ [25]. Thus, since X_Q is compact, one has

$$\tilde{\lambda} = \max_{i=1, \dots, n} \max_{x \in X} |\lambda_i(x)| \in \mathbb{R}.$$

Therefore, and $x \in X_Q$, one has $v(x) < 0$ for all eigenvalues $v(x)$ of $\hat{H}(x)$ so that $f + \mu\psi$ is strictly concave on X_Q . \square

It is interesting to observe that the second part of this theorem shows that every integer problem (16) with a sufficiently regular objective function can be transformed into a *concave* continuous problem, for which several solution methods have been developed (see, e.g., [11, 46, 62]).

For problem (16), in addition to the function $\psi(x) = \sum_{i=1}^n x_i(1-x_i)$ of [36], the following concave penalty functions $\varphi(\cdot; \epsilon) : \mathbb{R}^n \rightarrow \mathbb{R}$ have been proposed in [65]:

$$\varphi(x; \epsilon) = \sum_{i=1}^n \left(\log(x_i + \epsilon) + \log((1-x_i) + \epsilon) \right), \quad (21)$$

$$\varphi(x; \epsilon) = \sum_{i=1}^n \left(-(x_i + \epsilon)^{-p} - [(1-x_i) + \epsilon]^{-p} \right), \quad (22)$$

$$\varphi(x; \epsilon) = \frac{1}{\epsilon} \sum_{i=1}^n \left(\left[1 - e^{(-\alpha x_i)} \right] + \left[1 - e^{(-\alpha(1-x_i))} \right] \right), \quad (23)$$

$$\varphi(x; \epsilon) = \frac{1}{\epsilon} \sum_{i=1}^n \left([x_i + \epsilon]^q + [(1-x_i) + \epsilon]^q \right). \quad (24)$$

Furthermore, the following has been introduced in [56]:

$$\varphi(x; \epsilon) = \frac{1}{\epsilon} \sum_{i=1}^n \left([1 + e^{(-\alpha x_i)}]^{-1} + [1 + e^{(-\alpha(1-x_i))}]^{-1} \right), \quad (25)$$

where $\epsilon, \alpha, p > 0$, and $0 < q < 1$.

Proposition 1 ([56]) *For every penalty term (21)–(25), there exists a value $\bar{\epsilon}$ such that, for any $\epsilon \in]0, \bar{\epsilon}[$, problems (16) and (18) have the same set of global minimum points.*

A penalty function for mixed-integer problems is introduced in [3]. More precisely, consider the mixed-integer problem:

$$\inf f(u, v) \quad \text{s.t. } (u, v) \in R, \quad (26)$$

where $f : \mathbb{R}^n \times \mathbb{R}^k \rightarrow \mathbb{R}$, $R := \{(u, v) \in \{0, 1\}^n \times \mathbb{R}^k : g(u, v) \in D\}$, $g : \mathbb{R}^n \times \mathbb{R}^k \rightarrow \mathbb{R}^m$, and $D \subseteq \mathbb{R}^m$.

For a closed convex set $A \subseteq \mathbb{R}^h$, let $\text{proj}_A : \mathbb{R}^h \rightarrow \mathbb{R}^h$ be the function defined by

$$\|x - \text{proj}_A(x)\| = \min_{y \in A} \|y - x\|;$$

let $\Pi : R^n \times \mathbb{R}^k \rightarrow \mathbb{R}^k$ be the canonical projection on \mathbb{R}^k :

$$\Pi(u, v) = v.$$

Proposition 2 (Antoni [3]) Suppose R is closed. If, for all $z \in \{0, 1\}^n$, the set $A_z = \{(u, v) : (u, v) \in R, u = z\}$ is a closed convex set, then the function $\Psi : \mathbb{R}^n \times \mathbb{R}^k \times \mathbb{R} \rightarrow \mathbb{R}$ defined by

$$\Psi(u, v; \epsilon) = \left[\sum_{i=1}^n u_i(1 - u_i) + \|v - \Pi(\text{proj}_R(u, v))\| \right] / \epsilon$$

satisfies Assumptions (A2) and (A3) of [Theorem 1](#).

When the equivalence between (16) and (18) holds, properties and methods valid for one of the two problems can be transferred to the other one. As an instance of this, consider the quadratic programming problem

$$\min \left[\langle c, x \rangle + \frac{1}{2} \langle x, Cx \rangle \right] \text{ s.t. } x \in \{Ax \geq b\} \cap \{0, 1\}^n, \quad (27)$$

where $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $C \in \mathbb{R}^{n \times n}$ is a symmetric matrix.

Theorem 4 ([36]) There is a real μ_2 such that, for all $\mu > \mu_2$, THE problem (27) and the linear complementarity problem

$$\min \langle \bar{c}, \xi \rangle \text{ s.t. } (\xi, \eta) \in D, \quad (28)$$

where

$$D := \{(\xi, \eta) \in \mathbb{R}^{2n+m} \times \mathbb{R}^{2n+m} : \bar{A}\xi + \eta = \bar{b}, \xi \geq 0, \eta \geq 0, \langle \xi, \eta \rangle = 0\}$$

$$\bar{A} = \begin{pmatrix} -C + 2\mu I_n & A^T & -I_n \\ A & 0 & 0 \\ I_n & 0 & 0 \end{pmatrix},$$

$$\bar{c}^T = (1/2)(c^T + \mu e^T + e^T, b^T, e^T), \quad \bar{b}^T = (c^T, b^T, e^T),$$

have the same set of global minimum points.

Proof Because of [Theorem 3](#), whose hypotheses are trivially satisfied, (27) is equivalent to the quadratic problem

$$\min \left[\langle c + \mu e, x \rangle + \frac{1}{2} \langle x, (C - 2\mu I_n)x \rangle \right] \text{ s.t. } x \in \{Ax \geq b\} \cap X_Q, \quad (29)$$

where $e = (1, \dots, 1) \in \mathbb{R}^n$, if μ is large enough. The well-known Karush–Kuhn–Tucker necessary condition for (29) is

$$\begin{aligned} c^T + \mu e^T + (C - 2\mu I_n)x - A^T y + t - u &= 0 \\ Ax - v = b; \quad x + w = e; \quad x, y, t, u, v, w \geq 0 \\ \langle x, u \rangle = \langle y, v \rangle = \langle t, w \rangle = 0, \end{aligned} \tag{30}$$

where y, t, u are vectors of multipliers associated to the inequalities $Ax \geq b, x \leq e$, and $x \geq 0$, respectively, and v, w are slack variables. Solving (29) is equivalent to finding, among the solutions of the complementarity system (30) (stationary points), those which minimize the function in square brackets of (29). Such a function, evaluated at the stationary points, becomes “linear”:

$$\frac{1}{2}[\langle c + \mu e, x \rangle + \langle b, y \rangle + \langle e, t \rangle].$$

Indeed, (30) implies

$$\begin{aligned} Cx - 2\mu x &= -(c + \mu e) + A^T y - t + u, \\ \langle x, u \rangle &= 0, \\ 0 &= \langle y, v \rangle = \langle y, b - Ax \rangle, \\ 0 &= \langle t, w \rangle = \langle t, e - x \rangle, \end{aligned}$$

and therefore

$$\begin{aligned} \langle y, b \rangle &= \langle y, Ax \rangle, \\ \langle t, e \rangle &= \langle t, x \rangle. \end{aligned}$$

Now, to achieve the thesis, it is sufficient to set

$$\xi^T := (x^T, y^T, t^T), \quad \eta^T := (u^T, v^T, w^T).$$

This completes the proof. \square

Note that no assumption has been made on the matrix C , so the convex case, as well the nonconvex one, has been considered. See also [61] for a reduction of a mixed-integer feasibility problem to a linear complementarity system.

2.3 On the Estimate of Penalization Parameters

For stability reasons, it is often useful to be able to choose a penalization parameter μ not too large. This section contains some theoretical results on the comparison between the parameters μ given in [36, 49]. More precisely, in [49] the following linear optimization problem is considered:

$$\min[-\langle c, x \rangle] \text{ s.t. } x \in R \cap Z, \quad (31)$$

where $x, c \in \mathbb{R}^n$, R is a polyhedron of \mathbb{R}^n , and $Z = \{0, 1\}^n$ is the set of vertices of the hypercube $X = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1\}$. Without any loss of generality, suppose c have nonnegative components.

Kalantari and Rosen [49] find a solution x_0 of

$$\min[-\langle c, x \rangle] \text{ s.t. } x \in R \cap X.$$

If $x_0 \in Z$, then x_0 solves (31). Otherwise, they consider the following maximization problem

$$\max g(x) \text{ s.t. } x \in \hat{F}, \quad (32)$$

where $g : \mathbb{R}^n \rightarrow \mathbb{R}$, $g(x) = \sum_{i=1}^n (x_i - 1/2)^2$, and \hat{F} denotes the set of the vertices of $R \cap X$ not in Z . Denoting by \bar{x} a solution of (32), they prove the equivalence between (31) and

$$\min[f(x) - \mu g(x)] \text{ s.t. } x \in R \cap X,$$

when

$$\mu > \mu_R := \frac{\langle c, x_0 \rangle}{n/4 - g(\bar{x})}.$$

Observe that the strictly concave function

$$\psi(x) = \sum_{i=1}^n x_i(1 - x_i) \quad (33)$$

is related to g by the relation

$$\psi(x) = n/4 - g(x).$$

So that the denominator of μ_R is $\psi(\bar{x})$.

The parameter μ_R is now compared with the parameter μ_G introduced in [36] and denoted by μ_0 in [Theorem 3](#), which establishes equivalence between (31) and

$$\min[-\langle c, x \rangle + \mu \psi(x)] \text{ s.t. } x \in R \cap X, \quad (34)$$

for $\mu > \mu_G = \mu_0$.

The following notation is adopted: $q := (1/2, \dots, 1/2) \in \mathbb{R}^n$; $d(x, A)$ denotes the distance between the vector x and the set A of \mathbb{R}^n ; $B(y, r) := \{x \in \mathbb{R}^n : d(x, y) < r\}$; for a subset A of \mathbb{R}^n , $cl A$ and A^c denote, respectively, the closure and the complement of A .

In order to verify the hypotheses of [Theorem 3](#), observe that the objective function $x \mapsto -\langle c, x \rangle$ is a Lipschitz function with constant $L = \|c\|$, and ψ satisfies the hypothesis (H_2) . Indeed, let ρ be a value of $]0, 1[$, and, for every $z_i \in R \cap Z$, let the set $S(z_i)$ be a neighborhood of z_i with $S(z_i) \subseteq B(z_i, \rho)$. Then, ψ satisfies (H_2) for $\alpha = 1$ and $\epsilon = 1 - \rho$. Recall the definition of parameter μ_G :

$$\mu_G := \max \left(\frac{\|c\|}{1 - \rho}, \frac{(\sup_{x \in R \cap X} [-\langle c, x \rangle]) - (\inf_{x \in R \cap X} [-\langle c, x \rangle])}{\min_{x \in X_{0,\rho}} \psi(x)} \right),$$

where

$$X_{0,\rho} := (X \cap R) \setminus \left(\cup_{i=1}^k S(z_i) \right).$$

Thanks to [Theorem 3](#), the equivalence between (31) and (34) holds for any $\mu > \mu_G$. Observe the dependence of μ_G on the radius ρ ; below, to highlight this dependence, the notation $\mu_{G,\rho}$ will be used instead of μ_G .

Before giving sufficient conditions to establish a comparison, observe the following:

Remark 1 The hypothesis (H_2) of [Theorem 3](#) and the continuity of ψ imply that $\rho < 1$.

For the sake of simplicity, put

$$\lambda_{0,\rho} := \frac{(\sup_{x \in R \cap X} [-\langle c, x \rangle]) - (\inf_{x \in R \cap X} [-\langle c, x \rangle])}{\min_{x \in X_{0,\rho}} \psi(x)}.$$

The following result gives sufficient conditions assuring that there exist a $\bar{\rho} > 0$ and a neighborhood $S(z_i) \subseteq B(z_i, \bar{\rho})$ such that

$$\mu_{G,\bar{\rho}} \leq \mu_R.$$

Theorem 5 ([5]) *Let \bar{x} be a solution of (32). Suppose that*

$$d(\bar{x}, q) \geq \frac{\sqrt{n-1}}{2}$$

and

$$\frac{\sup_{x \in R \cap X} \langle c, x \rangle}{\|c\|} \geq \frac{1}{2}. \quad (35)$$

Then, there exist $\bar{\rho} \in]0, 1[$ and, $\forall z_i \in Z$, a neighborhood $S(z_i) \subseteq B(z_i, \bar{\rho})$ such that

$$\mu_{G, \bar{\rho}} \leq \mu_R. \quad (36)$$

Proof Consider the one-dimensional face of X given by

$$F_1 := \{x \in \mathbb{R}^n : x = (x_1, 0, \dots, 0), x_1 \in [0, 1]\}.$$

Since $d(\bar{x}, q) \geq \sqrt{n-1}/2$ and $d(q, F_1) = \sqrt{n-1}/2$, necessarily the intersection $A := \text{cl } B(q, d(\bar{x}, q)) \cap F_1$ is nonempty. Put $\bar{\rho} = d(0, A)$ and denote by x^* the point of A at minimum distance from the origin of \mathbb{R}^n . Finally, let $S(z_i) := X \cap B(z_i, \bar{\rho})$. Since $X_{0, \bar{\rho}} = (X \cap R) \setminus \left(\bigcup_{i=1}^k S(z_i) \right)$, by construction,

$$X_{0, \bar{\rho}} \subseteq \text{cl } B(q, d(q, \bar{x})) = \{x \in \mathbb{R}^n : \psi(x) \geq \psi(\bar{x})\};$$

then one has

$$\min_{x \in X_{0, \bar{\rho}}} \psi(x) \geq \psi(\bar{x}),$$

so

$$\lambda_{0, \bar{\rho}} \leq \frac{(\sup_{x \in R \cap X} [-\langle c, x \rangle]) - (\inf_{x \in R \cap X} [-\langle c, x \rangle])}{\psi(\bar{x})}.$$

Moreover, obviously,

$$\frac{(\sup_{x \in R \cap X} [-\langle c, x \rangle]) - (\inf_{x \in R \cap X} [-\langle c, x \rangle])}{\psi(\bar{x})} \leq -(\inf_{x \in R \cap X} [-\langle c, x \rangle]) \psi(\bar{x}) = \mu_R;$$

thus

$$\lambda_{0, \bar{\rho}} \leq \mu_R. \quad (37)$$

Besides, the inequality

$$\frac{\|c\|}{1 - \bar{\rho}} \leq \mu_R \quad (38)$$

is equivalent to

$$\frac{\psi(\bar{x})}{1 - \bar{\rho}} \leq \frac{\sup_{x \in R \cap X} \langle c, x \rangle}{\|c\|}.$$

The last inequality is true because $\psi(\bar{x}) = \psi(x^*) = x_1^*(1 - x_1^*) = \bar{\rho}(1 - \bar{\rho})$, and thanks to the inequality in (35); indeed, one has

$$\frac{\psi(\bar{x})}{1 - \bar{\rho}} = \bar{\rho} \leq \frac{1}{2} \leq \frac{\sup_{x \in R \cap X} \langle c, x \rangle}{\|c\|}.$$

So (36) follows from (37) and (38) and the theorem is thus proved. \square

2.4 On the Equivalence of Lagrangian Duality and Penalization

Based on results by Larsen and Tind [55], it is shown here that the penalization approach to a constrained optimization problem described in [Theorem 1](#) can be interpreted in terms of Lagrangian duality with zero gap for an associated problem.

More precisely, consider the sets $W \subseteq X \subset \mathbb{R}^n$ and a (penalty) function $g : X \rightarrow \mathbb{R}$ such that $g(x) \geq 0$ for all $x \in X$ and $g(x) = 0$ for all $x \in W$. Then, problem [\(2\)](#) can be equivalently restated as

$$\min f(x) \text{ s.t. } x \in W = \{x \in X, g(x) \leq 0\}. \quad (39)$$

Its Lagrangian dual problem is

$$\sup w(\mu) \text{ s.t. } \mu \in \mathbb{R}_+, \quad (40)$$

where

$$w(\mu) = \min[f(x) + \mu g(x)] \text{ s.t. } x \in X. \quad (41)$$

Then, requiring the existence of a penalization parameter $\bar{\mu}$ such that, for all $\mu > \bar{\mu}$, problems [\(2\)](#) and [\(3\)](#) have the same solutions is equivalent to requiring that problem [\(39\)](#) has no duality gap and, more precisely, that for all $\mu > \bar{\mu}$ problem [\(41\)](#) has the same optimal solutions as [\(39\)](#). Thus any known result for zero duality gap can be immediately rephrased as an exact penalty result and vice versa. As an example, recall the following results by Larsen and Tind [55], rephrased in the current setting, with the notation $B(\bar{x}, \epsilon) := \{x \in \mathbb{R}^n : \| \bar{x} - x \| < \epsilon\}$.

Theorem 6 ([55]) *Let $W \subseteq X$ be nonempty compact sets in \mathbb{R}^n and let f and g be continuous functions on X such that $g(x) \geq 0$ for all $x \in X$ and $g(x) = 0$ for all $x \in W$. Assume that there exist constants $\epsilon > 0$ and $H > 0$ such that $f(\bar{x}) - f(x) \leq H g(x)$ for all solutions \bar{x} of [\(39\)](#) and for all $x \in B(\bar{x}, \epsilon) \cap (X \setminus W)$. Then, there exists $\mu_0 \geq 0$ such that, $\forall \mu \geq \mu_0$,*

$$\min_{x \in W} f(x) = w(\mu).$$

Consider, now, the following facial disjunctive special case of [\(39\)](#):

$$\min f(x) \text{ s.t. } x \in W = \{x \in X : \langle a_1^h, x \rangle \geq b_1^h \vee \langle a_2^h, x \rangle \leq b_2^h, h \in H\}, \quad (42)$$

where $a_1^h, a_2^h \in \mathbb{R}^n \setminus \{0\}$, $b_1^h, b_2^h \in \mathbb{R}$ for all $h \in H$, and H is a finite index set. According to Balas [10], the facial requirement for [\(42\)](#) is

$$(A) \quad \forall x \in X, \quad \langle a_1^h, x \rangle \leq b_1^h \wedge \langle a_2^h, x \rangle \geq b_2^h, \quad \forall h \in H.$$

Indeed, for the function

$$g(x) = \sum_{h \in H} (b_1^h - \langle a_1^h, x \rangle)((\langle a_2^h, x \rangle - b_2^h)), \quad (43)$$

one has $g(x) \geq 0$ for all $x \in X$ and $g(x) = 0$ for all $x \in W$. So the feasible set of (42) has the desired form $W = \{x \in X : g(x) \leq 0\}$ and g satisfies the assumptions of [Theorem 6](#).

On the feasible set W , the following further assumption is made:

$$(B) \quad \exists \delta > 0 \forall \bar{x} \in W \forall h \in H, \frac{b_1^h - \langle a_1^h, \bar{x} \rangle}{\|a_1^h\|} \geq \delta \text{ or } \frac{\langle a_2^h, \bar{x} \rangle - b_2^h}{\|a_2^h\|} \geq \delta.$$

Theorem 7 ([55]) *Let Assumptions (A) and (B) hold for the feasible set of (42) and choose $\rho \in]0, \delta[$. Then, the constraint function g in (43) satisfies, for all $\bar{x} \in W$ and $x \in B(\bar{x}, \rho) \cap (X \setminus W)$, the inequality*

$$g(x) \geq ((\delta - \rho)/\delta^2) \inf \left\{ \sum_{h \in H} |\langle a_1^h, u \rangle| |\langle a_2^h, u \rangle| : u \in U(\delta) \right\} \|x - \bar{x}\|,$$

where $U(\delta) := \{u \in \mathbb{R}^n : u = \delta(x - \bar{x})/\|x - \bar{x}\|, \bar{x} \in W, x \in X \setminus W\}$.

Now, the previous results are applied to the 0–1 problem given by

$$\min f(x) \text{ s.t. } x \in W = R \cap \{0, 1\}^n. \quad (44)$$

Obviously it holds that

$$W = R \cap [0, 1]^n \cap \left\{ \bigcap_{j=1}^n (\{x_j \geq 1\} \cup \{x_j \leq 0\}) \right\}.$$

Putting $X = R \cap [0, 1]^n$, (44) is a facial disjunctive program satisfying the facial requirement (A). Moreover it becomes the facial program

$$\min f(x) \text{ s.t. } x \in W = \{x \in R \cap [0, 1]^n, g(x) \leq 0\} \quad (45)$$

using the realization of g given by

$$g(x) = \sum_{j=1}^n x_j (1 - x_j).$$

Therefore the Lagrangian dual of (45) is

$$\sup_{M \geq 0} \min_{x \in R \cap [0, 1]^n} \left[f(x) + M \sum_{j=1}^n x_j (1 - x_j) \right]. \quad (46)$$

The next proposition, first proved in [36], can be stated as a direct application of [Theorem 6](#).

Theorem 8 ([55]) *Assume R and W are, respectively, a closed and a nonempty set of \mathbb{R}^n ; assume f is a Lipschitz continuous function on $R \cap [0, 1]^n$. Then the duality gap between (45) and (46) is closed with a finite dual multiplier.*

Proof Problem (44) is a facial disjunctive problem; Assumption (A) holds with $X = R \cap [0, 1]^n$; Assumption (B) holds with $\delta \leq 1$; finally, for all $u \in U(\delta)$, $\sum_{h \in H} |\langle a_1^h, u \rangle| |\langle a_2^h, u \rangle| = \delta^2$. Thus, [Theorem 7](#) implies the inequality $g(x) \geq (\delta - \rho) \|x - \bar{x}\|$ for all $x \in B(\bar{x}, \rho) \cap (X \setminus W)$ and $\bar{x} \in W$. Therefore [Theorem 6](#) can be applied. \square

3 Generalized Concavity and Extreme Point Minimizers

3.1 Global Minimizers at Extreme Points

In many cases it is possible to restrict the search for the global solution of a nonlinear program to a finite set of points. One way of doing this consists in considering only the set of points that satisfy some kind of first- or second-order necessary condition for local optimality. This set is often finite, but in general it is not practical to compute all its elements or to minimize the objective function over it. However, there are some interesting cases where this simple idea leads to useful connections between nonlinear and combinatorial problems.

This section describes another important case where the search for a solution of a nonlinear program can be restricted to a finite set, namely, when at least one global minimizer is guaranteed to be in the set of extreme points of a polyhedral feasible region.

The following definitions of extreme point and of concavity of a function in the general setting on nonconvex domains will be needed in the sequel.

Definition 1 (Extreme Point) Let X be a subset of \mathbb{R}^n . A point $x \in X$ is called an *extreme point* of X iff $y, z \in X$, $\alpha \in]0, 1[$, and $x = \alpha y + (1 - \alpha)z$ imply $y = z$. A point $x \in X$ is called a *convex hull extreme point* of X iff it is an extreme point of the *convex hull* (denoted by $\text{conv}(X)$) of X .

Note that the set of extreme points of a polyhedron is finite and coincides with the set of its vertices. Let $\mathcal{E}(X)$ denote the set of extreme points of a set X . Then the set of convex hull extreme points is given by $\mathcal{E}(\text{conv}(X))$. Since $X \subseteq \text{conv}(X)$, it follows from the above definition that $\mathcal{E}(\text{conv}(X)) \subseteq \mathcal{E}(X) \subseteq X$.

It is well known (see, e.g., [66]) that the convex hull of a compact set is compact. Furthermore, the Krein–Millman Theorem establishes the equality $X = \text{conv}(\mathcal{E}(X))$ for a nonempty convex compact set X . From this it easily follows that a nonempty compact set X has a nonempty set of convex hull extreme points and,

furthermore, that every extreme point of X is in the convex hull of the convex hull extreme points of X .

Proposition 3 *For a nonempty compact set X , one has*

$$\mathcal{E}(\text{conv}(X)) \neq \emptyset \quad \text{and} \quad \text{conv}(\mathcal{E}(\text{conv}(X))) = \text{conv}(\mathcal{E}(X)).$$

Proof The results follow from Krein–Millman Theorem and from the inclusions

$$\text{conv}(X) = \text{conv}(\mathcal{E}(\text{conv}(X))) \subseteq \text{conv}(\mathcal{E}(X)) \subseteq \text{conv}(X).$$

□

Definition 2 (Concavity on Nonconvex Sets) Let X be a subset of \mathbb{R}^n . A function $f : X \rightarrow \mathbb{R}$ is *concave* (resp., *quasi-concave*) on X iff, for every $x \in X$ and for every set of points $\{x^1, \dots, x^m\} \subseteq X$ and of coefficients $\alpha_1, \dots, \alpha_m \in]0, 1[$ such that $\sum_{i=1}^m \alpha_i = 1$ and $x = \sum_{i=1}^m \alpha_i x^i$, one has

$$f(x) \geq \sum_{i=1}^m \alpha_i f(x^i) \quad (\text{resp. } f(x) \geq \min_i f(x^i)). \quad (47)$$

A function is called *strictly concave* or *strictly quasi-concave* iff the above relations are satisfied with a strict inequality whenever $x^i \neq x$ for at least one index i .

Note that if the set X coincides with $\mathcal{E}(\text{conv}(X))$ (like, e.g., in the case of a circle), then every function is strictly concave on X with the above definition. However, when X is convex, **Definition 2** is equivalent to the standard definition of concavity and of strict concavity of a function.

Several conditions for the validity of the following property will be investigated in this section.

Property E

At least one (or all) global minimizer of f on a set X belongs to $\mathcal{E}(X)$.

The Fundamental Theorem of linear programming is the most popular result that guarantees the validity of Property E.

Theorem 9 (Fundamental Theorem of LP) *If a linear function f is bounded below on a polyhedron X , then at least one face of X of minimum dimension must be formed by global minimum points of f . In particular, whenever $\mathcal{E}(X) \neq \emptyset$, at least one global minimum point of f belongs to $\mathcal{E}(X)$.*

The Fundamental Theorem of LP establishes an important bridge between a (continuous) Linear Programming problem and many important combinatorial problems that can be viewed as problems of minimizing a linear function on the vertices of some polyhedron.

It is well known that Property E also holds for classes of feasible sets X and of objective functions f that are considerably larger than the classes of polyhedra and of linear functions. Notable examples are the cases where f is concave and X is a closed convex set, or where f is quasi-concave and X is a compact convex set (see, e.g., [45, 77]). Note, however, that Property E does not hold in general for quasi-concave functions on *unbounded* closed convex sets. A simple counterexample is provided by the quasi-concave (and convex) function $f : X = [0, +\infty[\rightarrow \mathbb{R}$ defined by $f(x) = -x$, for $x \in [0, 1]$, and $f(x) = -1$, for $x \in]1, +\infty[$.

3.2 The Case of Nonconvex Domains

It will now be shown that Property E still holds (and in a stronger form), for the minimization of concave or quasi-concave functions on nonconvex domains.

Definition 3 (Concave Envelope) Let X be a subset of \mathbb{R}^n . The *concave envelope* (or *concave hull*) of a function $f : X \rightarrow \mathbb{R}$, denoted f^{cv} , is the smallest concave function on $\text{conv}(X)$ which is minorized by f on X . The *convex envelope* (or *convex hull*) of f , denoted f^{cx} , is defined as the greatest convex function on $\text{conv}(X)$ which is majorized by f on X .

The convex and concave envelopes of f can be obtained as follows (see also [66]):

$$\begin{aligned} f^{cx}(x) &:= \inf\{y : (x, y) \in \text{conv}(\text{Epi } f)\}, \\ f^{cv}(x) &:= \sup\{y : (x, y) \in \text{conv}(\text{Hypo } f)\}, \end{aligned} \quad (48)$$

where $\text{Epi } f = \{(x, y) \in X \times \mathbb{R} : y \geq f(x)\}$ and $\text{Hypo } f = \{(x, y) \in X \times \mathbb{R} : y \leq f(x)\}$ denote the epigraph and the hypograph of f , respectively. The convex envelope of f satisfies the following properties (see, e.g., [52, 62]):

$$\inf_{x \in X} f(x) = \inf_{x \in \text{conv}(X)} f^{cx}(x) \quad \text{and} \quad \arg \min_{x \in X} f(x) \subseteq \arg \min_{x \in \text{conv}(X)} f^{cx}(x), \quad (49)$$

where $\arg \min_{x \in S} g(x)$ denotes the set of global minimum points of a function g on a set S . Thanks to properties (49), convex envelopes have been employed by several authors to develop algorithms for the minimization of some classes of nonconvex functions (see [19, Sect. 3.2] for a brief survey of these approaches).

When X is any subset of \mathbb{R}^n and f is a concave function on X , the *concave envelope* $f^{cv}(x)$ satisfies the same properties as the convex envelope.

Theorem 10 *Let X be a subset of \mathbb{R}^n and let $f : X \rightarrow \mathbb{R}$ be any real-valued function on X . Then*

$$f^{cv}(x) = f^{cx}(x) = f(x), \quad \forall x \in \mathcal{E}(\text{conv}(X)).$$

Furthermore, if f is concave on X , then

$$\begin{aligned} f^{\text{cv}}(x) &= f(x), \quad \forall x \in X, \\ \inf_{x \in X} f(x) &= \inf_{x \in \text{conv}(X)} f^{\text{cv}}(x), \\ \arg \min_{x \in X} f(x) &\subseteq \arg \min_{x \in \text{conv}(X)} f^{\text{cv}}(x). \end{aligned} \tag{50}$$

Proof Let $x \in \mathcal{E}(\text{conv}(X))$. Then, as observed earlier, $x \in X$. Furthermore, $f^{\text{cv}}(x) \leq f(x)$ by definition of f^{cv} . Assume, ab absurdo, that $f^{\text{cv}}(x) < f(x)$. Then, by (48), there exists a set of points $\{x^1, \dots, x^m\} \subseteq X$ and coefficients $\alpha_1, \dots, \alpha_m \in]0, 1[$ such that $\sum_{i=1}^m \alpha_i = 1$, $\sum_{i=1}^m \alpha_i x^i = x$, and $\sum_{i=1}^m \alpha_i f(x^i) < f(x)$. However, this contradicts the fact that x is a convex hull extreme point of X , which implies $x^1 = x^2 = \dots = x^m = x$. The proof of the equality $f^{\text{cv}}(x) = f(x)$, $\forall x \in \mathcal{E}(\text{conv}(X))$, is analogous.

Assume now that f is concave, and take any $x \in X$. From the concavity of f it follows that $f(x) \geq \sum_{i=1}^m \alpha_i f(x^i)$ for every set of points $\{x^1, \dots, x^m\} \subseteq X$ and coefficients $\alpha_1, \dots, \alpha_m \in]0, 1[$ such that $\sum_{i=1}^m \alpha_i = 1$ and $\sum_{i=1}^m \alpha_i x^i = x$. This clearly implies $f(x) \geq y$ for every y such that $(x, y) \in \text{conv}(\text{Hypo } f)$ and hence $f(x) \geq f^{\text{cv}}(x)$ by (48). Since $f(x) \leq f^{\text{cv}}(x)$ by definition, the equality $f(x) = f^{\text{cv}}(x)$ follows.

Let $m = \inf_{x \in X} f(x)$. If $m = -\infty$, then clearly $\inf_{x \in \text{conv}(X)} f^{\text{cv}}(x) = -\infty$ since $f(x) = f^{\text{cv}}(x)$ on X . If $m > -\infty$, consider the constant function $h(x) = m$ defined on X and note that $h^{\text{cv}}(x) = m$ for every $x \in \text{conv}(X)$. Furthermore, since $f(x) \geq h(x)$ for every $x \in X$, one has $f^{\text{cv}}(x) \geq h^{\text{cv}}(x)$ on $\text{conv}(X)$. Hence, it follows that $\inf_{x \in \text{conv}(X)} f^{\text{cv}}(x) = m = \inf_{x \in X} f(x)$.

The inclusion $\arg \min_{x \in X} f(x) \subseteq \arg \min_{x \in \text{conv}(X)} f^{\text{cv}}(x)$ is a trivial consequence of the equality $\inf_{x \in \text{conv}(X)} f^{\text{cv}}(x) = \inf_{x \in X} f(x)$ and of the coincidence of f and f^{cv} on X . \square

It is well known [45] that a concave function that achieves its minimum on a closed convex set X that contains no line has at least one global minimum point among the extreme points of X . This result can be extended to the case where X is not convex.

Theorem 11 *Let $X \subseteq \mathbb{R}^n$ and $f : X \rightarrow \mathbb{R}$.*

- (i) *If either $\text{conv}(X)$ is closed and contains no line and f is concave on X , or X is compact and f is quasi-concave on X , then*

$$\arg \min_{x \in X} f(x) \neq \emptyset \Leftrightarrow \arg \min_{x \in X} f(x) \cap \mathcal{E}(\text{conv}(X)) \neq \emptyset. \tag{51}$$

- (ii) *If either $\text{conv}(X)$ is closed and contains no line and f is strictly concave on X , or X is compact and f is strictly quasi-concave on X , then*

$$\arg \min_{x \in X} f(x) \subseteq \mathcal{E}(\text{conv}(X)). \tag{52}$$

Proof Note that, by Theorem 10, the concave envelope f^{cv} coincides with f on X and the minimum of f^{cv} on $\text{conv}(X)$ coincides with the minimum

f on X . Observe also that if X is compact, then $\text{conv}(X)$ is closed. Let x be any point in $X \subseteq \text{conv}(X)$. Then, by Theorem 18.5 in [66], x can be expressed as a convex combination of extreme points and extreme directions of $\text{conv}(X)$. Hence, there exist $y \in \mathcal{E}(\text{conv}(X))$, $z \in \mathbb{R}^n$, and $\lambda \geq 0$ such that $x = y + \lambda z$ and $y + \lambda z \in \text{conv}(X)$ for all $\lambda \geq 0$. Thus, taking $\beta = \frac{\lambda}{\lambda+1}$, one has $y + (\lambda+1)z \in \text{conv}(X)$ and $x = (1-\beta)y + \beta(y + (\lambda+1)z)$. Since $y \in \mathcal{E}(\text{conv}(X))$ and $y + (\lambda+1)z \in \text{conv}(X)$, this implies that x can be expressed as a convex combination of points $y^1, \dots, y^h \in \mathcal{E}(\text{conv}(X))$ and $w^1, \dots, w^k \in \text{conv}(X)$ with $h \geq 1$ and $k \geq 0$ ($k = 0$ if X is compact). Assume now that x is a global minimum point for f on X . Then:

- (i) If f is concave on X , then $f(y^1) = \dots = f(y^h) = f(w^1) = \dots = f(w^k) = f(x)$ and if f is quasi-concave on X , then $f(x) = \min\{f(y^1), \dots, f(y^h)\}$. Hence, at least one point in $\mathcal{E}(\text{conv}(X))$ is a global minimum point for f on X .
- (ii) If f is strictly concave or strictly quasi-concave on X , then one must have $h = 1$ and $k = 0$, i.e., x must belong to $\mathcal{E}(\text{conv}(X))$; otherwise a contradiction would arise. \square

3.3 The Case of Polyhedral Domains

In the remainder of this section, the attention is restricted to the minimization of a function on a polyhedron, and in order to avoid trivial cases, it is also assumed that all (unbounded) polyhedra contain no line and thus are *pointed*, i.e., they have at least one vertex.

It is easily seen that the class of quasi-concave functions is the most general class of functions for which Property E holds for *every* bounded polyhedron S . Indeed, by applying Property E to the special polyhedron that coincides with the line segment joining two points x^1 and x^2 , one trivially obtains the inequality defining quasi-concave functions.

For a given polyhedron, or class of polyhedra, it is however possible to find classes of functions that properly include quasi-concave functions and satisfy Property E. Several results of this type that have been obtained by Tardella [72, 73, 75, 76] and by Hwang and Rothblum [47] are described next.

One first needs to introduce some definitions and notation. The *affine hull* of a subset S of \mathbb{R}^n is the smallest affine manifold containing S . $\text{tng}(S)$ denotes the smallest linear subspace of \mathbb{R}^n containing $S - \{x^0\} = \{x - x^0 : x \in S\}$, where x^0 is any point of S . Note that $\text{tng}(S)$ coincides with the set $\{\alpha(x - y) : x, y \in S, \alpha \in \mathbb{R}\}$ and can also be viewed as the linear subspace obtained by translating in 0 the affine hull of S . The *relative interior* $\text{rint}(S)$ and the *relative boundary* $\text{rbd}(S)$ of S are the interior and the boundary of S for the topology relative to the affine hull of S , respectively.

Given a polyhedron P , a point x in P , and a direction $d \in \mathbb{R}^n \setminus \{0\}$, let F_x denote the face of P of minimum dimension containing x , and consider the set

$$P_d(x) := \{z \in P : z = x + \lambda d, \lambda \in \mathbb{R}\},$$

obtained intersecting P with the line passing through x with direction d . Since P is pointed, $P_d(x)$ is either a segment or a half line for every x and d .

Consider the following conditions:

$$P_d(x) \text{ is bounded and } f \text{ is quasi-concave on } P_d(x). \quad (53)$$

$$f \text{ is concave and bounded below on } P_d(x). \quad (54)$$

$$f \text{ is strictly quasi-concave on } P_d(x). \quad (55)$$

A face F of P is said to satisfy *Property A*, *B*, or *C*, if for every $x \in \text{rint } F$ there exists $d \in \text{tng}(F) \setminus \{0\}$, such that:

Property A: Condition (53) or (54) or (55) holds.

Property B: Condition (53) or (54) holds.

Property C: Condition (55) holds.

Let \mathcal{F}_A , \mathcal{F}_B , and \mathcal{F}_C denote the sets of all faces of P that *do not* satisfy Properties *A*, *B*, and *C*, respectively. By definition, each set \mathcal{F}_A , \mathcal{F}_B , and \mathcal{F}_C contains all the vertices V_P of P . Furthermore, if f is concave and bounded below on P , or if P is bounded and f is quasi-concave on P , then \mathcal{F}_A and \mathcal{F}_B contain *only* the vertices of P . If f is strictly quasi-concave on P , then \mathcal{F}_A and \mathcal{F}_C contain *only* the vertices of P . Note also that $\mathcal{F}_A \subseteq \mathcal{F}_B$ and $\mathcal{F}_A \subseteq \mathcal{F}_C$.

The following lemma shows that the faces in \mathcal{F}_B contain all the best candidate points for minimizing f over P .

Lemma 1 *If $x \in P \setminus \bigcup_{F \in \mathcal{F}_B} F$, then there exists $x' \in \bigcup_{F \in \mathcal{F}_B} F$ such that $f(x') \leq f(x)$. Furthermore, if T_1 is the time required to find a direction d satisfying condition (53) or (54) in a face $F \notin \mathcal{F}_B$, and T_2 is the time required to find the extreme points of $P_d(x)$ for x in F , then x' can be determined in $O(n(T_1 + T_2))$ time.*

Proof Let $x \in P \setminus \bigcup_{F \in \mathcal{F}_B} F$ and let F_x be the face of P of minimum dimension containing x . Then $x \in \text{rint } F_x$, and thus one can find, in T_1 time, a direction $d \in \text{tng}(F) \setminus \{0\}$ such that (53) or (54) holds. Note that, since $x \in \text{rint } F_x$, one has $x \in \text{rint } P_d(x)$. If $P_d(x)$ is bounded and y, z are its extreme points, then $y, z \in \text{rbd } F_x$ and $f(x) \geq \{f(y), f(z)\}$ by quasi-concavity of f on $P_d(x)$. On the other hand, if $P_d(x)$ is unbounded and y is its only extreme point, then $y \in \text{rbd } F_x$ and $f(x) \geq f(y)$. Indeed, if $f(x) < f(y)$ and f is concave on $P_d(x)$, then f must be unbounded on $P_d(x)$ contradicting (54). Thus, in both cases one can find, in T_2 time, a point $x^1 \in \text{rbd } F$ satisfying $f(x^1) \leq f(x)$. If $x \in \bigcup_{F \in \mathcal{F}_B} F$, then set $x' = x^1$ and stop. Otherwise, consider the face F_{x^1} of P of minimum dimension containing x^1 , and iterate. Since $\dim(F_{x^1}) < \dim(F_x) \leq n$ and $V_P \subseteq \mathcal{F}_B$, after at most n iterations, one must find a point $x' \in \bigcup_{F \in \mathcal{F}_B} F$ satisfying $f(x') \leq f(x)$. \square

The next theorem shows that the infimum of f on P coincides with the infimum over the union of the faces in \mathcal{F}_B , and if one infimum is attained, the other is also attained. Furthermore, if the infimum is attained, at least one global minimizer of f on P must belong to a face in \mathcal{F}_A . Finally, the set of all global minimizers of f on P must be contained in the union of the faces in \mathcal{F}_C .

Theorem 12 (Existence and Location of Minima)

$$\inf_{x \in P} f(x) = \min_{F \in \mathcal{F}_B} \inf_{x \in F} f(x), \quad (56)$$

where the first infimum is attained if and only if the second infimum is attained. If the infimum is attained, then

$$\min_{x \in P} f(x) = \min_{F \in \mathcal{F}_A} \min_{x \in F} f(x) \quad (57)$$

and

$$\arg \min_{x \in P} f(x) \subseteq \bigcup_{F \in \mathcal{F}_C} F. \quad (58)$$

Proof The inequality $\inf_{x \in P} f(x) \leq \min_{F \in \mathcal{F}_B} \inf_{x \in F} f(x)$ is trivial. The converse inequality follows easily from Lemma 1. Furthermore, if f attains its minimum at a point $x \in P$, then by Lemma 1 it also attains its minimum at a point $x' \in \bigcup_{F \in \mathcal{F}_B} F$.

Assume now that all global minimum points of f on P belong to $\bigcup_{F \in \mathcal{F}_B} F \setminus \bigcup_{F \in \mathcal{F}_A} F$, and let F' be a face of minimum dimension in $\mathcal{F}_B \setminus \mathcal{F}_A$ containing a minimum point \bar{x} for f on P . Since $F' \in \mathcal{F}_B \setminus \mathcal{F}_A$, condition (55) must hold at \bar{x} . Let $d \in \text{tng}(F') \setminus \{0\}$ be such that f is strictly quasi-concave on $P_d(\bar{x})$. Then \bar{x} must be an extreme point of $P_d(\bar{x})$. Otherwise, \bar{x} could be obtained as a convex combination of two points $x^1, x^2 \in P_d(\bar{x})$ so that $f(\bar{x}) > \min\{f(x^1), f(x^2)\}$, contradicting the minimality of \bar{x} . On the other hand, if \bar{x} is an extreme point of $P_d(\bar{x})$, then $\bar{x} \in \text{rbd } F'$, contradicting the minimality of the dimension of F' . Hence one must have $\bar{x} \in \bigcup_{F \in \mathcal{F}_A} F$, so that (57) holds.

Let x be a global minimum point of f on P and assume that $x \notin \bigcup_{F \in \mathcal{F}_C} F$. Let F_x be the face of P of minimum dimension containing x . Then $x \in \text{rint } F_x$ and hence $x \in \text{rint } P_d(x)$ for every $d \in \text{tng}(F_x) \setminus \{0\}$. Since $F_x \notin \mathcal{F}_C$, there is a direction $d \in \text{tng}(F_x) \setminus \{0\}$ such that f is strictly quasi-concave on $P_d(x)$. Furthermore, since $x \in \text{rint } P_d(x)$, there exist points $x^1, x^2 \in P_d(x)$ such that $x = \frac{1}{2}(x^1 + x^2)$. Hence, by strict quasi-concavity, $f(x) > f(x^1) + f(x^2)$, contradicting the minimality of $f(x)$. \square

A further extension of the class of functions that attain their minima on P at a point in $\bigcup_{F \in \mathcal{F}_A} F$ (and in particular at a vertex of P , when $\bigcup_{F \in \mathcal{F}_A} F = V_P$) can be obtained with the following simple result.

Theorem 13 Let $f, g : X \rightarrow \mathbb{R}$ and $Y \subseteq X$ satisfy $f(x) \leq g(x)$ for all x in X , and $f(x) = g(x)$ for all x in Y . Then, if f attains its minimum on X at a point of Y , also g attains its minimum on X at a point of Y .

Proof

$$\min_{x \in Y} f(x) = \min_{x \in Y} g(x) \geq \inf_{x \in X} g(x) \geq \min_{x \in X} f(x) = \min_{x \in Y} f(x). \quad \square$$

[Theorem 12](#) can be used to decompose the problem of minimizing a function f on a polyhedron P into two subproblems:

$$\varphi(F) = \min_{x \in F} f(x) \quad \text{and} \quad \min_{F \in \mathcal{F}_A} \varphi(F). \quad (59)$$

The first subproblem is similar to the main problem of minimizing f on P . However, such subproblem could be easier to solve due to additional structure of f on some faces of P (like, e.g., convexity). On the other hand, the second subproblem has a combinatorial nature. In [Sects. 3.4](#) and [3.6](#), it is shown that, in some interesting cases, one or both subproblems can be solved efficiently, thereby providing efficient (polynomial) methods for some classes of problems, or polynomial time reductions from a continuous formulation of a problem to a combinatorial one and vice versa.

Equality [\(57\)](#) can also be fruitfully applied in some cases to solve, exactly or approximately, the combinatorial problem $\min_{F \in \mathcal{F}_A} \varphi(F)$ by means of the continuous problem $\min_{x \in P} f(x)$. This approach will be used in [Sects. 3.4](#) and [3.6](#) to obtain continuous formulations for the maximum independent set problem, for the maximum clique problem, and for other combinatorial problems.

Besides the Fundamental Theorem of Linear Programming, [Theorem 12](#) extends several well-known results, which can be easily derived from it. A first example is the following existence result in indefinite Quadratic Programming due to Frank and Wolfe [\[30\]](#), for which a number of (more complex) alternative proofs and extensions are already available [\[12, 27, 50, 58, 63\]](#).

Theorem 14 ([30]) If f is a quadratic function that is bounded below on P , then f attains its minimum on P .

Indeed, from [Theorem 12](#), one easily derives the following common extension of the Frank–Wolfe Theorem and of the Fundamental Theorem of LP:

Theorem 15 If f is a quadratic function that is bounded below on P , then f attains its minimum on P at a point contained in a face of P where f is strictly convex.

Proof Note that, in this case, the family of faces \mathcal{F}_B defined above coincides with the family of faces of P where f is strictly convex. Furthermore, by coercivity, the

minimum of a strictly convex quadratic function on a closed set always exists. The conclusion then follows from [Theorem 12](#). \square

Note that when f is linear, the only faces of P where f is strictly convex are the vertices of P . Thus the Fundamental Theorem of LP is a special case of [Theorem 15](#).

One difficulty with the direct application of [Theorem 12](#) is due to the hardness of the determination of the classes \mathcal{F}_A , \mathcal{F}_B , and \mathcal{F}_C in general. However, in many cases it is sufficient to use some superclasses of \mathcal{F}_A , \mathcal{F}_B , and \mathcal{F}_C that can be determined more efficiently by using only the directions parallel to the edges of P , as described below.

Let D^B and D^U be two sets of vectors parallel to all the bounded edges and to all the extreme rays (unbounded edges) of P , respectively, and let $S^B \subseteq D^B$, $S^U \subseteq D^U$ and $S \subseteq D^B \cup D^U$.

Lemma 2 *Assume that, for every $x \in P$, f is quasi-concave on $P_d(x)$ for all $d \in S^B$ and concave or strictly quasi-concave on $P_d(x)$ for all $d \in S^U$. Then \mathcal{F}_A is contained in the set of faces of P that do not have any edge parallel to a vector in $S^B \cup S^U$. Furthermore, if, for every $x \in P$, f is strictly quasi-concave on $P_d(x)$ for all $d \in S$, then \mathcal{F}_C is contained in the set of faces of P that do not have any edge parallel to a vector in S .*

Proof The proof follows from the definition of \mathcal{F}_A and \mathcal{F}_C , and from the fact that for every face F of P and every vector d parallel to an edge of F (bounded or unbounded) one has $d \in \text{tng}(F) \setminus \{0\}$. \square

From [Lemma 2](#) and [Theorem 12](#) one can immediately deduce the following extensions of the Fundamental Theorem of Linear Programming.

Theorem 16 *Assume that, for every x in P , f is quasi-concave on $P_d(x)$ for all d in S^B and concave or strictly quasi-concave on $P_d(x)$ for all d in S^U . Then, if f attains its minimum on P , at least one global minimizer belongs to a face that does not have any edge parallel to a vector in $S^B \cup S^U$. Furthermore, if, for every x in P , f is strictly quasi-concave on $P_d(x)$ for all d in S , then all global minimizers belong to a face that does not have any edge parallel to a vector in S .*

Theorem 17 *Assume that, for every x in P , f is quasi-concave on $P_d(x)$ for all d in D^B :*

- (i) *If f is concave and bounded below on $P_d(x)$ for all x in P and d in D^U , then f attains its minimum on P at a vertex of P .*
- (ii) *If f attains its minimum on P , and f is concave or strictly quasi-concave on $P_d(x)$ for all x in P and d in D^U , then at least one global minimizer of f on P is in a vertex of P .*

Furthermore, if, for every x in P , f is strictly quasi-concave on $P_d(x)$ for all d in $D^B \cup D^U$, then every global minimizer of f on P is in a vertex of P .

Clearly, in order to apply [Theorems 16](#) and [17](#), one must have a list of vectors d parallel to some or all the edges of P . Furthermore, one should be able to check if f is concave or (strictly) quasi-concave on $P_d(x)$ for all x in P .

When the polyhedron P is described by a system of m inequalities in n variables, and both m and n are not too large, a list of vectors parallel to all its edges can be obtained computationally in a reasonable time (see, e.g., [8, 32] and references therein). On the other hand, if the polyhedron has a special structure, a list of vectors parallel to all its edges may be readily available or could be obtained by means of theoretical results. The next sections are concerned with optimization problems over some polyhedra for which the vectors parallel to the edges are known: rectangles, simplices, and polymatroids.

In the case where the function f is twice continuously differentiable, and, in particular, when it is a quadratic function, some simple necessary and sufficient conditions for its concavity on the sets $P_d(x)$ are described in the following Remark.

Remark 2 When f is twice continuously differentiable, the concavity of f on the sets $P_d(x)$ for all $x \in P$ is equivalent to $d^T H_f(x)d \leq 0$ for all $x \in P$, where $H_f(x)$ is the Hessian matrix of f at x . Furthermore, if $f(x) = \frac{1}{2}x^T Cx + q^T x$, then the (strict) quasi-concavity of f on $P_d(x)$ for all $x \in P$ is equivalent to requiring that at least one of the following conditions hold:

- (i) $d^T Cd \leq 0 (< 0)$.
- (ii) $\min_{x \in P} (Cx + q)^T d \geq 0 (> 0)$.
- (iii) $\max_{x \in P} (Cx + q)^T d \leq 0 (< 0)$.

The equivalence is justified by the property (see, e.g., [9]) that a function is (strictly) quasi-concave on a line if and only if it is either (strictly) monotone, or nondecreasing (increasing) up to a certain point and then nonincreasing (decreasing) on the half line starting that point.

When the edge directions of P are not explicitly known, one can still guarantee the validity of Property E by making some more restrictive assumptions on f .

Definition 4 Let f be a real function defined on a convex set $X \subset \mathbb{R}^n$ and let $m \leq n$. f is called *m-concave* (*m-quasi-concave*) on X iff

$$\begin{aligned} f(\alpha x^1 + (1 - \alpha)x^2)) &\geq \alpha f(x^1) + (1 - \alpha)f(x^2) \\ (\text{resp. } f(\alpha x^1 + (1 - \alpha)x^2)) &\geq \min\{f(x^1), f(x^2)\}, \end{aligned}$$

for every $\alpha \in [0, 1]$ and for every $x^1, x^2 \in X$ such that $x_i^1 = x_i^2$ for at least $n - m$ indices i in $\{1, \dots, n\}$.

Assume now that the polyhedron P is expressed in one of the following ways:

$$\begin{aligned} P &= \{x \in \mathbb{R}^n : Ax = b, \ell \leq x \leq u\} \\ \text{or} \\ P &= \{x \in \mathbb{R}^n : Ax \leq b, \ell \leq x \leq u\}, \end{aligned} \tag{60}$$

where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, \ell = (\ell_1, \dots, \ell_n), u = (u_1, \dots, u_n)$, $-\infty \leq \ell_i < u_i \leq +\infty$ and $\min\{|\ell_i|, |u_i|\} < +\infty$. Note that the assumptions on ℓ and u imply that, if $P \neq \emptyset$, then P has at least one vertex.

Theorem 18 *Let P be a polyhedron defined by (60) and assume that A has rank $m - 1$. If f is m -concave on P or f is m -quasi-concave on P and P is bounded, then Property E holds.*

Proof Let $s = (s_1, \dots, s_n)$ be any vector parallel to an edge of P . Observe that every edge of P lies in the intersection of $n - 1$ linearly independent hyperplanes taken from among those defining P in (60). Since $\text{rank}(A) = m - 1$, one has $s_i = 0$ for at least $n - m$ indices i . From the m -(quasi)-concavity of f one then derives that f is m -(quasi)-concave on the sets $P_s(x)$ for every $x \in P$. Hence, the conclusion follows from Theorem 17. \square

Some results concerning the location of global minimizers for strictly quasi-convex differentiable functions on a polytope are described next. From these results, one obtains new conditions for the existence of global minimizers at the vertices. It is interesting to observe that these conditions are obtained under assumptions of (quasi) convexity of f rather than concavity.

First, recall the following two results by Scozzari and Tardella [70] that establish the existence of a sequence of nested faces with interior minimizers for this type of problems.

Theorem 19 *Let f be a strictly quasi-convex differentiable function that attains its minimum on a polytope P at an interior point x^* of P . Then there exists a facet F of P such that the minimum of f on F is attained at a point in its relative interior.*

Corollary 1 *Let f be a strictly quasi-convex differentiable function that attains its minimum on a polytope P at an interior point x^* of P . Then there exists a nested sequence of faces $F^1 \subset F^2 \subset \dots \subset F^n$ such that $F^n = P$, $\dim(F^i) = i$, and $x_{F^i}^* \in \text{rint}(F^i)$ for $i = 1, \dots, n$.*

Using this result, one can easily derive conditions on the location of global minimizers and, in particular, conditions for the existence of vertex minimizers.

Theorem 20 *Let f be a strictly quasi-convex differentiable function, and let \mathcal{G} be a family of faces of P such that:*

- (i) *For every face F of P that is not strictly contained in a face of \mathcal{G} , and for every nested sequence of faces $F^1 \subset F^2 \subset \dots \subset F^k = F$ with $\dim(F^i) = i$, there is an index i for which $F^i \in \mathcal{G}$.*
- (ii) *For every face $G \in \mathcal{G}$, the restriction of f to G does not attain its minimum at a point in the relative interior of G .*

Then the global minimum of f over P is attained in the relative interior of a face that is strictly contained in a face of \mathcal{G} .

Proof Let x be the (unique) global minimizer of f over P . Clearly, x is in the relative interior of F_x , the smallest face of P containing x . Furthermore, by Corollary 1, there exists a nested sequence of faces $F^1 \subset F^2 \subset \dots \subset F^k$ such that $F^k = F_x$, $\dim(F^i) = i$, and the restriction of f to each F^i attains its minimum in the relative interior of F^i . Thus, by Assumptions (i) and (ii), F_x must be strictly contained in a face of \mathcal{G} . \square

When \mathcal{G} is the set of faces of dimension k , Assumption (i) in the previous theorem holds trivially. Thus one obtains the following consequence:

Corollary 2 *Let f be a strictly quasi-convex differentiable function. If the restriction of f to every face of dimension k does not attain its minimum in the relative interior of such face, then the global minimum of f on P is attained in a face of dimension less than k .*

In particular, for $k = 1$, vertex optimality follows.

Corollary 3 *Let f be a strictly quasi-convex differentiable function. If the restriction of f to every edge of P is minimized at one of its endpoints, then the global minimum of f on P is attained at a vertex of P .*

3.4 Pseudo-Boolean Optimization and Optimization with Box Constraints

A simple class of polyhedra to which Theorems 16 and 17 can be easily applied are the (generalized) rectangles (or *box constraints*) $R = [\ell, u] = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$, where one allows the values $\ell_i = -\infty$ or $u_i = +\infty$ (but not $\ell_i = -\infty$ and $u_i = +\infty$). In this case $D^B \cup D^U = \{e^1, \dots, e^n\}$. Furthermore, a function f is concave or (strictly) quasi-concave on $R_{e^i}(x)$ for all x in R , if and only if the function $f_i(t) = f(x + te^i)$ is concave or (strictly) quasi-concave on the interval $[\ell_i - x_i, u_i - x_i]$, respectively. This is equivalent to requiring that f is concave or (strictly) quasi-concave with respect to each variable. In particular, if f is twice continuously differentiable, then f is (strictly) quasi-concave on $R_{e^i}(x)$ if and only if there exists $\bar{t} \in [\ell_i - x_i, u_i - x_i]$ such that $f'_i(t) \geq 0$ (> 0) for all $t < \bar{t}$, and $f'_i(t) \leq 0$ (< 0) for all $t > \bar{t}$ (see, e.g., [9]). Furthermore, f is concave on $R_{e^i}(x)$ for all x in R if and only if the second partial derivative $f_{x_i x_i}(x)$ is nonpositive for every x in R .

When $f(x) = \frac{1}{2}x^T C x + q^T x$ is a quadratic function, these conditions can be stated as follows:

f is (strictly) concave on $R_{e^i}(x)$ for all $x \in R \Leftrightarrow c_{ii} \leq 0$ (< 0).

f is (strictly) quasi-concave on $R_{e^i}(x)$ for all $x \in R \Leftrightarrow c_{ii} \leq 0$ (< 0).

$\min_{x \in R} (C_i^T x + q_i) \geq 0$ (> 0), or $\max_{x \in R} (C_i^T x + q_i) \leq 0$ (< 0).

where C_i is the i th column vector of the matrix C . Note that

$$\min_{x \in R} (C_i^T x + q_i) = C_i^T x^{\min} + q_i \quad \text{and} \quad \max_{x \in R} (C_i^T x + q_i) = C_i^T x^{\max} + q_i,$$

where

$$x_j^{\min} = \begin{cases} \ell_j & \text{if } c_{ij} \geq 0 \\ u_j & \text{if } c_{ij} < 0 \end{cases} \quad \text{and} \quad x_j^{\max} = \begin{cases} \ell_j & \text{if } c_{ij} \leq 0 \\ u_j & \text{if } c_{ij} > 0 \end{cases}.$$

Let S^B denote the set of all unit vectors e^i such that ℓ_i and u_i are finite and f is quasi-concave on $R_{e^i}(x)$ for all x in R . Similarly, let S^U denote the set of all unit vectors e^i such that $\ell_i = -\infty$ or $u_i = +\infty$ and f is concave or strictly quasi-concave on $R_{e^i}(x)$ for all x in R . Let also $I = \{i : e^i \in S^B\}$, $J_\ell = \{i : e^i \in S^U \text{ and } u_i = +\infty\}$, $J_u = \{i : e^i \in S^U \text{ and } \ell_i = -\infty\}$, and $J = J_\ell \cup J_u$. For every $K \subseteq I$, set

$$R(K) := \{x \in R : x_i = \ell_i \text{ for } i \in J_\ell \cup K \text{ and } x_i = u_i \text{ for } i \in J_u \cup (I \setminus K)\},$$

$$\psi(K) := \min_{x \in R(K)} f(x). \quad (61)$$

Theorem 21 *If f attains its minimum on R , then*

$$\min_{x \in R} f(x) = \min_{K \subseteq I} \psi(K). \quad (62)$$

Proof The proof follows from [Theorem 16](#) by observing that the faces of R that do not have any edge parallel to a vector in $S^B \cup S^U$ are precisely the sets $R(K)$. \square

A *pseudo-Boolean function* is a real-valued function f on the Boolean hypercube $\mathbb{B}^n = \{0, 1\}^n$. By viewing every element of \mathbb{B}^n as the characteristic vector of a subset N of n elements, one can also view f as a function defined on the set 2^N of all subsets of N . The problem of minimizing a pseudo-Boolean function on \mathbb{B}^n , or on a subset thereof, has been extensively studied (see, e.g., [17] for a recent survey). The general problem is *NP*-complete, even in the quadratic case, but several polynomial classes of pseudo-Boolean optimization problems are known.

[Theorem 21](#) establishes the equivalence between the problem of minimizing a function f on R (minimization under box constraints) and the pseudo-Boolean problem of minimizing $\psi(K)$ on the subsets of I . In the special case where R is the *continuous* unit hypercube $[0, 1]^n$, [Theorems 17](#) and [21](#) can be specialized as follows (see also [72, 73]).

Corollary 4 *If f is (strictly) quasi-concave with respect to each variable on $[0, 1]^n$, then at least one (all) global minimum point of f on $[0, 1]^n$ belongs to the Boolean hypercube \mathbb{B}^n .*

[Corollary 4](#) provides a link between the pseudo-Boolean optimization problem of minimizing f on \mathbb{B}^n and the nonlinear problem of minimizing f on $[0, 1]^n$. This can be used to exploit nonlinear programming techniques to solve pseudo-Boolean optimization problems and vice versa. This link has also been used in control theory to efficiently perform a stability test for a system of linear differential equations with uncertain real parameters [33]. Later, [Corollary 4](#) is used to provide a continuous formulation for the maximum-weight independent set problem in a graph. The equivalence between minimization on the Boolean hypercube \mathbb{B}^n and on the continuous hypercube $[0, 1]^n$ had been already established by Rosenberg [67] for *multilinear* functions, i.e., functions that are linear with respect to each variable.

In the general case, the faces $R(K)$ of R are obtained by fixing the values of the variables with indices in $I \cup J$, and all have the same dimension $n - |I| - |J|$. Hence, the complexity of problem (61) decreases with $|I|$, while that of problem (62) increases. When f has some additional structure, problems (61) or (62) may be solved in polynomial time even if $|I|$ is small or large, respectively. For example, if a vector $x \in R$ is partitioned in the form $x = (x_M, x_{I \cup J})$ and f is convex with respect to x_M for every fixed value of $x_{I \cup J}$, then the minimum in (61) can be computed in polynomial time for every $K \subseteq I$. A case where problem (62) can be solved in polynomial time is described below.

Recall that a subset $S \subseteq \mathbb{R}^n$ is a sublattice of \mathbb{R}^n if, for all $x^1, x^2 \in S$, $x^1 \vee x^2 \in S$ and $x^1 \wedge x^2 \in S$, where $(x^1 \vee x^2)_i := \max\{x_i^1, x_i^2\}$ and $(x^1 \wedge x^2)_i := \min\{x_i^1, x_i^2\}$.

Note that a rectangle is trivially a sublattice of \mathbb{R}^n , and so is the Boolean hypercube \mathbb{B}^n . A real-valued function f is submodular on a sublattice S if

$$f(x^1 \vee x^2) + f(x^1 \wedge x^2) \leq f(x^1) + f(x^2),$$

for all $x^1, x^2 \in S$.

In the case where f is twice continuously differentiable, it is known [78] that f is submodular on a rectangle R if and only if the second-order mixed partial derivatives $f_{x_i x_j}(x)$ are nonpositive for all $x \in R$ and $i \neq j$.

Grötschel et al. [40] proved that the problem of minimizing a submodular function on \mathbb{B}^n (that is equivalent to the problem of minimizing a submodular function on the family of subsets of a set with n elements) can be solved in polynomial time (see also [48, 68] for *combinatorial* polynomial algorithms for this problem). Furthermore, if f is a submodular function on R , then a result of Topkis [78, Theorem 4.3] implies that $\varphi(K)$ is a submodular function on 2^I . This proves the following:

Theorem 22 *If f is a submodular function on $R = [\ell, u]$, and $\psi(K)$ can be evaluated in polynomial time for every $K \subseteq I$, then f can be minimized in polynomial time over R .*

This result can be specialized to the case where f is a quadratic function and R is bounded. Given an $n \times n$ matrix C , and a subset of indices $J \subseteq N = \{1, \dots, n\}$,

let C_{JJ} be the submatrix of C formed by the elements with row and column indices in J .

Corollary 5 Let $R = [\ell, u]$, with $\ell, u \in \mathbb{R}^n$, and let $f(x) = \frac{1}{2}x^T C x + q^T x$. Assume that $c_{ij} \leq 0$ for all $i \neq j$ that there exists a subset M of indices such that C_{MM} is positive semidefinite and that $c_{ii} \leq 0$ for all $i \in I = N \setminus M$. Then f can be minimized in polynomial time over R .

Proof Since C_{MM} is positive semidefinite, f is convex on all faces $R(K)$ for $K \subseteq I$. Hence, $\psi(K) = \min_{x \in R(K)} f(x)$ can be evaluated in polynomial time. Furthermore, f is submodular on R by the assumption $c_{ij} \leq 0$ for all $i \neq j$. Thus, the result follows from [Theorem 22](#). \square

[Corollary 4](#) can be used to provide a continuous formulation of the weighted maximum independent set problem in a graph. Such formulation includes as a special case the one proposed in [1] for the unweighted case.

Let $G = (V, E)$ be a simple undirected graph, where $V = \{1, \dots, n\}$ is the vertex set and E is the set of edges. A subset $S \subseteq V$ is an *independent set* if there is no edge joining vertices in S . Let w_i denote the weight of node i in V , and $w(S) = \sum_{i \in S} w_i$ denote the weight of the subset S of nodes. An independent set S has *maximum weight* if $w(S)$ is maximum among all independent sets.

Let $\alpha_1, \dots, \alpha_n$ and β_1, \dots, β_n be quasi-convex real-valued functions on $[0, 1]$ satisfying the following properties. For every $i = 1, \dots, n$,

$$\alpha_i(1) = \beta_i(0) = 0, \quad \alpha_i(0) = \beta_i(1) = 1, \quad \text{and } \alpha_i(x), \beta_i(x) > 0, \quad \text{for all } x \in (0, 1).$$

Consider the function $f : [0, 1]^n \rightarrow \mathbb{R}$ defined by

$$f(x) = \sum_{i \in V} w_i \alpha_i(x_i) \prod_{(i,j) \in E} \beta_j(x_j).$$

Note that for every $x \in \{0, 1\}^n$

$$f(x) = \sum_{i \in V} w_i (1 - x_i) \prod_{(i,j) \in E} x_j.$$

To every subset $S \subseteq V$ associate the point $x^S \in \{0, 1\}^n$ defined by $x_i^S = 0$ iff $i \in S$ (i.e., x^S is the complement of the characteristic vector of S).

Theorem 23 Let T be a maximum-weight independent set in G . Then

$$f(x^T) = w(T) = \max_{x \in [0, 1]^n} f(x) = \max_{x \in \{0, 1\}^n} f(x). \quad (63)$$

Proof Since f is quasi-convex with respect to each variable, the last equality in (63) follows directly from Corollary 4. If S is an independent set of G , then it can be easily verified that $f(x^S) = w(S)$. On the other hand, if S is any subset of V , then set $S_I = \{i \in S : (i, j) \in E \Rightarrow j \notin S\}$ is an independent set of G . Furthermore, if $i \in S \setminus S_I$, then $\prod_{(i,j) \in E} x_j^S = 0$. Thus, $f(x^S) = f(x_I^S)$ for every $S \subseteq V$. Therefore,

$$\max_{x \in \{0,1\}^n} f(x) = \max\{f(x^S) : S \text{ is an independent set of } G\} = w(T)$$

□

3.5 Convex–Concave Problems

Consider a problem of the form:

$$\min f(x, y) \quad \text{s.t. } (x, y) \in S \times T, \quad (64)$$

where $S, T \subset \mathbb{R}^n$ and $f : S \times T \rightarrow \mathbb{R}$.

When $f(x, y) = c^T x + x^T Q y + d^T y$, with $Q \in \mathbb{R}^{p \times q}$, $c, x \in \mathbb{R}^p$, $d, y \in \mathbb{R}^q$, and S and T are polyhedra, problem (64) is called a *bilinear programming problem*. This kind of problems, which can be reduced to concave minimization problems, have been studied by several authors (see, e.g., [19, 46, 62] and references therein). It is well known that at least one solution of a bilinear programming problem is achieved at a vertex of its feasible region. It is shown below that this property holds for a more general class of functions.

It is straightforward to prove that, if $\min_{x \in S} f(x, y)$ exists for every $y \in T$, then problem (64) is equivalent to the following:

$$\min_{y \in T} \varphi(y), \quad (65)$$

where $\varphi(y) := \min_{x \in S} f(x, y)$.

Hence, the solution of problem (64) may be reduced to the solution of the two subproblems $\min_{x \in S} f(x, y)$ and $\min_{y \in T} \varphi(y)$. In the case where T is a polyhedron, Theorem 17 can be applied to ensure that $\varphi(y)$ attains its minimum at one of the vertices of T .

Theorem 24 *If T is a polyhedron and the function $y \mapsto f(x, y)$ is quasi-concave on all directions parallel to bounded edges of T and concave or strictly quasi-concave on all directions parallel to unbounded edges of T , then, if the function $\varphi(y)$ attains its minimum on T , at least one global minimum point lies in a vertex of T .*

Proof It is sufficient to notice that, since $\varphi(y)$ is defined as a minimum of functions that are concave or quasi-concave on the directions parallel to the edges of T ,

φ shares the same properties of such functions and hence [Theorem 17](#) may be employed to complete the proof. \square

Clearly, if $\min_{x \in S} f(x, y)$ and $\min_{y \in T} \varphi(y)$ can be evaluated efficiently, then problem (64) can also be solved in an efficient manner. This is also the case for a special class of indefinite quadratic programs. Consider the problem

$$\begin{cases} \min f(x, y) = x^T Bx + x^T Cy + y^T Dy + c^T x + d^T y \\ x \in S = \{x : A_1 x \leq b_1\} \\ y \in T = \{y : A_2 y \leq b_2\}, \end{cases} \quad (66)$$

where $B \in \mathbb{R}^{p \times p}$, $C \in \mathbb{R}^{p \times q}$, $D \in \mathbb{R}^{q \times q}$, $A_1 \in \mathbb{R}^{m_1 \times p}$, $A_2 \in \mathbb{R}^{m_2 \times q}$, $c, x \in \mathbb{R}^p$, $d, y \in \mathbb{R}^q$, $b_1 \in \mathbb{R}^{m_1}$, and $b_2 \in \mathbb{R}^{m_2}$. Note that, if B is positive semidefinite, then $\varphi(y) = \min_{x \in S} f(x, y)$ can be evaluated in polynomial time [53] and, if $s^T D s \leq 0$ for all vectors s parallel to an edge of T , then $\varphi(y)$ achieves its minimum on a vertex of T . Hence, problem (66) can also be solved in polynomial time if the set V_T of vertices of T is small or if $\varphi(y)$ can be minimized in polynomial time over V_T . This is the case, e.g., if $T = \{y \in \mathbb{R}^n : \ell \leq y \leq u\}$ and $b_{ij} \leq 0$, $c_{ij} \leq 0$, $d_{ij} \leq 0$ for all $i \neq j$. In fact, these assumptions imply submodularity of $f(x, y)$ which in turn implies submodularity of $\varphi(y)$ by a result of Topkis [78]. This proves the following result:

Theorem 25 Problem (66) can be solved in polynomial time if $T = \{y \in \mathbb{R}^n : \ell \leq y \leq u\}$, B is positive semidefinite, $b_{ij}, d_{ij} \leq 0$ for all $i \neq j$, and $c_{ij} \leq 0$ for all i and j .

3.6 Max Clique and Optimization Over Simplices and Polymatroids

A polyhedron P is a k -dimensional simplex in \mathbb{R}^n if it is the convex hull of $k+1$ affinely independent vectors v^1, \dots, v^{k+1} . This is equivalent to requiring that P is k -dimensional and has $k+1$ vertices v^1, \dots, v^{k+1} , with $k \leq n$. The set D of directions parallel to all the edges of such simplex is $D = \{v^i - v^j : i, j = 1, \dots, k+1, i \neq j\}$. If f is a twice continuously differentiable function on P , and $d_{ij} = v^i - v^j \in D$, then concavity of f on all the segments $P_{d_{ij}}(x)$, for $x \in P$ is equivalent to

$$d_{ij}^T H_f(x) d_{ij} \leq 0, \text{ for all } x \in P, \quad (67)$$

where $H_f(x)$ is the Hessian matrix of f in x . If (67) holds for every $d_{ij} \in D$, then, by [Theorem 17](#), f attains its minimum at a vertex of P . Hence, in this case, the minimum of f on P can be trivially obtained by comparing the values of f at the vertices of P .

By means of an affine transformation, one can always reduce the problem of minimizing a function on a simplex to the standard form

$$\min f(x) \text{ s.t. } x \in \Delta, \quad (68)$$

where $\Delta = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, x_i \geq 0, i = 1, \dots, n\}$ is the *standard simplex* in \mathbb{R}^n . Hence, one can restrict the attention to the problem of minimizing a function on the standard simplex. In the special case where f is a quadratic function, problem (68) becomes the so-called *standard quadratic optimization problem*, an *NP-hard* problem with a number of applications which has been the object of several recent studies (see [13–16] and references therein).

When $P = \Delta$, condition (67) becomes

$$f_{x_i x_i}(x) + f_{x_j x_j}(x) - 2f_{x_i x_j}(x) \leq 0, \text{ for all } x \in P. \quad (69)$$

In particular, in the case where $f(x) = \frac{1}{2}x^T C x + q^T x$, condition (69) simplifies to

$$c_{ii} + c_{jj} - 2c_{ij} \leq 0. \quad (70)$$

Recall that a matrix M is a Monge matrix [18] if $m_{ih} + m_{jk} \leq m_{ik} + m_{jh}$ for all $i < j$ and $h < k$. Clearly a symmetric Monge matrix satisfies (70). Hence, the standard quadratic optimization problem can be trivially solved by comparing the values of f at the vertices of Δ when C is Monge.

Let us now return to the general minimization problem on the standard simplex (68). Let S be the set of all directions $d_{ij} = e^i - e^j$ in D such that (69) holds. Consider the graph $G = (V, E)$, where

$$V = \{1, \dots, n\} \text{ and } (i, j) \in E \Leftrightarrow d_{ij} \notin S. \quad (71)$$

To every subset Q of V associate the face $F_Q = \{x \in \Delta : \sum_{i \in Q} x_i = 1\}$ of Δ . Note that the mapping $Q \mapsto F_Q$ is a bijection between the subsets of V and the faces of Δ . Also consider the following weight function defined on the subsets Q of V :

$$W(Q) = \min\{f(x) : x \in F_Q\}.$$

Recall that a subset Q of V is a *clique* of G if every pair of nodes in Q is joined by an edge in E . A clique is *maximal* if it is not strictly contained in any larger clique. Let \mathcal{C} denote the set of all cliques of G and \mathcal{C}^M the set of all maximal cliques of G . Given a real-valued function $W(Q)$ on the subsets of V , a *maximum (minimum) weight clique* is a clique that maximizes (minimizes) $W(Q)$ on the set \mathcal{C} of all cliques of G . Note that there exists a maximum (minimum) weight clique that is also a maximal clique if the function W is *isotone (antitone)*, i.e., $W(Q) \leq W(Q')$ ($W(Q) \geq W(Q')$) whenever $Q \subseteq Q'$. In the classical maximum weight clique problem, the function W is modular, i.e., $W(Q) = \sum_{i \in Q} w_i$. The following result shows that the minimization problem on the standard simplex is equivalent to the minimum weight clique problem on the associated graph. Furthermore, one can restrict the search for the optimal solutions of (68) to the faces F_Q of Δ corresponding to maximal cliques Q of G .

Theorem 26

$$\min_{x \in \Delta} f(x) = \min\{f(x) : x \in \bigcup_{Q \in \mathcal{C}^M} F_Q\} = \min_{Q \in \mathcal{C}} W(Q) = \min_{Q \in \mathcal{C}^M} W(Q).$$

Proof First note that the function $W(Q)$ is antitone, since $Q \subseteq Q'$ implies $F_Q \subseteq F_{Q'}$. Thus, the proof follows easily from [Theorem 16](#) by observing that a face F_Q of Δ does not contain any edge parallel to a direction in S if and only if Q is a clique of G . \square

In view of [Theorem 26](#), one can solve problem (68) in polynomial time in those cases where all the maximal cliques of G can be listed in polynomial time (e.g., when G is sparse), and the weight function $W(Q)$ can also be evaluated in polynomial time for every $Q \in V$.

A well-known result of Motzkin and Straus [60] establishes the equivalence between the maximum clique problem and a special quadratic minimization problem on the standard simplex. This result has been extended to the maximum weight clique problem by Gibbons et al. [39]. A generalized version of the result of Gibbons et al. can be easily obtained as a special case of [Theorem 26](#).

Theorem 27 Let $f(x) = \frac{1}{2}x^T C x$ and let $G = (V, E)$ be defined as in (71) (i.e., $E = \{(i, j) : c_{ii} + c_{jj} - 2c_{ij} > 0\}$). Assume that $c_{ij} = 0$ for every $(i, j) \in E$. Then:

- (i) If $c_{ii} \leq 0$ for some index i , then $\min\{f(x) : x \in \Delta\} = c_{kk} = \min\{c_{ii} : i = 1, \dots, n\}$. Furthermore, a global minimum point for f on Δ is given by $x^* = e^k$.
- (ii) If $c_{ii} > 0$ for every $i \in V$, then

$$\min\{f(x) : x \in \Delta\} = \min_{Q \in \mathcal{C}} \left(\sum_{i \in Q} c_{ii}^{-1} \right)^{-1} = \left(\max_{Q \in \mathcal{C}} \sum_{i \in Q} c_{ii}^{-1} \right)^{-1}.$$

Furthermore, $x_i^* = c_{ii}^{-1} (\sum_{i \in Q} c_{ii}^{-1})^{-1}$ for $i \in Q$ and $x_i^* = 0$ for $i \notin Q$ is a global minimum point for f on Δ .

Proof Recall that $W(Q) = \min\{f(x) : x \in F_Q\}$. When Q is a clique of G one has

$$W(Q) = \min \left\{ \sum_{i \in Q} c_{ii} x_i^2 : \sum_{i \in Q} x_i = 1, x_i \geq 0, i = 1, \dots, n \right\}. \quad (72)$$

Hence, the optimal solution x^* of the minimization problem in (72) is given by $x^* = e^k$ with $k \in Q$ such that $c_{kk} = \min\{c_{ii} : i \in Q\}$, if $c_{kk} \leq 0$, and by $x_i^* = c_{ii}^{-1} (\sum_{i \in Q} c_{ii}^{-1})^{-1}$ for $i \in Q$ and $x_i^* = 0$ for $i \notin Q$, if $c_{kk} > 0$. In the first case $W(Q) = \min\{c_{ii} : i \in Q\}$, while in the second case $W(Q) = (\sum_{i \in Q} c_{ii}^{-1})^{-1}$. The proof then follows from [Theorem 26](#). \square

When all the c_{ii} are positive, this theorem transforms the problem of minimizing the special quadratic function f on Δ into a maximum weight clique problem. However, the transformation can also be applied in the reverse direction, as shown in the following corollary.

Corollary 6 ([39]) *Let $G = (V, E)$ be a simple graph with weight $w_i > 0$ associated to each node i in V , and let $W(Q) = \sum_{i \in Q} w_i$. Let C be any symmetric matrix satisfying $c_{ii} = w_i^{-1}$, for all $i \in V$, $c_{ij} = 0$ for all $(i, j) \in E$, and $c_{ii} + c_{jj} - 2c_{ij} \leq 0$ for all $(i, j) \notin E$. Then a clique of maximum weight in G is given by a subset Q of V such that the point $x_i^* = c_{ii}^{-1}(\sum_{i \in Q} c_{ii}^{-1})^{-1}$ for $i \in Q$ and $x_i^* = 0$ for $i \notin Q$ is a global minimum point of f on Δ . Furthermore,*

$$\left(\max_{Q \in \mathcal{C}} W(Q) \right)^{-1} = \min\{f(x) : x \in \Delta\}.$$

The maximum weight clique formulation for the standard quadratic optimization problem has been used by Scozzari and Tardella [70] to efficiently solve the latter problem by means of clique algorithms on the associated graph. This approach has led to the solution of problems of up to two orders of magnitude greater than those previously solved in the literature.

This section ends with the description of a class of polyhedra that includes both rectangles and simplices and still has a very small and simple set of directions parallel to all edges: polymatroids and base polyhedra.

Thus, in this class, one can still prove the existence of a vertex optimal solution by checking only a small number of fairly simple conditions.

A *polymatroid* is a polyhedron P in \mathbb{R}^n described by the following inequalities:

$$\begin{cases} \sum_{i \in I} x_i \leq g(I) & \text{for all } I \subseteq N = \{1, \dots, n\}, \\ x_i \geq 0 & \text{for all } i \in N, \end{cases}$$

where the function $g : 2^N \rightarrow \mathbb{R}$ is

Isotone: $g(S) \leq g(T)$ for all $S, T \subseteq N$.

Submodular: $g(S \cap T) + g(S \cup T) \leq g(S) + g(T)$ for all $S, T \subseteq N$.

Normalized: $g(\emptyset) = 0$.

A *base polyhedron* is the facet of a polymatroid determined by the additional constraint $\sum_{i \in N} x_i = g(N)$. It is well known from the characterization of adjacency of vertices in polymatroids and base polyhedra (see, e.g., [31, 79]) that every edge of a polymatroid is parallel to a vector of the set $\{e^i : i \in N\} \cup \{e^i - e^j : i, j \in N, i \neq j\}$, while every edge of a base polyhedron is parallel to a vector of the set $\{e^i - e^j : i, j \in N, i \neq j\}$.

Thus, a twice differentiable function f attains its minimum at a vertex of a base polyhedron P if

$$f_{x_i x_i}(x) + f_{x_j x_j}(x) - 2f_{x_i x_j}(x) \leq 0, \forall x \in P \text{ and } \forall i, j \in N, \text{ with } i \neq j.$$

For a polymatroid one must add the condition

$$f_{x_i x_i}(x) \leq 0, \forall x \in P \text{ and } \forall i \in N.$$

Note that the polyhedra described by nonnegativity constraints $x \geq 0$, upper bounds $x \leq u$, and a single knapsack constraint $\sum_i a_i x_i \leq b$ or $\sum_i a_i x_i = b$, with $a_i > 0$, $a_i u_i \leq b$ for all i , and $b \leq \sum_i a_i u_i$ can be transformed into polymatroids and base polyhedra by setting $y_i = a_i x_i$ and taking the submodular function g defined by $g(\emptyset) = 0$, $g(\{e^i\}) = a_i u_i$ for all i , and $g(S) = b$ for all other subsets $S \subseteq N$.

4 Piecewise Concavity and Applications

4.1 Piecewise Concavity

In this section the equivalence between some continuous and discrete optimization problems is established by exploiting another form of generalized concavity of the objective functions involved. More precisely, *piecewise concave* or *piecewise quasi-concave* functions are minimized on a set X . Such functions, which arise naturally from several applications, are required to be concave or quasi-concave on each element of a family of subsets of X whose union covers X . Some applications to location theory and to various types of minimax problems are described in the next sections. For piecewise concave functions, the search for a global minimum point can be restricted to a special subset of X that, under suitable assumptions, is guaranteed to be finite.

Let $X = \bigcup_{i \in I} X_i \subset \mathbb{R}^n$, where each X_i is a closed subset of \mathbb{R}^n . Consider a family $\{f_i\}_{i \in I}$ of functions with $f_i : X_i \rightarrow \mathbb{R}$ and $f_i(x) = f_j(x)$ for every $x \in X_i \cap X_j$ and for all $i, j \in I$. Let $g : X \rightarrow \mathbb{R}$ be defined by

$$g(x) := f_i(x), \quad \text{for } x \in X_i, i \in I. \quad (73)$$

The function g is called (*strictly*) *piecewise concave* or (*strictly*) *piecewise quasi-concave* iff all the functions f_i are (*strictly*) concave or (*strictly*) quasi-concave, respectively. Note that the sets X_i are not required to be convex and that the notions of concavity and quasi-concavity on X_i are those specified in [Definition 2](#). In fact, the sets X_i are not necessarily convex, e.g., in the important problem of minimizing a function defined as the maximum of a family of concave or quasi-concave functions on a convex set. Clearly, *piecewise convexity* and *piecewise quasi-convexity* of a function can be defined in a similar manner.

Consider the subset \mathcal{E} of X containing all the convex hull extreme points of all sets X_i , and let D be the subset of \mathcal{E} formed by those points that are convex hull extreme points of every set X_i to which they belong. More formally

$$\mathcal{E} = \bigcup_{i \in I} \mathcal{E}(\text{conv}(X_i)) \text{ and } D = \{x \in \mathcal{E} : x \in X_i \Rightarrow x \in \mathcal{E}(\text{conv}(X_i))\}. \quad (74)$$

Let X_{\min} be the set of global minimum points of g over X . The main result of this section, which generalizes and strengthens some well-known results for piecewise concave functions and for minimax problems [74] is the following:

Theorem 28 (i) If f_i is quasi-concave on X_i and X_i is compact for all $i \in I$, then

$$X_{\min} \neq \emptyset \Leftrightarrow X_{\min} \cap \mathcal{E} \neq \emptyset.$$

(ii) If f_i is strictly quasi-concave on X_i and X_i is compact for all $i \in I$, then

$$X_{\min} \neq \emptyset \Leftrightarrow X_{\min} \cap D \neq \emptyset.$$

(iii) If X is compact, g is lower semicontinuous on X and f_i is concave on X_i for all $i \in I$, then

$$X_{\min} \cap D \neq \emptyset \text{ and } \mathcal{E}(\text{conv}(X_{\min})) \subseteq D.$$

Proof (i) Let x^* be a global minimum point for g over X and assume that $x^* \in X_i$. Since X_i is compact, $\text{conv}(X_i)$ is also compact and hence $\text{conv}(X_i) = co(\mathcal{E}(\text{conv}(X_i)))$, by the Krein–Millman Theorem.

Thus $X_i \subseteq co(\mathcal{E}(\text{conv}(X_i)))$, so that there exist points $x^1, \dots, x^m \in \mathcal{E}(\text{conv}(X_i))$ and coefficients $\alpha_1, \dots, \alpha_m \in]0, 1[$ such that $x^* = \sum_{i=1}^m \alpha_i x^i$ and $\sum_{i=1}^m \alpha_i = 1$. Then, by the quasi-concavity of f_i , at least one of the points x^1, \dots, x^m must be a global minimum point for g .

- (ii) Let x^* be a global minimum point for g over X and assume that $x^* \in X_i$. Using an argument similar to the one employed in the proof of (i), for $x^* \notin \mathcal{E}(\text{conv}(X_i))$, there exist points $x^1, \dots, x^m \in \mathcal{E}(\text{conv}(X_i))$ such that $f_i(x^*) > \min\{f_i(x^1), \dots, f_i(x^m)\}$. This contradicts the minimality of x^* .
- (iii) First note that, by the lower semicontinuity of g , the set X_{\min} is nonempty, closed, and hence compact since it is contained in the compact set X . Hence, $\mathcal{E}(\text{conv}(X_{\min})) \neq \emptyset$ and, by the Krein–Millman Theorem, $\text{conv}(X_{\min}) = \text{conv}(\mathcal{E}(\text{conv}(X_{\min})))$. To complete the proof, it suffices to show that if $x^* \in \mathcal{E}(\text{conv}(X_{\min}))$, then $x^* \in \mathcal{E}(\text{conv}(X_i))$ for every i such that $x^* \in X_i$. Ab absurdo, if $x^* \in X_i \setminus \mathcal{E}(\text{conv}(X_i))$, then there exist $y, z \in \text{conv}(X_i)$ and $\lambda \in]0, 1[$ such that $x^* = \lambda y + (1-\lambda)z$. Then, by definition of $\text{conv}(X_i)$, there exist $y^1, \dots, y^{m_1}, z^1, \dots, z^{m_2}$ in X_i and coefficients $\alpha_1, \dots, \alpha_{m_1}, \beta_1, \dots, \beta_{m_2} \in]0, 1[$ such that $y = \sum_{i=1}^{m_1} \alpha_i y^i$, $z = \sum_{i=1}^{m_2} \beta_i z^i$, $\sum_{i=1}^{m_1} \alpha_i = 1$, and $\sum_{i=1}^{m_2} \beta_i = 1$. Setting $\gamma_i = \lambda \alpha_i$ for $i = 1, \dots, m_1$ and $\delta_i = (1-\lambda)\beta_i$ for $i = 1, \dots, m_2$, one has $x^* = \sum_{i=1}^{m_1} \gamma_i y^i + \sum_{i=1}^{m_2} \delta_i z^i$ and $\sum_{i=1}^{m_1} \gamma_i + \sum_{i=1}^{m_2} \delta_i = 1$. From the concavity of f_i , it then follows that $f_i(x^*) = f_i(y^i) = f_i(z^i)$ for all indices i . Hence, $g(x^*) = g(y) = g(z)$ and therefore $y, z \in X_{\min}$, contradicting $x^* \in \mathcal{E}(\text{conv}(X_{\min}))$.

□

Remark 3 Note that if $\text{conv}(X_i)$ is a polytope, then, taking into account [Theorem 17](#), the Assumption of (i) in [Theorem 28](#) can be weakened by requiring only quasi-concavity of f_i on all line segments in $\text{conv}(X_i)$ which are parallel to some edge of $\text{conv}(X_i)$.

Remark 4 The result of [Theorem 28\(i\)](#) holds also when some set X_i is an unbounded polyhedron. Also in this case, taking into account [Theorem 17](#), assumption (i) in [Theorem 28](#) can be replaced by the requirement of quasi-concavity of f_i on all line segments in X_i which are parallel to some bounded edge of X_i and concavity or strict quasi-concavity of f_i on all line segments in X_i which are parallel to some unbounded edge of X_i .

4.2 Minimax Problems

In several applications (see, e.g., [21, 23]) one has to solve minimax problems of the form

$$\min_{x \in X} \max_{i \in I} f_i(x), \quad (75)$$

where X is a compact subset of \mathbb{R}^n (often a polytope) and each f_i is (quasi-)concave on X . In this case, [Theorem 28](#) can be applied to the piecewise (quasi-)concave function $g(x) := \max_{i \in I} f_i(x)$ which is (quasi-)concave on the sets $X_i := \{x \in X : g(x) = f_i(x)\}$. Thus one obtains the following result first stated by Zangwill [80] for the concave case with a somewhat incomplete proof.

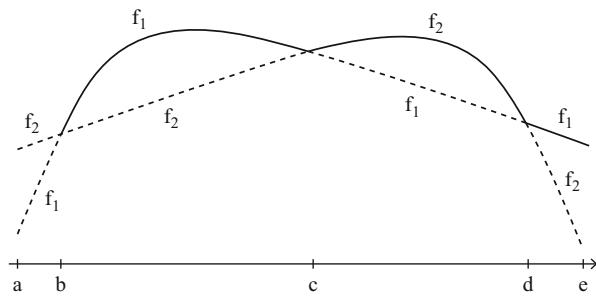
Theorem 29 *If X is a compact subset of \mathbb{R}^n and the functions $f_i : X \rightarrow \mathbb{R}$ are continuous and (quasi-)concave, then at least one solution of (75) belongs to the set D (resp. \mathcal{E}).*

Suppose now that X is a polytope described by a set of linear inequalities $a_j^T x \leq b_j$, $j \in J$ and, for every $x \in X$, define $I(x) = \{i \in I : g(x) = f_i(x)\}$ and $J(x) = \{j \in J : a_j^T x = b_j\}$. It can be easily verified that a point $x \in X$ is a vertex (extreme point) of X if $J(x)$ is maximal, i.e., there does not exist any $y \in X$ such that $J(x)$ is a proper subset of $J(y)$. Analogously, a point $x \in X$ is called a *g-vertex* of X iff $I(x) \cup J(x)$ is maximal. This notion has been introduced by Du and Hwang [22] (see also [21]) who also proved the following result that has been extended to the case of an infinite index set I by Du and Pardalos [24].

Theorem 30 *If X is a polytope in \mathbb{R}^n and the functions $f_i : X \rightarrow \mathbb{R}$ are continuous and concave, then at least one solution of (75) belongs to the set G of g-vertices of X .*

[Theorems 29](#) and [30](#) establish an interesting equivalence between the continuous problem (75) and the problem of minimizing $g(x)$ over the sets D , \mathcal{E} , or G which are often finite. Note however that [Theorem 29](#) provides a stronger restriction for

Fig. 1 Example for strict inclusions



global minimum points than [Theorem 30](#). Indeed, it is clear that $D \subseteq \mathcal{E}$, and it has been proved in [74] that $D \subseteq G$. Furthermore, simple examples like the following one show that there are cases where the inclusions $D \subset \mathcal{E} \subset G$ can be strict.

Example 1 Consider the interval $X = [a, e] \subset \mathbb{R}$ and the functions $f_1, f_2 : X \rightarrow \mathbb{R}$ illustrated in [Fig. 1](#). Then $X_1 = [a, b] \cup [c, d]$, $X_2 = [b, c] \cup [d, e]$, $D = \{a, e\}$, $\mathcal{E} = \{a, b, d, e\}$, and $G = \{a, b, c, d, e\}$.

4.3 Location Problems

Piecewise concavity has important applications in location theory. Consider, e.g., the rectilinear distance facility location problem, which consists in finding the coordinates of m new facilities x^1, \dots, x^m in an n -dimensional space so as to minimize a weighted sum of the rectilinear (or L^1) distances among them and between them and a set of p existing facilities located at the points a^1, \dots, a^p . Formally, one wants to minimize over $\mathbb{R}^{n \times m}$ the function

$$g(x^1, \dots, x^m) := \sum_{k=1}^n \sum_{i=1}^m \sum_{j=1}^m w_{ijk} |x_k^i - x_k^j| + \sum_{k=1}^n \sum_{i=1}^m \sum_{j=1}^p v_{ijk} |x_k^i - a_k^j|,$$

where w_{ijk} and v_{ijk} are nonnegative weights. Note that the function g is separable, i.e., it can be written in the form $g(x) = \sum_{k=1}^n g_k(x_k^1, \dots, x_k^m)$, where

$$g_k(x_k^1, \dots, x_k^m) = \sum_{i=1}^m \sum_{j=1}^m w_{ijk} |x_k^i - x_k^j| + \sum_{j=1}^p v_{ijk} |x_k^i - a_k^j|.$$

Hence, the problem of minimizing g over $\mathbb{R}^{n \times m}$ can be solved by minimizing each function g_k over \mathbb{R}^m separately. For this reason one can restrict the attention to the one-dimensional case where $x^i, a^i \in \mathbb{R}$. It is well known [20] that a global minimum point of g must be contained in the finite set $A := \{a^1, \dots, a^p\}^m$. One

can show that this is true also in a more general setting and that it is a consequence of the piecewise concavity of g .

Consider the function g defined by

$$g(x) = \sum_{i=1}^m \sum_{j=1}^m \varphi_{ij}(|x^i - x^j|) + \sum_{i=1}^m \sum_{j=1}^p \psi_{ij}(|x^i - a^j|), \quad (76)$$

where $\varphi_{ij}, \psi_{ij} : \mathbb{R}_+ \rightarrow \mathbb{R}$ are nondecreasing continuous concave functions satisfying $\varphi_{ij}(0) = \psi_{ij}(0) = 0$ and $\lim_{t \rightarrow +\infty} \psi_{ij}(t) = +\infty$ for all i and j . Assume, without loss of generality, that $a^1 < a^2 < \dots < a^p$. Let H denote the set of all bijective mappings (permutations) π from the set of indices $L = \{1, \dots, m+p\}$ into itself that verify $\pi^{-1}(i) < \pi^{-1}(i+1)$ for all $i = m+1, \dots, m+p-1$.

Lemma 3 *For every $i \in L$, define $y_i = x^i$ if $i \leq m$ and $y_i = a^{i-m}$ if $i > m$. Then for every $\pi \in H$ the function g defined in (76) is concave on $X_\pi := \{x \in \mathbb{R}^n : y_{\pi(i)} \leq y_{\pi(i+1)}, i = 1, \dots, m+p-1\}$.*

Proof Indeed, on X_π one has $|y_i - y_j| = y_i - y_j$ if $\pi^{-1}(i) < \pi^{-1}(j)$ and $|y_i - y_j| = y_j - y_i$ if $\pi^{-1}(i) > \pi^{-1}(j)$. Hence, on X_π the functions $\varphi_{ij}(|x^i - x^j|)$ and $\psi_{ij}(|x^i - a^j|)$ are concave for all i, j , since they are obtained as the composition of a concave and of a linear function. Therefore, the function g is concave on X_π because it is a sum of concave functions. \square

Theorem 31 *At least one global minimum point of g on \mathbb{R}^n is achieved at a point in A .*

Proof Note that the sets X_π are polyhedra with vertices in A and that $\mathbb{R}^n = \cup_{\pi \in H} X_\pi$. Furthermore, since g is continuous and $\lim_{t \rightarrow +\infty} \psi_{ij}(t) = +\infty$ for all i and j , there must be a global minimum point of g in a bounded neighborhood of 0. Hence, by Theorem 28, a global minimum point of g must be contained in A . \square

Another continuous location problem that can be solved by restricting the search to a finite set of candidate solutions is the problem of locating some undesirable facilities at locations $x^1, \dots, x^m \in \mathbb{R}^n$ in a way that minimizes the maximum of some decreasing nuisance cost functions of the distances among them and between them and a given set of points $b^1, \dots, b^p \in \mathbb{R}^n$ (see [28] for a survey on this type of problems). Formally, one wishes to minimize the function

$$g(x) = g(x^1, \dots, x^m) := \max\{\max_{i \in I, j \in J} \alpha_{ij}(\|x^i - b^j\|), \max_{i \neq j \in I} \beta_{ij}(\|x^i - x^j\|)\},$$

where α_{ij} and β_{ij} are continuous nonincreasing functions, $I = \{1, \dots, m\}$, $J = \{1, \dots, p\}$, and $\|\cdot\|$ is any norm in \mathbb{R}^n . In this case, the function g is piecewise

quasi-concave since it is the maximum of a finite family of continuous quasi-concave functions. Clearly, the infimum of g on $\mathbb{R}^{n \times m}$ is obtained when $\|x^i\| \rightarrow \infty$ for all i . Hence, this location problem becomes meaningful only when x is required to be in a compact set X . Under this assumption, [Theorem 29](#) can be applied to establish the following result.

Theorem 32 *At least one global minimum point of g on a compact set X is achieved in the set $\mathcal{E} = \bigcup_{i,j \in I} \mathcal{E}(\text{conv}(X_{ij})) \cup \bigcup_{i \in I, j \in J} \mathcal{E}(\text{conv}(Y_{ij}))$, where*

$$\begin{aligned} X_{ij} &:= \{x \in X : \alpha_{ij}(\|x^i - b^j\|) \\ &\geq \max\{\max_{h \in I, k \in J} \alpha_{hk}(\|x^h - b^k\|), \max_{h \neq k \in I} \beta_{hk}(\|x^h - x^k\|)\}\}, \\ Y_{ij} &:= \{x \in X : \beta_{ij}(\|x^i - x^j\|) \\ &\geq \max\{\max_{h \in I, k \in J} \alpha_{hk}(\|x^h - b^k\|), \max_{h \neq k \in I} \beta_{hk}(\|x^h - x^k\|)\}\}. \end{aligned}$$

When only one new undesirable facility has to be established, the objective function $g : \mathbb{R}^n \rightarrow \mathbb{R}$ becomes

$$g(x) = \max_{j \in J} \alpha_j(\|x - b^j\|).$$

In this case, [Theorem 32](#) implies that a global minimum point of g on a compact set X is attained at a point in $\mathcal{E} = \bigcup_{j \in J} \mathcal{E}(\text{conv}(X \cap X_j))$, where $X_j = \{x \in \mathbb{R}^n : \alpha_j(\|x - b^j\|) \leq \alpha_k(\|x - b^k\|) \text{ for all } k \neq j\}$. The family of sets X_j forms the (generalized) Voronoi diagram with respect to the functions $\alpha_j(\|x\|)$. This geometric structure has received much attention in recent years especially in the case where $\alpha_j(\|x\|) = \|x\|$ or $\alpha_j(\|x\|) = \lambda_j \|x\|$ and $\|\cdot\|$ is the Euclidean norm (see [6] and references therein). In particular, when $\alpha_j(\|x\|) = \|x\|$ and $\|\cdot\|$ is the Euclidean norm, the sets X_j are polyhedra described by

$$X_j = \left\{ x \in \mathbb{R}^n : (b^k - b^j)^T \left(x - \frac{b^j + b^k}{2} \right) \leq 0 \text{ for all } k \neq j \right\}.$$

In this case, the maximum number N of points in $\mathcal{E} = \bigcup_{j \in J} \mathcal{E}(X_j)$ satisfies the following bounds [51]:

$$n/2! p^{n/2} \leq N \leq 2(n/2! p^{n/2}) \text{ for } n \text{ even}$$

and

$$\frac{(\lceil n/2 \rceil - 1)!}{e} p^{\lceil n/2 \rceil} \leq N \leq 2(\lceil n/2 \rceil! p^{\lceil n/2 \rceil}) \text{ for } n \text{ odd.}$$

Furthermore, the points of \mathcal{E} can be computed in $O(p^{\lceil n/2 \rceil + 1})$ time in the general case [7], and in time $O(p \log p)$ for $n = 2$. Thus the minimum of g on a polyhedron X can be found by simple enumeration of all the vertices of the sets $X_i \cap X$. This yields a finite algorithm, while in [71] an algorithm is presented which requires some additional assumptions to solve the same problem in finitely many iterations.

Facility location on networks is an important field in location theory where much research has concentrated in identifying a *finite* set of points that necessarily contain an optimal solution. Good surveys on this topic are [43, 54, 59].

Location problems on networks can be formulated using some definitions and notation taken from [54]. An *edge* of length $\ell > 0$ is the image of an interval $[0, \ell]$ by a continuous mapping f from $[0, \ell]$ to \mathbb{R}^d such that $f(\theta) \neq f(\theta')$ for any $\theta \neq \theta'$ in $[0, \ell]$. A *network* is defined as a subset N of \mathbb{R}^d that satisfies the following conditions: (a) N is the union of a finite number of edges; (b) any two edges intersect at most at their extremities; (c) N is connected. The *vertices* of the network are the extremities of the edges defining N and are denoted by $V = \{v^1, \dots, v^n\}$. The set of edges defining the network is denoted by E . For every edge $[v^i, v^j] \in E$, let f_{ij} from $[0, \ell_{ij}]$ to \mathbb{R}^d be the defining mapping and denote by θ_{ij} the inverse of f_{ij} which maps $[v^i, v^j]$ into $[0, \ell_{ij}]$. Given two points $x^1, x^2 \in [v^i, v^j]$, the subset of points of $[v^i, v^j]$ between and including x^1, x^2 is a subedge $[x^1, x^2]$. The length of $[x^1, x^2]$ is given by $|\theta_{ij}(x^1) - \theta_{ij}(x^2)|$. A *path* joining two points $x^1 \in N$ and $x^2 \in N$ is a minimal connected subset of N containing x^1 and x^2 . The length of a path is equal to the sum of all its constituent edges and subedges. A metric d on N is defined by setting $d(x^1, x^2)$ equal to the length of a shortest path joining x^1 and x^2 .

For any point $z \in N$ consider the function on $[0, \ell_{ij}]$ defined by $d(z, f_{ij}(\theta))$. Note that, by the definition of the distance, one has

$$d(z, f_{ij}(\theta)) = \min\{d(z, v^i) + \theta, d(z, v^j) + \ell_{ij} - \theta\}.$$

Hence, $d(z, f_{ij}(\theta))$ is the minimum of two linear functions in θ . Therefore, the following properties hold [54]:

- (i) $d(z, f_{ij}(\theta))$ is continuous and concave on $[0, \ell_{ij}]$.
- (ii) $d(z, f_{ij}(\theta))$ is linearly increasing with slope $+1$ on $[0, \theta_{ij}(z)]$ and linearly decreasing with slope -1 on $[\theta_{ij}(z), \ell_{ij}]$, where $\theta_{ij}(z) = \frac{1}{2}[\ell_{ij} + d(z, v^j) - d(z, v^i)]$.

The (single) median problem on the network N consists of finding a point in N that minimizes the function

$$F(x) = \sum_{v^i \in V} w_i((d(v^i, x))),$$

where $w_i(t)$ are concave nondecreasing functions from \mathbb{R}_+ to \mathbb{R} . In 1964, Hakimi [41] showed that at least one solution of this problem belongs to V when $w_i(t) = \lambda_i t$ with $\lambda_i \geq 0$. This result was extended in [59] to the case where all $w_i(t)$ are concave nondecreasing functions. The proof is based on the remark that if $w_i(t)$ is

concave nondecreasing, then the function $w_i((d(v^i, f_{ij}(\theta))))$ is concave on $[0, \ell_{ij}]$ and hence $F(f_{ij}(\theta))$ is also concave on $[0, \ell_{ij}]$. Therefore, the global minimum of F must be attained at one of the points $f_{ij}(0)$ or $f_{ij}(\ell_{ij})$, i.e., at a vertex of N .

In the (single) center problem on the network N , one seeks to minimize the function

$$G(x) = \max_{v^i \in V} (d(v^i, x)).$$

In this case, the function $G(f_{ij}(\theta))$ is not concave on the whole segment $[0, \ell_{ij}]$, but only on subsegments thereof. However, the number of such subsegments is bounded by $|V|^2|E|$ and their extremities can be explicitly determined (see [54] for details). Hence, also the center problem on a network can be solved by enumerating a finite (and polynomially bounded) number of points.

5 Generalized Systems

5.1 Introduction

In the first half of the last century, there have been several important achievements: a great growth of the theory of extrema in the wake of the so-called Peano Function, the parallel development of the Theorems of the Alternative and of the Separation Theorems, the gathering of important results to form Convex Analysis, and a great development of the Calculus of Variations. Due to these achievements and also to some important problems arisen in Mathematical Physics, like the so-called Signorini Contact Problem, at the beginning of the second half of the last century, the theory of variational inequalities and, in parallel, that of Complementarity Systems have been formulated, the former mainly in infinite dimensional spaces, and the latter in finite dimensional spaces.

All these great developments have lead to search for models, which embody parallel theories. In particular, this has happened in the case of extremum problems, variational inequalities, and generalized systems. A possible unification has been found with the formulation of the so-called generalized systems. As concerns constrained extremum problems, the connection with a system is, nowadays, clear: while “optimization” is merely a language for looking at real problems, the mathematical core lies in stating whether or not a system be impossible.

The results contained in the following Sections are based on [2].

5.2 General Results

Given positive integers n and l , a convex cone $\mathcal{C} \subseteq \mathbb{R}^l$, sets $R, Z \subset \mathbb{R}^n$, and a vector-valued function $F : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^l$ consider the problem \mathcal{P} which consists in finding $y \in R \cap Z$ such that the system, in the unknown x ,

$$F(x; y) \in \mathcal{C}, \quad x \in R \cap Z \tag{77}$$

be impossible.

Given a set $X \subseteq \mathbb{R}^n$ such that $Z \subseteq X$, the replacement of $R \cap Z$ with $R \cap X$ represents a relaxation of the domain of (77); of course, this may change the set of solutions of \mathcal{P} . This drawback can be overcome through a suitable change of the system. To this end, consider a vector-valued function $\Phi : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^l$ and the family $\{\mathcal{P}(\mu)\}_{\mu \in \mathbb{R}}$ of problems where $\mathcal{P}(\mu)$ consists in finding $y \in R \cap X$ such that the system (in the unknown x)

$$F(x; y) + \mu \Phi(x; y) \in \mathcal{C}, \quad x \in R \cap X \quad (78)$$

be impossible. System (78) shows, with respect to (77), a relaxation of the domain and a penalization of F .

In [2] conditions are given under which \mathcal{P} and $\mathcal{P}(\mu)$ are equivalent in the sense that they have the same (possibly empty) set of solutions. To this end, one needs a preliminary proposition.

As concern notation, for every cone $\mathcal{C} \subset \mathbb{R}^l$ and for all $x, y \in \mathcal{C}$, let

$$\begin{aligned} x \geq_{\mathcal{C}} y &\iff x - y \in \mathcal{C}, \\ x \not\geq_{\mathcal{C}} y &\iff x - y \notin \mathcal{C}, \\ x >_{\mathcal{C}} y &\iff x - y \in \text{int } \mathcal{C}, \end{aligned}$$

where $\text{int } \mathcal{C}$ denotes the interior of \mathcal{C} . The notation \leq , $\not\leq$, and $<$ is defined in analogous way. Finally, for the sake of simplicity, and without any fear of confusion, $\|\cdot\|$ denotes either a norm in \mathbb{R}^l or in \mathbb{R}^n , or a matrix norm.

Lemma 4 Let $\mathcal{C}, \mathcal{C}^0 \subset \mathbb{R}^l$ be cones such that \mathcal{C}^0 be closed and $\{0\} \neq \mathcal{C}^0 \subseteq \text{int } \mathcal{C}$. Let $B_\rho = \{x \in \mathbb{R}^l : \|x\| \leq \rho\}$ with $\rho \geq 0$, and $U = \{x \in \mathbb{R}^l : \|x\| = 1\}$. Then, there exists η_0 such that

$$V_1 + \eta V_2 \in \text{int } \mathcal{C}, \quad \forall \eta > \eta_0, \quad \forall V_1 \in B_\rho, \quad \forall V_2 \in \mathcal{C}^0 \cap U \quad (79)$$

Proof Since $\|V_2\| = 1, \forall V_2 \in \mathcal{C}^0 \cap U, \forall \eta > \rho$, one has

$$1 \geq \frac{\langle V_1 + \eta V_2, V_2 \rangle}{\|V_1 + \eta V_2\|} \geq \frac{\langle V_1, V_2 \rangle + \eta \|V_2\|^2}{\|V_1\| \|V_2\| + \eta \|V_2\|} \geq \frac{-\rho + \eta}{\rho + \eta}.$$

Since the scalar product of vectors of unitary norm is 1 if and only if they coincide, passing to the limit as $\eta \rightarrow +\infty$, one obtains

$$\lim_{\eta \rightarrow +\infty} \frac{V_1 + \eta V_2}{\|V_1 + \eta V_2\|} = V_2.$$

Since $\mathcal{C}^0 \cap U$ is a compact set included in $\text{int } \mathcal{C}$, then $U \setminus \mathcal{C}$ and $\mathcal{C}^0 \cap U$ have positive distance (induced by the norm considered). Hence (79) follows. This completes the proof. \square

Note that, when $l = 1$, the assumptions of Lemma (4) are fulfilled only by $\mathcal{C} = \mathbb{R}_+$ and by $\mathcal{C} = \mathbb{R}_+ \setminus \{0\}$ (and, of course, by their opposites); in both cases $\mathcal{C}^0 = \mathbb{R}_+$ necessarily (or $\mathcal{C}^0 = \mathbb{R}_-$). In the sequel, $\forall x \in X$, $p(x)$ will denote a vector belonging to the set $\text{proj}_Z(x)$, where $\text{proj}_Z : X \rightarrow Z$ is the multivalued function which projects x on the compact set Z .

Theorem 33 *Let $R \subseteq \mathbb{R}^n$ be a closed set, $Z \subset X \subset \mathbb{R}^n$, Z and X be compact sets, and let $F, \Phi : X \times X \rightarrow \mathbb{R}^l$ satisfy the following assumptions:*

(H_1) *F is bounded on $X \times X$, and there exist positive reals L and α , and an open set $\Omega \supset Z$, such that*

$$\|F(p(x); x)\| \leq L \|x - p(x)\|^\alpha, \quad \forall x \in \Omega \cap X, \quad \forall p(x) \in \text{proj}_Z(x);$$

(H_2) (i) *Φ is continuous on $X \times X$.*

(ii) *$\forall x, y \in Z$, $\Phi(x, y) = 0$.*

(3i) *there exists a closed cone \mathcal{C}^+ with $\emptyset \neq (\mathcal{C}^+ \setminus \{0\}) \subseteq \text{int } \mathcal{C}$, such that*

$$\Phi(x, y) \in (\mathcal{C}^+ \setminus \{0\}), \quad \forall x \in Z, \quad \forall y \in X \setminus Z.$$

(4i) *$\forall z \in Z$ there exists a neighborhood $S(z)$ of z , and a real $\epsilon(z)$ such that*

$$\|\Phi(p(x); x)\| \geq \epsilon(z) \|x - p(x)\|, \quad \forall x \in S(z) \cap (X \setminus Z), \quad \forall p(x) \in \text{proj}_Z(x).$$

Then a real μ_1 exists, such that, $\forall \mu > \mu_1$, a solution of $\mathcal{P}(\mu)$ is a solution of \mathcal{P} .

Proof To prove the thesis it is sufficient to show that $\exists \mu_0 \in \mathbb{R}$ such that, $\forall \mu > \mu_0$, a solution of $\mathcal{P}(\mu)$ is achieved necessarily at point $z \in R \cap Z$. Because of (H_2)(i), this claim assures that a solution of $\mathcal{P}(\mu)$ is also a solution of \mathcal{P} . Consider the sets $\hat{X} = R \cap X$, $\hat{Z} = R \cap Z$, and $\hat{S}(z) = \Omega \cap S(z)$, where $S(z)$ is precisely that of (H_2)(4i). The family $\{\hat{S}(z), z \in \hat{Z}\}$ is obviously a cover of \hat{Z} . Since \hat{Z} is a closed subset of Z , there is a finite subcover, say $\{\hat{S}(z_i), i = 1, \dots, k\}$, of \hat{Z} . Set $S = \cup_{i=1}^k \hat{S}(z_i)$ and let $\rho = \max\{L/\epsilon(z_i), i = 1, \dots, k\}$. Because of (H_1) and (H_2)(4i), it holds

$$\frac{\|F(p(x); x)\|}{\|\Phi(p(x); x)\|} \leq \rho, \quad \forall x \in S \cap (\hat{X} \setminus \hat{Z}).$$

Consider the set $U = \{x \in \mathbb{R}^l : \|x\| = 1\}$; because of $(H_2)(3i)$ one has

$$\frac{\Phi(p(x); x)}{\|\Phi(p(x); x)\|} \in \mathcal{C}^+ \cap U, \quad \forall x \in \hat{X} \setminus \hat{Z}. \quad (80)$$

Apply [Lemma 4](#), where V_1 , V_2 , \mathcal{C}^0 , and \mathcal{C} are identified with $F(p(x); x)/\|\Phi(p(x); x)\|$, $\Phi(p(x); x)/\|\Phi(p(x); x)\|$, \mathcal{C}^+ , and \mathcal{C} , respectively. The assumptions of [Lemma 4](#) being fulfilled, one obtains the existence of a real η_1 , such that (79) holds, namely, for all $\eta > \eta_1$ and $x \in S \cap (\hat{X} \setminus \hat{Z})$

$$\frac{F(p(x); x)}{\|\Phi(p(x); x)\|} + \eta \frac{\Phi(p(x); x)}{\|\Phi(p(x); x)\|} \in \mathcal{C}. \quad (81)$$

It follows that, $\forall \mu > \eta_1$, $\mathcal{P}(\mu)$ cannot have solutions in $S \cap (\hat{X} \setminus \hat{Z})$.

Now, consider the compact set $X_0 = \hat{X} \setminus S$, and fix $\hat{z} \in \hat{Z}$. Because of $(H_2)(i, 3i)$, Φ is continuous and different from the null vector on the compact set $\{\hat{z}\} \times X_0$. Thus

$$M_\Phi = \min_{x \in X_0} \|\Phi(\hat{z}; x)\| > 0.$$

Apply again [Lemma 4](#), where $\rho = M_F/M_\Phi$, and V_1 , V_2 , \mathcal{C}^0 , and \mathcal{C} are identified with $F(\hat{z}; x)/\|\Phi(\hat{z}; x)\|$, $\Phi(\hat{z}; x)/\|\Phi(\hat{z}; x)\|$, \mathcal{C}^+ , and \mathcal{C} , respectively, and $M_F = \sup_{(x,y) \in X \times X} \|F(x; y)\|$. Then, the hypotheses of [Lemma 4](#) being satisfied, one obtains the existence of η_2 , such that, for all $\eta > \eta_2$ and $x \in X_0$,

$$\frac{F(\hat{z}; x)}{\|\Phi(\hat{z}; x)\|} + \eta \frac{\Phi(\hat{z}; x)}{\|\Phi(\hat{z}; x)\|} \in \mathcal{C}. \quad (82)$$

Hence, $\forall \mu > \eta_2$, $\mathcal{P}(\mu)$ cannot have solutions in X_0 . If $\mu > \mu_1 = \max\{\eta_1, \eta_2\}$, account taken of (81) and (82), $\mathcal{P}(\mu)$ cannot have solutions in $\hat{X} \setminus \hat{Z}$. This completes the proof. \square

Remark 5 Consider the special case where

$$F(x; y) = f(y) - f(x), \quad (83)$$

with $f : X \rightarrow \mathbb{R}^l$ bounded. It is trivial to check that, if f fulfills the Hölder condition on the set Ω , i.e., there exist $L, \alpha > 0$ such that

$$\|f(y) - f(x)\| \leq L \|x - y\|^\alpha, \quad \forall x, y \in \Omega \cap X, \quad (84)$$

then the function F defined in (83) satisfies (H_1) . The converse is not true, as shown by the [Example 2](#).

Note that, for F given by (83), y is a solution of \mathcal{P} if and only if f is a solution of the vector minimum problem

$$\min_{\mathcal{C}} f(x) \quad \text{s.t. } x \in R \cap Z, \quad (85)$$

where \mathcal{C} defines now the partial order and $\min_{\mathcal{C}}$ marks vector minimum: the impossibility of (77) defines y as a solution of (85); y is called vector minimum point, in short VMP.

Example 2 Let $n = l = 1$, $R = \mathbb{R}$, $Z = [0, 1]$, and $X = [-1, 1]$. Consider (83) with $f(x) = x \sin(1/x)$ if $x \neq 0$ and $f(0) = 0$. (H_1) is satisfied at $\alpha = 1$, $L = 1$, and $\Omega = [-2, 2]$. In fact, if $x \in Z$ so that $p(x) = x$, then $F(p(x); x) = 0$; if $x \in (\Omega \cap X) \setminus Z = [-1, 0[$, then $|F(0; x)| = |f(x)| = |x \sin(1/x)| \leq |x|$. Thus (H_1) holds. Obviously f does not fulfill (84).

Remark 6 Consider two special cases. For this, introduce $G : \mathbb{R}^n \rightarrow \mathbb{R}^{l \times n}$, and let $\langle G(u), v \rangle_l$ denote the vector whose i th component is the scalar product between the i th row of the matrix $G(u)$ and the vector v . The former special case is

$$F(x; y) = \langle G(y), y - x \rangle_l; \quad (86)$$

the second case is

$$F(x; y) = \langle G(x), y - x \rangle_l. \quad (87)$$

If G is bounded on \mathbb{R}^n , then both functions F in (86) and (87) fulfill (H_1) at $\alpha = 1$ and $L = \|G\| = \sup_{\|x\|=1} \|G(x)\|$, as it is easy to check. *Example 3* shows that the converse is not true.

Note that y is a solution of \mathcal{P} in case (86) (or (87)) if and only if it is a solution of the Vector Inequality of Stampacchia type [34]: find $y \in R \cap Z$ such that

$$\langle G(y), x - y \rangle_l \not\leq_C 0, \quad \forall x \in R \cap Z, \quad (88)$$

or of the vector variational inequality of Minty type [34]: find $y \in R \cap Z$ such that

$$\langle G(x), y - x \rangle_l \not\geq_C 0, \quad \forall x \in R \cap Z. \quad (89)$$

Example 3 Let $n = l = 1$, $\mathcal{C} = \mathbb{R}^+ \setminus \{0\}$, $R = \mathbb{R}$, $Z = [0, 1]$, and $X = [0, 2]$. Let F be defined by (86) with $G(y) = 1/\sqrt{|1-y|}$, with $y \neq 1$ and $G(1) = 0$; it holds

$$|F(p(x); x)| \leq |x - p(x)|^{1/2}, \quad \forall x \in]1, 2[;$$

indeed, if $x \in Z$, then $p(x) = x$ so that $F(p(x); x) = 0$; if $x \in]1, 2[$, then $p(x) = 1$ so that $|F(p(x); x)| = \sqrt{x-1}$.

Example 4 Let $F : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, $F(x; y) = \sqrt{|x-y|}(x-1)$, $Z = [0, 1]$, and $X = [0, 2]$. In this case a constant L and an open set $\Omega \supset Z$ does not exist such that

$$|F(x; y)| \leq |x-y|, \quad \forall x \in \Omega \cap X, \quad \forall y \in Z.$$

However, the hypothesis (H_1) of [Theorem 33](#) holds with $\Omega = \mathbb{R}$, $L = \sqrt{2}$, and $\alpha = 3/2$.

Example 5 Let $n = l = 1$, $\mathcal{C} = \mathbb{R}^+ \setminus \{0\}$, $R = \mathbb{R}$, $Z = [0, 1]$ and $X = [0, 2]$, $F : X \times X \rightarrow \mathbb{R}$, and $F(x; y) = (x-y)^2(1-y)(x-1)$. Then F fulfills (H_1) with $\Omega =]-1, 2[$, $L = 1$, $\alpha = 2$. In fact, F is bounded; moreover, if $x \in]1, 2[$, $p(x) = 1$ and $F(1; x) = 0$; if $x \in Z$, $p(x) = x$ and $F(x; x) = 0$. Each $y \in [0, 1]$ is a solution of the following problem \mathcal{P} : find $y \in Z$ such that

$$F(x; y) \in \mathcal{C}, \quad x \in [0, 1]$$

be impossible.

Let $\Phi : X \times X \rightarrow \mathbb{R}$ be a penalty function defined as follows:

$$\Phi(x; y) = \begin{cases} -(1-x)^2, & \text{if } (x, y) \in]1, 2] \times [0, 1] \\ 0, & \text{if } (x, y) \in [0, 1] \times [0, 1] \\ (1-y)^2, & \text{if } (x, y) \in [0, 1] \times]1, 2[\\ -(x-y)^2, & \text{if } (x, y) \in]1, 2] \times]1, 2[, y \leq x \\ (x-y)^2, & \text{if } (x, y) \in]1, 2] \times]1, 2[, y > x \end{cases}$$

Then Φ fulfills condition (H_2) choosing $\mathcal{C}^+ = \mathbb{R}_+$, $\epsilon(z) = 1$, $\forall z \in Z$, $\alpha = 2$, and $L = 1$. Note that, $\forall \mu \in \mathbb{R}_+$, $y \in [0, 1]$ is not a solution for $\mathcal{P}(\mu)$. In fact, $\forall x \in]1, 2]$

$$F(x; y) + \mu \Phi(x; y) = (x-1)[(x-y)^2(1-y) - \mu(x-1)].$$

Observe that $\lim_{x \downarrow 1} (x-y)^2(1-y) - \mu(x-1) = (1-y)^3 > 0$. Thus $y \in [0, 1]$ is not a solution of $\mathcal{P}(\mu)$. Furthermore, F does not fulfill (H_3) of the following [Theorem 34](#) at $\alpha = 2$, $L = 1$. In fact, $p(x) = 1$ for $x \geq 1$ and the inequality

$$|F(p(x); y) - F(x; y)| \leq (x-1)^2, \quad \forall x \in X \cap \Omega, \text{ and } \forall p(x) \in \text{proj}_Z(x)$$

holds if and only if

$$(x-y)^2(1-y) \leq (x-1), \quad \forall x > 1, \quad \forall y \in [0, 1].$$

This is impossible for $x = 5/4$, $y = 1/4$.

Moreover, observe that $\mathcal{P}(\mu)$ has no solution: for each $y \in [1, 2]$, $x \in [0, 1]$, and $\mu > 0$

$$F(x; y) + \mu\Phi(x; y) \in \mathcal{C}.$$

If Z is finite, then the inequality in $(H_2)(4i)$ can be equivalently replaced (in the sense that the thesis of the [Theorem 33](#) is still valid and the class of the penalty functions Φ which satisfy it is nonempty) by the following condition:

$\forall z \in Z$ there exist a neighborhood $S(z)$ and a real $\epsilon(z) > 0$, such that

$$\|\Phi(z; x)\| \geq \epsilon(z) \|x - z\|, \quad \forall x \in S(z) \cap (X \setminus Z).$$

In fact, choosing a suitable neighborhood $S(z)$ of z , one has $p(x) = z$.

If Z is not finite, then the above condition might be in contrast with Assumptions $(H_2)(i, ii)$. In this case, the latter is replaced by the following one: $(H_2)'$ There exists a vector-valued function $\varphi : X \rightarrow \mathbb{R}^l$, such that:

- (i) φ is continuous on X .
- (ii) $\forall x \in Z$, $\varphi(x) = 0$.
- (3i) There exists a closed cone \mathcal{C}^+ with $\{0\} \neq \mathcal{C}^+ \subseteq \text{int } \mathcal{C}$, such that

$$\varphi(x) \in \mathcal{C}^+ \setminus \{0\}, \quad \forall x \in X \setminus Z.$$

(4i) $\forall z \in Z$; there exist a neighborhood $S(z)$ and a real $\epsilon(z) > 0$ such that

$$\|\varphi(x)\| \geq \epsilon(z) \|x - p(x)\|^\alpha, \quad \forall x \in S(z) \cap (X \setminus Z), \text{ and } \forall p(x) \in \text{proj}_Z(x).$$

Note that if $\Phi(x; y) = \varphi(x) - \varphi(y)$, $\forall x, y \in X$, then (H_2) of [Theorem 33](#) is fulfilled if $(H_2)'$ is fulfilled.

The following theorem deals with the special class of problem \mathcal{P} where the cone \mathcal{C} is equal to \mathbb{R}_+^l ; it gives a condition guaranteeing that a solution of \mathcal{P} is also a solution of a suitable problem $\mathcal{P}(\mu)$, for μ large enough.

Theorem 34 *Let $R \subseteq \mathbb{R}^n$ be a closed set, $Z \subset X \subset \mathbb{R}^n$, Z , and X be compact set, and let $F, \Phi : X \times X \rightarrow \mathbb{R}^l$ satisfy the following hypotheses:*

(H_3) *F is bounded on $X \times X$ and there exist positive reals L , α and an open set $\Omega \supset Z$, such that*

$$\|F(x; y) - F(p(x); y)\| \leq L \|x - p(x)\|^\alpha, \quad \forall x, y \in \Omega \cap X, \quad \forall p(x) \in \text{proj}_Z(x)$$

(H_4)

- (i) Φ is continuous on $X \times X$.
- (ii) $\forall x, y \in Z$, $\Phi(x; y) = 0$, $\forall x \in X$, $\Phi(x; \cdot)$ is constant on Z .
- (3i) There exists a closed cone \mathcal{C}^- , with $\{0\} \neq \mathcal{C}^- \subseteq \text{int } (-\mathcal{C})$, such that

$$\Phi(x; y) \in \mathcal{C}^- \setminus \{0\}, \quad \forall x \in X \setminus Z, \quad \forall y \in Z.$$

(4i) $\forall z \in Z$; there exists a neighborhood $S(z)$ of z and a positive real $\epsilon(z)$ such that

$$\|\Phi(x; p(x))\| \geq \epsilon(z) \|x - p(x)\|^\alpha, \quad \forall x, y \in S(z) \cap (X \setminus Z), \quad \forall p(x) \in \text{proj}_Z(x).$$

Then, there exists $\mu_2 \in \mathbb{R}$, such that, $\forall \mu > \mu_2$, a solution of \mathcal{P} is a solution of $\mathcal{P}(\mu)$.

Note that hypothesis (H_3) of [Theorem 34](#) can be weakened by replacing the requirement $x, y \in \Omega \cap X$ with $x \in \Omega \cap X, y \in Z$.

5.3 The Case of a Discrete Domain

This section deals with a special but very important case, namely, that of a discrete problem. More precisely, consider the case

$$Z = \mathbb{B}^n, \quad \mathcal{C} = \mathbb{R}_+^l \setminus \{0\}, \quad \mathcal{C}^+ = \{v \in \mathbb{R}_+^l : v_1 = \dots = v_l\}, \quad \mathcal{C}^- = -\mathcal{C}^+, \quad (90)$$

where $\mathbb{B} = \{0, 1\}$; the case where Z is a bounded subset of \mathbb{Z}^n can be reduced to the above one by well-known transformations [61].

The relaxation of $Z = \mathbb{B}^n$ is $X = X_Q = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, i = 1, \dots, n\}$ and the penalty term $\Phi : X \times X \rightarrow \mathbb{R}^l$ is defined by

$$\Phi(x; y) = \begin{pmatrix} \varphi(y) - \varphi(x) \\ \vdots \\ \varphi(y) - \varphi(x) \end{pmatrix} \quad (91)$$

where $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$, $\varphi(x) = \langle x, e - x \rangle$, with $e = (1, \dots, 1) \in \mathbb{R}^n$. Under Assumptions (90), the function defined in (91) fulfills, with $\alpha = 1$, the conditions (H_2) of [Theorem 33](#) and (H_4) of [Theorem 34](#). In fact, $(H_2)(i)$, (ii) , $(3i)$, $(H_4)(i)$, (ii) , $(3i)$ are obvious. As concerns $(H_2)(4i)$ and $(H_4)(4i)$, note that if $S(z)$ is small enough so that $p(x) = z$, then, as proved in [36], φ satisfies the following conditions: $\forall z \in Z$ there is a neighborhood $S(z)$ of z and a real $\hat{\epsilon}(z)$, such that

$$\varphi(x) \geq \hat{\epsilon} \|x - z\|, \quad \forall x \in S(z) \cap (X \setminus Z).$$

Therefore for all $z \in Z$ and $x \in S(z) \cap (X \setminus Z)$

$$\|\Phi(p(x); x)\| = \|\Phi(x; p(x))\| = \begin{pmatrix} \varphi(x) \\ \vdots \\ \varphi(x) \end{pmatrix} = \sqrt{l}|\varphi(x)| \geq \sqrt{l}\hat{\epsilon}(z) \|x - z\|,$$

which proves $(H_2)(4i)$ and $(H_4)(4i)$ by setting $\epsilon(z) = \sqrt{l}\hat{\epsilon}(z)$. Hence, the following result has been established:

Theorem 35 *For the case (90)–(91), let the function F verify with $\alpha = 1$ the hypothesis (H_3) of Theorem 34, and assume that $F(x; x) = 0$ for all $x \in \Omega$. Then, there exists $\mu_3 \in \mathbb{R}$ such that, $\forall \mu > \mu_3$, \mathcal{P} and $\mathcal{P}(\mu)$ have the same set of solutions.*

Remark 7 Any function $F : X \times X \rightarrow \mathbb{R}^l$ of the form

$$F(x; y) = f(x) - f(y),$$

where $f : X \rightarrow \mathbb{R}^l$, or

$$F(x; y) = \langle G(y), y - x \rangle,$$

where $G : \mathbb{R}^n \rightarrow \mathbb{R}^{l \times n}$, fulfills the condition $F(x; x) = 0$ for all $x \in \Omega$.

In Theorem 35 the hypothesis (H_3) with $\alpha = 1$ was sufficient for the special Φ chosen. For $\alpha = 1$, the above theorem is still valid for all strictly concave functions $\varphi_1, \dots, \varphi_l : \mathbb{R}^n \rightarrow \mathbb{R}$ such that $\varphi_i(x) = 0$ for all $i = 1, \dots, l$ and $x \in Z$, and there exist a neighborhood $S_i(y)$ and a real constant $\epsilon_i(y)$ such that

$$|\varphi_i(x)| \geq \epsilon_i(y) \|x - y\|, \quad \forall x \in S_i(y) \cap (X \setminus Z).$$

Note that for $i = 1, \dots, l$, the above condition is a slight generalization of the condition on the function φ in [36], and it is equivalent to (H_2) of Theorem 33 when $l = 1$. Then one can put $h = (\varphi_1, \dots, \varphi_l)$, and, $\forall x, y \in X$,

$$\Phi(x; y) = h(y) - h(x). \tag{92}$$

Condition $(H_2)(4i)$ for Φ follows choosing $\forall z \in Z$, $S(z) = \cap_{i=1}^l S_i(z)$, and $\epsilon(z) = \sqrt{l} \min\{\epsilon_i(z), i = 1, \dots, l\}$.

The following theorem gives a condition which assures that, $\forall y \in X$; the function $F(\cdot; y) + \mu\Phi(\cdot; y)$ is component-wise strictly convex. This is a straightforward extension of Theorem 3.2 of [36] and Theorem 2 of [37].

Theorem 36 *In the case (90)–(91), let $F : X \times X \rightarrow \mathbb{R}^l$ fulfill the hypotheses of Theorem 35 with $\alpha = 1$. If $F \in \mathcal{C}^2(X \times X)$, then there exists a real μ_4 such that, $\forall \mu > \mu_4$, \mathcal{P} and $\mathcal{P}(\mu)$ have the same solutions, and $\forall y \in X$, the function $F(\cdot; y) + \mu\Phi(\cdot; y)$ is component-wise strictly convex.*

Proof $\forall y \in X$, let $H_i(x; y)$ and $\hat{H}_i(x; y)$ be the Hessian matrices at x , of the i -th component of $F(\cdot; y)$ and of $F(\cdot; y) + \mu\Phi(\cdot; y)$ respectively. Hence, for $i = 1, \dots, l$

$$\hat{H}_i(x; y) = H_i(x; y) + 2\mu I_n,$$

where I_n denotes the $n \times n$ identity matrix. $\forall i \in \{1, \dots, l\}$, $\forall r \in \{1, \dots, n\}$, let $\lambda_{ir} : X \times X \rightarrow \mathbb{R}$ be the function where $\lambda_{ir}(x; y)$, $r = 1, \dots, n$ are the eigenvalues of the Hessian $H_i(x; y)$. Because of the continuity of F , λ_{ir} is continuous. Hence

$$\eta_4 := \max_{i,r} \max_{X \times X} |\lambda_{ir}(x; y)| < +\infty.$$

Moreover, observe that, $\forall x, y$, $v_i(x; y)$ is an eigenvalue of $\hat{H}_i(x; y)$ if and only if $v_i(x; y) - 2\mu$ is an eigenvalue of $H_i(x; y)$. Then, if

$$\mu > \mu_4 = \frac{1}{2} \max\{\mu_1, \mu_2, \eta_4\}$$

it follows that, for all $(x, y) \in X \times X$, $v_{ir}(x; y) = \lambda_{ir}(x; y) + 2\mu$ is an eigenvalue of $\hat{H}_i(x; y)$ and it results $v_{ir}(x; y) > 0$. This completes the proof. \square

5.4 Discrete Vector Optimization and Variational Inequalities

In this section, equivalence results for vector optimization problems and vector variational inequalities are obtained as a direct application of the preceding theorems.

Indeed, consider the special case of (85), where $Z = \mathbb{B}^n$ and $\mathcal{C} = \mathbb{R}_+^l \setminus \{0\}$, namely, consider the vector minimization problem:

$$\min_{\mathcal{C}} f(x) \quad \text{s.t. } x \in R \cap \mathbb{B}^n. \quad (93)$$

It is well known that a point y is a solution of (93) if and only if the system

$$f(y) - f(x) \in \mathcal{C}, \quad x \in R \cap \mathbb{B}^n, \quad (94)$$

is impossible. (When $l = 1$ and $\mathcal{C} = (0, +\infty)$ the problem (93) becomes a scalar 0–1 optimization problem, and the impossibility of (94) becomes the impossibility of the inequality $f(y) - f(x) > 0$, $x \in R \cap \mathbb{B}^n$.)

Then, consider the vector minimum problem:

$$\min_{\mathcal{C}} [f(x) + \mu\psi(x)] \quad \text{s.t. } x \in R \cap X_Q, \quad (95)$$

where $\psi : \mathbb{R}^n \rightarrow \mathbb{R}^l$, $\psi = (\varphi, \dots, \varphi)$, $\varphi(x) = \langle x, e - x \rangle$, with $e = (1, \dots, 1)$, $\mu \in \mathbb{R}$, $X_Q = \{x \in \mathbb{R}^n : 0 \leq x_i \leq 1, i = 1, \dots, n\}$.

Corollary 7 *Let the following hypotheses be satisfied:*

(H_5) $f : \mathbb{R}^n \rightarrow \mathbb{R}^l$ is bounded on X_Q , and there exist a positive real L and an open set $\Omega \supset \mathbb{B}^n$ such that

$$|f_i(x) - f_i(y)| \leq L \|x - y\|, \quad \forall x, y \in \Omega \cap X_Q, \quad i = 1, \dots, l,$$

where f_i denotes the i -th component of f . Then, there exists a real $\mu_5 \in \mathbb{R}$, such that, $\forall \mu > \mu_5$, (93) and (95) have the same solutions. If, in addition, $f \in C^2(X)$, then there exists $\mu_6 \in \mathbb{R}$ such that, $\forall \mu > \mu_7 = \max\{\mu_5, \mu_6\}$, $f + \mu\psi$ is component-wise strictly concave.

Proof Put $\Phi : X \times X \rightarrow \mathbb{R}^l$, $\Phi(x; y) = \psi(y) - \psi(x)$. According to Remark 5, under Assumptions (H_5), the function $F(x; y) = f(y) - f(x)$ satisfies (H_1) of Theorem 33 and (H_3) of Theorem 34. Moreover, Assumption (H_2) of Theorem 33 and (H_4) of Theorem 34 is fulfilled by the present Φ given by (91), as shown in Sect. 5.3. As concerns the second part of the thesis, it is enough to note that (93) and (95) are equivalent to \mathcal{P} and $\mathcal{P}(\mu)$, respectively. Hence, Theorem 36 can be applied. This completes the proof. \square

In the special—but important—case where R is a polyhedron, Corollary 7 shows that the class of vector minimization problems with component-wise strictly concave objective function (95) has all vector minimum points necessarily at the vertices of the feasible region. This is not true in general. Indeed, while the next theorem shows that component-wise quasi-concavity guarantees the existence of a vector minimum point at a vertex of a polytope, the following Examples 6 and 7 show that component-wise strict concavity need not imply that *all* vector minimum points must be among the vertices of a polytope.

Theorem 37 *Let $P \subset \mathbb{R}^n$ be a nonempty polytope and let $g := (g_1, \dots, g_l) : \mathbb{R}^n \rightarrow \mathbb{R}^l$ be such that either g_i is quasi-concave for all i (component-wise quasi-concavity) or there exists an index i_0 for which g_{i_0} is strictly quasi-concave. Then, at least one vector minimum point of the problem*

$$\min_C g(x) \quad \text{s.t. } x \in P \tag{96}$$

is attained at a vertex of P .

Proof Without loss of generality assume that $i_0 = 1$. For $i = 1, \dots, l$, consider the problem

$$a_i := \min g_i(x) \quad \text{s.t. } x \in S_{i-1},$$

where

$$S_0 = P, \quad S_i := \operatorname{argmin}_{x \in S_{i-1}} g_i(x), \quad i = 1, \dots, l.$$

Note that

$$S_i \subseteq S_{i-1}, \quad i = 1, \dots, l.$$

If g_i is quasi-concave for all i , then the sets S_1, \dots, S_l are unions of faces of P . On the other hand, if g_1 is strictly concave, then S_1 , and thus also every S_2, \dots, S_l , is a subset of the vertices of P . It will be now shown that each element of S_l is a VMP of (96) so that the thesis will follow. Consider any $x^0 \in S_l$. Ab absurdo, suppose that x^0 be not solution of (96). Then, $\exists y_{x^0} \in P$, such that

$$g(y_{x^0}) \leq_c g(x^0),$$

so that $\exists i_{x^0} \in \{i = 1, \dots, l\}$ such that

$$g_{i_{x^0}}(y_{x^0}) < g_{i_{x^0}}(x^0) = a_{i_{x^0}}, \quad (97)$$

$$g_{i_{x^0}}(y_{x^0}) \leq g_i(x^0), \quad \forall i = 1, \dots, l, \quad i \neq i_{x^0}. \quad (98)$$

Then y_{x^0} must belong to S_l . Otherwise, $\forall i = 0, \dots, l-1$,

$$y_{x^0} \in S_i \setminus S_{i+1} \implies g_{i+1}(y_{x^0}) > g_{i+1}(x^0) = a_{i+1},$$

which contradicts (97) and (98). Since

$$g(x) = (a_1, \dots, a_l), \quad \forall x \in S_l,$$

it follows that $g(y_{x^0}) = (a_1, \dots, a_l)$, which contradicts (97). This completes the proof. \square

Remark 8 If, for any $k \in \{1, \dots, l\}$, S_k is a singleton, then obviously its (unique) element is a VMP of (96), and the subsequent S_i coincide with S_k . This observation suggests a method for finding a solution of (96). However, this method does not necessarily find all VMPs; for instance, even if g is component-wise strictly concave, the method cannot find the VMPs (if any) which fall in $\text{int } P$ whatever the ordering of the component of g may be. The following examples show some strictly concave cases where there are non-vertex solutions even under the further assumption that the (global) maximum points of all g_i fall into the interior of P .

Example 6 In (96), put $n = 1$, $P = [0, 1]$, $\mathcal{C} = \mathbb{R}_+ \setminus \{0\}$, $g_1(x) = 1 - x^2$, and $g_2(x) = x(2 - x)$; it is easy to check that every element of P is a VMP.

Example 7 In (96), put $n = 1$, $P = [-3, 3]$, $\mathcal{C} = \mathbb{R}_+ \setminus \{0\}$, $g_1(x) = (x + 3)(7 - x)$, and $g_2(x) = (3 - x)(x + 7)$. It is easy to check that the VMP are now ± 3 and all the elements of $]-1, 1[$.

Now, consider the vector variational inequality

$$\text{find } y \in R \cap Z \text{ s.t. } \langle G(y), x - y \rangle_l \not\leq_C 0, \forall x \in R \cap Z, \quad (99)$$

where $Z = \mathbb{B}^n$ and $C = \mathbb{R}_+^l \setminus \{0\}$. As noted in [Remark 6](#), y solves [\(Eq. 99\)](#) if and only if it solves the problem \mathcal{P} given by the following: find $y \in R \cap Z$ such that

$$\langle G(y), y - x \rangle_l \in C, x \in R \cap Z$$

is impossible.

Similarly, y solves the vector variational inequality:

$$\text{find } y \in R \cap X_Q \text{ s.t. } \langle G(y), x - y \rangle_l - \mu \Phi(x; y) \not\leq_C 0, \forall x \in R \cap X_Q, \quad (100)$$

where Φ is the function in [\(91\)](#) and $\mu \in \mathbb{R}_+$, if and only if y solves the problem $\mathcal{P}(\mu)$ given by the following: find $y \in R \cap X_Q$ such that

$$\langle G(y), y - x \rangle_l + \mu \Phi(x; y) \in C, x \in R \cap X_Q$$

is impossible. So, next result is a direct application of [Theorems 33](#) and [34](#).

Corollary 8 *Let $G : \mathbb{R}^n \rightarrow \mathbb{R}^{l \times n}$ be bounded on X_Q . Then, there exists a real $\mu_8 \in \mathbb{R}$, such that, $\forall \mu > \mu_8$, [\(99\)](#) and [\(100\)](#) have the same set of solutions.*

Proof The function F defined by $F(x; y) = \langle G(y), y - x \rangle$ fulfills (H_1) of [Theorem 33](#) according to [Remark 6](#). Moreover, Assumption (H_3) of [Theorem 34](#) holds, since

$$\begin{aligned} \| F(x_1; y) - F(x_2; y) \| &= \| \langle G(y), y - x_1 \rangle - \langle G(y), y - x_2 \rangle \| \\ &\leq \| G \| \| x_1 - x_2 \|, \forall x_1, x_2, y \in X_Q. \end{aligned}$$

The present Φ fulfills (H_2) of [Theorem 33](#) and (H_4) of [Theorem 34](#) as shown in [Sect. 5.3](#). Hence, [Theorems 33](#) and [34](#) guarantee the existence of a real $\mu_8 \in \mathbb{R}$ such that \mathcal{P} and $\mathcal{P}(\mu)$, and then [\(99\)](#) and [\(100\)](#) have the same set of solutions. \square

In the special—but important—case where R is a polyhedron, [Corollary 8](#) provides a class of vector variational inequalities with bounded operator, i.e., [\(100\)](#), having—because of the equivalence with [\(99\)](#)—a solution necessarily at a vertex of the domain. This is not true in general, as simple examples show.

6 Conclusion

The developments of the previous sections suggest some interesting topics that should be further investigated. A first question concerns the identification of a class of constrained extremum problems, or Complementarity Systems, or generalized

systems, or Variational Inequalities for which the exact penalty parameter can be improved. Some results on this topic have been presented in Sect. 2.3.

Some of the assumptions of Theorems 2 or 3 might be weakened; in the general case, \mathbb{R}^n might be replaced by a metric space. For instance, isoperimetric problems or optimal control problems, where the feasible region has only a discrete or even finite number of curves, might be equivalently transformed into a classic differentiable problems which admits Euler equation as necessary condition.

In Sect. 2.2, it has been observed that integer programs can be reduced to (16). However, such a reduction implies, in general, an enormous increase in the number of variables. Hence, it is interesting to extend the analysis to the case where $\{0, 1\}^n$ is replaced by \mathbb{Z}^n . In this case, functions like $\sum_{j=1}^n \sin(\pi x_j)$ or $\sum_{j=1}^n \prod_{s=0}^{L_j} (x_j - s)^2$, with L_j upper bound on x_j , might be used as penalty functions. Furthermore, for problem (16)—or, more generally, when Z is finite—it is interesting to investigate other types of penalty functions, e.g., in the field of gauge functions. The idea of resolvent for a Boolean problem [42] might be transferred to a concave minimization problem through the equivalence established in Sects. 2.2 and 3.4.

Theorem 4 states a connection between 0–1 programming problems and complementarity problems, which are a special type (when the domain is a cone) of variational inequalities. Hence, one has the possibility of connecting two fields very far from each other. For instance, fixed-point methods or gap function methods, which have been designed for variational inequalities, may be adapted to 0–1 problems. On the other hand, cutting plane methods and the related group theory, as well as other methods conceived for integer programs, may be transferred to variational inequalities.

It is well known that the concept of saddle of the Lagrangian function associated to a problem like (18) can be used to achieve sufficient optimality conditions (beside necessary ones). Through the equivalence such a concept can be introduced for 0–1 programs.

The equivalence between nonlinear and 0–1 programs can be used also in the context of duality. For instance, it can be used to close the duality gap when Lagrangian duality is applied to facial constraint [55]. Since the optimality of a constrained extremum problem can be put in terms of the impossibility of a suitable system of inequalities, it might be useful to extend the connections between continuous and discrete problems to a system of inequalities, recovering constrained extremum problems as special cases. This could also embrace vector extremum problems.

General classes of functions that attain their minimum at the vertices of a polyhedron have been described in Sect. 3. It is interesting to investigate the possibility of extending the methods employed for concave minimization problems [11, 46, 62] to this more general case.

Consider a special case of (85), where $R = \mathbb{R}^n$, $Z = \cup_{k=1}^r Z_k$, with Z_k convex and compact, $f : \mathbb{R}^n \rightarrow \mathbb{R}^l$ component-wise convex, and $\mathcal{C} = \mathbb{R}_+^l \setminus \{0\}$, namely, the problem

$$\min_{\mathcal{C}} f(x), \quad \text{s.t. } x \in \cup_{k=1}^r Z_k; \quad (101)$$

y is a solution of (101) if and only if the system (in the unknown x)

$$f(y) - f(x) \in \mathcal{C}, \quad x \in \cup_{k=1}^r Z_k \quad (102)$$

is impossible. Consider compact sets $X_k \supseteq Z_k$ and functions $h_k : \cup_{k=1}^r X_k \rightarrow \mathbb{R}$, $k = 1, \dots, r$, such that there is an $\alpha \in \mathbb{R}$ for which each h_k fulfills $(H_2)'$ of Sect. 5.2 with $l = 1$, $\mathcal{C}^+ = \mathbb{R}_+$. It is easily seen that

$$H(x) := \prod_{k=1}^r h_k(x)^{\alpha/r} \quad (103)$$

fulfills $(H_2)'$. In fact, it is trivial to verify $(H_2)'(i)$, (ii) , and $(3i)$. In the following formulas, k as index will denote that one is referring to Z_k and X_k instead of Z and X . H fulfills also $(4i)$:

$$\begin{aligned} H(x) &= \prod_{k=1}^r h_k(x)^{\alpha/r} \geq \prod_{k=1}^r [\epsilon_k(z)^{\alpha/r} \|x - p_k(x)\|]^{\alpha/r} \\ &= \prod_{k=1}^r \epsilon_k(z)^{\alpha/r} \prod_{k=1}^r \|x - p_k(x)\|^{\alpha/r} \geq \prod_{k=1}^r \epsilon_k(z)^{\alpha/r} \|x - p(x)\|^\alpha \\ &= \tilde{\epsilon}(z) \|x - p(x)\|^\alpha, \quad \tilde{\epsilon}(z) = \prod_{k=1}^r \epsilon_k(z)^{\alpha/r}, \end{aligned}$$

where the first inequality comes from $(H_2)'$ at $\varphi = h_k$ and the second inequality is due to the fact that $\|x - p(x)\| = \min\{\|x - p_k(x)\|, k = 1, \dots, r\}$. The function $\Phi : X \times X \rightarrow \mathbb{R}^l$ defined by

$$\Phi(x; y) = \begin{pmatrix} H(y) - H(x) \\ \vdots \\ H(y) - H(x) \end{pmatrix} \quad (104)$$

can be chosen as ‘‘penalty term.’’ The decomposition (103) may help in setting up H and in conceiving solution methods. For instance, if Z_k is a polytope defined by

$$Z_k = \{x \in \mathbb{R}^n; A^k x \geq b^k\}, \quad k = 1, \dots, r,$$

where $A^k \in \mathbb{R}^{m_k \times n}$, $b^k \in \mathbb{R}^{m_k}$, then one can set

$$h_k(x) = \max \left\{ 0, \exp \left(-\alpha_i^k \sum_{j=1}^n a_{ij}^k x_j - b_i^k \right) - 1, i = 1, \dots, m_k \right\},$$

where $A^k = (a_{ij}^k, i = 1, \dots, m_k, j = 1, \dots, n)$, $b^k = (b_1^k, \dots, b_{m_k}^k)^T$, α_i^k being positive parameters. A particular but interesting case is that where Z is not convex, while the sets Z_i , X_i , and $\cup_{i=1}^r X_i$ are all convex.

[Corollary 7](#) suggests a method for solving (93) that is based on the theory introduced in [46, 77] and that will be shortly outlined. To this end, consider the special, but common, case where R is a polytope. Because of [Corollary 7](#), the combinatorial vector problem (93) can be replaced by the continuous vector problem (95). If μ is large enough (i.e., $\mu > \mu_7$), then, because of [Theorem 37](#) (see [Remark 8](#)), a VMP of (95) is a vertex of $P \cap X_Q$. Hence, a method could be the following. The first step must consist in finding a local VMP of (95). Then, consider the family of strictly concave (scalar) problems:

$$\min g_i(x; \mu), \quad x \in P \cap X_Q; i = 1, \dots, l, \quad (105)$$

where $g_i(x; \mu) := f_i(x) + \mu \varphi(x)$. If i is such that x^0 is a local¹ (scalar) minimum point of (105), then Tuy's Theory [46, 77] provides a “cutting halfspace,” say H_i , such that

$$x^0 \notin H_i; \quad \left\{ \begin{array}{l} x \in P \cap X_Q \\ g_i(x; \mu) < g_i(x^0; \mu) \end{array} \right\} \implies x \in (P \cap H_i) \cap X_Q. \quad (106)$$

This condition means that, if there exists an x at which g_i takes a value less than $g_i(x^0; \mu)$, then x must belong to H_i . For all other indices i Tuy's “cutting halfspace” collapses to a supporting halfspace of $P \cap X_Q$; this will be denoted again by H_i . The former (latter) set of indexes will be denoted by I^+ (respectively I^-). If $I^+ \neq \emptyset$, then from (106) one can easily deduce that

$$\left\{ \begin{array}{l} x \in P \cap X_Q \\ g(x; \mu) <_c g(x^0; \mu) \end{array} \right\} \implies x \in P \cap \left(\cap_{i \in I^+} H_i \right) \cap X_Q. \quad (107)$$

This condition means that, if there exists an x at which g takes a value less (in vector sense, with respect to C) than $g(x^0; \mu)$, then x must belong to $\cap_{i \in I^+} H_i$; hence, such an intersection plays a role for vector problems as Tuy's cut does for scalar ones. The case $I^+ = \emptyset$ is a degenerate one for all Tuy's cuts and requires a special analysis: the present vertex can be replaced by any of the adjacent vertices, since they are alternative local VMP. From (107) one derives the condition:

¹In the sense of not necessarily global.

$$I^+ \neq \emptyset, \quad P_1 \cap X_Q = \emptyset, \quad (108)$$

where

$$P_1 := P \cap \left(\cap_{i \in I^+} H_i \right),$$

is a sufficient condition for x^0 to be a VMP of (93) at $R = P$. If (108) is not satisfied, then one can replace P with P_1 in (105), and repeat the above argument. Noting that the set

$$P \cap \left(\cap_{i \in I^+} (\mathbb{R}^n \setminus H_i) \right) \cap X_Q$$

does not contain any alternative VMP of (95), while they might belong to the set

$$P \cap (\mathbb{R}^n \setminus H_r) \cap \left(\cap_{i \in I^+ \setminus \{r\}} H_i \right) \cap X_Q, \quad r \in I^+.$$

An interesting application of the above decomposition should be to the case where (101) is replaced by a vector variational inequality.

According to Remark 8, an alternative method for finding a VMP of (93) may consists in solving l scalar problems, having a strictly concave objective function and a union of vertices of P as feasible region.

Cross-References

- ▶ [Algorithms and Metaheuristics for Combinatorial Matrices](#)
- ▶ [Dual Integrality in Combinatorial Optimization](#)
- ▶ [Energy Efficiency in Wireless Networks](#)
- ▶ [Fault-Tolerant Facility Allocation](#)
- ▶ [Network Optimization](#)
- ▶ [Partition in High Dimensional Spaces](#)
- ▶ [Probabilistic Verification and Non-approximability](#)

Recommended Reading

1. J. Abello, S. Butenko, P.M. Pardalos, M.G.C. Resende, Finding independent sets in a graph using continuous multivariable polynomial formulations. *J. Global Optim.* **21**, 111–137 (2001)
2. C. Antoni, F. Giannessi, On the equivalence, via relaxation-penalization, between vector generalized systems. *Acta Vietnam.* **22**, 567–588 (1997)
3. C. Antoni, Constrained optimization: equivalence theorems for mixed-integer problems via exact penalty. Technical report, Naval Academy, Livorno, Viale Italia 72, 2011
4. C. Antoni, Duality for generalized systems, in *The Proceedings of the Workshop “Equilibrium Problems with Side Constraint. Lagrangean Theory and Duality II”*, Scilla, Reggio Calabria, 1996, pp. 11–18

5. C. Antoni, M. Pedrazzoli, Estimate of penalty parameters. Technical report, Naval Academy, Livorno, Viale Italia 72, 2011
6. F. Aurenhammer, Voronoi diagrams – a survey of a fundamental geometric data structure. *Comput. Surv.* **23**, 345–405 (1991)
7. D. Avis, B.K. Bhattacharya, Algorithms for computing d-dimensional Voronoi diagrams and their duals, in *Advances in Computing Research, Computational Geometry*, vol. 1, ed. by F.P. Preparata (JAI Press, 1983), pp. 159–180
8. D. Avis, K. Fukuda, Reverse search for enumeration. *Discrete Appl. Math.* **65**, 21–46 (1996)
9. M. Avriel, W.E. Diewert, S. Schaible, I. Zang, *Mathematical Concepts and Methods in Science and Engineering*, vol. 36 (Plenum, New York, 1988)
10. E. Balas, Disjunctive programming. *Ann. Discrete Math.* **5**, 3–51 (1979)
11. H.P. Benson, Concave minimization: theory, applications and algorithms, in *Handbook of Global Optimization*, ed. by R. Horst, P.M. Pardalos (Kluwer Academic, Boston, 1995), pp. 43–148
12. E. Blum, W. Oettli, Direct proof of the existence theorem in quadratic programming. *Oper. Res.* **20**, 165–167 (1972)
13. I.M. Bomze, On standard quadratic optimization problems. *J. Global Optim.* **13**, 369–387 (1998)
14. I.M. Bomze, Copositivity aspects of standard quadratic optimization problems, in *Optimization, Dynamics, and Economic Analysis* (Physica, Heidelberg, 2000), pp. 1–11
15. I.M. Bomze, Branch-and-bound approaches to standard quadratic optimization problems. *J. Global Optim.* **22**, 17–37 (2002)
16. I.M. Bomze, E. De Klerk, Solving standard quadratic optimization problems via linear, semidefinite and copositive programming. *J. Global Optim.* **24**, 163–185 (2002)
17. E. Boros, P.L. Hammer, Pseudo-Boolean optimization. *Discrete Appl. Math.* **123**, 155–225 (2002)
18. R. Burkard, B. Klinz, R. Rudolf, Perspective of monge properties in optimization. *Discrete Appl. Math.* **70**, 95–161 (1996)
19. R.E. Burkard, H. Hamacher, J. Tind, On abstract duality in mathematical programming. Report Mathematisches Institut Universitatzukoln - Angewandte Mathematik, 1981
20. V. Cabot, R.L. Francis, M.A. Stuart, A network flow solution to a rectilinear distance facility location problem. *AIIE Trans.* **2**, 132–141 (1970)
21. D.-Z. Du, Minimax and its applications, in *Handbook of Global Optimization*, ed. by R. Horst, P.M. Pardalos (Kluwer Academic, Boston, 1995), 339–367
22. D.-Z. Du, F.K. Hwang, A proof of Gilbert-Pollak conjecture on the Steiner ratio. *Algorithmica* **7**, 121–135 (1992)
23. D.-Z. Du, P.M. Pardalos (eds.), *Minimax and Applications* (Kluwer Academic, Boston, 1995)
24. D.-Z. Du, P.M. Pardalos, A continuous version of a result of Du and Hwang. *J. Global Optim.* **5**, 127–130 (1994)
25. N. Dunford, J.T. Schwartz, *Linear Operator, Part I* (Interscience, New York, 1958)
26. F. Giannessi, Theorems of alternative, quadratic programs and complementarity problems, in *Variational Inequalities and Complementarity Problems*, ed. by R.W. Cottle, F. Giannessi, J.-L. Lions (Wiley, New York, 1980), pp. 151–186
27. B. Eaves, On quadratic programming. *Manag. Sci.* **17**, 698–711 (1971)
28. E. Erkut, S. Neumann, Analytical models for locating undesirable facilities. *Eur. J. Oper. Res.* **40**, 275–291 (1989)
29. P. Favati, F. Tardella, Convexity in nonlinear integer programming. *Ricerca Oper.* **53**, 3–44 (1990)
30. M. Frank, P. Wolfe, An algorithm for quadratic programming. *Naval Res. Logist. Q.* **3**, 95–110 (1956)
31. S. Fujishige, *Annals of Discrete Mathematics*, vol. 47 (North-Holland, Amsterdam, 1991)
32. K. Fukuda, T.M. Liebling, F. Margot, Analysis of backtrack algorithms for listing all vertices and all faces of a convex polyhedron. *Comput. Geom.* **8**, 1–12 (1997)

33. P. Gahinet, P. Apkarian, M. Chilali, Affine parameter-dependent Lyapunov functions and real parametric uncertainty. *IEEE Trans. Autom. Control* **41**, 436–442 (1996)
34. F. Giannessi, On minty variational principle, in *Trend Mathematical Programming, Proceeding of a Conference* held in Matrafured, 1996, pp. 161–176
35. F. Giannessi, On some connections among variational inequalities, combinatorial and continuous optimization. *Ann. Oper. Res.* **58**, 181–200 (1995)
36. F. Giannessi, F. Niccolucci, Connections between nonlinear and integer programming problems, in *Symposia Mathematica*, vol. 19 (Academic, New York, 1976), pp. 161–176
37. F. Giannessi, F. Tardella, Connections between nonlinear programming and discrete optimization, in *Handbook of Combinatorial Optimization*, vol. 1, ed. by D.Z. Du, P.M. Pardalos (Kluwer Academic, Boston, 1998), pp. 149–188
38. F. Giannessi, E. Tomasin, Non convex quadratic programs, linear complementarity problems and integer linear programs, in *Proceedings of the NATO School on “Mathematical Programming in Theory and in Practice”*, North Holland, Amsterdam, 1994
39. L. Gibbons, D. Hearn, M. Ramana, P.M. Pardalos, A continuous characterization of the maximum clique problem. *Math. Oper. Res.* **22**, 754–768 (1997)
40. M. Gröschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1**, 169–187 (1981)
41. S.L. Hakimi, Optimum location of switching centers and the absolute centers and medians of a graph. *Oper. Res.* **12**, 450–459 (1964)
42. P.L. Hammer, S. Rudeanu, *Méthodes Booléennes en Recherche Opérationnelle* (Dunod, Paris, 1970)
43. G.Y. Handler, P.B. Mirchandani, *Location on Networks: Theory and Algorithms* (MIT, Cambridge, 1979)
44. J.-B. Hiriart-Urruty, C. Lemarechal, Testing necessary and sufficient conditions for global optimality in the problem of maximizing a convex quadratic function over a convex polyhedron. Technical report, University Paul Sabatier of Toulouse, Seminar of Numerical Analysis, 1990
45. W.M. Hirsch, A.J. Hoffman, Extreme varieties, concave functions and the fixed charge problem. *Comm. Pure Appl. Math.* **14**, 355–369 (1961)
46. R. Horst, H. Tuy, *Global Optimization. Deterministic Approaches* (Springer, Berlin, 1990)
47. F.K. Hwang, U.G. Rothblum, Directional quasi-convexity, asymmetric Schur-convexity and optimality of consecutive partitions. *Math. Oper. Res.* **21**, 540–554 (1996)
48. S. Iwata, L. Fleisher, S. Fujishige, A strongly polynomial algorithm for minimizing submodular functions. *J. Assoc. Comput. Mach.* **48**, 761–777 (2001)
49. B. Kalantari, J.B. Rosen, Penalty for zero-one integer equivalent problem. *Math. Program.* **24**, 229–232 (1982)
50. G. Kéri, On the minimum value of a quadratic function under linear constraints. *Stud. Sci. Math. Hung.* **6**, 193–196 (1971)
51. V. Klee, On the complexity of d-dimensional Voronoi diagrams. *Arkiv Math.* **34**, 75–80 (1980)
52. K. Kleibohm, Bemerkungen zum Problem der nichtkonvexen Programmierung. *Unternehmensforschung* **11**, 49–60 (1967)
53. M.K. Kozolov, S.P. Tarasov, L.G. Hačjan, Polynomial solvability of convex quadratic programs. *Sov. Math. Doklady* **20**, 1108–1111 (1979)
54. M. Labb  , D. Peeters, J.F. Thisse, Location on networks, in *Handbooks in Operations Research and Management Science*, vol. 8, ed. by M.O. Ball, et al. (Elsevier, Amsterdam, 1995), pp. 551–624
55. C. Larsen, J. Tind, Lagrangean duality for facial programs with applications to integer and complementarity problems. *Oper. Res. Lett.* (North-Holland) **11**, 293–302 (1992)
56. S. Lucidi, F. Rinaldi, Exact penalty functions for nonlinear integer programming problems. *J. Optim. Theory Appl.* **145**, 479–488 (2010)
57. S. Lucidi, F. Rinaldi, An exact penalty global optimization approach for mixed-integer programming problems. Technical report no. 17, DIS, University of Rome, 2010

58. Z.Q. Luo, S. Zhang, On extensions of the Frank-Wolfe theorems. *Comput. Optim. Appl.* **13**, 87–110 (1999)
59. P.B. Mirchandani, R.L. Francis (eds.), *Discrete Location Theory* (Wiley, New York, 1990)
60. T.S. Motzkin, E.G. Straus, Maxima for graphs and a new proof of a theorem of Turán. *Can. J. Math.* **17**, 533–540 (1965)
61. P.M. Pardalos, Continuous approaches to discrete optimization problems, in *Nonlinear Optimization and Applications*, ed. by G. Di Pillo, F. Giannessi (Plenum, New York, 1996), pp. 313–328
62. P.M. Pardalos, J.B. Rosen, *Constrained Global Optimization: Algorithms and Applications* (Springer, Berlin, 1987)
63. A.F. Perold, A generalization of the Frank-Wolfe theorem. *Math. Program.* **18**, 215–227 (1980)
64. M. Ragavachari, On the connection between zero-one integer programming and concave programming under linear constraints. *Oper. Res.* **17**, 680–683 (1969)
65. F. Rinaldi, New results on the equivalence between zero-one programming and continuous concave programming. *Opt. Lett.* **3**, 377–386 (2009)
66. R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, 1970)
67. I.G. Rosenberg, 0 – 1 optimization and non-linear programming. *Rev. Française Autom. Inform. Opér.* **6**, 95–97 (1972)
68. A. Schrijver, A combinatorial algorithm for minimizing submodular functions in strongly polynomial time. *J. Combin. Theory Ser. B* **80**, 346–355 (2000)
69. A. Schrijver, *Combinatorial Optimization. Polyhedra and Efficiency* (Springer, Berlin, 2003)
70. A. Scozzari, F. Tardella, A clique algorithm for standard quadratic programming. *Discrete Appl. Math.* **156**, 2439–2448 (2008)
71. J. Shi, Y. Yoshitsugu, A D.C. approach to the largest empty sphere problem in higher dimension, in *State of the Art in Global Optimization*, ed. by C.A. Floudas, P.M. Pardalos (Kluwer Academic, Boston, 1996), pp. 395–411
72. F. Tardella, On a class of functions attaining their maximum at the vertices of a polyhedron. *Discrete Appl. Math.* **22**, 191–195 (1988–1989)
73. F. Tardella, On the equivalence between some discrete and continuous optimization problems. *Ann. Oper. Res.* **25**, 291–300 (1990)
74. F. Tardella, Piecewise concavity and minimax problems, in *Approximation and Complexity in Numerical Optimization. Continuous and Discrete Problems*, (Series: Nonconvex Optim. Appl. n. 42), ed. by P.M. Pardalos (Kluwer Academic, Boston, 2000), pp. 511–524
75. F. Tardella, Connections between continuous and combinatorial optimization problems through an extension of the fundamental theorem of linear programming. *Electron. Note Discrete Math.* **17C**, 57–262 (2004)
76. F. Tardella, The fundamental theorem of linear programming: extensions and applications. *Optimization* **60**, 15–27 (2011)
77. H. Tuy, Concave programming under linear constraints. *Sov. Math.* **5**, 1437–1440 (1964)
78. D.M. Topkis, Minimizing a submodular function on a lattice. *Oper. Res.* **26**, 305–321 (1978)
79. D.M. Topkis, Adjacency on polymatroids. *Math. Program.* **30**, 229–237 (1984)
80. W.I. Zangwill, The piecewise concave function. *Manag. Sci.* **13**, 900–912 (1967)

Coverage Problems in Sensor Networks

Mihaela Cardei

Contents

1	Introduction	900
2	Sensor Coverage Problems and Design Choices	901
2.1	Coverage Types	901
2.2	Design Choices	902
3	Area Coverage	904
3.1	Energy-Efficient Coverage	904
3.2	Energy-Efficient Connected Coverage	906
3.3	Area Coverage Using Multiple Mobile Sinks	909
3.4	Area Coverage for Composite Event Detection	910
3.5	Adaptive Sensor Scheduling for Area Coverage	912
4	Point Coverage	914
4.1	Energy-Efficient Coverage	914
4.2	Point Coverage in Sensor Networks with Adjustable Sensing Ranges	915
4.3	Node Coverage as Approximation	917
4.4	Connected Point Coverage	919
4.5	Point Coverage in Heterogeneous Wireless Sensor Networks	922
4.6	Minimum Coverage Breach Under Bandwidth Constraints	924
5	Conclusion	924
	Cross-References	925
	Recommended Reading	925

Abstract

Wireless sensor networks have many applications in national security, environmental monitoring, surveillance, military, and health care. Coverage problems have become an important research topic in wireless sensor networks in recent years. They are addressing the following fundamental question: how well do

M. Cardei

Computer and Electrical Engineering and Computer Science Department, Florida Atlantic University, Boca Raton, FL, USA
e-mail: mihaela@cse.fau.edu

the sensors observe the physical space? Many coverage problems have been addressed by the research literature due to a variety of sensors and applications. They vary on the subject to be covered (area, discrete points), sensor deployment mechanism (random, deterministic), sensor mobility (stationary, mobile), event structure (simple, composite), and network properties (connectivity, energyefficiency). This chapter surveys various coverage formulations, their assumptions, and an overview of the solutions proposed.

1 Introduction

Wireless sensor networks provide rapid, untethered access to information and computing, eliminating the barriers of distance, time, and location for many applications in national security, surveillance, health care, environmental monitoring, and many more. A wireless sensor network (WSN) is a deployment of a large number of small, inexpensive, self-powered devices that can sense, compute, and communicate with other devices for the purpose of gathering local information used to make global decisions about a physical environment. Research on sensor networks was originally motivated by military applications, such as acoustic surveillance and target detection. However, WSNs have many other applications such as [14]:

- Infrastructure security: instrument critical buildings and facilities such as power plants with networks of acoustic, video, and other sensors in order to provide early detection of any potential threat.
- Environmental and habitat monitoring: environmental sensors can be used to study vegetation response to climatic changes and diseases, while acoustic and imaging sensors can be used to identify, track, and measure the population of birds and other species.
- Traffic monitoring: sensors are deployed in intersections for traffic monitoring and control and to control traffic lights. Another concept is deploying sensors along the highways and attaching them to each vehicle. Such a network can be used to avoid traffic jams and plan alternative routes.

WSNs are application specific; thus, sensors nodes are equipped with sensors accordingly. Some applications (e.g., building monitoring) require a smaller number of sensors that can be placed individually. Others (e.g., surveillance of a battlefield) require a large number of sensors (e.g., thousands or even millions) that will be deployed ad hoc. Using a larger number of sensors increases network robustness and fault tolerance. The IrisNet (Internet-scale Resource-Intensive Sensor Network Services) project [19] at Intel Research is envisioning a worldwide sensor web of millions of widely distributed, heterogeneous sensors.

Sensor nodes are tiny devices equipped with one or more sensors, one or more transceivers, processing and storage resources, and, possibly, actuators. Sensor nodes have limited resources: they have finite battery resources, low CPU speed, little memory, and small transmission range. For example, the Crossbow MICAz mote [16] operates on the 2.4 GHz ISM band, uses two AA batteries, and has an ATmega 128 L processor at 8 MHz, 4 Kbytes RAM, and a transmission range up to 30 m (indoor)/100 m (outdoor).

An important topic researched in literature is the sensor coverage problem. This problem addresses an important question: *how well do the sensors observe the physical space?* As pointed out in [28], the coverage concept is a measure of the quality of service (QoS) of the sensing function and is subject to a wide range of interpretations due to a large variety of sensors and applications. The goal is to have each location in the physical space of interest within the sensing range of at least one sensor.

This chapter presents various coverage problem formulations. [Section 2](#) presents the main types of coverage and their design choices. [Sections 3](#) and [4](#) discuss relevant contributions addressing area coverage and point coverage. These works are characterized based on the design choices introduced in [Sect. 2](#). The chapter is concluded in [Sect. 5](#).

2 Sensor Coverage Problems and Design Choices

2.1 Coverage Types

Two main types of coverage are (i) area coverage and (ii) point coverage. They depend on the objective to be covered.

In the *area coverage*, the main objective of the sensor network is to cover (monitor) a given area of interest. An area is covered by a WSN if every point in the area is located within communication range of an active sensor which is part of the network. [Figure 1a](#) shows an example of a square area which is covered by a WSN.

In the *point coverage*, the main objective of the sensor network is to cover a set of points. These points may represent targets that have to be monitored continuously by the sensors in the network. Yet in some other scenarios, these points can approximate an area or a region. [Figure 1b](#) shows an example of point coverage.

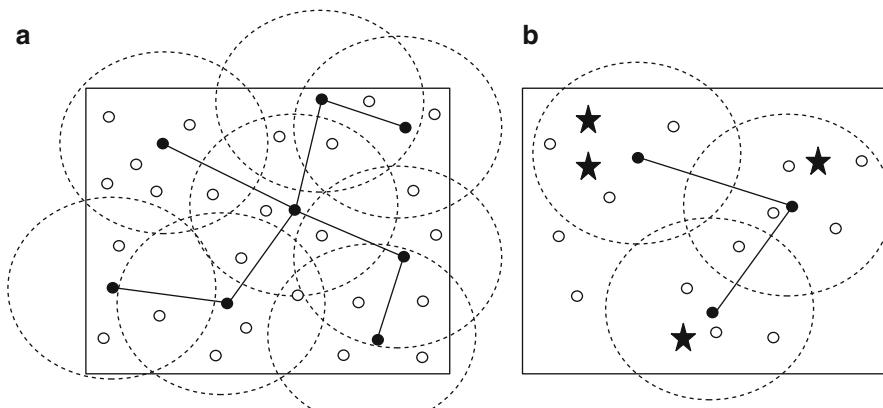


Fig. 1 Coverage types. (a) Area coverage. (b) Point coverage

2.2 Design Choices

Various coverage problems have been formulated based on different design choices, such as:

- *Sensor deployment method*: deterministic versus random. A deterministic sensor placement may be feasible in friendly and accessible environments and when the network size is relatively small. Random sensor distribution is generally considered in remote or inhospitable areas, for military applications, and for applications involving a large population of sensors.
- *Sensor type*: homogeneous versus heterogeneous. Some applications assume that all sensor nodes in the network have the same characteristics (e.g., sensing range, communication range, types of sensors on the sensor board), while other applications assume that the sensors are heterogeneous.
- *Sensor mobility*: static versus mobile. Majority of studies assume that the sensors are stationary, while some works consider mobile sensors. In mobile sensor networks, sensors can self-propel via springs [11], wheels [18], or they can be attached to transporters, such as robots [18] and vehicles [23].
- *Network type*: simple or heterogeneous. Some problems consider a simple (or standard) architecture consisting of sensor nodes that report data to a sink, see [Fig. 2a](#). Other works consider a heterogeneous architecture. One architecture which has been recently explored contains two types of wireless devices, as presented in [Fig. 2b](#). Lower layer is formed by sensor nodes with size and weight restrictions, low cost (projected to be less than \$1), limited battery power, short transmission range, low data rate (up to several hundred Kbps), and low duty cycle. Main tasks performed by a sensor node are sensing, data processing, and data transmission/relaying. The dominant power consumer is the radio transceiver. Upper layer consists of resource-rich supernodes overlaid over the sensor network, as illustrated in [Fig. 2b](#). Supernodes can have two radio transceivers, one for communication with sensor nodes and the other for communication with other supernodes. Supernodes have more power reserves and better processing and storage capabilities than sensor nodes. Wireless communication links between supernodes have considerably longer range and higher data rates, allowing supernode network to bridge remote regions of the interest area. One or more supernodes are sinks, that is, they relay data between the heterogeneous WSN and users. Supernodes are more expensive, and, therefore, fewer are used than sensor nodes. One of the main tasks performed by a supernode is to transmit/relay data from sensor nodes to/from the sinks. Other tasks can include sensor data aggregation, complex computations, and decision making. One example of such an architecture consists of Crossbow MPR400 motes [16] communicating on 916 MHz and Crossbow Stargate Xscale platform [16] used as supernodes. A Stargate can be connected to a mote MPR400 and can use a IEEE 802.11 wireless card on 2.4 GHz. Using two transceivers, a Stargate is part of the WSN and can also communicate with the other Stargate devices on 2.4 GHz.

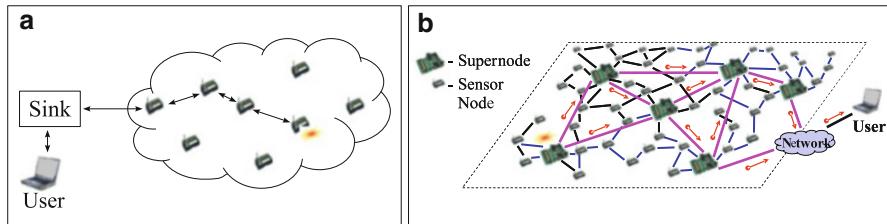


Fig. 2 Network types. (a) Simple WSN. (b) Heterogeneous WSN

- *Sink mobility*: static or mobile. Some works assume that the sinks are stationary, while others assume they are mobile. Recent advances in the field of robotics [3, 22] make it possible to integrate robots as sinks (or gateways) in WSNs [32]. Adding mobile devices to WSNs infrastructure has attracted increased attention recently. Much of the work has been conducted on data gathering applications, where the mobile sinks move randomly [30, 34], using predetermined paths [10, 25, 32], or autonomously [2]. A random moving sink is not aware of the sensor residual energy and thus might threaten the energy balance among the sensors. The predetermined path models lack flexibility and scalability with network size. The moving strategies where the sinks take the moving decisions autonomously can better adapt to various network conditions.
- *Event type*: simple or composite. Some works assume only one sensing measure (e.g., acoustic sensor), while others address composite events. For example, a fire event is better detected if temperature, humidity, and smoke sensor measurements are combined. This will form a composite event. In such a situation, there are coverage requirements for each of the atomic events forming the composite event.
- *Fault tolerance*: simple or k-coverage. Some works require that each point of interest be simply covered, that is, to be monitored by at least one sensor. Other works require that each point of interest be k-covered, that is, it will be monitored by at least k sensors at each time. k-coverage improves fault tolerance, as some of the sensors may die over time due to physical damage or by depleting their energy resources.
- *Coverage breach*: some applications allow coverage breach. If a target (or monitor region) is not covered by an active sensor, then it is called breached. The objective is to minimize the coverage breach (e.g., minimize the maximum accumulated breach time of any target and minimize the number of breached targets).
- *Additional requirements*: energy efficiency, connectivity, maximum network lifetime, etc. There are usually additional requirements needed for a healthy functioning of the WSN. Sensor nodes are resource constraints, and they can function a limited amount of time before depleting their energy resource. When sensors are remotely deployed, it is infeasible and sometimes impossible to replace their battery. Thus, developing mechanisms which are energy-efficient is one of the main requirements in WSN design. A sensor node's radio can be in one of the following four states: transmit, receive, idle, or sleep. The idle

state is when the transceiver is neither transmitting nor receiving, and the sleep mode is when the radio is turned off. As presented in [29], an analysis of the power usage for Rockwell WINS seismic sensor indicates power consumption for the transmit state between 0.38 and 0.7 W, for the receive state 0.36 W, for the idle state 0.34 W, and for the sleep state 0.03 W. The receive and idle modes may require as much energy as transmitting, while the sleep mode requires the less energy. Another observation is the communication/computation power usage ratio, which can be higher than 1,000 (e.g., for Rockwell WINS [29] is from 1,500 to 2,700); therefore, local data processing, data fusion, and data compression are highly desirable. Judiciously selecting the state of each sensor node's radio is accomplished through a *scheduling* mechanism. Power saving techniques can generally be classified in the following categories:

1. Schedule the wireless nodes to alternate between active and sleep mode
2. Power control by adjusting the transmission range of wireless nodes
3. Energy-efficient routing, data gathering
4. Reduce the amount of data transmitted and avoid useless activity

Energy-efficiency mechanisms have also direct impact on WSN lifetime. Yet another important requirement is WSN connectivity. Active sensors must form a connected topology to allow sensor data to be relayed to the user. Sometimes, k -connectivity is enforced, where each sensor node has k distinct paths to a sink. This improves reliability: in case some paths or sensors become unavailable, data can be relayed on alternate paths. There are also cases when sensors do not have to be connected. In this case, all sensors are within communication range with the sink (or base station), or a mobile sink can be used to collect sensor data.

3 Area Coverage

In the area coverage problem, the main objective of the sensor network is to cover (monitor) an area (also referred sometimes as region). [Figure 1a](#) shows an example of a random deployment of sensors to cover a given rectangular-shaped area, where circles represent the sensing range. Each point of the area is monitored by at least one sensor. The connected black nodes form the set of active sensors, the result of a scheduling mechanism. Several energy-efficient area coverage problem formulations, their models and assumptions, and solutions proposed are surveyed next.

3.1 Energy-Efficient Coverage

The works in [6, 31] consider a large population of sensors, deployed randomly for area monitoring. The goal is to achieve an energy-efficient design that maintains area coverage. As the number of sensors deployed is greater than the optimum required to perform the monitoring task, the solution proposed is to divide the sensor nodes into disjoint sets, such that every set can individually perform the

area monitoring tasks. These sets are activated successively, and while the current sensor set is active, all other nodes are in a low-energy sleep mode. The goal of this approach is to determine a maximum number of disjoint sets, as this has a direct impact on conserving sensor energy resources as well as on prolonging the network lifetime. The solutions proposed are centralized.

Slijepcevic and Potkonjak [31] model the area as a collection of fields, where every field has the property that any enclosed point is covered by the same set of sensors. The most-constrained least-constraining algorithm [31] computes the disjoint covers successively, selecting sensors that cover the critical element (field covered by a minimal number of sensors), giving priority to sensors that cover a high number of uncovered fields, cover sparsely covered fields, and do not cover fields redundantly.

Cardei et al. [6] model the disjoint sets as disjoint dominating sets in an undirected graph, where sensors form the vertex set and an edge joins any two vertices that are within each other's sensing range. The maximum disjoint dominating sets computation is NP-complete, and any polynomial-time approximation algorithm has a lower bound of 1.5. A graph-coloring mechanism is proposed for computing the disjoint dominating sets. First, disjoint sets are formed by coloring all nodes, using the sequential coloring algorithm. Then, each non-dominating set is considered in an increasing color number and transformed into a dominating set by recoloring a smallest number of higher-color vertices. When this process ends and no more dominating sets can be formed, the remaining nodes are added to the sets where they have the greatest contribution in covering parts of the uncovered given area. Simulations have shown that the number of sets computed is between 1.5 and 2 times greater than by using the algorithm in [31], with lapses in area coverage less than 5 %, on average.

Another energy-efficient node-scheduling-based coverage mechanism is proposed by Tian and Georganas in [33]. The protocol proposed is distributed and localized. Power savings are obtained both by scheduling the sensor nodes activity as well as by considering an adjustable sensing range. The off-duty *eligibility rule* determines whether a node's sensing area is included in its neighbors' sensing area. Solutions for determining whether a node's coverage can be sponsored by its neighbors (sponsored coverage calculation) are provided for several cases: when nodes have the same sensing range and know their location, when nodes have the same sensing range and can obtain neighboring node's directional information, or in particular scenarios when nodes have different sensing ranges. The node-scheduling scheme is divided into rounds, where each round has a self-scheduling phase followed by a sensing phase. In the self-scheduling phase, the nodes investigate the off-duty eligibility rule. Eligible nodes turn off their communication and sensing units, while all other nodes will perform sensing tasks in the sensing phase. In order to obtain neighboring information, each node broadcasts a position advertisement message at the beginning of each round. This message contains the node ID and node location. If the off-duty eligibility rule is tested simultaneously by neighboring nodes, a node and its sponsor may decide to turn off simultaneously, triggering the occurrence of *blind points*. To avoid this, a back-off scheme is used, where every

node starts the evaluation rule after a random time and then broadcasts a status advertisement message to announce if it is available for turning off. Before turning off, a node waits another T_w time to listen for neighboring nodes update. This work does not specify synchronization mechanisms in detail. It is implemented as an extension of the data gathering LEACH protocol [20], and simulation results show an increase of 1.7 on average in system lifetime.

A probing-based, node-scheduling solution for the energy-efficient coverage problem is proposed in [43] by Ye et al. Here, all sensors are characterized by the same sensing range, and coverage is measured as the ratio between the area under monitoring and total size of the network field. The off-duty eligibility rule is based on a probing mechanism. Basically, a sensor broadcasts a probing message PRB within a probing range r . Any working node that hears this message responds with a PRB_RPY . If at least one reply is received, the node enters the sleep mode. Probing range is selected based on the desired working node density (number of sensors per unit area) or based on the desired coverage redundancy, whereas the wake-up time is based on the tolerable sensing intermittence. This protocol is distributed, localized, and has low complexity but still does not preserve the original coverage area.

3.2 Energy-Efficient Connected Coverage

An important issue in WSNs is connectivity. A network is connected if any active node can communicate with any other active node, possibly using intermediate nodes as relays. Once the sensors are deployed, they organize into a network that must be connected so that the information collected by sensor nodes can be relayed back to data sinks or controllers. An important, frequently addressed objective is to determine a minimal number of working sensors required to maintain the initial coverage area as well as connectivity. Selecting a minimal set of working nodes reduces power consumption and prolongs network lifetime. Several connected coverage mechanisms are presented next.

An important, but intuitive result was proved by Zhang and Hou [44], which states that if the communication range R_c is at least twice the sensing range R_s , a complete coverage of a convex area implies connectivity of the working nodes. If the communication range set up is too large, radio communication may be subject to excessive interference. Therefore, if the communication range can be adjusted, a good approach to assure connectivity is to set transmission range as twice the sensing range. Two nodes are neighbors if they have overlapping sensing areas. By intuition, when $R_c \geq 2R_s$, two neighbors are within their communication ranges, as shown in Fig. 3.

Based on this result, Zhang and Hou [44] further discussed the case $R_c \geq R_s$. An important observation is that an area is completely covered if there are at least two disks that intersect and all crossings are covered. Here, a disk refers to a node's sensing area, and a crossing is an intersection point of the circle boundaries of two disks. In the ideal case, in which node density is sufficiently high, the full

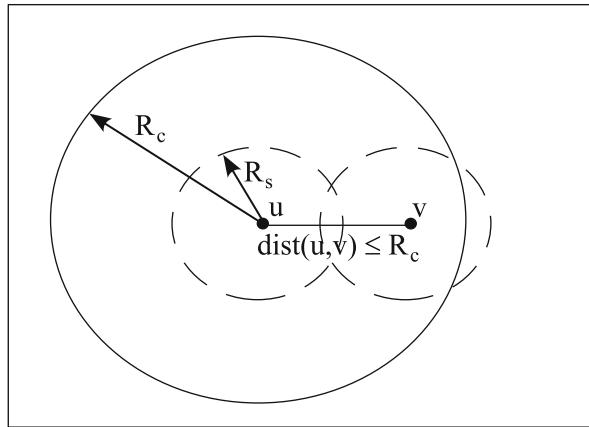


Fig. 3 Two sensors, u and v , with overlapping sensing areas can directly communicate with each other if $R_c \geq 2R_s$

coverage can be obtained by optimally placing the subset of working nodes at the vertices of regular hexagonal plane tiling. Based on these results, authors proposed a distributed, localized algorithm, called Optimal Geographical Density Control (OGDC). At any time, a node can be in one of the states: UNDECIDED, ON, and OFF. The algorithm runs in rounds, and at the beginning of each round, a set of one or more starting nodes are selected as working nodes. After a back-off time, a starting node broadcasts a power-on message and changes its state to ON. The power-on message contains (1) the position of the sender and (2) the direction along which a working node should be located. The direction indicated by the power-on message of a starting node is randomly distributed. Having starting nodes randomly selected at the beginning of each round ensures uniform power consumption across the network. Also, the back-off mechanism avoids packet collisions. At the beginning of each round, all nodes are UNDECIDED and will change either to ON or OFF state until the beginning of the next round. This decision is based on the power-on messages received. Every node keeps a list with neighbor information. When a node receives a power-on message, it checks whether its neighbors cover its sensing area, and if so, it will change to OFF state. A node decides to change into the ON state if it is the closest node to the optimal location of an ideal working node selected to cover the crossing points of the coverage areas of two working neighbors. NS-2-based simulation shows good results in terms of percentage of coverage, number of working nodes, and system lifetime.

Some applications may require different degrees of coverage while still maintaining working node connectivity. A network has a coverage degree k (k -coverage) if every location is within the sensing range of at least k sensors. Networks with a higher coverage degree can obtain higher sensing accuracy and be more robust to sensor failure. Wang et al. [36] generalized the result in [44] by showing that, when the communication range R_c is at least twice the sensing range R_s ,

a k-covered network will result in a k-connected network. A k-connected network has the property that removing any $k - 1$ nodes will still maintain the network connectivity. The following discussion addresses the case when $R_c \geq 2R_s$. To define the k-coverage eligibility mechanism, the problem of determining the coverage degree of a region is reduced to a simpler problem of determining the coverage degrees of all the intersection points. Given a coverage region A , a point p is called an *intersection point* if (1) p is an intersection point of the sensing cycles of any two nodes u and v , for example, $p \in u \cup v$, and (2) for any node v , $p \in v \cap A$ and $|pv| = R_s$, where R_s is the sensing range. The authors proved that a convex region is k-covered if it contains intersection points, and all these intersection points are k-covered. Based on this, a sensor is ineligible to turn active if all the intersection points inside its sensing circle are at least k-covered.

Wang et al. proposed the coverage configuration protocol (CCP) [36] that can dynamically configure the network to provide different coverage degrees requested by applications. To facilitate the computation of intersection points, every node maintains a table with neighbor information (location, status: active/inactive) and periodically broadcasts a HELLO beacon with its current location and status. A node can be in one of the three states: SLEEP, LISTEN, and ACTIVE. All nodes start in the SLEEP state for a random time. When a node wakes up, it enters LISTEN state, and based on the outcome of the eligibility rule over a time interval, it will enter either SLEEP or ACTIVE state. Once a node is in the ACTIVE state, it will reevaluate the coverage eligibility every time it receives a HELLO message and decide whether to go into the SLEEP state or remain in the ACTIVE state.

For the case when $R_c < 2R_s$, CCP does not guarantee network connectivity. The solution adopted in [36] is to integrate CCP with SPAN [12] to provide both sensing coverage and network connectivity. The combined eligibility rule is as follows: (1) an inactive node goes into the active state if it satisfies the eligibility rules of SPAN and CCP and (2) an active node withdraws if it satisfies neither the eligibility rule of SPAN nor CCP. With this combined mechanism, k-coverage can be obtained using CCP and 1-connectivity using SPAN. The algorithm was implemented and tested using NS-2 and showed good performance results in terms of coverage, active nodes, and system lifetime.

Carle and Simplot [9] propose another mechanism for energy-efficient connected area coverage for the case when all sensor nodes have the same sensing range and the communication range equals the sensing range. The goal of the algorithm is to select an area-dominating set of nodes of minimum cardinality, such that the selected set covers the given area. The main idea is to use one of the existing protocols for building a connected dominating set (e.g., Dai and Wu's algorithm in [17]) but instead of providing the node coverage to assure the area coverage. Once a node has decided its status (active or sleeping) for the next round, it transmits a message to its neighbors. The message sent is referred to as positive or negative advertising, depending on whether the node has decided to be active or in the sleep mode, respectively. Each node sets its priority depending on the remaining battery life and employs a back-off mechanism, such that the nodes with larger energy resources decide their status first. The protocol run by a node u is as follows: (1) node u

determines whether its monitoring area is already covered by its neighbors, based on the positive or negative advertising messages. If the scheme by Tian and Georganas [33] is used for determining whether the neighbor set covers u 's monitoring area, then each node needs to know its neighbors' locations; (2) at the end of the back-off interval, node u computes a subgraph of its one-hop active neighbors; (3) if the subgraph is connected and fully covers u 's area, then node u will be in the sleep mode; otherwise, it will be in the active mode during the next round. Since this algorithm follows the steps in [17], the resulting area-dominating set is connected.

3.3 Area Coverage Using Multiple Mobile Sinks

The work [26] considers a WSN consisting of sensor nodes deployed randomly in large number and several mobile sinks used to collect data from the sensors. Sinks have two transceivers, one to communicate with the sensors and another to communicate with the other sinks. Sensors send their data to the sinks using multi-hop communication. In WSNs, sensors closer to a sink tend to consume more energy than those farther away from the sinks. This is mainly because, besides transmitting their own packets, they forward packets on behalf of other sensors that are located farther away. As a result, the sensors closer to the sink will drain their energy resources first, resulting in holes in the WSN. This uneven energy consumption will reduce network lifetime.

One method to alleviate this problem is using mobile sinks which move to a new location when the energy of the sensors near the sink becomes low. Article [26] presents an algorithm where sinks move autonomously such that (1) the sinks remain interconnected all the time forming a virtual sink backbone and (2) network lifetime is maximized. This sink movement mechanism is then combined with the area coverage application. Sensors are scheduled to alternate between active and sleep states such that the active nodes satisfy the area coverage requirement. The problem requirements can be summarized as follows: (1) the set of active sensors provides area coverage, (2) the sinks remain interconnected all the time, and (3) network lifetime is maximized.

The algorithm [26] runs in rounds. The steps at the beginning of the first round are:

1. Sensors run a scheduling mechanism to decide which sensors stay active and which sensors go to sleep such that the active sensors ensure the area coverage.
2. Sensors form cluster-based data collection trees, where mobile sinks are the clusterheads.
3. Start data collection phase.

At the beginning of each other round (excluding the first round), each sink S_i executes the following steps:

1. Sink S_i decides if it has to move. If not, then nothing has to be done until the beginning of the next round; go to step 5. Otherwise, the steps below are executed.

2. Sink S_i announces the sensors in its cluster area to change the set of active sensors (e.g., alternate the set of active sensors with some sleep sensors with more residual energy), such that the set of active sensors ensure area coverage and connectivity.
3. After deciding the set of active sensors in S_i 's cluster area, form a data collection tree for the sink S_i .
4. Sink S_i decides if it has to move based on the energy in the new tree. If it does not need to move, then nothing has to be done until the beginning of the next round; go to the step 5. Otherwise, S_i invokes the mechanism for deciding the new sink location (see [26]).
5. Start data collection phase.

Sensors compute a scheduling mechanism to decide which of them stay active to satisfy the area coverage and sensor connectivity requirements. Two such mechanisms are discussed. The first approximates the area coverage by point coverage, thus approximating the requirement of area coverage with the requirement that every sensor node needs to be covered by the set of active sensor nodes (see Sect. 4.3). The covering nodes are formed by finding a connected dominating set using Rule k [17]. The second method divides the area into grids, such that any sensor active in a grid can cover that grid area and can communicate with any of the neighboring grids. Then one sensor (e.g., the node with highest residual energy) will be active in each grid.

Starting with the second and subsequent rounds, sinks might move in order to prolong network lifetime. A sink S_i decides to move if p% of its 1-hop sensors have $E \leq E_{th}$. If the sink has to move, then before it actually moves, the set of active nodes in its cluster is recomputed based on the residual energy in the sensor nodes. After the new active nodes are decided, they form a data collection tree. The sink S_i rechecks the moving criteria, and if p% of its 1-hop sensors have $E \leq E_{th}$, then it moves; this means that the neighboring sensors have low-energy resources.

3.4 Area Coverage for Composite Event Detection

A WSN can detect single (or atomic) events or composite events [27]. Considering the sensors manufactured by Crossbow Technology, Inc. [16] as an example, a sensor equipped with MTS400 multi-sensor board can sense temperature, humidity, barometric pressure, and ambient light. Thus, it can detect multiple atomic events.

For a single sensing component, for example, the temperature, if the temperature rises above some predefined threshold, then an *atomic event* is detected. A *composite event* is the combination of several atomic events. For example, a fire-detection application uses sensors to measure various parameters of the environment. A composite event fire might be defined as the combination of the atomic events $\text{temperature} > th_1$, $\text{light} > th_2$, and $\text{smoke} > th_3$, where “th” denotes a threshold for the corresponding attribute. That is, $\text{fire} = (\text{temperature} > th_1) \wedge (\text{light} > th_2) \wedge (\text{smoke} > th_3)$. It is more accurate to report the fire when all these atomic events occur, instead of the case when only one attribute is above the threshold.

The definitions for k-watching an atomic event and a composite event are as follows:

- *k-Watched Atomic Event*: An atomic event is k-watched by a set of sensors if at any time this event occurs at any point within the interested area, at least k sensors in the network can detect this occurrence.
- *k-Watched Composite Event*: A composite event is k-watched by a set of sensors if every atomic event part of the composite event is k-watched by the set of sensors.

The problem studied in [27] is defined as follows: given a set of N sensors with different sensing capabilities, a monitored area A, a composite event which is a combination of M atomic events involving sensing components x_1, x_2, \dots, x_M , and the energy constraint of each sensor E_{init} design a sensor scheduling mechanism such that (1) the composite event is k-watched in the area A, (2) the set of active sensor nodes is connected, and (3) network lifetime is maximized.

The mechanism proposes using sensor scheduling to achieve energy efficiency. Network activity is organized in rounds. Each round has two phases: initialization phase and event detection phase. In the initialization phase, sensor nodes decide if they will be active or if they go to sleep during the next round. The active sensor set must provide a connected topology and must ensure k-watching property across the deployment area. Each node u has a priority $p(u) = (E(u), ID(u))$, where $E(u)$ is the node u's residual energy and $ID(u)$ is the node u's ID. A node with higher residual energy has a higher priority. Note that a node priority change over time: if the node is active, then its residual energy decreases, so it has a lower priority in the next round.

The h-hop neighborhood of a node u is defined as $N_h(u) = \{v | \text{distance}(u, v) \leq h \text{ hops}\}$. The set of nodes within the h-hop neighborhood of u that have higher priority is defined as $N'_h(u) = \{v \in N_h(u) | p(v) > p(u)\}$. A node u decides its status (active/sleep) for the next round by using the following rule:

The default status of a sensor node is active. A sensor u is in sleep mode if the following two conditions hold:

1. (k-Watching property) the set of nodes $N'_h(u)$ provides the k-watching of each of u's sensing components.
2. (Connectivity property) any two of u's neighbors in $N_1(u)$, w and v, are connected by a path with all intermediate nodes in $N'_h(u)$.

The intuition behind this rule is that a sensor u can go to sleep if the nodes with higher priority in its h-hop neighborhood can provide the k-watching property and the connectivity property on behalf of u. Note that a node with higher priority can also go to sleep if the rule holds true in its h-hop neighborhood. To avoid inconsistencies, a mechanism that assigns global priorities to the nodes is used. If a node u has only one neighbor, then the connectivity requirement is fulfilled, and u goes to sleep if the k-watching property holds.

Work [35] proposes a centralized approach that builds the detection sets (or data collection trees) using the breadth-first search algorithm [15] starting from a gateway, which can be any sensor node with richer energy resources. One drawback of the proposed approach is the global knowledge required by the algorithm constructing the detection sets. The sets are computed by the gateway node, and

the decision if an event (simple or composite) occurs is also made by the gateway node. Simulation results show that the centralized tree-based approach has a longer network lifetime (more detection sets), but the trade-off consists in using a single point of failure and large overhead for a large network size.

3.5 Adaptive Sensor Scheduling for Area Coverage

Article [41] considers a publish-subscribe scenario similar to [21]. The sink sends a query as interest toward sensors in the monitored area. When sensors receive a query, if they can satisfy the interests and meet the query, then they activate to perform the sensing task. The intermediate sensors help route sensed data toward the sink, and the results are finally reported to the sink. For example, assume that the sink is interested in whether a fire will happen in the next 2h; then it sends the interest query:

```
type = temperature ∧ smoke
interval = 1 min
rect = [(50, 70), (90, 90)]
timestamp = 01 : 30 : 00
expiresAt = 03 : 30 : 00
```

The sensors that can sense the temperature and/or smoke and are within the rectangle with the lower left end (50, 70) and the upper right end (90, 90) are activated and report data every 1 min from time 01: 30: 00 to 03: 30: 00. In general, the *type* field in the interest query is the combination of one or more sensing components, that is, $x_1 \wedge x_2 \wedge \dots \wedge x_l$.

More than one query can be sent to the monitored area. For example, another query can be issued as follows:

```
type = temperature
interval = 2 min
rect = [(30, 60), (70, 80)]
timestamp = 03 : 00 : 00
expiresAt = 04 : 00 : 00
```

The areas of interest for the two queries intersect; thus, some sensors might report data for both queries. The active sensors have to satisfy both the coverage and the global connectivity requirements.

The objective in [41] is to design a sensor scheduling algorithm that allows sensors not actively participating in sensing or data relaying to go to sleep in order to conserve energy and to prolong the network lifetime. The sensor scheduling mechanism adaptively decides the set of active sensor nodes such that both the coverage and the connectivity requirements are met:

- *Coverage requirement*: as queries propagate in the monitored area, related sensors in the area of interest are activated for the sensing tasks. This is an adaptive mechanism: as new query requests arrive, new sensors are activated as

needed; as queries expire, sensors in the area of interest go to sleep if they are not sensing any other request.

- *Connectivity requirement:* the set of active sensors must be connected all the time. This is needed for the communication between sensors and the sink in operations such as data reporting, query propagation, and forwarding of control messages.

The problem studied in [41] is as follows: Given a WSN with sensors equipped with one or more sensing components (attributes) from the set $\{x_1, x_2, \dots, x_M\}$, design an adaptive sensor scheduling mechanism such that the set of active sensors change over time such that to satisfy the *coverage* and *connectivity* requirements, and the WSN lifetime is maximized.

The adaptive sensor scheduling in WSNs (ASW) localized mechanism proposed in [41] forms an adaptive connected dominating set (CDS) that change over time as new requests arrive or expire. When a new request regarding sensing a particular rectangular area is received/expires, sensors adaptively update the active nodes in the required area to satisfy both coverage requirement and global connectivity.

Initially, a CDS backbone is selected so that the requests can be propagated to the monitored area along the backbone. The CDS is updated periodically after each time interval T . Before rerunning the CDS, sensors update their priority based on the remaining energy. Then for a time T , priority and the CDS stays unchanged. All the nodes participate in the new CDS selection and update their fields accordingly depending on whether they will be part of the CDS or not. The nodes serving queries will continue to be active and to perform their sensing tasks. Nodes which are not in the CDS or serving queries go to sleep.

Every sensor u decides whether it will be a CDS node or not during the next period T as follows. Initially, a node u sets $CDS(u) = \text{TRUE}$. Then sensor u sets $CDS(u) = \text{FALSE}$ if the following condition holds: for any two of u 's neighbors in $N_1(u)$, w and v , w and v are connected by a path with all intermediate nodes in $N'_h(u)$. See Sect. 3.4 for a definition of $N_h(u)$ and $N'_h(u)$.

Consider that the following query is sent by the BS: *Query(type, interval, area R, Tstart, Tend)*. The field *type* contains the attributes of interest that have to be reported, for example, $type = x_1 \wedge x_3$. The CDS nodes awake the 1-hop sleeping nodes in area R and forward the query information. An awaken node checks if it has at least one of the sensing components from the query type field, then it remains active for the duration of the query and will report data according to the reporting interval in the query. When a query time request ends, the reporting sensors check their role field. If a sensor is part of the CDS or is serving another query, then it remains active. Otherwise, it returns to sleep.

One case that can occur regards overlapping queries. A sensor may serve multiple queries if their reporting areas intersect. The frequency of reporting is done according to the requesting queries, and this field updates as queries expire or new queries arrive.

Compared with directed diffusion [21], ASW is more energy efficient, it generates less overhead; however, it may have a longer delivery path. This is because in the directed diffusion, all sensors are active and they may form shorter data delivery paths, while in ASW, only part of the sensors are active.

4 Point Coverage

In the point coverage problem, the objective is to cover a set of points. [Figure 1b](#) shows an example of a set of sensors randomly deployed to cover a set of points (star nodes). The connected black nodes form the set of active sensors, the result of a scheduling mechanism. Some important cases discussed in this section are energyefficiency, sensors with adjustable sensing ranges, multiple point coverage, connectivity, heterogeneous WSNs, and coverage breach.

4.1 Energy-Efficient Coverage

Article [5] addresses the point coverage problem in which a limited number of points (targets) with known locations need to be monitored. A large number of sensors are dispersed randomly in close proximity of the targets and send the monitored information to a central processing node. The requirement is that every target is monitored at all times by at least one sensor, assuming that every sensor is able to monitor all targets within its sensing range.

One method for extending sensor network lifetime through energy efficiency is to divide the set of sensors into disjoint sets such that every set completely covers all targets. These disjoint sets are activated successively, such that at any moment in time, only one set is active. Using this approach, all targets are monitored by every sensor set. The goal of this approach is to determine a maximum number of disjoint sets, such that the time interval between two activations for any given sensor is longer. By decreasing the fraction of time, a sensor is active, the overall time until power runs out for all sensors is increased, and the application lifetime is extended proportionally by a factor equal to the number of disjoint sets.

A solution for this application is proposed in [5], where disjoint sets are modeled as disjoint set covers, such that every cover completely monitors all the target points. Article [5] shows that the disjoint set coverage problem is NP-complete and any polynomial-time approximation algorithm has a lower bound of 2. The disjoint set cover problem is reduced to a maximum flow problem, which is modeled as a mixed integer programming. Simulations show better results in terms of the number of disjoint set covers computed compared with most-constrained least-constraining algorithm [31], where every target is modeled as a field.

In [7], the scheduling mechanism also forms sets of sensor nodes that monitor the targets successively. In contrast with the previous approach, a sensor can be part of multiple sets (e.g., the sets are not necessarily disjoint) and the sets can be operational for different time intervals. The objective of the maximum set covers (MSC) problem is to organize the sensors in sets and to determine the time interval of each set such that the sum of all time intervals is maximized (e.g., maximum network lifetime), subject to the constraints that each sensor cover monitors all targets and the time a sensor is active is less than the sensor lifetime. The paper assumes all sensors can be active for the same amount of time (e.g., they have the same starting energy). Article [7] shows that the MSC problem is NP-hard by reduction from the 3-SAT problem.

Two heuristic solutions are proposed. The first one formulates the MSC problem using integer programming (IP) and then uses the relaxation technique to design a linear programming (LP)-based heuristic. The total complexity is $O(p^3n^3)$, where p is an upperbound on the number of covers and n is the number of sensor nodes. The second heuristic builds the covers successively. When building a new cover, sensors with greatest contribution (e.g., covers more uncovered targets and has more residual energy) are added first. Sensors are added to a cover until all the targets are covered. The complexity is $O(dm^2n)$, where d is the number of sensors that cover the most sparsely covered target, m is the number of targets, and n is the number of sensors. Simulation results show that the IP approach returns a longer network lifetime but has a larger complexity than the greedy mechanism.

4.2 Point Coverage in Sensor Networks with Adjustable Sensing Ranges

Article [8] addresses the case when sensor nodes can adjust their sensing range. In the adjustable range set covers (AR-SC) problem, a set of points (or targets) and a set of sensors with adjustable sensing ranges are given. The objective is to find a family of set covers such that (1) the number of covers is maximized, (2) each set monitors all targets, and (3) a sensor consumes at most E energy. The problem assumes a discrete set of sensing ranges for the sensors.

The first approach uses IP to determine the set covers $c_1, c_2 \dots c_K$ and the sensors in each cover x_{ikp} . Given:

- N sensor nodes s_1, \dots, s_N .
- M targets t_1, t_2, \dots, t_M .
- P sensing ranges r_1, r_2, \dots, r_P and the corresponding energy consumption e_1, e_2, \dots, e_P .
- Initial sensor energy E .
- The coefficients showing the relationship between sensor, radius, and target: $a_{ipj} = 1$ if sensor s_i with radius r_p covers the target t_j .

Maximize $c_1 + \dots + c_K$

subject to

$$\begin{aligned} \sum_{k=1}^K (\sum_{p=1}^P x_{ikp} e_p) &\leq E && \text{for all } i = 1 \dots N \\ \sum_{p=1}^P x_{ikp} &\leq c_k && \text{for all } i = 1..N, k = 1 \dots K \\ \sum_{i=1}^N (\sum_{p=1}^P x_{ikp} * a_{ipj}) &\geq c_k && \text{for all } k = 1 \dots K, j = 1 \dots M \\ x_{ikp} &\in \{0, 1\} \text{ and } c_k \in \{0, 1\} \end{aligned} \quad (1)$$

The variables used are:

- c_k , boolean variable, for $k=1..K$; $c_k = 1$ if this subset is a set cover; otherwise, $c_k = 0$.
- x_{ikp} , boolean variable, for $i = 1..N$, $k = 1..K$, $p = 1..P$; $x_{ikp} = 1$ if sensor i with range r_p is in cover k ; otherwise, $x_{ikp} = 0$.

The first constraint guarantees that the energy consumed by each sensor i is less than or equal to E , which is the starting energy of each sensor. The second constraint ensures that if sensor i is part of the cover k , then exactly one of its P sensing ranges is set. The third constraint guarantees that each target t_j is covered by each set c_k .

Since the complexity of the IP is large for large number of sensors, the IP is relaxed to a LP and then rounded to get a solution to the original problem AR-SC. In the relaxed LP, the variables x_{ikp} and c_k are relaxed to $0 \leq x_{ikp} \leq 1$ and $0 \leq c_k \leq 1$. To form a set cover c_k (taken in decreasing order), sensors are added in decreasing order of x_{ikp} as long as they have enough energy resources and new uncovered targets are monitored.

The second solution is a centralized greedy algorithm, which repeatedly constructs set covers and sets sensors' sensing ranges as long as each target is covered by at least one sensor with enough energy resources. In forming a set cover, sensors are selected based on their contribution. ΔB_{ip} is the incremental contribution of the sensor i when its sensing range is increased to r_p . $\Delta B_{ip} = \Delta T_{ip}/\Delta e_p$, where $\Delta T_{ip} = |T_{ip}| - |T_{iq}|$ and $\Delta e_p = e_p - e_q$. The range r_q is the current sensing range of the sensor i , thus $r_p > r_q$. Initially, all the sensors have assigned a sensing range $r_0 = 0$, and the corresponding energy is $e_0 = 0$. Sensors with higher contribution are added first (e.g., by increasing their sensing range) until all the targets are covered. Intuitively, a smaller sensing range is preferable as long as the target coverage objective is met, since energy resources are conserved, allowing the sensor to be operational longer. Once the set cover is formed (e.g., all targets are covered by the selected set of sensors), the sensors with a sensing range greater than zero form the set of active sensors, while all other sensors with sensing range $r_0 = 0$ will be in sleep mode.

The third method proposed in [8] is a distributed and localized method. In a *distributed and localized* method, the decision process at each node makes use of only information in a neighborhood within a constant number of hops. A distributed and localized algorithm is desirable in WSN since it adapts better to dynamic and large topologies. The algorithm runs in rounds. Each round begins with an initialization phase, where sensors decide whether they will be in an active or sleep mode during the current round.

Each sensor maintains a waiting time, after which it decides its status (sleep or active) and its sensing range, and then it broadcasts the list of targets it covers to its one-hop neighbors. The waiting time of each sensor s_i depends on s_i 's contribution, and it is set up initially to $W_i = \left(1 - \frac{B_{ip}}{B_{\max}}\right) \times W$ where B_{\max} is the largest possible contribution, defined as $B_{\max} = M/e_1$, where M is the number of targets. B_{ip} is the contribution of sensor i with range r_p , defined as $B_{ip} = |T_{ip}|/e_p$, where T_{ip} is the set of uncovered targets within the sensing range r_p of sensor s_i .

When its waiting time expires, a sensor broadcasts its status (active or sleep) as well as the list of targets it covers. The neighbors receiving the message update their contribution and waiting time accordingly. If a sensor has all the targets in its range already covered, then it sets its range to $r_0 = 0$. In the initialization phase, nodes which are active during the current round are determined. If, after the initialization time, one of the sensors cannot cover a target in their range, then it announces a

failure message to the base station (BS). Network lifetime is computed as the time elapsed until the BS detects the first failure.

Simulation results show that the centralized greedy algorithm performs the best, followed by the LP-based mechanism, and then the distributed greedy one. There is also high improvement in network lifetime when sensors can adjust their sensing ranges. Compared to the base case when sensors have only one sensing range, the most noticeable improvements are for up to four different sensing range values. Between 4 and 6, the network lifetime improvement was not significant.

4.3 Node Coverage as Approximation

This section has two parts. [Section 4.3.1](#) discusses the case when each point has to be monitored by at least one sensor, and [Sect. 4.3.2](#) discusses the case when each point has to be covered by at least k sensors.

4.3.1 Simple Point Coverage

When a large and dense sensor network is randomly deployed for area monitoring, the area coverage can be approximated by the coverage of the sensor locations. That is, based on the assumption of a large and dense population of sensors, covering each sensor location approximates the coverage of each point in the given area. One method to assure coverage and connectivity is to design the set of active sensors as a connected dominating set (CDS). A distributed and localized protocol for constructing the CDS is proposed by Wu and Li, using the *marking process* in [38]. A node is a coverage node if there are two neighbors that are not connected (i.e., not within the transmission range of each other). Coverage nodes (also called gateway nodes) form a CDS. A pruning process can be used to reduce the size of coverage node set while keeping the CDS property. A generalized pruning rule called *pruning rule k* is proposed in [17]. Basically, a coverage node can be withdrawn if its neighbor set can be collectively covered by those of k coverage nodes. In addition, these k coverage nodes have higher priority and are connected. Pruning rule k ensures a constant approximation ratio. The CDS derived from the marking process with Rule k can be locally maintained, when sensors switch on/off.

Wu et al. [40] also discuss the energy-efficient dominating set coverage approach. In general, nodes in the connected dominating set consume more energy in order to handle various bypass traffic than nodes outside the set. To prolong the life span of each node and, hence, the network by balancing the energy consumption in the network, nodes should be alternated in being chosen to form a connected dominating set. A set of power-aware pruning rules is proposed, where coverage nodes are selected based on their energy levels. Simulation results in [40] show that the modified power-aware marking process outperforms several existing approaches in terms of life span of the network.

The node coverage problem can be related to the broadcasting problem, where a small set of forwarding nodes is selected [37]. Forwarding set selection in broadcasting is similar to the point coverage problem, where both try to find a small coverage set. Note that directed diffusion [21] also uses this platform to

collect information through broadcasting. As the interest is propagated through the network, sensor nodes set up reverse gradients to the sink in a decentralized way. The difference is that in the directed diffusion, all sensors forward the data. One difference between node coverage and area coverage is that neighbor set information is sufficient for node coverage, while in area coverage, geometric/directional information is needed.

4.3.2 Multiple Point Coverage

Article [42] considers that the sensor network is sufficiently dense so that covering the sensors can approximate area coverage. Thus, area coverage is reduced to the point coverage. The desired level of coverage is defined as a multiple coverage for the purpose of reliability in case of failure or for other applications related to security (e.g., localized intrusion detection) or localization (e.g. triangulation-based positioning).

The k-coverage set (k-CS) problem is defined as follows: given a constant $k > 0$ and an undirected graph $G = (V, E)$, find a subset of nodes $C \subseteq V$ such that (1) each node in V is dominated (covered) by at least k different nodes in C and (2) the number of nodes in C is minimized. The k-connected coverage set (k-CCS) problem is defined similarly, with the additional requirement (3) the nodes in C are connected.

First, a centralized IP-based approach with performance ratio ρ is proposed for the k-CS problem. Given:

- n nodes s_1, \dots, s_n
- a_{ij} , the coefficients showing the coverage relationship between nodes. These coefficients are defined as follows:

$$a_{ij} = \begin{cases} 1 & \text{if node } s_i \text{ is covered by node } s_j \\ 0 & \text{otherwise} \end{cases}$$

Variables: x_j , boolean variable, for $j = 1 \dots n$:

$$x_j = \begin{cases} 1 & \text{if node } s_j \text{ is selected in the subset } C \\ 0 & \text{otherwise} \end{cases}$$

Integer programming:

$$\text{Minimize } x_1 + x_2 + \dots + x_n$$

$$\text{subject to } \sum_{j=1}^n a_{ij} x_j \geq k \quad \text{for all } i = 1, \dots, n \quad (2)$$

$$x_j \in \{0, 1\} \quad \text{for } i = 1, \dots, n$$

The constraint $\sum_{j=1}^n a_{ij} x_j \geq k$, for all $i = 1, \dots, n$, guarantees that each node in V is covered by at least k nodes in C . Relaxation and rounding technique are used and provide an algorithm with approximation ratio ρ , where $\rho = \Delta + 1$ and Δ the maximum node degree in G . The IP is relaxed to an LP by requiring that $0 \leq x_j \leq 1$

for $j = 1, \dots, n$. As part of the rounding technique, a sensor s_j is added to the set C if its corresponding variable $x_j \geq 1/\rho$.

The second approach [42] is called the cluster-based k-CCS/k-CS algorithm:

1. Using a clustering algorithm to select clusterheads, set C_1 , and the selected clusterheads are marked and removed from the network.
2. Repeat step 1 for k times; mark and remove C_i , $i = 2, \dots, k$.
3. Use a gateway selection approach to select gateways to connect clusterheads in the first set C_1 ; denote the gateway set by D . (This step is removed for without connectivity.)
4. For each node in $C_1 \cup C_2 \cup \dots \cup C_k \cup D$ (clusterhead or gateway), if the number of its marked neighbors t is smaller than k , then it designates $k - t$ unmarked neighbors to be marked.

The clustering algorithm divides the network into several clusters; each cluster has a clusterhead and several neighbors of this clusterhead as members. Any two clusterheads are not neighbors, and the clusterhead set is a maximum independent set of the network in addition to a dominating set. Such a clustering algorithm is provided in [39]. The gateway selection can be tree-based, whereby gateways are selected globally to make the connected domination set (CDS) a tree, or mesh-based, whereby each clusterhead is connected to all of its neighboring clusterheads, and thus the CDS is a mesh structure. For coverage without connectivity, the third step, gateway selection, can be removed.

The third approach [42] is a completely localized solution, where each node determines its status (marked or unmarked) based on its 2-hop neighborhood information:

1. Each sensor node u is given a unique priority $L(u)$, and each node u is represented by tuple $(L(u), ID(u))$, where $ID(u)$ is the ID of u .
2. Each node broadcasts its neighbor set $N(u)$, where $N(u) = \{v | v \text{ is a neighbor of } u\}$.
3. At node u , build a subset: $C(u) = \{v | v \in N(u), L(v) > L(u)\}$. Node u is unmarked if
 - a. Subset $C(u)$ is connected by nodes with higher priorities than u (This constraint is removed for a solution without connectivity)
 - b. For any node $w \in N(u)$, there are k distinct nodes in $C(u)$, say v_1, v_2, \dots, v_k , such that $w \in N(v_i)$

After the algorithm terminates, all the marked nodes form the k-CCS/k-CS. Simulation results show that (i) cluster-based algorithm has better performance in terms of the size of the set C of nodes selected to be active, and (ii) cluster and localized algorithms have better scalability than the IP approach, especially when the network is relatively dense.

4.4 Connected Point Coverage

One of the important issues regards sensor connectivity which forms a network that allows the gathering of the sensed data to the sink. Article [24] addresses the adjustable sensing range connected sensor cover (ASR-CSC) problem: given a set of

targets (or monitoring points) and a set of sensors with adjustable sensing ranges in a WSN, schedule sensors' sensing ranges such that the WSN lifetime is maximized, under the conditions that both target coverage and network connectivity are satisfied, and each sensor energy consumption is upperbounded by the initial energy E .

Two localized algorithms are proposed [24] depending on the order in which connectivity and coverage are ensured. The first algorithm first ensures connectivity, then the target coverage. In the second algorithm, the order is reversed: coverage first, then connectivity.

The first algorithm starts by building a *virtual backbone* consisting of sensor nodes that will be active and participate in data forwarding. It has the property that any sensor node which is not in the backbone is one hop away from a node in the backbone. Thus, if it has to be active for the coverage task, then it can send its sensing data through the backbone. The algorithm executes in rounds, and each round has the following steps: (1) construct a virtual backbone for the WSN; (2) for each sensor in the virtual backbone, set its transmission range d_c and calculate its transmission energy consumption; (3) all sensors including inactive sensors (dominatees) together with active sensors in the virtual backbone (dominators) iteratively adjust their sensing ranges based on contribution until a full coverage is found; and (4) each sensor i active for sensing and connectivity consumes $e p(i) + g_e$ from its residual energy, while sensors active only for connectivity consume g_e .

The virtual backbone is constructed by forming a connected dominating set and pruning redundant sensors by applying the Rule k in [17]. If a sensor node meets the rule's requirement, then it is not part of the backbone in the next round; otherwise, it is part of the backbone for the next round. The coverage mechanism is the same as the one in [8], which means sensors use a waiting time based on the sensor contribution, and when this expires, they decide the coverage responsibilities.

The second algorithm starts by deciding the sensing sensors first. These sensors are connected using Rule k, where sensing nodes have the highest priority; thus, it is ensured they will be part of the backbone. The simulation study shows that the two algorithms have comparable network lifetime and that using sensors with adjustable sensing ranges has a good impact on network lifetime.

Another connected point-coverage approach is presented in [4]. Here the objective is that the sensing nodes are connected to a base station (BS) which collects sensed data, see Fig. 4. This is different from requiring that two sensor nodes be connected to each other, as discussed in the article [24].

The connected set cover (CSC) problem [4] is defined as follows: given a set of sensors s_1, s_2, \dots, s_N , a base station s_0 , and a set of targets r_1, r_2, \dots, r_M , find a family of set covers c_1, c_2, \dots, c_K such that (1) K is maximized, (2) sensors in each set c_k ($k = 1, \dots, K$) are BS-connected, (3) each sensor set monitors all targets, and (4) each sensor appearing in the sets c_1, c_2, \dots, c_K consumes at most E energy.

The first method proposed in [4] is a centralized approach using IP. The target coverage and the energy constraints have been introduced in the previous discussions. The connectivity requirement has been modeled using flows. Each edge has capacity M . Sensing nodes are the nodes in charge with monitoring the targets and they must be able to transmit data to the sink. In Fig. 4, s_1, s_3, s_5 , and s_6 are the

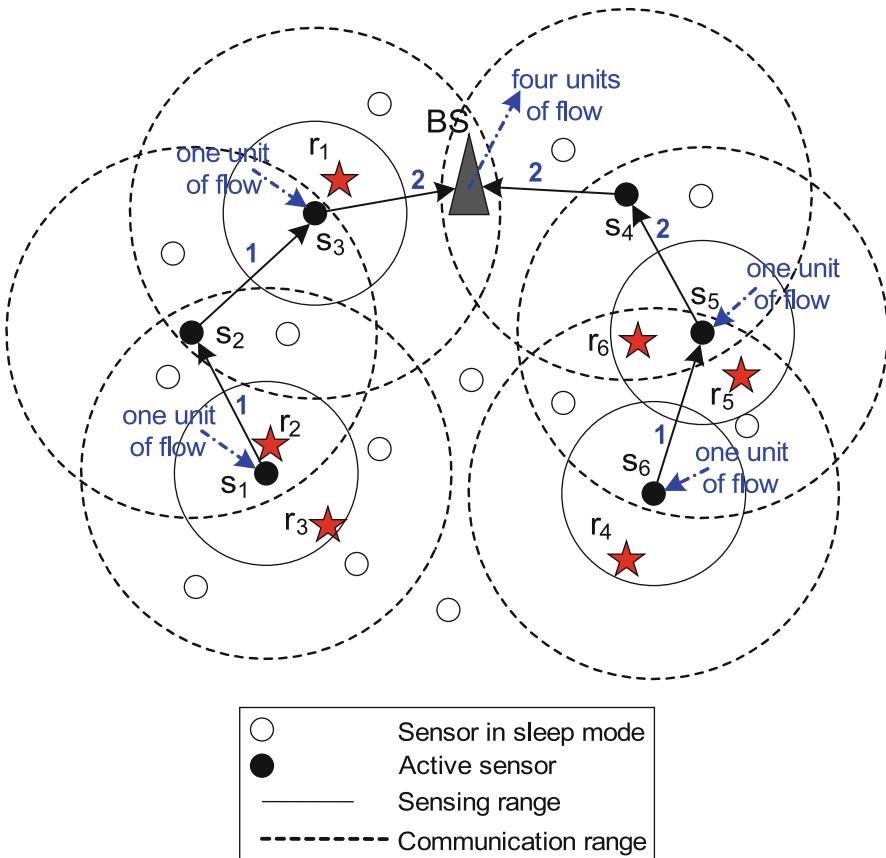


Fig. 4 Connected point coverage

sensing nodes. Some of the sensing nodes (e.g., s_1) may not be able to communicate directly with the BS, so some relay nodes must be used to ensure BS connectivity. In the IP, one unit of flow is inserted at each sensing node (which are the flow sources), and the flow is collected at the BS (which is the flow sink). In Fig. 4, one unit of flow is inserted at each of the four sensing nodes, and four units of flow are collected at the BS. Flow conservation property is also ensured for all nodes in the network.

Secondly, a centralized greedy algorithm [4] is proposed which is executed at the BS. Once the sensors are deployed, they send their coordinates to the BS. The BS computes the connected set covers and broadcasts back the sensors' schedule. The algorithm builds each set consecutively, as long as remaining sensor energy resources allow building a new set. The mechanism to build a set is as follows. At each step, a critical target is selected to be covered. This can be, for example, the target most sparsely covered, both in terms of number of sensors as well as with regard to the residual energy of those sensors. The sensors considered as candidates must have sufficient residual energy. Once the critical target has been

selected, the heuristic selects the sensor with the greatest contribution that covers the critical target. Various sensor contribution functions can be defined. For example, a sensor can be defined to have a greater contribution if it covers a larger number of uncovered targets and if it has more residual energy available. Once a sensor s has been selected, it is added to the current set cover and all additionally targets within range of s are marked as covered.

Once the sensing sensors have been determined, the BS connectivity is ensured by running the breadth-first search algorithm [15] starting from the BS. The resulting breadth-first tree is pruned: only the paths from the sensing nodes to the BS are kept. The sensors on these paths become relay nodes for the current round.

The third mechanism [4] is a localized approach, which runs in rounds. Each round starts with an initialization phase where sensors decide whether they will be in active mode (sensing or relaying) or sleep mode during the current round. First, the sensing nodes are selected. Each node with uncovered targets has a waiting time which is inverse proportional with the residual energy and the number of uncovered targets. When the timer expires, the node declares itself as a sensing node for the current round and broadcasts this decision and the set of covered targets to its 2-hop neighborhood.

In the second part, the relay nodes are selected. Each sensing node builds a virtual tree where the set of vertices is comprised of the BS and the set of targets. The tree is a minimum spanning tree built using Prim's algorithm [15], where the BS is the root. Each target t has a supervisor sensor, which is the first sensing sensor which covers it. Each supervisor selects relay nodes to allow it to connect with the supervisor of t 's parent in the virtual tree. Simulation results show that the IP algorithm finds the largest number of covers, but it has a long run time and therefore is not feasible for large scenarios. In general, the distributed algorithm performs better than the greedy approach because it employs a virtual tree-based mechanism for finding relay nodes, and this promotes the reuse of already active supervisor nodes, while the greedy one uses a breadth-first tree that results in shorter sensor-BS paths, involving more relay nodes.

4.5 Point Coverage in Heterogeneous Wireless Sensor Networks

Article [1] considers a heterogeneous WSNs that contains two types of wireless devices: resource-constrained wireless sensor nodes deployed randomly in large number and several resource-rich, predeployed supernodes, see Fig. 2b. Sensor nodes transmit and relay measurements. Once data packets encounter a supernode, they are forwarded using fast supernode to supernode communication toward the user application. Additionally, supernodes could process sensor data before forwarding.

The heterogeneous connected set covers problem is defined as follows: given a set of points (or targets) t_1, t_2, \dots, t_T , a set of supernodes g_1, g_2, \dots, g_M , and a set of randomly deployed sensors s_1, s_2, \dots, s_N , find a family of sensor set covers

c_1, c_2, \dots, c_p , such that (1) P is maximized, (2) sensors in each set cover c_p ($p = 1, \dots, P$) are connected to supernodes, (3) each sensor set monitors all targets, and (4) each sensor appearing in the sets c_1, c_2, \dots, c_p consumes at most E energy.

Network activity is organized in rounds. Each round has two phases: initialization and data collection. During the initialization phase, a set of active sensors (e.g., the set cover c_i) is established such that conditions (2), (3), and (4) are satisfied. During the data collection phase, sensors in the set cover c_i are active, while all other sensors are in the sleep mode for the rest of the round, and they will wake up for the next initialization phase.

Building a set cover c_i has two steps: (1) choosing the sensing nodes and (2) choosing the relay nodes. To select the sensing nodes, each node defines a waiting time which is smaller for sensors that have a higher residual energy and cover a larger number of uncovered targets. When a timer expires and a sensor node decides to be a sensing node, then it broadcasts this decision to its 2-hop neighborhood. Nodes that receive this message update the waiting time accordingly.

Three mechanisms are proposed to select the relay nodes: two of them use a cluster-based approach, and the third one uses a greedy mechanism. For the clustering mechanisms, supernodes serve as clusterheads. Each supernode serves as clusterhead, and it broadcasts a *CLUSTER-INIT* message containing the supernode id and the number of hops which is initially zero. Each sensor node maintains information about the closest supernode and forwards only messages from which it learns about a closer supernode. Once the clusters have been constructed, there are two algorithms for relay nodes selection: the shortest-path mechanism and the Rule k mechanism.

In the shortest-path mechanism, each sensing node broadcasts a special control message *RELAY-REQ* to the supernode of the cluster where it belongs to. The message is sent along the parent path which was set up during cluster formation. All nodes that participate in relaying the *RELAY-REQ* message become relay nodes.

In the Rule k mechanism, each cluster selects first a backbone using the Rule k algorithm [37]. Then, similar with the previous mechanism, a *RELAY-REQ* message is sent by the supernode, but this time, the message is forwarded only by the backbone nodes. Sensing nodes send back *RELAY-REQ* messages that help set up the relay nodes.

In the greedy approach, one or more connected components are formed such that the sensing nodes in each component are connected through relay nodes to a supernode. The goal is to activate a minimum number of relay nodes in order to satisfy the supernode connectivity requirement. Components are built in a greedy fashion, similar to Kruskal's algorithm [15], by successively merging two components connected by a minimum number of relay nodes. Components merge successively until each component contains one supernode. More details are presented in [1].

Simulation results show that the greedy algorithm performs better than the cluster Rule k which performs better than cluster shortest-path. Still the complexity of the greedy is higher since it operates over the whole network. Greedy gets the best

results since it minimizes the number of relay nodes added on the whole network, while the two other algorithms minimize the number of relay nodes added per cluster.

4.6 Minimum Coverage Breach Under Bandwidth Constraints

An efficient method used for data reporting is to divide sensor nodes in sets; then only one set is active in each round, while all other sensors are in sleep mode to conserve their energy resources. Consider the case when the BS is within direct communication of all sensors. If each active sensor sends the sensed data to the BS, this requires that there must be sufficient bandwidths for this activity. The bandwidth could be the total number of time slots if a time division scheme is used on a single shared channel or the total number of channels if multiple channels are available.

The issue occurs when there are W channels available and a subset has more than W sensors. Then some sensors cannot have channel access for data transmission. To properly allocate time slots or channels to sensors, the scheduling techniques must select sensors in each set such that they satisfy the coverage requirement while being fully restricted by the bandwidth constraints. If a target (or monitor region) is not covered by any active sensor, it is called *breached*. The objective is to minimize the total breach of all targets.

The minimum breach problem has been introduced in [13]: given a set A of fixed points, and a set S of sensor nodes, organize sensor nodes into disjoint subsets $C_i = \{s_{i_1}, s_{i_2}, \dots\}$, $i = 1, \dots, K$, where each subset $|C_i| \leq W$ and the overall breach is minimized. Article [13] shows that the decision version of the minimum breach problem is NP-complete. Then two solutions are proposed using IP and LP approaches. Simulation results show that to improve the coverage performance in WSNs, the bandwidth also needs to be increased; in bandwidth constrained networks, increasing the number of sensors alone does not always improve the coverage results.

5 Conclusion

This chapter categorizes and describes recent area and point coverage problems proposed in literature, their formulations, assumptions, and proposed solution. Sensor coverage is an important element of QoS in applications with WSNs. Coverage is, in general, associated with energyefficiency and network connectivity, two important properties of a WSNs. To accommodate a large WSN with limited resources and a dynamic topology, coverage control algorithms and protocols perform best if they are distributed and localized. Various interesting formulations for sensor coverage have been proposed in literature. Scheduling sensor nodes to alternate between sleep and active mode is an important method to conserve energy resources. Such mechanisms that efficiently organize or schedule the sensor activity after deployment are very efficient and have a direct impact on prolonging the

network lifetime. Sensor coverage with various design choices, such as breach coverage, composite event coverage, heterogeneous WSNs, and so on, poses interesting research problems. The literature on coverage problems is more vast than the papers surveyed here, and the readers are invited to explore them in the research literature.

Acknowledgements This work was supported in part by NSF grant CCF 0545488.

Cross-References

► Connected Dominating Set in Wireless Networks

Recommended Reading

1. W. Awada, M. Cardei, Energy-efficient data gathering in heterogeneous wireless sensor networks, in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob'06)*, Montreal, 2006
2. Y. Bi, L. Sun, J. Ma, N. Li, I.A. Khan, C. Chen, HUMS: an autonomous moving strategy for mobile sinks in data-gathering sensor networks. *EURASIP J. Wirel. Commun. Netw.* **2007**, 1–16 (2007)
3. J. Butler, Robotics and microelectronics: mobile robots as gateways into wireless sensor networks, in *Technology@Intel Magazine*, 2003
4. I. Cardei, M. Cardei, Energy-efficient connected-coverage in wireless sensor networks. *Int. J. Sens. Netw.* **3**(3), 201–210 (2008)
5. M. Cardei, D.-Z. Du, Improving wireless sensor network lifetime through power aware organization. *ACM Wirel. Netw.* **11**(3), 333–340 (2005)
6. M. Cardei, D. MacCallum, X. Cheng, M. Min, X. Jia, D. Li, D.-Z. Du, Wireless sensor networks with energy efficient organization. *J. Interconnect. Netw.* **3**(3–4), 213–229 (2002)
7. M. Cardei, M. Thai, Y. Li, W. Wu, Energy-efficient target coverage in wireless sensor networks, in *IEEE INFOCOM*, Miami, USA, 2005
8. M. Cardei, J. Wu, M. Lu, Improving network lifetime using sensors with adjustable sensing ranges. *Int. J. Sens. Netw.* **1**(1/2), 41–49 (2006)
9. J. Carle, D. Simplot, Energy efficient area monitoring by sensor networks. *IEEE Comput.* **37**(2), 40–46 (2004)
10. A. Chakrabarti, A. Sabharwal, B. Aazhang, Using predictable observer mobility for power efficient design of sensor networks, in *Proceedings of the 2nd IEEE IPSN*, Palo Alto, 2003
11. S. Chellappan, X. Bai, B. Ma, D. Xuan, C. Xu, Mobility limited flip-based sensor networks deployment. *IEEE Trans. Parallel Distrib. Syst.* **18**(2), 199–211 (2007)
12. B. Chen, K. Jamieson, H. Balakrishnan, R. Morris, Span: an energy-efficient coordination algorithm for topology maintenance in Ad Hoc wireless networks, in *ACM/IEEE International Conference on Mobile Computing and Networking*, Rome, 2001, pp. 85–96
13. M.X. Cheng, L. Ruan, W. Wu, Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks, in *IEEE INFOCOM*, Miami, 2005, pp. 2638–2645
14. C.-Y. Chong, S.P. Kumar, Sensor networks: evolution, opportunities, and challenges. *Proc. IEEE* **91**(8), 1247–1256 (2003)
15. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 3rd edn. (MIT, Cambridge, 2009). ISBN:0262033844
16. Crossbow Technology (2011), <http://www.xbow.com:81/Products/wproductsoverview.aspx>. Accessed Dec 2011

17. F. Dai, J. Wu, Distributed dominant pruning in Ad Hoc wireless networks, in *Proceedings of IEEE International Conference on Communications*, Anchorage, 2003
18. K. Dantu, M.H. Rahimi, H. Shah, S. Babel, A. Dhariwal, G.S. Sukhatme, Robomote: enabling mobility in sensor networks, in *IPSN*, Los Angeles, 2005, pp. 404–409
19. P.B. Gibbons, B. Karp, Y. Ke, S. Nath, S. Seshan, IrisNet: an architecture for a worldwide sensor web. *IEEE Pervasive Comput.* **2**, 22–33 (2003)
20. W.R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocols for wireless microsensor networks, in *Proceedings of HICSS*, Maui, 2000
21. C. Intanagonwiwat, R. Govindan, D. Estrin, Directed diffusion: a scalable and robust communication paradigm for sensor networks, in *Proceedings of ACM MobiCom*, Boston, 2000, pp. 56–67
22. A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. Sigurdsson, K. Sikorski, D. Fox, G. Borriello, in *Making Sensor Networks Practical with Robots*, ed. by F. Mattern, M. Naghshineh. Lecture Notes in Computer Science (Springer, New York, 2002), pp. 152–166
23. U. Lee, E.O. Magistretti, B.O. Zhou, M. Gerla, P. Bellavista, A. Corradi, Efficient data harvesting in mobile sensor platforms, in *PerCom Workshops*, 2006, pp. 352–356
24. M. Lu, J. Wu, M. Cardei, M. Li, Energy-efficient connected coverage of discrete targets in wireless sensor networks. *Int. J. Ad Hoc. Ubiquitous Comput.* **4**(3/4), 137–147 (2009)
25. J. Luo, J.-P. Hubaux, Joint mobility and routing for lifetime elongation in wireless sensor networks, in *IEEE INFOCOM*, Miami, 2005
26. M. Marta, M. Cardei, Improved sensor network lifetime with multiple mobile sinks. *J. Pervasive Mobile Comput.* **5**(5), 542–555 (2009). Elsevier
27. M. Marta, Y. Yang, M. Cardei, Energy-efficient composite event detection in WSNs, in *International Conference on Wireless Algorithms, Systems and Applications (WASA'09)*, Boston, 2009
28. S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. Srivastava, Coverage problems in wireless Ad-Hoc sensor networks, in *IEEE INFOCOM*, Anchorage, 2001, pp. 1380–1387
29. V. Raghunathan, C. Schurgers, S. Park, M.B. Srivastava, Energy-aware wireless microsensor networks. *IEEE Signal Process. Mag.* **19**, 40–50 (2002)
30. R.C. Shah, S. Roy, S. Jain, W. Brunette, Data MULEs: modeling a three-tier architecture for sparse sensor networks, in *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03)*, Anchorage, 2003
31. S. Slijepcevic, M. Potkonjak, Power efficient organization of wireless sensor networks, in *Proceedings of IEEE International Conference on Communications*, Kyoto, 2001, pp. 472–476
32. A.A. Somasundara, A. Kansal, D.D. Jea, D. Estrin, M.B. Srivastava, Controllably mobile infrastructure for low energy embedded networks. *IEEE Trans. Mobile Comput.* **5**, 958–973 (2006)
33. D. Tian, N.D. Georganas, A coverage-preserving node scheduling scheme for large wireless sensor networks, in *Proceedings of the 1st ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, 2002
34. L. Tong, Q. Zhao, S. Adireddy, Sensor networks with mobile agents, in *Proceedings of IEEE Military Communications Conference (MILCOM'03)*, Boston, 2003
35. C.T. Vu, R.A. Beyah, Y. Li, Composite event detection in wireless sensor networks, in *IEEE International Performance, Computing, and Communications Conference*, New Orleans, 2007
36. X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, C.D. Gill, Integrated coverage and connectivity configuration in wireless sensor networks, in *ACM Conference on Embedded Networked Sensor Systems*, 2003
37. J. Wu, F. Dai, Broadcasting in Ad Hoc networks based on self-pruning, in *Proceedings of the 22nd Annual Joint Conference of IEEE Communication and Computer Society*, San Franciso, 2003

38. J. Wu, H. Li, On calculating connected dominating set for efficient routing in Ad Hoc wireless networks, in *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Seattle, 1999, pp. 7–14
39. J. Wu, W. Lou, Forward node set based broadcast in clustered mobile ad hoc networks. *Wirel. Commun. Mobile Comput.* **3**(2), 141–154 (2003)
40. J. Wu, F. Dai, M. Gao, I. Stojmenovic, On calculating power-aware connected dominating sets for efficient routing in Ad Hoc wireless networks. *J. Commun. Netw.* **1**, 1–12 (2002)
41. Y. Yang, M. Cardei, Adaptive energy efficient sensor scheduling for wireless sensor networks. *Optim. Lett.* **4**(3), 359–369 (2010). Springer, ISSN:1862–4472
42. S. Yang, F. Dai, M. Cardei, J. Wu, F. Patterson, On connected multiple point coverage in wireless sensor networks. *Int. J. Wirel. Inf. Netw.* **13**(4), 289–301 (2006)
43. F. Ye, G. Zhong, S. Lu, L. Zhang, Energy efficient robust sensing coverage in large sensor networks. Technical Report UCLA, 2002
44. H. Zhang, J.C. Hou, Maintaining sensing coverage and connectivity in large sensor networks. Technical Report UIUC, UIUCDCS-R-2003-2351, 2003

Data Correcting Approach for Routing and Location in Networks

Boris Goldengorin

Contents

1	Introduction	930
2	The Asymmetric Traveling Salesman Problem.....	934
2.1	The Data-Correcting Algorithm.....	934
2.2	Computational Experience with ATSP Instances.....	938
3	The Simple Plant Location Problem.....	940
3.1	A Pseudo-Boolean Formulation of the SPLP.....	941
3.2	Preprocessing SPLP Instances.....	943
3.3	The Data-Correcting Algorithm.....	946
3.4	Computational Experience with SPLP Instances.....	947
4	The p -Median Problem	955
4.1	Introduction.....	957
4.2	Brief Overview of PMP Formulations.....	959
4.3	The Mixed Boolean Pseudo-Boolean Model (MBpBM).....	967
4.4	Reduction Tests for the p -Median Problem.....	972
4.5	Computational Results.....	973
4.6	Summary and Future Research Directions.....	977
5	Maximization of General Submodular Functions.....	978
5.1	A Simple Data-Correcting Algorithm.....	979
5.2	A Data-Correcting Algorithm Based on Multi-level Search.....	981
5.3	Computational Experience with Quadratic Cost Partition Instances.....	986
6	Conclusion	989
	Cross-References	990
	Recommended Reading.....	990

B. Goldengorin (✉)

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: goldengorin@ufl.edu; goldengorin@gmail.com

Abstract

A data-correcting (DC) algorithm is a branch-and-bound-type algorithm, in which the data of a given problem is “heuristically corrected” at the various stages in such a way that the new instance will be polynomially solvable and its optimal solution is within a prespecified deviation (called prescribed accuracy) from the optimal solution to the original problem. The DC approach is applied to determining exact and approximate global optima of NP-hard problems. DC algorithms are designed for various classes of NP-hard problems including the quadratic cost partition (QCP), simple plant location (SPL), and traveling salesman problems based on the algorithmically defined polynomially solvable special cases. Results of computational experiments on the publicly available benchmark instances as well as on random instances are presented. The striking computational result is the ability of DC algorithms to find exact solutions for many relatively difficult instances within fractions of a second. For example, an exact global optimum of the QCP problem with 80 vertices and 100 % density was found within 0.22 s on a PC with 133-Mhz processor, and for the SPL problem with 200 sites and 200 clients, within 0.2 s on a PC with 733-Mhz processor.

1 Introduction

Let us consider the following combinatorial minimization problem (CMP) (E, C, D, f_C) which is the problem of finding

$$S^* \in \arg \min\{f_C(S) \mid S \in D\},$$

where $C : E \rightarrow \Re$ is the given *instance* of the problem with a *ground set* E satisfying $|E| = m$ ($m \geq 1$), $D \subseteq 2^E$ is the *set of feasible solutions*, and $f_C : 2^E \rightarrow \Re$ is the *objective function* of the problem. By $D^* = \arg \min\{f_C(S) \mid S \in D\}$, the set of optimal solutions is denoted. It is assumed that $D^* \neq \emptyset$ and that $S \neq \emptyset$ for some $S \in D$.

Classic examples of routing and location problems include, among others, the following:

- (A) The asymmetric traveling salesman problem (ATSP) [29]: it is given $n \geq 3$ cities and distances $C = ||c(i, j)||$ between i -th and j -th cities, find a Hamiltonian cycle of minimum length, which enters and leaves each city exactly once. Here in the CMP (E, C, D, f_C) , $E = A$ is the set of arcs in a directed weighted simple graph $G = (V, A, C)$ with the set of vertices $V = \{1, 2, \dots, n\}$, and $A \subset V \times V$ is the set of arcs $(i, j) \in A$ with nonnegative weights (distances) $c(i, j)$ connecting vertices i and j ; D is the set of all Hamiltonian cycles, and the objective function of ATSP f_C is given by $f_C(S) = \sum_{(i,j) \in S} c(i, j)$ for $S \in D$.

- (B) The assignment problem (AP): it is given a directed graph $G = (V, A, C)$ as above and a nonnegative cost function C on $E = A$, find a vertex permutation $\pi_a^* : V \rightarrow V$ such that $\sum_{i=1}^n c(i, \pi_a(i))$ is minimized on the set of all possible permutations of V . An optimal solution to the AP is denoted by π_a^* and its optimal value $f_C(\pi_a^*)$. Note that a set $S \subset A$ is a feasible solution of the AP if it is of the form $S = \{(1, \pi(1)), (2, \pi(2)), \dots, (n, \pi(n))\}$ for some permutation π of V . Clearly, a Hamiltonian cycle corresponds to the cyclic permutation π_c of V and vice versa.
- (C) The simple plant location problem (SPLP): it is given a weighted bipartite graph $G = (V, A, C, F)$ with the set of vertices $V = I \cup J$, such that $I = \{1, \dots, m\}$ is the set of sites in which plants can be located and $J = \{1, \dots, n\}$ is the set of clients, each having a unit demand. Thus, $A \subseteq I \times J$ is the set of arcs, $F = (f_i)$ are fixed costs for setting up plants at sites $i \in I$, and $C = ||c(i, j)||$ is a matrix of transportation costs from $i \in I$ to $j \in J$ as input. The SPLP computes a set $S^* : \emptyset \subset S^* \subseteq I$, at which plants can be located so that the total costs of satisfying all client demands are minimal. The costs involved in meeting the client demands include the fixed costs of setting up plants and the transportation costs of supplying clients from the plants that are set up. An instance of the problem is described by an m -vector $F = (f_i)$ and an $m \times n$ matrix $C = ||c(i, j)||$. It is assumed that F and C are nonnegative, that is, $F \in \Re_+^m$ and $C \in \Re_+^{mn}$. The $m \times (n + 1)$ augmented matrix $[F|C]$ will be used as a shorthand for describing an instance of the SPLP. The total cost $f_{[F|C]}(S)$ associated with a solution S consists of two components, the fixed costs $\sum_{i \in S} f_i$ and the transportation costs $\sum_{j \in J} \min\{c(i, j) : i \in S\}$, that is,

$$f_{[F|C]}(S) = \sum_{i \in S} f_i + \sum_{j \in J} \min\{c(i, j) : i \in S\},$$

and the SPLP is the problem of finding

$$S^* \in \arg \min\{f_{[F|C]}(S) : \emptyset \subset S \subseteq I\}. \quad (1)$$

- (D) The p-median problem (PMP): the PMP is defined on a weighted bipartite graph $G = (V, A, C)$ and related to the SPLP as follows. Given a set I of m potential facilities, a set J of n clients (or customers), a distance function $c : I \times J \rightarrow \Re_+$, and a constant $p \leq m$, determine which p facilities to open so as to minimize the sum of the distances from each client to its closest open facility. The PMP and SPLP differ in the following details. First, the SPLP involves a fixed cost for locating a facility at a given location, while the PMP does not. Second, unlike the PMP, SPLP does not have a constraint p on the maximum number of facilities. Typical SPLP formulations separate the set of candidate sites from the set of clients. In the PMP, these sets are identical, that is, $I = J$. The objective function f_C is given by $f_C(S) = \sum_{j \in J} \min\{c(i, j) : i \in S\}$, and the PMP is the problem of finding

$$S^* \in \arg \min \{f_C(S) : \emptyset \subset S \subseteq I, |S| = p\}. \quad (2)$$

The adopted approach is to develop what is termed the data-correcting algorithm (DCA), which forms a class of algorithms, introduced in Goldengorin [33, 37] for the solution of NP-hard problems. Crucial in these algorithms is the fact that the data of a given problem instance (E, C, D, f_C) is corrected to obtain a new problem instance (E, H, D, f_H) belonging to a polynomially solvable class of instances \mathcal{P} . A polynomially solvable class of instances that is used in DCA might be determined either algorithmically or analytically [29].

For example, let us consider the DC algorithm applied to the traveling salesman problem (TSP) with the following algorithmically defined polynomially solvable class. A nonnegative square $n \times n$ -matrix $H = ||h(i, j)||$ is called a *Hungarian matrix* (see [44]) if the Hungarian algorithm (see, e.g., [64]) with input algorithm (see, e.g., [64]) and with input H results in an optimal Hamiltonian cycle: notation $H \in \mathcal{P} = \mathcal{H}$. If a nonnegative square $n \times n$ -matrix $C = ||c(i, j)||$ is not Hungarian, then by correcting some entries of C , a Hungarian matrix H that is as close as possible to C will be tried to be found for some *proximity measure*

$$\rho(C, H) = \min\{\mu(C, H), \nu(C, H)\} \quad (3)$$

of C and H , with $\mu(C, H) = \sum_{i=1}^n \max\{|c(i, j) - h(i, j)| : j = 1, \dots, n\}$ and $\nu(C, H) = \sum_{j=1}^n \max\{|c(i, j) - h(i, j)| : i = 1, \dots, n\}$.

The proximity measure $\rho(C, H)$ is used in the framework of the DCA for finding, by means of a heuristic procedure, an instance H that is as close as possible to C . Usually, this heuristic can be easily constructed by a simple modification of a polynomial algorithm by which a polynomially solvable class is defined. In case of the Hungarian algorithm, the so-called Hungarian matrix is obtained by patching the subcycles [51].

In the following theorem, which is first published in Goldengorin [37], it is formulated that the proximity measure is an upper bound for the difference of the lengths of shortest Hamiltonian cycles of C and H , denoted by $f_C(\pi_C^*)$ and $f_H(\pi_H^*)$, respectively, with the corresponding optimal permutations π_C^* and π_H^* .

Theorem 1 *Let $C, H \in R^{n \times n}$. Then the following holds: $|f_C(\pi_C^*) - f_H(\pi_H^*)| \leq \rho(C, H)$.*

Proof This proof is split into two cases:

Case 1. $f_C(\pi_C^*) \geq f_H(\pi_H^*)$ and Case 2. $f_C(\pi_C^*) < f_H(\pi_H^*)$.

Theorem 1 will be proved because the proof of Case 2 might be done in similar lines as follows: $0 \leq f_C(\pi_C^*) - f_H(\pi_H^*) = |f_C(\pi_C^*) - f_H(\pi_H^*)| \leq f_C(\pi_C^*) - f_H(\pi_H^*) = \sum_{(i,j) \in \pi_H^*} c(i, j) - \sum_{(i,j) \in \pi_H^*} h(i, j) = \sum_{(i,j) \in \pi_H^*} [c(i, j) - h(i, j)] \leq \sum_{(i,j) \in \pi_H^*} |c(i, j) - h(i, j)| = A$. Note that $A \leq \sum_{i=1}^n \max\{|c(i, j) - h(i, j)| : j = 1, \dots, n\} = \rho(C, H)$.

$j = 1, \dots, n\} = \mu(C, H)$ and $A \leq \sum_{j=1}^n \max\{|c(i, j) - h(i, j)| : i = 1, \dots, n\} = v(C, H)$, and hence, $A \leq \min\{\mu(C, H), v(C, H)\} = \rho(C, H)$. \square

If $\rho(C, H)$ is smaller than a prescribed accuracy ε_0 , then the ATSP solution with respect to H is also an ε_0 solution with respect to C . Otherwise, the DCA decreases the current value of $\rho(C, H)$ by means of a binary branching procedure which cancels the largest contribution $0 < |c(i, j) - h(i, j)|$ to $\rho(C, H)$ by an inclusion of the arc (i, j) in all Hamiltonian cycles with $h(i, j) = c(i, j)$ and an exclusion of the arc (i, j) from all Hamiltonian cycles with $h(i, j) = \infty$. In other words, the DCA gives an answer to the most challenging question for any branch-and-bound algorithm: by means of which element (arc in case of the ATSP) one should branch such that to arrive to a feasible solution with the prescribed accuracy within the minimum number of branchings. Moreover, [Theorem 1](#) provides virtual lower lb and upper ub bounds to the unknown optimal value $f_C(\pi_C^*)$ as follows.

Corollary 1 *Let $C, H \in R^{n \times n}$. Then the following holds: $lb = f_H(\pi_H^*) - \rho(C, H) \leq f_C(\pi_C^*) \leq f_H(\pi_H^*) + \rho(C, H) = ub$.*

Another example can be described as follows. Let Φ be a function defined on a set D , and let Ψ belong to a polynomially solvable class of functions which is a subclass of a given class of functions Φ defined also on D , and let $\rho(\Phi, \Psi) = \max\{|\Phi(S) - \Psi(S)| : S \in D\}$ be the proximity measure. If the optimal values of $\Phi(S)$ and $\Psi(S)$ are denoted on D by $\Phi^*(D)$ and $\Psi^*(D)$, respectively, then an analogue of the above-mentioned inequality is read as follows.

Theorem 2 *Let Φ and Ψ be functions on the set D with optimal values $\Phi^*(D)$ and $\Psi^*(D)$. Then the following holds: $|\Phi^*(D) - \Psi^*(D)| \leq \rho(\Phi, \Psi)$.*

Again, if $\rho(\Phi, \Psi)$ is smaller than a prescribed accuracy ε_0 , then the problem of finding $\Phi^*(D)$ with the given prescribed accuracy ε_0 is solved by $\Psi^*(D)$. More information about DC algorithms (general scheme, comparison with branch-and-bound-type algorithms, steps of construction, methods, etc.) can be found in Goldengorin [33, 37].

This chapter describes a step in the direction of incorporating polynomially solvable special cases into approximation algorithms. *Data-correcting algorithms* are reviewed – approximation algorithms that make use of polynomially solvable special cases to arrive at high-quality solutions. The basic insight that leads to these algorithms is the fact that it is often easy to compute a bound on the difference between the costs of optimal solutions to two instances of a problem, even though it may be hard to compute optimal solutions for the two instances. These algorithms were first reported in the Russian literature (see [32–36]) and later on in the Western literature (see, e.g., [27, 28, 38, 40–42, 44, 46]).

The approximation in data-correcting algorithms is determined in terms of an *accuracy parameter*, which is an upper bound on the difference between the

objective value of an optimal solution to the instance and that of a solution returned by the data-correcting algorithm. Note that this is *not* expressed as a fraction of the optimal objective value for this instance as in common ε -optimal algorithms but as actual deviations from the cost of optimal solutions.

Although it is suggested to use the data-correcting algorithms to solve NP-hard routing and location problems, they form a general problem-solving tool and can be used for functions defined on a continuous domain with a finite range as well (see [27, 28, 38]).

The next four sections contain a description of actual implementation of data-correcting algorithms to four problems: the asymmetric traveling salesman problem, simple plant location problem, p-median problem, and maximization of a general submodular function. It means that the problem examples cover the most widely used models in routing and location problems, namely, problems defined either on a subset of permutations (see, e.g., [30]) or on a subset of all sets of the finite (ground) set (see, e.g., [22]). This chapter is an essentially revised chapter of Ghosh et al. [27] which is concentrated on data-correcting algorithms to solve routing [30] and location problems.

2 The Asymmetric Traveling Salesman Problem

In an asymmetric traveling salesman problem (ATSP) instance there are given a weighted digraph $G = (V, A, C)$ and a $|V| \times |V|$ distance matrix $C = ||c(i, j)||$, and the objective is to output a least cost Hamiltonian cycle in this graph. This is one of the most studied problems in combinatorial optimization (see Lawler et al. [58] and Gutin and Punnen [47] for a detailed introduction).

2.1 The Data-Correcting Algorithm

[Theorem 1](#) presents the idea of data-correcting algorithm (DCA) for solving the ATSP instances.

Before presenting a pseudocode for the DCA, let us illustrate the data-correcting step for the ATSP with an example. Consider the six-city ATSP instance with the distance matrix $C = ||c(i, j)||$ shown below.

C	1	2	3	4	5	6
1	-	10	16	19	25	22
2	19	-	10	13	13	10
3	10	28	-	22	16	13
4	19	25	13	-	10	19
5	16	22	19	13	-	11
6	13	22	15	13	10	-

If subtours in solutions to the ATSP are allowed, the assignment problem relaxation is obtained. Solving the assignment problem on C , using the Hungarian method, the following reduced distance matrix $C^H = [c^H(i, j)]$ is got.

C^H	1	2	3	4	5	6
1	-	0	6	7	16	12
2	9	-	0	1	4	0
3	0	18	-	10	7	3
4	8	14	2	-	0	8
5	5	11	8	0	-	0
6	2	11	4	0	0	-

This leads to a solution with two cycles (1231) and (4564) (corresponding to π_H^*). Using patching techniques (see, e.g., [58]), a feasible ATSP solution (1245631) is obtained. Notice that (1245631) would be an optimal solution to the assignment problem if $c^H(2, 4)$ and $c^H(6, 3)$ had been set to zero in C^H , and that would have been the situation if $c(2, 4)$ and $c(6, 3)$ were initially reduced by 4 and 1, respectively, that is, if the distance matrix in the original ATSP instance was a Hungarian matrix H defined below.

H	1	2	3	4	5	6
1	-	10	16	19	25	22
2	19	-	10	9	13	10
3	10	28	-	22	16	13
4	19	25	13	-	10	19
5	16	22	19	13	-	11
6	13	22	14	13	10	-

Therefore, H is the distance matrix of the correction of the instance with distance matrix C . The proximity measure $\rho(C, H) = \sum_{i=1}^6 \sum_{j=1}^6 |c(i, j) - h(i, j)| = |c(2, 4) - h(2, 4)| + |c(6, 3) - h(6, 3)| = 4 + 1 = 5$.

A proximity measure (3) is an upper bound to the difference between the costs of two solutions for a problem instance, so the stronger the bound, the better would be the performance of any enumeration algorithm dependent on such bounds.

The pseudocode for a recursive version of the DCA for the ATSP is given below.

Note that a good lower bound can be incorporated into DCA-ATSP to make it more efficient.

Algorithm DCA-ATSP(G, ε)

Output: A tour π_C^* such that the difference between the cost of π_C^* and the cost of an optimal tour π_C^* is not more than ε

Code:

```

1. begin
2.    $\pi_C :=$  an arbitrary tour in  $G$ ;
3.    $lb :=$  a lower bound on the cost of an optimal tour;
4.   if  $f_C(\pi_C) = lb$  return  $\pi_C$ ;
5.    $\pi_a^* :=$  an optimal solution to the assignment problem on  $G$ ;
6.   construct a solution  $\pi_p$  from  $\pi_a^*$  through patching;
7.   using  $\pi_a^*$ , compute the proximity measure  $\rho$ ;
8.   if  $\rho \leq \alpha$ 
9.     return  $\pi_C$ ;
10.    else begin
11.      branch according to a pre-decided branching rule;
12.      for each subproblem  $i$  generated
13.         $\pi_C(i) :=$  the solution output by DCA-ATSP on subproblem  $i$ ;
14.      return the best solution from among all  $\pi_C(i)$ 's;
15.    end;
16. end.

```

(* Branch *)

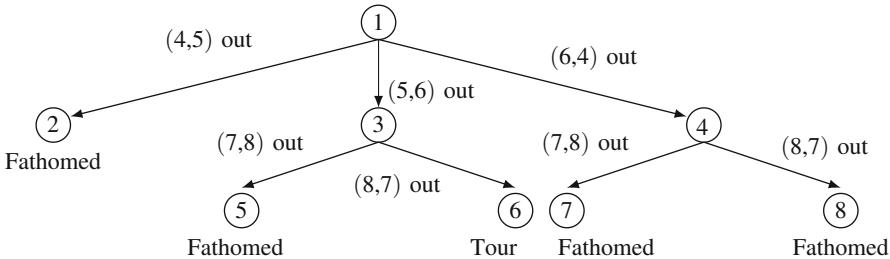
Next, the DCA-ATSP algorithm above will be illustrated on an instance of the ATSP. Consider the eight-city ATSP instance with the distance matrix $D = [d(ij)]$ shown below (this example was taken from Balas and Toth [5], p. 381).

D	1	2	3	4	5	6	7	8
1	-	2	11	10	8	7	6	5
2	6	-	1	8	8	4	6	7
3	5	12	-	11	8	12	3	11
4	11	9	10	-	1	9	8	10
5	11	11	9	4	-	2	10	9
6	12	8	5	2	11	-	11	9
7	10	11	12	10	9	12	-	3
8	7	10	10	10	6	3	1	-

The following are used:

- The proximity measure ρ , see (3), for data correction
- The assignment algorithm to compute lower bounds for subproblems
- A patching algorithm to create feasible solutions and compute proximity measures
- The patched solution derived from the assignment solution as a feasible solution

The branching rule used in this example is as follows. At each subproblem, the assignment problem solution is constructed and then patched. Also, the original matrix is corrected to a new matrix that would output the patched solution if the assignment problem is solved on it. Next, the arc corresponding to the entry in the new matrix that had to be corrected by the maximum amount is identified. The tail



Subproblem at node	Upper bound	Lower bound	Assignment solution	Patched tour	Cost of patching	Revised bound
1	∞	17	(1231)(4564)(787)	(123786451)	9	26
2	26	29		Fathomed by bounds	–	26
3	26	25	(12631)(454)(787)	(126453781)	6	26
4	26	25	(12631)(454)(787)	(126453781)	6	26
5	26	31		Fathomed by bounds	–	26
6	26	26	(123786451)	Patching not required	26	26
7	26	31		Fathomed by bounds	–	26
8	26	27		Fathomed by bounds	–	26

Fig. 1 Branch-and-bound tree for the instance in the example

of this arc is identified, and it should be branched on all the arcs in the subtour containing that vertex. For example, in this problem, the assignment solution is (1231)(4564)(787), which is patched to (123786451), and the cost of patching is 9. If the problem data is corrected, it will be seen that the entry $d(5, 1)$ (corresponding to arc (5, 1)) contributes the maximum amount (7) to the patching. Hence, it should be branched on each arc in the cycle (4564), and three subproblems should be constructed, the first with the additional constraint that arc (4, 5) be excluded from the solution, the second with the additional constraint that arc (5, 6) be excluded from the solution, and the third with the additional constraint that arc (4, 5) be excluded from the solution.

The polynomially solvable special case under consideration is the set of all ATSP instances for which the assignment procedure gives rise to a cyclic permutation.

Using the branching rule described above, depth-first branch and bound generates the enumeration tree of Fig. 1. The nodes are labeled according to the order in which the problems at the corresponding nodes were evaluated.

Since the cost of patching equals the value of ρ , the performance of data correcting on this example can now be evaluated. If the allowable accuracy parameter α is set to 0, then the enumeration tree constructed by DC will be the one shown in Fig. 1, and it will evaluate eight subproblems. However, if the value of α is set to 1, then enumeration stops after node 4, since the lower bound obtained is 25 which is one less than the solution at hand.

The previous example shows that the data-correcting algorithm can be a very attractive alternative to branch-and-bound algorithms. In the next subsection,

experiences of the performance of the data-correcting algorithm on ATSP instances from the TSPLIB [73] are presented [68].

2.2 Computational Experience with ATSP Instances

TSPLIB has 27 ATSP instances, out of which for the experiments 12 have been chosen. These 12 can be solved to optimality within 5 h using an ordinary branch-and-bound algorithm. Eight of these belong to the “*ftv*” class of instances, while four belong to the “*rbg*” class. DCA-ATSP in C was implemented on an Intel Pentium-based computer running at 666 MHz with 128-MB RAM.

The results of the experiments are presented graphically in Figs. 2 through 5. In computing accuracies (Figs. 2 and 4), the accuracy and deviation of the solution output by the data-correcting algorithm from the optimal (called “achieved accuracy” in the figures) have been plotted as a fraction of the cost of an optimal solution to the instance. It was observed that for each of the 12 instances studied, the achieved accuracy is consistently less than 80 % of the prespecified accuracy.

There was a wide variation in the CPU time required to solve the different instances. For example, *ftv70* required 17,206 s to solve it to optimality, while *rbg323* required just 5 s. Thus, in order to maintain uniformity while demonstrating the variation in execution times with respect to changes in α values, the execution

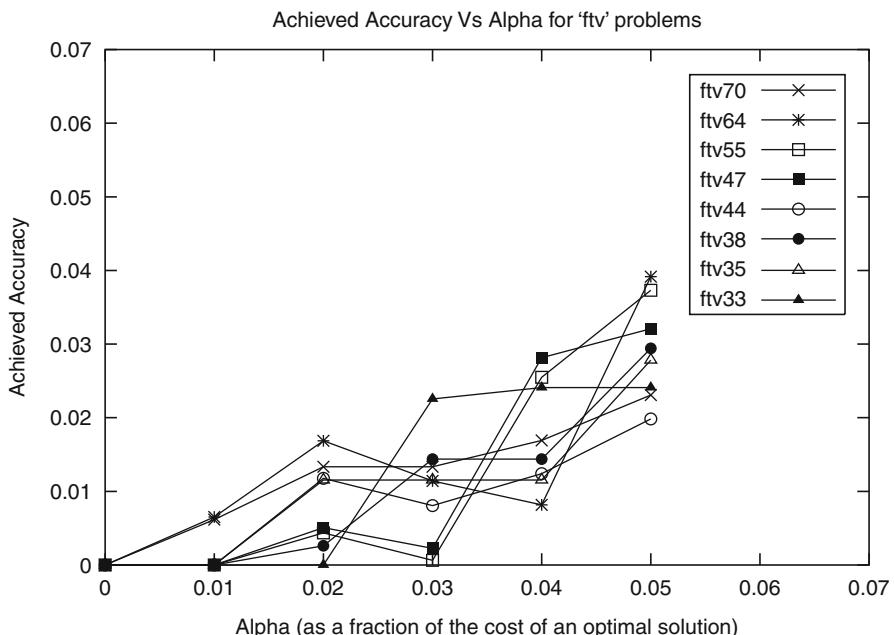


Fig. 2 Accuracy achieved versus α for *ftv* instances

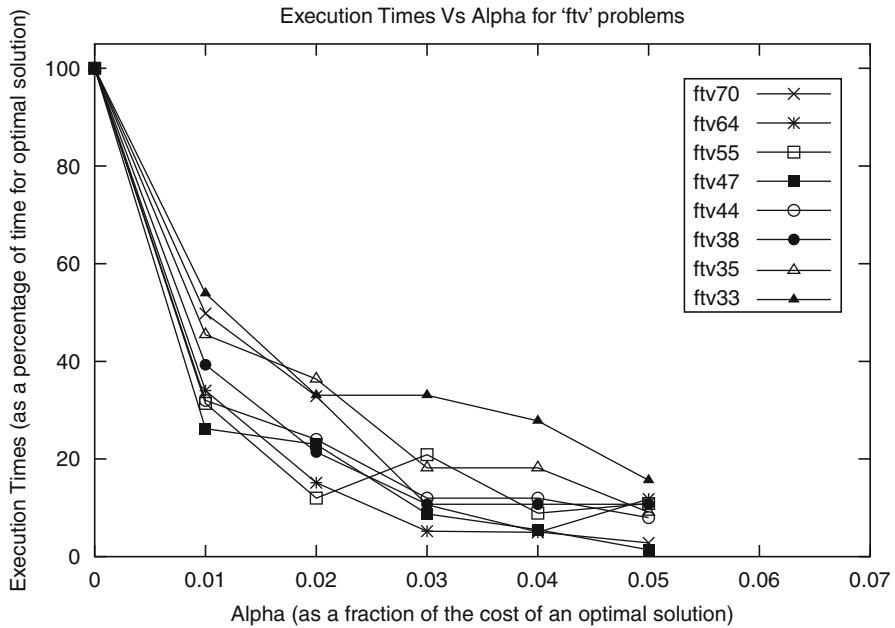


Fig. 3 Variation of execution times versus α for *ftv* instances

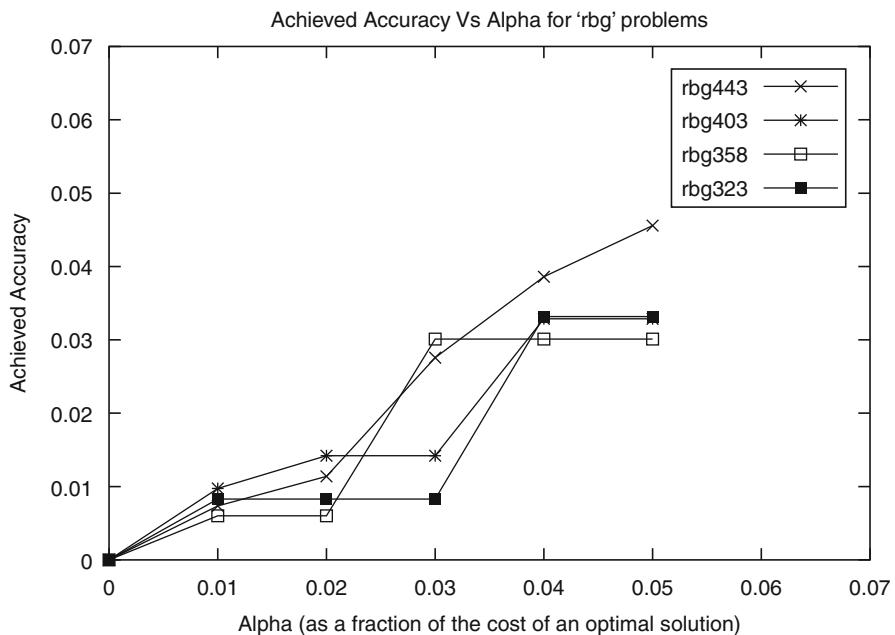


Fig. 4 Accuracy achieved versus α for *rbg* instances

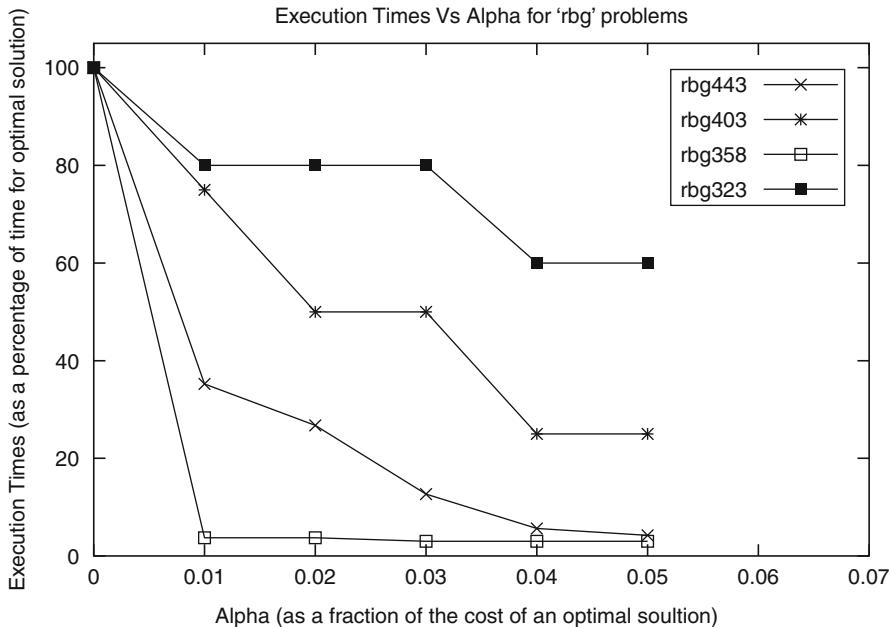


Fig. 5 Variation of execution times versus α for rbg instances

times for each instance for each α value were represented as a percentage of the execution time required to solve that instance to optimality. Notice that for all the f_{tv} instances when α was 5 % of the cost of the optimal solution, the execution time was reduced to 20 % of that required to solve the respective instance to optimality. The reduction in execution times for rbg instance was equally steep, with the exception of $rbg323$ which was in any case an easy instance to solve.

In summary, it is quite clear that data correcting is an effective methodology for solving ATSP instances. There are usually steep reductions in execution times even when the allowed accuracy is very small. This makes the method very useful for solving real-world problems where a near-optimal solution is often acceptable provided the execution times are not too long.

3 The Simple Plant Location Problem

The simple plant location problem (SPLP) takes a set $I = \{1, 2, \dots, m\}$ of sites in which plants can be located; a set $J = \{1, 2, \dots, n\}$ of clients, each having a unit demand; a vector $F = (f_i)$ of fixed costs for setting up plants at sites $i \in I$; and a matrix $C = [c_{ij}]$ of transportation costs from $i \in I$ to $j \in J$ as input. It computes a set $S^*, \emptyset \subset S^* \subseteq I$, at which plants can be located so that the total cost of satisfying all client demands is minimal. The costs involved in meeting the

client demands include the fixed costs of setting up plants and the transportation cost of supplying clients from the plants that are set up. A detailed introduction to this problem has appeared in Cornuejols et al. [21], which also classifies the problem as NP-hard.

In applying data correcting to the SPLP, a pseudo-Boolean formulation of the problem is taken. It is shown how data correcting can be used to preprocess SPLP instances efficiently and then to solve the problem.

3.1 A Pseudo-Boolean Formulation of the SPLP

The pseudo-Boolean approach to solving the SPLP [11, 50] is a penalty-based approach that relies on the fact that any instance of the SPLP has an optimal solution in which each client is supplied by exactly one plant. This implies that in an optimal solution, each client will be served fully by the plant located closest to it. Therefore, it is sufficient to determine the sites where plants are to be located and then use a minimum cost assignment of clients to plants.

An instance of the SPLP can be described by an m -vector $F = (f_i)$ and an $m \times n$ matrix $C = [c_{ij}]$: $m, n \geq 1$. The $m \times (n + 1)$ *augmented matrix* $[F|C]$ will be used as a shorthand for describing an instance of the SPLP. The total cost $f_{[F|C]}(S)$ associated with a subset S of I consists of two components, namely, the fixed costs $\sum_{i \in S} f_i$ and the transportation costs $\sum_{j \in J} \min\{c_{ij} | i \in S\}$; that is,

$$f_{[F|C]}(S) = \sum_{i \in S} f_i + \sum_{j \in J} \min\{c_{ij} | i \in S\},$$

and the SPLP is the problem of finding

$$S^* \in \arg \min\{f_{[F|C]}(S) | \emptyset \subset S \subseteq I\}. \quad (4)$$

In the remainder of this subsection, there will be described the pseudo-Boolean formulation of the SPLP due to Beresnev [11] and Hammer [50].

An $m \times n$ *ordering matrix* $\Pi = [\pi_{ij}]$ is a matrix each of whose columns $\Pi_j = (\pi_{1j}, \dots, \pi_{mj})^T$ defines a permutation of $1, \dots, m$. Given a transportation matrix C , the set of all ordering matrices Π such that $c[\pi_{1j}, j] \leq c[\pi_{2j}, j] \leq \dots \leq c[\pi_{mj}, j]$ for $j = 1, \dots, n$ is denoted by $\text{perm}(C)$.

Defining for each $i = 1, \dots, m$

$$y_i = \begin{cases} 0 & \text{if } i \in S \\ 1 & \text{otherwise,} \end{cases} \quad (5)$$

any solution P can be indicated by a vector $y = (y_1, y_2, \dots, y_m)$. The fixed cost component of the total cost can be written as

$$F_F(y) = \sum_{i=1}^m f_i(1 - y_i). \quad (6)$$

Corresponding to an ordering matrix $\Pi \in perm(C)$, an $m \times n$ difference matrix $\Delta = ||\Delta c(i, j)||$ can be constructed for each $j \in J$ as

$$\Delta c[0, j] = c[\pi_{1j}, j] \quad \text{and}$$

$$\Delta c[l, j] = c[\pi_{(l+1)j}, j] - c[\pi_{lj}, j], \quad l = 1, \dots, m-1. \quad (7)$$

Note that $\Delta c[l, j] \geq 0$, even if the transportation cost matrix C contains negative entries. The transportation costs of supplying any client $j \in J$ from any open plant can be expressed in terms of the $\Delta c[\cdot, j]$ values. It is clear that it is necessary to spend at least $\Delta c[0, j]$ in order to satisfy j 's demand, since this is the cheapest cost of satisfying j . If no plant is located at the site closest to j , that is, $y_{\pi_{1j}} = 1$, the demand is tried to be satisfied from the next closest site. In that case, an additional $\Delta c[1, j]$ is spent. Continuing in this manner, the transportation cost of supplying $j \in J$ is

$$\begin{aligned} \min\{c_{ij} | i \in S\} &= \Delta c[0, j] + \Delta c[1, j] \cdot y_{\pi_{1j}} + \Delta c[2, j] \cdot y_{\pi_{1j}} \cdot y_{\pi_{2j}} \\ &\quad + \cdots + \Delta c[m-1, j] \cdot y_{\pi_{1j}} \cdots y_{\pi_{(m-1)j}} \\ &= \Delta c[0, j] + \sum_{k=1}^{m-1} \Delta c[k, j] \cdot \prod_{r=1}^k y_{\pi_{rj}}, \end{aligned}$$

so that the transportation cost component of the cost of a solution y corresponding to an ordering matrix $\Pi \in perm(C)$ is

$$T_{C,\Pi}(y) = \sum_{j=1}^n \left\{ \Delta c[0, j] + \sum_{k=1}^{m-1} \Delta c[k, j] \cdot \prod_{r=1}^k y_{\pi_{rj}} \right\}. \quad (8)$$

Combining (6) and (8), the total cost of a solution y to the instance $[F|C]$ corresponding to an ordering matrix $\Pi \in perm(C)$ is given by the pseudo-Boolean polynomial

$$\begin{aligned} f_{[F|C],\Pi}(y) &= F_F(y) + T_{C,\Pi}(y) \\ &= \sum_{i=1}^m f_i(1 - y_i) \\ &\quad + \sum_{j=1}^n \left\{ \Delta c[0, j] + \sum_{k=1}^{m-1} \Delta c[k, j] \cdot \prod_{r=1}^k y_{\pi_{rj}} \right\}. \end{aligned} \quad (9)$$

It can be shown (see [1]) that the total cost function $f_{[F|C],\Pi}(\cdot)$ is identical for all $\Pi \in \text{perm}(C)$. This pseudo-Boolean polynomial is called the *Hammer-Beresnev function* $H_{[F|C]}(y)$ corresponding to the SPLP instance $[F|C]$ and $\Pi \in \text{perm}(C)$. In other words

$$H_{[F|C]}(y) = f_{[F|C],\Pi}(y) \text{ where } \Pi \in \text{perm}(C). \quad (10)$$

The SPLP (4) can be formulated in terms of Hammer-Beresnev functions as

$$y^* \in \arg \min \{H_{[F|C]}(y) | y \in \{0, 1\}^m, y \neq \bar{1}\}. \quad (11)$$

As an example, consider the SPLP instance:

$$[F|C] = \left[\begin{array}{c|ccccc} 9 & 7 & 12 & 22 & 13 \\ 4 & 8 & 9 & 18 & 17 \\ 3 & 16 & 17 & 10 & 27 \\ 6 & 9 & 13 & 10 & 11 \end{array} \right]. \quad (12)$$

Two possible ordering matrices corresponding to C are

$$\Pi_1 = \left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 1 \\ 4 & 4 & 2 & 2 \\ 3 & 3 & 1 & 3 \end{array} \right] \text{ and } \Pi_2 = \left[\begin{array}{cccc} 1 & 2 & 4 & 4 \\ 2 & 1 & 3 & 1 \\ 4 & 4 & 2 & 2 \\ 3 & 3 & 1 & 3 \end{array} \right], \quad (13)$$

and the difference matrix is

$$\Delta = \left[\begin{array}{cccc} 7 & 3 & 10 & 11 \\ 1 & 3 & 0 & 2 \\ 1 & 1 & 8 & 4 \\ 7 & 4 & 4 & 10 \end{array} \right]. \quad (14)$$

The Hammer-Beresnev function is $H_{[F|C]}(y) = \{9(1 - y_1) + 4(1 - y_2) + 3(1 - y_3) + 6(1 - y_4)\} + \{7 + 1y_1 + 1y_1y_2 + 7y_1y_2y_4\} + \{9 + 3y_2 + 1y_1y_2 + 4y_1y_2y_4\} + \{10 + 0y_3 + 8y_3y_4 + 4y_2y_3y_4\} + \{11 + 2y_4 + 4y_1y_4 + 10y_1y_2y_4\} = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 21y_1y_2y_4 + 4y_2y_3y_4$.

3.2 Preprocessing SPLP Instances

The first preprocessing rules for the SPLP involving both fixed costs and transportation costs appeared in Khumawala [54]. In terms of Hammer-Beresnev

functions, these rules are stated in the following theorem. It is assumed (without loss of generality) that I cannot be partitioned into sets I_1 and I_2 , and J into sets J_1 and J_2 , such that the transportation costs from sites in I_1 to clients in J_2 and from sites in I_2 to clients in J_1 are not finite. It is assumed too that the site indices are arranged in nonincreasing order of $f_i + \sum_{j \in J} c_{ij}$ values.

Theorem 3 *Let $H_{[F|C]}(y)$ be the Hammer-Beresnev function corresponding to the SPLP instance $[F|C]$ in which similar monomials have been aggregated. For each site index k , let a_k be the coefficient of the linear monomial corresponding to y_k and let t_k be the sum of the coefficients of all nonlinear monomials containing y_k . Then the following assertion holds:*

- RO:** *If $a_k \geq 0$, then there is an optimal solution y^* in which $y_k^* = 0$, else*
RC: *If $a_k + t_k \leq 0$, then there is an optimal solution y^* in which $y_k^* = 1$, provided that $y_i^* \neq 1$ for some $i \neq k$.*

Notice that RO and RC primarily try to either open or close sites. If it succeeds, it also changes the Hammer-Beresnev function for the instance, reducing the number of nonlinear monomials therein. In the remaining portion of this subsection, there will be described a completely new reduction procedure (RP), whose primary aim is to reduce the coefficients of terms in the Hammer-Beresnev function and, if it can be reduced to zero, to eliminate the term from the Hammer-Beresnev function. This procedure is based on fathoming rules of branch-and-bound algorithms and data-correcting principles.

Let us assume that there is an upper bound (UB) on the cost of an optimal solution for the given SPLP instance. This can be obtained by running a heuristic on the problem data. Now consider a nonlinear monomial $s \cdot \prod_{r=1}^k y_{\pi_{rj}}$ in the Hammer-Beresnev function. This term will contribute to the cost of a solution, only if plants are *not* located in any of the sites $\pi_{1j}, \dots, \pi_{kj}$. Let LB be a lower bound on the cost of solutions in which facilities are not located in sites $\pi_{1j}, \dots, \pi_{kj}$. If $LB \leq UB$, then any judgment about this term cannot be made. On the other hand, if $LB > UB$, then it is known that there cannot be an optimal solution with $y_{\pi_{1j}} = \dots = y_{\pi_{kj}} = 1$. In this case, if the coefficient s is reduced by $LB - UB - \varepsilon$ ($\varepsilon > 0$, small), then the new Hammer-Beresnev function and the original one have identical sets of optimal solutions. If after the reduction s is nonpositive, then the term can be removed from the Hammer-Beresnev function. Such changes in the Hammer-Beresnev function alter the values of t_k and can possibly allow us to use Khumawala's rules to close certain sites. Once some sites are closed, some of the linear monomials in the Hammer-Beresnev function change into constant terms, and some of the quadratic terms change into linear ones. These changes cause changes in both the a_k and the t_k values and can make further application of Khumawala's rules possible, thus preprocessing some other sites and making further changes in the Hammer-Beresnev function. A pseudocode of the reduction procedure (RP) is provided below.

Procedure RP($H_{[F|C]}(y)$)

Output: A preprocessed instance of the SPLP, i.e. an equivalent instance of reduced size, and decisions to either locate or not locate plants in some of the sites.

Code:

```

1. begin
2.   repeat
3.     compute an upper bound  $UB$  for the instance;
4.     for each nonlinear monomial  $s \cdot \prod_{r=1}^k y_{\pi_{rj}}$  in  $H_{[F|C]}(y)$  do
5.       begin
6.         compute lower bound  $LB$  on the cost of solutions in
7.         which plants are not located in sites  $\pi_{1j}, \dots, \pi_{kj}$ ;
8.         if  $LB > UB$  then
9.           reduce the coefficient of the term by
10.           $\max\{s, LB - UB - \varepsilon\}$ ;
11.         apply Khumawala's rules until no further preprocessing is possible;
12.         recompute the Hammer-Beresnev function  $H_{[F|C]}(y)$ ;
13.       until no further preprocessing of sites was achieved in the current iteration;
14. end;
```

Let us consider the application of all preprocessing rules to the example with the Hammer-Beresnev function $H_{[F|C]}(y) = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 21y_1y_2y_4 + 4y_2y_3y_4$. The values of a_k , t_k , and $a_k + t_k$ are as follows:

$k :$	1	2	3	4
$a_k :$	-8	-1	-3	-4
$t_k :$	27	27	12	37
$a_k + t_k :$	19	26	9	33

It is clear that neither RO nor RC is applicable here, since the coefficient of the term $21y_1y_2y_4$ is too large. Therefore, this coefficient is tried to be reduced by applying the RP.

An upper bound of $UB = 51$ to the original problem can be obtained by setting $y_1 = y_4 = 1$ and $y_2 = y_3 = 0$. A lower bound to the problem under the restriction $y_1 = y_2 = y_4 = 1$ is 73, since $H_{[F|C]}(1, 1, 0, 1) = 73$. Using RP therefore, the coefficient of $21y_1y_2y_4$ can be reduced by $73 - 51 - \varepsilon = 20$, so that the new Hammer-Beresnev function with the same set of optimal solutions as the original function becomes $H'(y) = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 1y_1y_2y_4 + 4y_2y_3y_4$. The updated values of a_k , t_k , and $a_k + t_k$ are presented below.

$k :$	1	2	3	4
$a_k :$	-8	-1	-3	-4
$t_k :$	7	7	12	17
$a_k + t_k :$	-1	6	9	13

RC can immediately be applied in this situation to set $y_3 = 1$. Updating $H'(y)$, RO and set $y_2 = y_4 = 0$ can be applied. This makes it possible to apply RC again to set $y_3 = 1$, thus giving an optimal solution (i.e., $(1, 0, 1, 0)$) to the instance, with a cost of 48.

3.3 The Data-Correcting Algorithm

The basic idea behind the data-correcting algorithm is to modify the Hammer-Beresnev function in a way such that the RO and RC rules can be applied to the modified instance. While modifying the instance, care is taken so that an optimal solution to the modified instance is not too suboptimal for the original instance. The following suitably modified version of [Theorem 1](#) for this problem is used:

Lemma 1 *Consider two Hammer-Beresnev functions $H_1(y)$ and $H_2(y)$. Let y_1^* and y_2^* be the optimal solutions to $H_1(y)$ and $H_2(y)$ respectively. Then,*

$$|H_1(y_1^*) - H_1(y_2^*)| \leq \sum_{i=0}^{m-1} \sum_{j=1}^n |\Delta c^1[i, j] - \Delta c^2[i, j]|.$$

Consider an SPLP instance with accuracy parameter α in which RO and RC cannot be applied. Clearly, in the Hammer-Beresnev function for this instance, $a_k < 0$ and $a_k + t_k > 0$ for all k . Let $k_0 = \arg \min\{|a_k|, a_k + t_k\}$. Also, let $|a_{k_0}| \leq a_{k_0} + t_{k_0}$. In this case, if the Hammer-Beresnev function of the instance is changed (corrected) by increasing the coefficient of y_{k_0} to zero, then RO can be applied to the corrected instance, and preprocessing can continue. However, this is allowed only if $\min\{|a_{k_0}|, a_{k_0} + t_{k_0}\} \leq \alpha$. In such a situation, if $|a_{k_0}| > a_{k_0} + t_{k_0}$, then the instance can be corrected by decreasing the coefficient of y_{k_0} by $a_{k_0} + t_{k_0}$; then, RC can be applied to the corrected instance, and preprocessing can continue. Notice that while correcting the instance, suboptimality to the extent of $|a_{k_0}|$ in the first case and $a_{k_0} + t_{k_0}$ in the second case is allowed for. Thus, the accuracy parameter for the corrected instance is reduced appropriately.

It may happen, however, that $\min\{|a_{k_0}|, a_{k_0} + t_{k_0}\} > \alpha$. In that case, correction is not possible at this stage, and the problem has to be broken down into subproblems. This is done by a branching operation. Goldengorin et al. [46] suggest that the algorithm branches on an index from $\arg \max\{t_k\}$.

The logic behind this rule is the following. A plant would have been located in this site in an optimal solution if the coefficient of linear monomial involving y_k in the Hammer-Beresnev function would have been increased by $-a_k$. It could have been predicted that a plant would not be located there if the same coefficient would have been decreased by $t_k + a_k$. Therefore, the average of $-a_k$ and $a_k + t_k$ could be used as a measure of the chance that the researcher will *not* be able to predict the fate of site k in any subproblem of the current subproblem [44]. If it is necessary to reduce the size of the branch-and-bound tree by assigning values to such variables, then it is possible to think of a branching function that branches on the index k_0 with the largest average value, that is, the largest value of $-a_k + (a_k + t_k)$, that is, the largest value of t_k .

On the basis of the discussion above, the pseudocode for a data-correcting algorithm for SPLP is given below. It works by maintaining three sets, Ω containing the sites where facilities are to be located, Λ containing the sites where facilities are not to be located, and Ψ containing the rest of the sites. RO and RC rules are assumed to be able to manipulate these three sets.

Algorithm DCA-SPLP($H_{[F|C]}(y)$, α)

Output: A solution y^α to the SPLP such that $H_{[F|C]}(y^\alpha) \leq H_{[F|C]}(y^*) + \alpha$.

Code:

```

1. begin
2.     apply RP to initialize  $\Omega$ ,  $\Lambda$  and  $\Psi$ ;
3.      $y^\alpha := \text{Int-DCA-SPLP}(\Omega, \Lambda, \Psi, H_{[F|C]}(\cdot), \alpha);$ 
4.     return  $y^\alpha$ ;
5. end.

```

Function Int-DCA-SPLP(Ω , Λ , Ψ , $H_{[F|C]}(\cdot)$, α)

```

1. begin
2.     while RO or RC is applicable
3.         update  $\Omega$ ,  $\Lambda$ ,  $\Psi$ , and  $H_{[F|C]}(\cdot)$  by applying RO and RC;
4.          $k^* \in \arg\{k \in \Psi \mid \min\{|a_k|, a_k + t_k\} =$ 
...            $\min\{\min\{|a_s|, a_s + t_s\} \mid s \in \Psi\}\};$ 
5.         if  $|a_{k^*}| \leq a_{k^*} + t_{k^*}$  then begin
6.             if  $|a_{k^*}| \leq \alpha$  then                                     (* Correct *)
7.                 return Int-DCA-SPLP( $\Omega + k^*$ ,  $\Lambda$ ,  $\Psi - k^*$ ,  $H_{[F|C]}(\cdot)$ ,  $\alpha - |a_{k^*}|$ );
8.             else begin                                              (* Branch *)
9.                  $k^b := \arg \max\{t_k \mid k \in \Psi\};$ 
10.                 $y^1 := \text{Int-DCA-SPLP}(\Omega + k^b, \Lambda, \Psi - k^b, H_{[F|C]}(\cdot), \alpha);$ 
11.                 $y^2 := \text{Int-DCA-SPLP}(\Omega, \Lambda + k^b, \Psi - k^b, H_{[F|C]}(\cdot), \alpha);$ 
12.                return  $\arg \max\{H_{[F|C]}(y^1), H_{[F|C]}(y^2)\};$ 
13.            end;
14.        else begin
15.            if  $a_{k^*} + t_{k^*} \leq \alpha$  then                               (* Correct *)
16.                return Int-DCA-SPLP( $\Omega, \Lambda + k^*, \Psi - k^*$ ,  $H_{[F|C]}(\cdot)$ ,  $\alpha - a_{k^*} - t_{k^*}$ );
17.            else begin                                              (* Branch *)
18.                 $k^b := \arg \max\{t_k \mid k \in \Psi\};$ 
19.                 $y^1 := \text{Int-DCA-SPLP}(\Omega + k^b, \Lambda, \Psi - k^b, H_{[F|C]}(\cdot), \alpha);$ 
20.                 $y^2 := \text{Int-DCA-SPLP}(\Omega, \Lambda + k^b, \Psi - k^b, H_{[F|C]}(\cdot), \alpha);$ 
21.                return  $\arg \max\{H_{[F|C]}(y^1), H_{[F|C]}(y^2)\};$ 
22.            end;
23.        end;
24.    end;

```

3.4 Computational Experience with SPLP Instances

The computational experience with the DCA-SPLP on several benchmark instances of the SPLP is reported in the remainder of this section. The performance of the algorithm is compared with that of the algorithms described in the papers that suggested these instances. One of two bounds was used in the

implementations of RP and DCA-SPLP: a combinatorial Khachaturov-Minoux bound [52, 61] and a much stronger Erlenkotter bound based on an LP dual-ascent algorithm [24]. There was implemented the DCA-SPLP in PASCAL, which was compiled using Prospero Pascal and which was run on a 733-MHz Pentium III machine. The computation times reported are in seconds on the machine used.

3.4.1 Testing the Effectiveness of the Reduction Procedure RP

Given an instance of the SPLP, the reduction procedure RP reduces it to a smaller core instance by making decisions to locate or not locate plants in several sites. The effectiveness of the RP can thus be measured either by computing the number of free locations in the core instance or by computing the number of nonzero nonlinear monomials present in the Hammer-Beresnev function of the core instance. Tables 1 and 2 show how the various methods of reduction perform on the benchmark SPLP instances in the OR-Library [7]. In the tables, procedure (a) refers to the use of the “delta” and “omega” rules from Khumawala [54], procedure (b) to the RP with the Khachaturov-Minoux combinatorial bound to obtain a lower bound, and procedure (c) to the RP with the Erlenkotter bound to obtain a lower bound.

The existing preprocessing rules due to Khumawala [54] and Goldengorin et al. [1] (i.e., procedure (a), which was used in the SPLP example in Goldengorin et al. [44]) cannot solve any of the OR-Library instances to optimality. However, the variants of the new reduction procedure (i.e., procedures (b) and (c)) solve a large number of these instances to optimality. Procedure (c), based on the Erlenkotter bound, is marginally better than procedure (b) in terms of the number of free locations (Table 1), but substantially better in terms of the number of nonzero nonlinear terms in the Hammer-Beresnev function (Table 2).

Tables 1 and 2 also demonstrate the superiority of the new preprocessing rule over the “delta” and “omega” rules. Consider, for example, the problem cap132. The

Table 1 Number of free locations after preprocessing SPLP instances in the OR-Library

Problem	m	n	Procedure		
			a	b	c
cap71	16	50	4	0	0
cap72	16	50	6	0	0
cap73	16	50	6	3	3
cap74	16	50	2	0	0
cap101	25	50	9	0	0
cap102	25	50	13	3	0
cap103	25	50	14	0	0
cap104	25	50	12	0	0
cap131	50	50	34	32	8
cap132	50	50	27	25	5
cap133	50	50	25	19	10
cap134	50	50	19	0	0

Table 2 Number of nonzero nonlinear monomials in the Beresnev function after preprocessing SPLP instances in the OR-Library

Problem	Nonzero terms before preprocessing	Procedure		
		a	b	c
cap71	699	6	0	0
cap72	699	12	0	0
cap73	699	13	2	2
cap74	699	1	0	0
cap101	1,147	24	0	0
cap102	1,147	33	2	0
cap103	1,147	38	0	0
cap104	1,147	29	0	0
cap131	2,389	163	135	8
cap132	2,389	112	92	3
cap133	2,389	101	60	11
cap134	2,389	62	0	0

“delta” and “omega” rules reduce the problem size from $m = 50$ and 2,389 nonzero nonlinear variables to $m = 27$ and 112 non-zero nonlinear variables. However, the new preprocessing rule reduces the same problem to one having $m = 5$ and 3 nonzero nonlinear variables!

3.4.2 Bilde- and Krarup-Type Instances

These are the earliest benchmark problems that are considered here. The exact instance data is not available, but the process of generating the problem instances is described in Bilde and Krarup [13]. There are 22 different classes of instances, and in this subsection, the nomenclature used in Bilde and Krarup [13] is applied. In the performed experiments, 10 instances for each of the types of problems were generated, and the mean values of the received solutions were used to evaluate the performance of the discussed algorithm with the one used in Bilde and Krarup [13]. In the implementation, the Khachaturov-Minoux combinatorial bound was used in the reduction procedure RP as well as in the DCA-SPLP.

The reduction procedure was not useful for these instances, but the DCA-SPLP could solve all the instances in reasonable time. The results of the experiments are presented in Table 3. The performance of the algorithm implemented in Bilde and Krarup [13] was measured in terms of the number of branching operations performed by the algorithm and its execution time in CPU seconds on an IBM 7094 machine. The number of branching operations is estimated by the suggested algorithm as the logarithm (to the base 2) of the number of subproblems it generated. From the table, it can be seen that the DCA-SPLP reduces the number of subproblems generated by the algorithm in Bilde and Krarup [13] by a factor of 1,000. This is especially interesting because Bilde and Krarup use a bound (discovered in 1967, see [12]) identical to the Erlenkotter bound in their algorithm (see [55]), and in the proposed algorithm, the Khachaturov-Minoux combinatorial bound is used. The CPU time required by the DCA-SPLP to solve these problems was too low to warrant the use of any $\alpha > 0$.

Table 3 Results from Bilde- and Krarup-type instances

Problem type	DCA		Bilde and Krarup	
	Branching	CPU time	Branching	CPU time [†]
B	11.72	0.67	43.3	4.33
C	17.17	14.81	*	>250
D1	13.80	0.65	216	11
D2	12.13	0.38	218	24
D3	10.87	0.19	169	19
D4	10.25	0.15	141	17
D5	9.24	0.07	106	14
D6	8.99	0.09	101	15
D7	8.79	0.09	83	13
D8	8.60	0.09	55	11
D9	8.15	0.07	47	11
D10	7.29	0.03	43	11
E1	18.66	35.28	1,271	202
E2	16.14	8.64	1,112	172
E3	14.59	3.81	384	82
E4	13.65	2.74	258	65
E5	12.73	2.01	193	53
E6	11.82	0.90	136	43
E7	10.82	0.53	131	42
E8	10.79	0.68	143	48
E9	10.62	0.76	117	44
E10	10.36	0.69	79	37

[†]IBM7094 seconds

*Could not be solved in 250 s

3.4.3 Galvão and Raggi-Type Instances

Galvão and Raggi [25] developed a general 0–1 formulation of the SPLP and presented a three-stage method to solve it. The benchmark instances suggested in this work are unique, in that the fixed costs are assumed to come from a normal distribution rather than the more commonly used uniform distribution. The reader is referred to Galvão and Raggi [25] for a detailed description of the problem data.

As with the data in Bilde and Krarup [13], the exact data for the instances are not known. So, 10 instances for each problem size were generated, and the mean values of the solutions were used for comparison purposes. In the DCA-SPLP implementation, the Khachaturov-Minoux combinatorial bound was used in the reduction procedure RP and in the DCA-SPLP. The comparative results are given in Table 4. Since the computers used are different, it is not possible to comment on the relative performance of the solution procedures. However, since the average number of subproblems generated by the DCA-SPLP is always less than 10 for each of these instances, it can be concluded that these problems are easy for the discussed algorithm. In fact, they are too easy for the DCA-SPLP to warrant $\alpha > 0$.

Notice that the average number of opened plants in the optimal solutions to the instances generated is quite close to the number of opened plants in the optimal solutions reported in Galvão and Raggi [25]. Also, notice that the reduction procedure was quite effective – it solved 35 of the 80 instances generated.

3.4.4 Instances from the OR-Library

The OR-Library [7] has a set of instances of the SPLP. These instances were solved in Beasley [6] using an algorithm based on the Lagrangian heuristic for the SPLP. Here too, the Khachaturov-Minoux combinatorial bound was used in the reduction procedure RP as well as in the DCA-SPLP. The problems was solved to optimality using the DCA. The results of the computations are provided in [Table 5](#). The

Table 4 Results from Galvão and Raggi-type instances

Problem size ($m = n$)	DCA				Galvão and Raggi	
	# solved by pre processing	# of sub problems [†]	CPU time [†]	# of open plants [†]	CPU time*	# of open plants
10	6	2.3	<0.001	4.7	<1	3
20	5	2.4	<0.001	9.0	<1	8
30	7	1.8	0.002	13.6	1	11
50	7	2.6	0.002	20.3	2	20
70	2	3.8	0.004	28.8	6	31
100	3	3.5	0.011	41.1	6	44
150	1	7.8	0.010	64.4	25	74
200	4	2.9	0.158	81.8	63	84

[†]Average over 10 instances

*IBM 4331 s

Table 5 Results from OR-Library instances

Problem name	m	n	DCA				
			m after pre processing	# of sub problems	CPU time	CPU time [6] [†]	# of open plants
cap71	16	50	*	0	<0.01	0.11	11
cap72	16	50	*	0	<0.01	0.08	9
cap73	16	50	*	0	<0.01	0.11	5
cap74	16	50	*	0	<0.01	0.05	4
cap101	25	50	9	6	<0.01	0.18	15
cap102	25	50	13	16	<0.01	0.16	11
cap103	25	50	14	16	<0.01	0.14	8
cap104	25	50	12	7	0.01	0.11	4
cap131	50	50	34	196	0.01	0.31	15
cap132	50	50	27	183	0.02	0.28	11
cap133	50	50	25	71	<0.01	0.29	8
cap134	50	50	19	25	<0.01	0.15	4

^{*}Instance solved by preprocessing only

[†]Cray-X-MP/28 seconds

execution times suggest that the DCA-SPLP is faster than the Lagrangian heuristic described in Beasley [6]. The reduction procedure was also quite effective for these instances, solving 4 of the 16 instances to optimality and reducing the number of free sites appreciably in the other instances. Once again, the use of $\alpha > 0$ cannot be justified, considering the execution times of the DCA.

3.4.5 Körkel-Type Instances with 65 Sites

Körkel [55] described several relatively large Euclidean SPLP instances ($m = n = 100$ and $m = n = 400$) and used a branch-and-bound algorithm to solve these problems. The bound used in that work is an improvement on a bound based on the dual of the linear programming relaxation of the SPLP due to Erlenkotter [24] and is extremely effective. In this subsection, instances that have the same cost structure as the ones in Körkel [55] but for which $m = n = 65$ were used. Instances of this size were not dealt with in Körkel [55]. The Khachaturov-Minoux combinatorial bound was implemented both for the reduction procedure RP and the DCA-SPLP.

In Körkel [55], 120 instances of each problem size are described. These can be divided into 28 sets (the first 18 sets contain 5 instances each, and the rest contain 3 instances each). All the 120 generated instances were solved, and it was found out that the instances in sets 1, 2, 3, 4, 10, 11, and 12 are more difficult to solve than others. Therefore, these instances were used in the experiments in this section. The transportation cost matrix for a Körkel instance of size $n \times n$ is generated by distributing n points in random within a rectangular area of size $700 \times 1,300$ and calculating the Euclidean distances between them. The fixed costs are computed as in [Table 6](#).

The value of the results that is presented for each set is the average of the values obtained for all the instances in that set. Interestingly, the preprocessing rules were found to be totally ineffective for all of these problems. Since the fixed costs are identical for all the sites, the sites are distributed randomly over a region, and the variable cost matrix is symmetric; no site presents a distinct advantage over any other. This prevents our reduction procedure to open or close any site. [Table 7](#) shows the variation in the costs of the solution output by the DCA-SPLP with changes in α , and [Table 8](#) shows the corresponding decrease in execution times.

The effect of varying the acceptable accuracy α on the cost of the solution output by the DCA-SPLP is also presented graphically in [Fig. 6](#). The *achieved accuracy* β is defined as

Table 6 Description of the fixed costs for instances in Körkel [55]

Problem set	# of instances	Fixed cost for i -th instance
Set 1	5	Identical, set at $141 + 6.6i$
Set 2	5	Identical, set at $174 + 6.6i$
Set 3	5	Identical, set at $207 + 6.6i$
Set 4	5	Identical, set at $174 + 66i$
Set 10	5	Identical, set at $7,170 + 660i$
Set 11	5	Identical, set at $7,120.5 + 333.3i$
Set 12	5	Identical, set at $8,787 + 333.3i$

Table 7 Costs of solution output by the DCA-SPLP on Körkel-type instances with 65 sites

Problem set	Optimal	Acceptable accuracy*				
		1 %	2 %	3 %	5 %	10 %
Set 1	6,370.0	6,404.8	6,450.6	6,480.6	6,569.2	6,781.0
Set 2	6,920.6	6,952.2	6,971.4	7,028.4	7,123.8	7,320.2
Set 3	7,707.4	7,738.0	7,770.2	7,797.6	7,854.6	8,053.8
Set 4	9,601.2	9,642.4	9,680.2	9,698.4	9,786.6	9,932.0
Set 10	146,691.2	146,896.6	146,909.6	147,543.6	148,062.0	151,542.2
Set 11	168,598.4	168,858.2	169,655.0	170,341.6	170,597.0	173,913.8
Set 12	186,386.3	186,729.7	187,112.0	188,002.7	188,854.2	192,528.7

*As a percentage of the optimal cost

Table 8 Execution times for the DCA-SPLP on Körkel-type instances with 65 sites

Problem set	Optimal	Acceptable accuracy*				
		1 %	2 %	3 %	5 %	10 %
Set 1	119.078	90.948	70.758	55.494	43.200	20.426
Set 2	290.388	225.108	172.422	145.828	96.240	36.966
Set 3	458.370	339.420	259.022	203.036	150.216	50.378
Set 4	158.386	129.694	109.754	89.666	65.548	30.058
Set 10	428.598	370.120	319.804	283.832	230.078	142.090
Set 11	542.530	476.350	418.628	408.594	290.338	160.744
Set 12	479.092	416.472	370.832	326.572	261.835	149.038

*As a percentage of the optimal cost

$$\beta = \frac{\text{cost of the DCA-SPLP output} - \text{cost of an optimal solution}}{\text{cost of optimal solution}},$$

and the *relative time* τ as

$$\tau = \frac{\text{execution time for the DCA-SPLP for acceptable accuracy } \alpha}{\text{execution time for the DCA-SPLP to compute an optimal solution}}.$$

Note that the achieved accuracy β varies almost linearly with α , with a slope close to 0.5. Also, note that the relative time τ of the DCA-SPLP reduces with increasing α . The reduction is slightly better than linear, with an average slope of -8 .

3.4.6 Körkel-Type Instances with 100 Sites

The benchmark instances in Körkel [55] with $m = n = 100$ were solved to optimality, and it was observed that the instances in sets 10, 11, and 12 required relatively longer execution times. So, further computations were restricted to instances in those sets. The fixed and transportation costs for these problems are computed in the procedure described in Sect. 3.4.5. Tables 9 and 10 show the results obtained by running the DCA-SPLP on these problem instances. In the

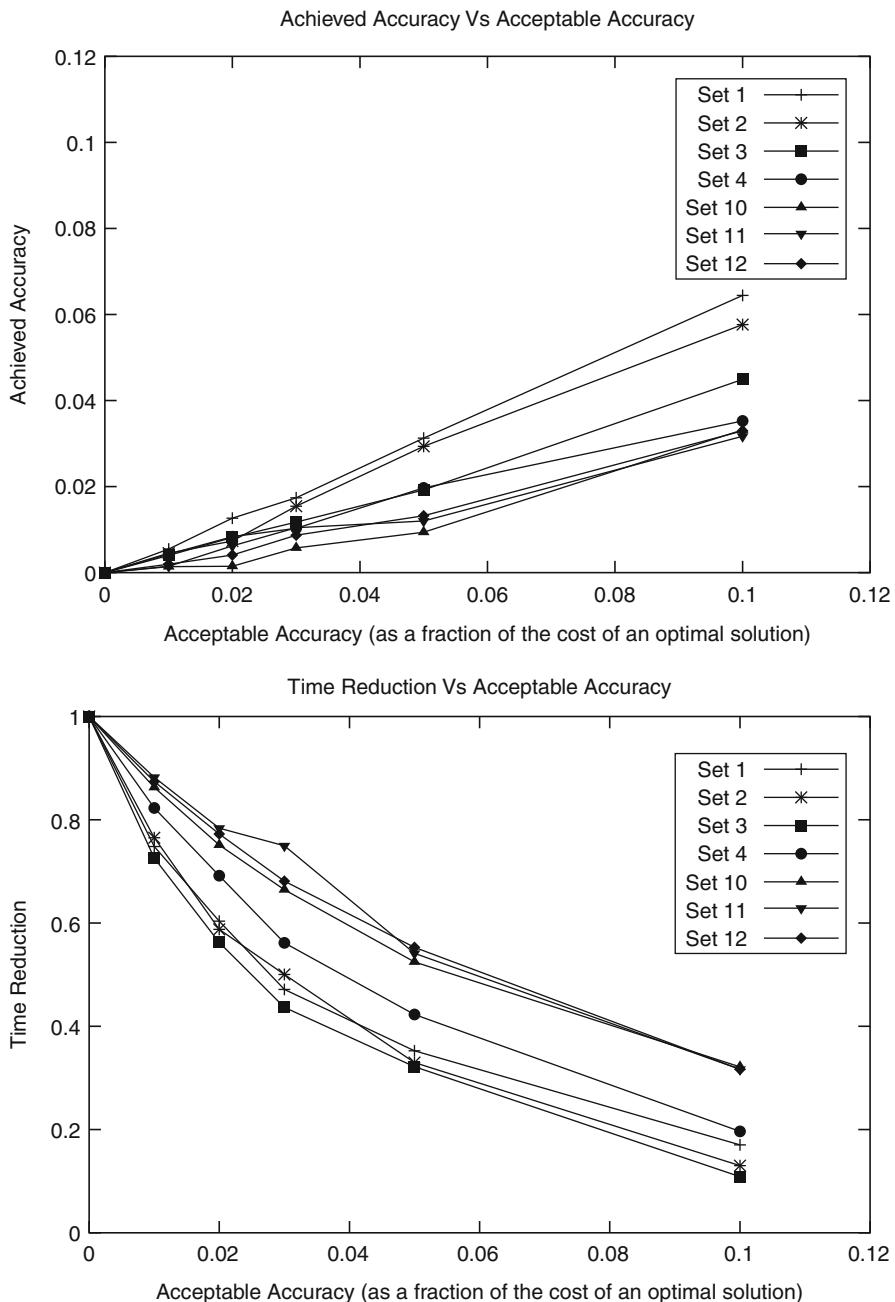


Fig. 6 Performance of the DCA-SPLP for Körkel-type instances with 65 sites

Table 9 Costs of solution output by the DCA-SPLP on Körkel-type instances with 100 sites

Problem set	Optimal	Acceptable accuracy*					
		1 %	2 %	3 %	5 %	10 %	
Set 10	190,782.0	191,550.8	192,755.4	192,080.6	195,983.2	203,934.2	
Set 11	219,583.4	220,438.8	222,393.6	221,947.2	228,467.2	235,963.4	
Set 12	240,402.4	241,609.6	243,336.8	244,209.4	247,417.6	259,168.6	

*As a percentage of the optimal cost

Table 10 Execution times for the DCA-SPLP on Körkel-type instances with 100 sites

Problem set	Optimal	Acceptable accuracy*					
		1 %	2 %	3 %	5 %	10 %	
Set 10	133.746	91.774	65.99	65.908	44.2	32.074	
Set 11	81.564	55.356	39.554	38.348	33.628	17.598	
Set 12	111.272	85.858	65.608	55.928	61.758	33.014	

*As a percentage of the optimal cost

DCA-SPLP implementation for solving these instances, the Erlenkotter bound in both the reduction procedure RP and the DCA-SPLP was used.

Figure 7 illustrates the effect of varying the acceptable accuracy α on the cost of the solution output by the DCA-SPLP for the instances mentioned above. The nature of the graphs is similar to those in Fig. 6. However, in several of the instances, it was observed that β is reduced when α is increased, and in some other instances, τ was increased when α was increased.

4 The p -Median Problem

The objective of the p -median problem (PMP) is to locate p facilities (medians) such that the sum of the distances from each demand point to its nearest facility is minimized. A justification for this research is that there is evidence that the PMP can provide better cluster recovery than alternative clustering procedures (the complete and average linkage, minimum variance, K-means, and p -cliques, see [16]). The PMP is formulated as a minimization problem of a pseudo-Boolean polynomial (pBp) and presents three types of data correcting for the given PMP instance, represented by an $m \times n$ input matrix and the fixed number $p : 1 \leq p \leq m$. The first type of data correcting is the truncation of the corresponding pBp and the columns of an input matrix, which leads to the preprocessing rule fixing at least one site closed. The second type of data correcting is based on the construction of pBp monomials and aggregation of similar monomials in pBp which leads to formation of some terms with zero coefficients and sometimes to preprocessing (reducing) the number of columns (clients) in the input matrix. The third data correction is similar to the data correcting applied in the SPLP and leads to the reduction of monomials at each of which an optimal solution to the PMP cannot be attained. All three types of data correcting are incorporated into the linearized pBp and corresponding mixed

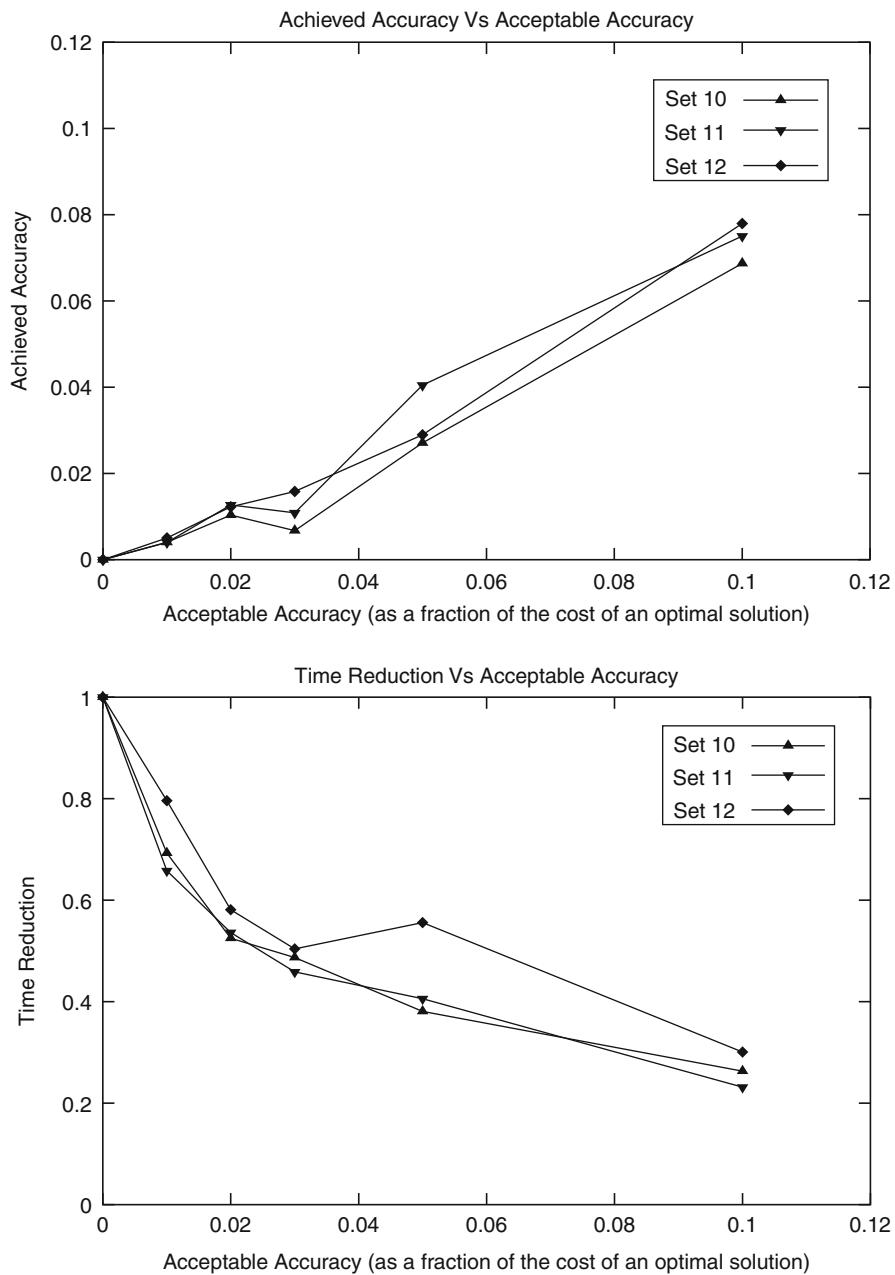


Fig. 7 Performance of the DCA-SPLP for Körkel-type instances with 100 sites

integer linear programming (MILP) PMP model. The computational experiment has showed that the new model is able to solve to optimality some of PMP benchmark instances which were intractable by general-purpose ILP software and even by the state-of-the-art algorithms (see [4, 9, 16, 67, 72]). Also, CPU times with the proposed model outperform CPU times for the best available PMP models (see [18, 19, 23, 70]).

4.1 Introduction

The p-median problem (PMP) is a well-known NP-hard problem which was originally defined by Hakimi [48, 49] and involves the location of p facilities on a network in such a manner that the total weighted distance of serving all demands is minimized. It has been widely studied in literature and applied in cluster analysis, quantitative psychology, marketing, telecommunications industry [16], sales force territory design [63], political districting [8] and references within, optimal diversity management [15], cell formation in group technology [75], vehicle routing [56], and topological design of computer communication networks [66].

The basic PMP model that has remained almost unchanged during the recent 30 years is the so-called ReVelle and Swain [69] integer linear programming formulation [19]. Note that in this formulation there are n^2 Boolean decision variables and, hence, this is a Boolean linear programming formulation. The PMP has since been the subject of considerable research involving the development of some different types of adjusted model formats (Rosing et al. [71], Cornuejols et al. [20], Church [18] and recently by Church [19], AlBdaiwi et al. [1], and Elloumi [23]) as well as the development of advanced solution approaches [67] and references within and some recent publications by Senne et al. [72], Beltran et al. [9], Avella et al. [4], Brusco and Köhn [16]. For a comprehensive list of references to the PMP, the reader is addressed to Reese [67] and Mladenovich et al. [62] and a bibliographical overview by ReVelle et al. [70].

A Boolean linear programming formulation of the PMP can be defined on a weighted bipartite graph $G = (V, A, C)$ with $V = I \cup J$, $A \subseteq I \times J$, and non-negative weights $C = \{c_{ij} : c_{ij} \geq 0, (i, j) \in A\}$ as follows. For the given sets $I = \{1, 2, \dots, m\}$ of sites at which plants (cluster centers) can be located, $J = \{1, 2, \dots, n\}$ of clients (cluster points) with unit demand at each client site, and a matrix $C = [c_{ij}]$ of non-negative costs (distances or some other dissimilarity measure) of supplying each $j \in J$ from each $i \in I$, the number p of plants to be opened, the PMP is

$$\text{minimize} \quad \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (15)$$

$$\text{subject to} \quad \sum_{i=1}^m x_{ij} = 1, \quad j = 1, \dots, n. \quad (16)$$

$$x_{ij} \leq \bar{y}_i, \quad i = 1, \dots, m; \quad j = 1, \dots, n. \quad (17)$$

$$\sum_{i=1}^m \bar{y}_i = p, \quad (18)$$

$$\bar{y}_i \in \{0, 1\}, \quad i = 1, \dots, m. \quad (19)$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \dots, m; \quad j = 1, \dots, n. \quad (20)$$

For any feasible solution (x_{ij}, \bar{y}_i) , $\bar{y}_i = 1$ if plant i is open, and $\bar{y}_i = 0$, otherwise; $x_{ij} = 1$ if client j is assigned to plant i , and $x_{ij} = 0$, otherwise. Constraints (16) assign each client to exactly one plant, constraints (17) forbid the assignment of a client to a closed plant, and constraint (18) fixes the number of opened plants to p .

A PMP instance is described by an $m \times n$ matrix $C = [c_{ij}]$ and the number $1 \leq p \leq |I|$. It is assumed that the entries of C are nonnegative and finite, that is, $C \in \mathbb{R}_+^{mn}$. If $I = J$, we have the classic ReVelle and Swain's PMP model [69] with n^2 Boolean decision variables.

In this section, the study of the classic p -median model represented as a Boolean linear programming model is started by posing the following questions: (i) What are the optimal numbers of decision variables partitioned into Boolean and non-Boolean variables? (ii) What is the optimal number of constraints? (iii) do the above-mentioned numbers of decision variables and constraints depend on the PMP input data, more specifically on the number p of medians?

Further progress with improvements of ReVelle and Swain's PMP model are done by Rosing et al. [71], Cornuejols et al. [20], Church [18, 19], and recently by AlBdaiwi et al. [1] and Elloumi [23]. All of them have incorporated in different ways the following properties of PMP:

- (i) Based on an ordering of the distances with respect to a given demand point, they have either reduced the number of clients or excluded from (15) to (20) a repetition of decision variables x_{pj} and x_{qj} corresponding to the equal costs $c_{pj} = c_{qj}$ for some $j \in J$.
- (ii) The $mn + m$ Boolean decision variables are replaced by m Boolean decision variables and mn nonnegative decision variables, that is, (20) is replaced by

$$x_{ij} \geq 0, \quad i = 1, \dots, m; \quad j = 1, \dots, n. \quad (21)$$

To the best of the author's knowledge based on available publications, there is no PMP model that adjusts the numbers of nonnegative decision variables and corresponding linear constraints depending on the number p of medians.

This section proposes a new model formulation for the PMP that contains all previously suggested improvements which have been incorporated in a concise and simplified notation including the proposed adjustment of decision variables and corresponding linear constraints depending on the number p of medians. This new p -median formulation is called *a mixed Boolean pseudo-Boolean model for the PMP*. It is shown that the suggested model can result in a substantially smaller

mixed Boolean linear programming formulation for a given application of the PMP and can be used either to find a global optimum by means of general-purpose ILP solvers or to develop new exact and approximate algorithms based on the well-known methods in mixed integer programming (see, e.g., [74]).

As sometimes happened, some of the above-mentioned improvements were separately done for the PMP without taking into account the ongoing progress with model formulations for another common model within minsum location-allocation problems, namely, the simple plant location problem (SPLP), often referred to as the uncapacitated facility location problem (UFLP) (see [21]) or the warehouse location problem (see, e.g., [70]). Such problems are well known in cluster analysis (see, e.g., [16] and references within). Both problems form underlying models in several combinatorial problems, like set covering, set partitioning, information retrieval, simplification of logical Boolean expressions, airline crew scheduling, vehicle dispatching (see [17]), and assortment (see, e.g., [37, 65]), and are subproblems of various location analysis problems (see [70]).

An instance of the SPLP has an optimal solution in which each client is satisfied by exactly one plant. A similar observation is valid for the PMP. In Hammer [50], this fact is used to derive a pseudo-Boolean representation of the SPLP. The pseudo-Boolean polynomial (PBP) developed in that work has terms that contain both a literal and its complement. Subsequently, in Beresnev [11], a different pseudo-Boolean form has been developed in which each term contains only literals or only their complements. This form was found easier to manipulate, and hence Beresnev's formulation of the SPLP was adjusted to the PMP in AlBdaiwi et al. [1].

The purpose of this section is twofold. First, a new model for the p -median problem is designed, and it is shown that the number of nonnegative decision variables and corresponding constraints depends on the number of p -medians and will be adjusted in the suggested model. Moreover, these numbers are minimal in the class of mixed integer linear programs for the PMP. Second, it is shown that the new model allows solving by means of a general-purpose solver on PC some PMP benchmark instances previously intractable neither by general-purpose solver nor by the state-of-the-art exact algorithms.

In Sect. 4.2, there is a brief overview of mathematical models for the p -median problems based on combinatorial, pseudo-Boolean, and classic Boolean programming models. Based on the overview of the p -median models and reported computational results, a new model is motivated and formulated in Sect. 4.3. Section 4.4 presents data-correcting reductions, and Sect. 4.5 reports the computational study. Section 4.6 concludes this section with a summary and future research directions.

4.2 Brief Overview of PMP Formulations

Recall that it is given sets $I = \{1, 2, \dots, m\}$ of sites in which plants can be located, $J = \{1, 2, \dots, n\}$ of clients, a nonnegative matrix $C = [c_{ij}]$ of costs of supplying each $j \in J$ from each $i \in I$, the number p of plants to be opened, and unit demand

at each client site. The p -Median Problem (PMP) is one of finding a set $S \subseteq I$ with $|S| = p$, such that the total cost

$$f_C(S) = \sum_{j \in J} \min\{c_{i,j} | i \in S\} \quad (22)$$

is minimized. An instance of the problem is described by an $m \times n$ matrix $C = [c_{ij}]$ and the number $1 \leq p \leq |I|$. The *combinatorial formulation of PMP* is to find

$$S^* \in \arg \min\{f_C(S) : \emptyset \subset S \subseteq I, |S| = p\}. \quad (23)$$

The PMP (23) can be reformulated in terms of Hammer-Beresnev polynomials as the *pseudo-Boolean formulation of PMP*.

$$y^* \in \arg \min\{\mathcal{B}_C(y) : y \in \{0, 1\}^m, \sum_{i=1}^m y_i = m - p\}. \quad (24)$$

Here is $\mathcal{B}_C(y) = T_{C,\Pi}(y)$ (see (8)).

Example 1 Consider a PMP instance with $m = 4, n = 5, p = 2$, and

$$C = \begin{bmatrix} 1 & 6 & 5 & 3 & 4 \\ 2 & 1 & 2 & 3 & 5 \\ 1 & 2 & 3 & 3 & 3 \\ 4 & 3 & 1 & 8 & 2 \end{bmatrix} \quad (25)$$

borrowed from [23]. A possible ordering matrix for this problem is given by

$$\Pi = \begin{bmatrix} 1 & 2 & 4 & 1 & 4 \\ 3 & 3 & 2 & 2 & 3 \\ 2 & 4 & 3 & 3 & 1 \\ 4 & 1 & 1 & 4 & 2 \end{bmatrix}, \quad (26)$$

and the difference matrix is

$$\Delta = \begin{bmatrix} 1 & 1 & 1 & 3 & 2 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 2 & 3 & 2 & 5 & 1 \end{bmatrix}. \quad (27)$$

The Hammer-Beresnev polynomial representing the total cost function for this instance in the form (8) is

$$\begin{aligned}\mathcal{B}_C(y) = & [1 + 0y_1 + 1y_1y_3 + 2y_1y_2y_3] + \\ & [1 + 1y_2 + 1y_2y_3 + 3y_2y_3y_4] + \\ & [1 + 1y_4 + 1y_2y_4 + 2y_2y_3y_4] + \\ & [3 + 0y_1 + 0y_1y_2 + 5y_1y_2y_3] + \\ & [2 + 1y_4 + 1y_3y_4 + 1y_1y_3y_4].\end{aligned}\quad (28)$$

After reduction of similar monomials in the obtained polynomial, the following pseudo-Boolean representation of the instance is obtained:

$$\begin{aligned}\mathcal{B}_C(y) = & 8 + 1y_2 + 2y_4 + 1y_1y_3 + 1y_2y_3 + 1y_2y_4 + 1y_3y_4 + \\ & 7y_1y_2y_3 + 1y_1y_3y_4 + 5y_2y_3y_4 \rightarrow \min \\ s.t. \quad & y_1 + y_2 + y_3 + y_4 = m - p = 2 \\ & y \in \{0, 1\}^m.\end{aligned}\quad (29)$$

Let us show that the pseudo-Boolean formulation (24) is a generalization of the so-called *alternative formulation (AF)* (see [20]) and *new formulation (NF)* (see [23]). The reader should be careful with two different Boolean decision variables \bar{y}_i and y_i , such that $\bar{y}_i = 1 - y_i$ for all $i \in I$.

For any client j , let K_j be the number of different distances $C_{1j} < C_{2j} < \dots < C_{K_j j}$, within sorted distances $c_{\pi_1 j} \leq c_{\pi_2 j} \leq \dots \leq c_{\pi_m j}$, from j to any facility; hence, $K_j \leq m$, and $j = 1, \dots, n$. For each client j , let us define a hierarchy of neighborhoods as follows. For $k = 1, \dots, K_j$, denote by V_{kj} the k -th neighborhood of client j , that is, the set of facility sites located within distance C_{kj} from j or, formally, $V_{kj} = \{i : c_{ij} \leq C_{kj}\}$. AF uses the same \bar{y} variables as classical formulation (CF) (15)–(20) and introduces new variables z_{kj} . For any client j and $k = 1, \dots, K_j$, $z_{kj} = 1$ if and only if all the sites in V_{kj} are closed and 0, otherwise. Note that similar to the pseudo-Boolean formulation of PMP, the AF uses an inversion of standard assignments of the Boolean variables z_{kj} , namely, an opened plant is 0 and a closed one is 1.

The AF of PMP is

$$\text{minimize} \quad \sum_{j=1}^n \left(C_{1j} + \sum_{k=1}^{K_j-1} (C_{k+1j} - C_{kj}) z_{kj} \right) \quad (30)$$

$$\text{subject to} \quad z_{kj} + \sum_{i : c_{ij} \leq C_{kj}} \bar{y}_i \geq 1, \quad j = 1, \dots, n; \quad k = 1, \dots, K_j; \quad (31)$$

$$\sum_{i=1}^m \bar{y}_i = p; \quad (32)$$

$$z_{K_j j} = 0, \quad j = 1, \dots, n; \quad (33)$$

$$z_{kj} \geq 0, \quad j = 1, \dots, n; \quad k = 1, \dots, K_j; \quad (34)$$

$$\bar{y}_i \in \{0, 1\}, \quad i = 1, \dots, m. \quad (35)$$

Constraints (31) say that for any client j and for any of its neighborhoods V_{kj} , either at least one facility is open in V_{kj} or $z_{kj} = 1$. Constraints (33) ensure that for every client j , at least one plant is open in $V_{K_j,j}$ that is precisely the set of all plants. Finally, the objective function (30) sums up the allocation distances of all clients up to the first open plant.

CF and AF have the same number of Boolean variables y_i , but the number of nonnegative variables x_{ij} in CF is mn , and that of z_{kj} in AF is $K = \sum_{j=1}^m K_j$. Similarly, the number of constraints in CF and AF is $m + 1 + mn$ and $m + K + 1$, respectively. Hence, AF has approximately $mn - K$ less variables and constraints. If all distances are pairwise different, that is, for each j , $K_j = m$, then both formulations have the same size. For the example instance considered above (25), the AF formulation is as follows:

$$\begin{aligned}
& [1 + 1z_{11} + 2z_{21}] + [1 + 1z_{12} + 1z_{22} + 3z_{32}] + \\
& [1 + 1z_{13} + 1z_{23} + 2z_{33}] + [3 + 5z_{14}] + \\
& [2 + 1z_{15} + 1z_{25} + 1z_{35}] \rightarrow \min \\
& s.t. \\
& \bar{y}_1 + \bar{y}_2 + \bar{y}_3 + \bar{y}_4 = 2 \\
& z_{11} + \bar{y}_1 + \bar{y}_3 \geq 1 \\
& z_{21} + \bar{y}_1 + \bar{y}_2 + \bar{y}_3 \geq 1 \\
& z_{12} + \bar{y}_2 \geq 1 \\
& z_{22} + \bar{y}_2 + \bar{y}_3 \geq 1 \\
& z_{32} + \bar{y}_2 + \bar{y}_3 + \bar{y}_4 \geq 1 \\
& z_{13} + \bar{y}_4 \geq 1 \\
& z_{23} + \bar{y}_2 + \bar{y}_4 \\
& z_{33} + \bar{y}_2 + \bar{y}_3 + \bar{y}_4 \geq 1 \\
& z_{14} + \bar{y}_1 + \bar{y}_2 + \bar{y}_3 \geq 1 \\
& z_{15} + \bar{y}_4 \geq 1 \\
& z_{25} + \bar{y}_3 + \bar{y}_4 \geq 1 \\
& z_{35} + \bar{y}_1 + \bar{y}_3 + \bar{y}_4 \geq 1 \\
& z_{k,j} \geq 0, \quad j = 1, \dots, n; k = 1, \dots, K_j \\
& \bar{y}_i \in \{0, 1\}, \quad i = 1, \dots, m.
\end{aligned}$$

As indicated in Elloumi [23], AF is not always better than CF in terms of the running time needed by a general-purpose ILP solver, for example, CPLEX 8.1. With the purpose to reduce the above-mentioned CPU times for AF, Elloumi [23] has replaced constraints (31) by the following constraints:

$$z_{1j} + \sum_{i : c_{ij} = C_{1j}} \bar{y}_i \geq 1, \quad j = 1, \dots, n. \quad (36)$$

$$z_{kj} + \sum_{i : c_{ij} = C_{kj}} \bar{y}_i \geq z_{k-1j}, \quad j = 1, \dots, n; \quad k = 2, \dots, K_j. \quad (37)$$

The obtained PMP formulation (30), (36), (37), (32), (33), (34), and (35) is called a *new formulation* (NF) of PMP (see [23]). Similarly, to the neighborhoods V_{kj} , Church [19] has incorporated into CF the set of h_j closest sites to demand j by using a penalty variable f_j if demand j must be assigned to a plant further than its h_j closest plant. Church [19] calls such a model BEAMR (both exact and approximate model representation) and has demonstrated that BEAMR is useful to solve PMPs exactly without fully specifying all of the assignment variables and constraints needed in the CF. Moreover, any feasible solution to BEAMR is a valid lower bound to the PMP, and when some f_j values are positive, the solution can be used to test how close to optimality the solution obtained is. Church's [19] BEAMR as well as COBRA [18] (condensed Balinski constraints with the reduction of assignment variables using equivalent variable substitution) are very closed variations of the AF and NF applied to create a set of PMP models with the purpose to use general-purpose ILP software. For the cost matrix (25), Church's model is

$$\begin{aligned} & (1+3)x_{11} + 2x_{21} + 1x_{31} + 1x_{22} + 2x_{32} + (3+1)x_{42} + \\ & 2x_{23} + 3x_{33} + 3x_{24} + 3x_{34} + 4x_{13} + 3x_{35} + 2x_{45} \rightarrow \min \\ & s.t. \\ & \bar{y}_1 + \bar{y}_2 + \bar{y}_3 + \bar{y}_4 = 2 \\ & x_{11} + x_{21} + x_{31} = 1 \\ & x_{22} + x_{32} + x_{42} = 1 \\ & x_{23} + x_{33} + x_{43} = 1 \\ & x_{11} + x_{24} + x_{34} = 1 \\ & x_{13} + x_{35} + x_{45} = 1 \\ & x_{11} + x_{11} + x_{13} \leq (4-2+1)\bar{y}_1 \\ & x_{21} + x_{22} + x_{23} + x_{24} \leq 3\bar{y}_2 \\ & x_{31} + x_{32} + x_{33} + x_{34} + x_{35} \leq 3\bar{y}_3 \\ & x_{41} + x_{42} + x_{45} \leq 3\bar{y}_4 \end{aligned}$$

$$x_{11} \leq \bar{y}_1$$

$$x_{12} \leq \bar{y}_1$$

$$x_{13} \leq \bar{y}_1$$

$$x_{21} \leq \bar{y}_2$$

$$x_{22} \leq \bar{y}_2$$

$$x_{23} \leq \bar{y}_2$$

$$x_{24} \leq \bar{y}_2$$

$$x_{25} \leq \bar{y}_2$$

$$x_{31} \leq \bar{y}_3$$

$$x_{32} \leq \bar{y}_3$$

$$x_{33} \leq \bar{y}_3$$

$$x_{34} \leq \bar{y}_3$$

$$x_{35} \leq \bar{y}_3$$

$$x_{41} \leq \bar{y}_4$$

$$x_{42} \leq \bar{y}_4$$

$$x_{45} \leq \bar{y}_4$$

$$x_{i,j}, y_i \in \{0, 1\} \quad i = 1, \dots, m; \quad j = 1, \dots, n.$$

Elloumi [23] proves that the above-mentioned three mixed Boolean programs CF, AF, and NF formulate the same PMP and have the same lower bound by linear relaxation. Anyway, Elloumi [23] shows by means of computational experiments that the NF leads to better CPU times compared to CF or AF formulations because the NF sometimes has smaller number of constraints. Further speedup of NF linear programming relaxation (abbreviated by NFExr, see Elloumi [23]) is done by introduction of three reduction rules R1–R3 which eliminate z_{kj} variables corresponding either to different columns with the same ordering of distances or within a column with equal distances corresponding to different plants. The example PMP instance (25) borrowed from Elloumi [23] illustrates the corresponding numbers of variables and constraints for the above-reviewed formulations of PMP: CF, NF, NFExr, and finally the mixed Boolean pseudo-Boolean model (MBpBM). For CF, 4 Boolean variables y_i , 20 nonnegative variables x_{ij} , and 26 constraints including 4 Boolean constraints; for NF, 4 Boolean variables y_i , 18 nonnegative variables x_{ij} (in Elloumi [23], 17 nonnegative variables x_{ij} are indicated), and 28 constraints (in Elloumi [23], 23 constraints are indicated) including 4 Boolean constraints; for NFExr, 4 Boolean variables y_i , 7 nonnegative variables x_{ij} , and 21 constraints (in Elloumi [23], 11 constraints are indicated) including 4 Boolean constraints.

In the following fragment, it is shown that the proposed MBpBM applied to the PMP instance (25) has 4 Boolean variables y_i , 4 nonnegative variables z_i , and 13 constraints including 4 Boolean constraints. Recall that the original Hammer-Beresnev polynomial $\mathcal{B}_C(y)$ has 20 monomials including the monomials with zero coefficients, and after reducing similar monomials, there are 10 monomials. The number of monomials in the Hammer-Beresnev representation of a PMP instance can be further reduced by exploiting the fact that for any feasible solution y to the PMP instance, $\sum_{i=1}^m y_i = m - p$. This implies that any monomial in the Hammer-Beresnev polynomial for a PMP instance which can be expressed as a constant multiplied with more than $m - p$ literals necessarily evaluates to zero. This is formalized in [Theorem 4](#) (see [2]).

Theorem 4 *For any PMP instance C with $p \leq m$, the following assertions hold:*

1. *The degree of truncated Hammer-Beresnev polynomial $\mathcal{B}_{C,p}(y)$ is at most $m - p$.*
2. *The truncated Hammer-Beresnev polynomial is equal to the Hammer-Beresnev polynomial for any feasible solution y to the PMP instance, that is, $\mathcal{B}_C(y) = \mathcal{B}_{C,p}(y)$.*

Proof Suppose some monomial of degree d ($m - p < d < m$) evaluates to 1 in a feasible solution. By definition of $\mathcal{B}_C(y)$, such a monomial contains exactly d different variables y_i and evaluates to 1 only if all these d variables are ones. However, this contradicts the restriction on the number of opened facilities (see (24)) that ensures that in any feasible solution, exactly $m - p$ variables are set to 1. Thus, any monomial of degree greater than $m - p$ is zero for any feasible solution and $\mathcal{B}_C(y) = \mathcal{B}_{C,p}(y)$. \square

Taking into account that $p = 2$, the reduced and truncated Hammer-Beresnev polynomial $\mathcal{B}_{C,p=2}(y) = 8 + 1y_2 + 2y_4 + 1y_1y_3 + 1y_2y_3 + 1y_2y_4 + 1y_3y_4$ and has just 7 monomials with the corresponding ($p = 2$)-truncated matrix (see [2]) $C_{(p=2)}$ of the given matrix (25) as follows:

$$C_{(p=2)} = \begin{bmatrix} 1 & 3 & 3 & 3 & 4 \\ 2 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 3 & 3 \\ 2 & 3 & 1 & 3 & 2 \end{bmatrix}.$$

After introduction of nonnegative variables z_i , $i = 5, \dots, 8$, the objective function (see, e.g., [74]) was linearized such that $\mathcal{B}_{C,p=2}(y) = 8 + y_2 + 2y_4 + y_1y_3 + y_2y_3 + y_2y_4 + y_3y_4 = 8 + y_2 + 2y_4 + z_5 + z_6 + z_7 + z_8$ and the MBpBM is

$$\text{Minimize } 8 + y_2 + 2y_4 + z_5 + z_6 + z_7 + z_8 \quad (38)$$

$$\text{subject to } z_5 + 1 \geq y_1 + y_3, \quad (39)$$

$$z_6 + 1 \geq y_2 + y_3, \quad (40)$$

$$z_7 + 1 \geq y_2 + y_4, \quad (41)$$

$$z_8 + 1 \geq y_3 + y_4, \quad (42)$$

$$\sum_{i=1}^4 y_i = m - p = 2, \quad (43)$$

$$z_i \geq 0, \quad i = 5, \dots, 8; \quad (44)$$

$$y_i \in \{0, 1\}, \quad i = 1, \dots, 4. \quad (45)$$

This MBpBM (38)–(45) has the same decision variables y_i as the pseudo-Boolean formulation (24), but its objective function is a linear function on y_i and new variables z_i , coefficients of which are reflecting some penalties depending on the values of y_i . For example, $z_5 = y_1 y_3$ and this relationship can be reflected by the following linear constraint $y_1 + y_3 - 1 \leq z_5$ which is equivalent to $z_5 + 1 \geq y_1 + y_3$. In other words, each nonlinear monomial in the original pseudo-Boolean polynomial induces a linear inequality which must be added to the set of constraints. In the restrictions of MBpBM, the new variables z_i must be just nonnegative because Boolean variables y_i imply that all z_i are Boolean. Informally, the linear constraints (39)–(42) indicate whether or not a specific penalty to the linear part of objective function (38) should be added depending on the subset of opened and closed sites. For the sake of completeness, the best Elloumi's model (NFexr) (46) is included for the same example (38)–(45) as follows.

$$\begin{aligned} \text{Minimize } & 8 + (1 - \bar{y}_2) + 2(1 - \bar{y}_4) + z_{11} + 7z_{21} + z_{22} + 5z_{32} \\ & + z_{23} + z_{25} + z_{35} \end{aligned} \quad (46)$$

$$\text{subject to } z_{11} + \bar{y}_1 + \bar{y}_3 \geq 1,$$

$$z_{21} + \bar{y}_2 \geq z_{11},$$

$$\bar{y}_4 \geq z_{21},$$

$$z_{22} + \bar{y}_3 \geq 1 - \bar{y}_2,$$

$$z_{32} + \bar{y}_4 \geq z_{22},$$

$$\bar{y}_1 \geq z_{32},$$

$$z_{23} + \bar{y}_2 \geq 1 - \bar{y}_4,$$

$$z_{25} + \bar{y}_3 \geq 1 - \bar{y}_4,$$

$$z_{35} + \bar{y}_1 \geq z_{25},$$

$$\bar{y}_2 \geq z_{35},$$

$$\sum_{i=1}^4 \bar{y}_i = p = 2,$$

$$\begin{aligned} z_{ij} &\geq 0, \quad i = 1, \dots, 4; \quad j = 1, \dots, 5; \\ \bar{y}_i &\in \{0, 1\}, \quad i = 1, \dots, 4. \end{aligned}$$

Remember that compared to Elloumi's paper [23] in the suggested PMP, a transpose of matrix C was used, and hence the order of indices indicated in the variables z_{ji} is changed to z_{ij} . As easy to conclude, the objective function (38) in MBpBM has only 7 nonzero coefficients compared to 10 nonzero coefficients in (46). Hence, the main distinction between the MBpBM and Elloumi's NFexr model is the truncation in the new model of all terms with degree at least $(m - p + 1) = 4 - 2 + 1 = 3$ in pBp and corresponding linear constraints. Actually, the reduced but non-truncated pBp is

$$8 + y_2 + 2y_4 + y_1y_3 + 1y_2y_3 + 1y_2y_4 + 1y_3y_4 + 7y_1y_2y_3 + 1y_1y_3y_4 + 5y_2y_3y_4$$

and has also 10 terms with the same set of nonzero coefficients, namely,

$$8 + y_2 + 2y_4 + z_{11} + z_{22} + z_{23} + z_{25} + 7z_{21} + z_{35} + 5z_{32}.$$

The number of linear constraints in (38)–(45) is 9 including 4 non-negativity constraints (44) compared to 18 linear constraints including 7 non-negativity constraints for Elloumi's NFexr model.

4.3 The Mixed Boolean Pseudo-Boolean Model (MBpBM)

As a consequence of [Theorem 4](#), the objective function of a PMP instance with $p \leq m$ and a cost matrix C can be described using the following *truncated Hammer-Beresnev polynomial*:

$$\mathcal{B}_{C,p}(y) = \sum_{j=1}^n \left\{ \Delta c[0, j] + \sum_{k=1}^{m-p} \Delta c[k, j] \cdot \prod_{r=1}^k y_{\pi_{rj}} \right\}. \quad (47)$$

A corollary to [Theorem 4](#) is a reformulation of the definition of PMPs in terms of truncated Hammer-Beresnev polynomials.

Corollary 2 A PMP instance with m possible facilities, n clients, and a cost matrix C can be represented as

$$\mathbf{y}^* \in \arg \min \left\{ \mathcal{B}_{C,p}(y) : y \in \{0, 1\}^m, \sum_{i=1}^m y_i = m - p \right\}. \quad (48)$$

Let us denote the truncated and reduced Hammer-Beresnev polynomial [14] $\mathcal{B}_{C,p}(y)$ by

$$\mathcal{A}_{C,p}(y) = \sum_{r=0}^k \alpha_r \prod_{i \in T_r} y_i, \quad (49)$$

with the set T_r of Boolean variables y_i included in term r and new variables $z_r = y_r$, for $r = 1, \dots, m$, and $z_r = \prod_{i \in T_r} y_i$ for $r = m+1, \dots, k$. Since $\alpha_r \geq 0$ and PMP is a minimization problem (48), it is possible to replace each nonlinear equality $\prod_{i \in T_r} y_i = z_r$ by an equivalent nonlinear inequality $\prod_{i \in T_r} y_i \leq z_r$ which is equivalent to the following linear inequalities $\sum_{i \in T_r} y_i - |T_r| + 1 \leq z_r$ and $z_r \geq 0$. In any optimal PMP solution, the variable $z_r = 0$ if and only if at least one variable $y_i = 0$, and $z_r = 1$ if and only if all $y_i = 1$, that is, $z_r \in \{0, 1\}$. Now, the pseudo-Boolean formulation of PMP with a nonlinear objective function (49) can be presented as the following mixed Boolean linear programming model:

$$\text{Minimize} \quad \left\{ \alpha_0 + \sum_{r=1}^m \alpha_r y_r + \sum_{r=m+1}^k \alpha_r z_r \right\} \quad (50)$$

$$\text{subject to} \quad \sum_{i=1}^m y_i = m - p; \quad (51)$$

$$\sum_{i \in T_r} y_i - |T_r| + 1 \leq z_r, \quad r = m+1, \dots, k; \quad (52)$$

$$z_i \geq 0, \quad i = m+1, \dots, k. \quad (53)$$

$$y_i \in \{0, 1\}, \quad i = 1, \dots, m. \quad (54)$$

The objective function (50) is split into three parts: the first part α_0 is the sum of all smallest entries δ_{1k} per column (client) k under assumption that all sites are opened, the second part reflects the penalties incurred by the next δ_{2k} to the smallest entries, and the third part represents all other penalties corresponding to δ_{rk} for $1 < r \leq m-p$. All other constraints are explained similarly to the example (38)–(45).

4.3.1 Further Data Correcting

Even though MBpBM is a very compact model, it can be further reduced by involving upper and lower bounds on the cost of optimal solutions similarly to the SPLP (see Preprocessing SPLP instances, in Sect. 3.2). Suppose we know from some “oracle” an (global) upper bound f^{UB} on the cost of optimal solutions. Let us now consider some term r and the set of indices of its variables T_r . If now a vector y^r is made by setting its i -th entry to 1 if $i \in T_r$ and to 0, otherwise, then the value of the pseudo-Boolean polynomial on y^r is a trivial lower bound for the subspace of solutions having locations from T_r closed. More formally,

$$f_r^{LB} = \mathcal{B}_{C,p}(y^r) \quad (55)$$

The essence of the reduction that is considered here can be expressed by the following lemma.

Lemma 2 *If for some term $r = r(y)$ of a truncated Hammer-Beresnev polynomial holds $f_r^{LB} > f^{UB}$, then for every optimal solution y^* holds $r(y^*) = 0$.*

Proof Taking into account that there is a truncated polynomial, any term r has a degree of at most $m - p$ and $|T_r| \leq m - p$. This means that to keep r equal to 1, at least p sites are to be opened, implying that the costs of any feasible solution with sites from T_r closed is at least $\mathcal{B}_{C,p}(y')$ (by closing additional sites, the objective value can only be increased). By condition of the lemma, there exists a cheaper feasible solution; thus, any feasible solution y with $y_i = 1$ for any $i \in T_r$ is not optimal. \square

The following counterexample shows that the inequality in Lemma 2 should be strict.

Example 2 Consider an instance defined by the following cost matrix C :

$$C = \begin{bmatrix} 0 & 6 & 6 \\ 1 & 0 & 8 \\ 2 & 9 & 9 \\ 5 & 4 & 0 \end{bmatrix}.$$

A possible permutation and difference matrices are

$$\Pi = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 4 & 1 \\ 3 & 1 & 2 \\ 4 & 3 & 3 \end{bmatrix} \Delta = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 4 & 6 \\ 1 & 2 & 2 \\ 3 & 3 & 1 \end{bmatrix}.$$

In case $p = 2$, the Hammer-Beresnev polynomial is

$$\mathcal{B}_{C,p=2}(y) = 1y_1 + 4y_2 + 6y_4 + 1y_1y_2 + 3y_1y_4 + 2y_2y_4.$$

Considering the first term $r = y_1$, there are $T_1 = \{1\}$, $y^1 = (1, 0, 0, 0)^T$, and $f_1^{LB} = (B)_C((y)^1) = 1$. Suppose the upper bound is the same $f^{UB} = 1$. If the inequality in Lemma 2 was non-strict, then this would imply that for any optimal solution $y_1 = 0$. However, it can be checked that for the unique optimal solution to the instance $y^* = (1, 0, 1, 0)^T$, this does not hold.

Now the above condition of the Lemma 2 for every term of the pseudo-Boolean polynomial can be checked, starting from terms with the lowest degree. Such order of checking terms is beneficial because if some term is found to be zero in any

optimal solution (let us call such terms *null terms*), then all terms of higher degree that contain it are also zero. There are two possibilities of dealing with null terms within the MBpBM model.

The first and the most straightforward approach is to set the variable that corresponds to a null term to zero throughout the formulation. This eliminates one term from the objective function but preserves the number of constraints. We call the model reduced according to this approach based on bounds MBpBMb. The following example shows how this reduction works. Having the cost matrix C (25), $p = 2$, and an MBpBM model (38)–(45), the global upper bound f^{UB} can be computed, for example, by greedy heuristics that work as follows. Starting with all facilities opened, at each iteration, the facility that minimizes the cost of obtained solution is closed until exactly p facilities remain opened. For the considered case, $f^{UB} = 9$. Now for every term r , f_r^{LB} is computed. The first term is y_2 , and the corresponding lower bound for all solutions having second facility closed is $f_2^{LB} = \mathcal{B}_{C,p=2}(0, 1, 0, 0) = 9$. Thus, this term does not pass the test. The next term is y_4 and $f_4^{LB} = \mathcal{B}_{C,p=2}(0, 0, 0, 1) = 10 > f^{UB}$, implying that for every optimal solution, $y_4 = 0$ and (38)–(45) can be reduced to

$$\begin{aligned} \text{minimize} \quad & 8 + y_2 + z_5 + z_6 + 0z_7 + 0z_8 \\ \text{subject to} \quad & z_5 + 1 \geq y_1 + y_3, \\ & z_6 + 1 \geq y_2 + y_3, \\ & 0 + 1 \geq y_2 + 0, \\ & 0 + 1 \geq y_3 + 0, \\ & y_1 + y_2 + y_3 = 2, \\ & y_4 = 0 \\ & z_i \geq 0, \quad i = 5, \dots, 8; \\ & y_i \in \{0, 1\}, \quad i = 1, \dots, 4. \end{aligned}$$

Here is used the fact that $z_7 = y_2y_4$ and $z_8 = y_3y_4$ both contain the variable that is set to 0 and thus can also be eliminated. It should be noted that third and fourth constraints are redundant and can be dropped. The next term is y_1y_3 , and the lower bound is $f_{1,3}^{LB} = \mathcal{B}_{C,p=2}(1, 0, 1, 0) = 9$. The last term is y_2y_3 , and the lower bound is $f_{2,3}^{LB} = \mathcal{B}_{C,p=2}(0, 1, 1, 0) = 10 > f^{UB}$ implying that for every optimal solution, the corresponding variable is $z_6 = 0$. This leads to the following MBpBMb model:

$$\begin{aligned} \text{Minimize} \quad & 8 + y_2 + z_5 \\ \text{subject to} \quad & z_5 + 1 \geq y_1 + y_3, \\ & 1 \geq y_2 + y_3, \\ & y_1 + y_2 + y_3 = 2, \end{aligned}$$

$$\begin{aligned} y_4 &= 0, \\ z_5 &\geq 0; \\ y_i &\in \{0, 1\}, \quad i = 1, \dots, 4. \end{aligned}$$

Another approach to dealing with null terms allows not only to reduce the size of the objective function but also the number of constraints. Its essence is expressed by the following lemmas.

Lemma 3 *Increasing a coefficient at a null term does not affect the cost of optimal solutions.*

Proof Straightforward from the definition of null terms. If some term is found to be equal to 0 in any optimal solution, then increasing the coefficient of the corresponding monomial will not result in new optimal solutions. \square

Lemma 4 *If for some term r_1 in the Hammer-Beresnev polynomial there exists an embedded term r_0 with infinitely large coefficient, then r_1 can be given a zero coefficient without affecting the cost of optimal solutions.*

Proof If there exists an optimal solution of finite cost, then closing locations from $T_{r_1} \setminus T_{r_0}$ does not change the cost of a solution. Thus, the coefficient at r_1 does not influence the objective value and can be set to an arbitrary number, for example, 0. \square

Thus, if a null term is found, it can be given an infinite (large enough) coefficient, and all terms for which it is embedded can be eliminated from the Hammer-Beresnev polynomial without a need for additional constraints. The model with this reduction is called MBpBMB1. Even though it allows smaller reduction of the objective function, it can benefit from the reduced number of constraints. For the considered above instance with cost matrix C (25) and $p = 2$, the MBpBMB1 model is as follows:

$$\begin{aligned} \text{Minimize } \quad & 8 + y_2 + 1000y_4 + z_5 + 1000z_6 \\ \text{subject to } \quad & z_5 + 1 \geq y_1 + y_3, \\ & z_6 + 1 \geq y_2 + y_3, \\ & y_1 + y_2 + y_3 + y_4 = 2, \\ & y_4 = 0, \\ & z_i \geq 0, \quad i = 5, \dots, 8; \\ & y_i \in \{0, 1\}, \quad i = 1, \dots, 4. \end{aligned}$$

Here, infinite coefficients were set to 1,000 for the sake of clarity, while in practice the value of $f^{UB} + 1 = 10$ suffices for this purpose. It is also clear that if a linear null term is found, then the corresponding y variable is fixed to 0, and it is eliminated from the objective function (instead of raising its coefficient to infinity). Moreover, if for some null term there are no terms into which it is embedded, then it is beneficial to eliminate it and add a corresponding constraint. If these exceptions are introduced into MBpBMb1, then for the considered example both formulations (MBpBMb and MBpBMb1) become equal, although the mechanisms by which some constraints were dropped are different. While in MBpBMb third and fourth constraints became redundant as most of the variables in them were set to 0, in MBpBMb1 these constraints did not exist at all because the corresponding terms were eliminated from the pseudo-Boolean polynomial.

Finally, it should be mentioned that instead of $f_r^{LB} = \mathcal{B}_{C,p}(y^r)$, a somewhat stronger lower bound can be used:

$$f_r^{LB} = f_C(\bar{T}_r) + \min_{k_i \in \bar{T}_r} \sum_{i=1}^{m-p-|\bar{T}_r|} [f_C(\bar{T}_r \setminus \{k_i\}) - f_C(\bar{T}_r)] \quad (56)$$

where $f_C(\cdot)$ – cost function of the PMP, that is, $f_C(\bar{T}_r) = \mathcal{B}_{C,p}(y^r)$ and \bar{T}_r denotes the complement of T_r , that is, $\bar{T}_r = \{1, \dots, m\} \setminus T_r$. In the computational experiments reported in the following section, lower bounds given by (56) were used.

4.4 Reduction Tests for the p -Median Problem

Looking back on the formulations described in the available literature and on the principles behind the formulation, it can be noticed that all improvements over the classical formulation by ReVelle and Swain [69] are done by application of various reduction tests to the instances of the problem. These tests can be classified into the following two broad groups:

- Pure
- Optimizational

The first group includes all kinds of tests exploiting structural redundancies in the input data. For example, presence of equal entries within a column of the cost matrix was first used by Cornuejols [20] in his alternative formulation (AF) (30) and is present in most of the recent models, including Elloumi's NF [23] and the MBpBM (50)–(54). Another pure reduction is an exclusion of the largest p entries from the formulation of each column of the cost matrix. This reduction was used by Avella et al. ([3], reduction test *FIX1*) and Church [18], and the proposed model is implemented via truncation of the Hammer-Beresnev polynomial. It is worth mentioning that this reduction has a universal nature in a sense that it always allows to exclude from consideration exactly p entries from each column of the cost matrix, irrespective of the particular instance data. The next structural peculiarity

that can be exploited for strengthening the formulation is related to the similarity of neighborhoods of the clients, that is, the order in which potential locations are listed if being sorted by increasing distance from a client. If two clients have equal neighborhoods, then they may be considered as one aggregated client. In the new model, this rule is reflected by combining similar monomials in the pseudo-Boolean polynomial, while in Elloumi's NF (reduction rules R2 and R3 in [23]) and Church's COBRA [18], this is done by "substitution of equivalent variables" (as it is called in [18]). Finally, for the sake of completeness, it is important to mention Elloumi's reduction rule R1 that is related to the obtained ILP formulation rather than to initial data. The essence of this rule is that some z variables can be expressed in terms of y variables in a linear way. In fact, this rule is implicitly present in the proposed model as no new variables were introduced for linear monomials of the Hammer-Beresnev polynomial. The effect of these pure reductions can be illustrated by [Table 11](#) where reduction tests were applied to several selected benchmark instances that are widely used for testing PMP-related algorithms. The first three columns contain the title of the instance, number of potential locations m , and the number of medians p , correspondingly. Next, two columns indicate the number of entries in the cost matrix $|C|$ and the number of nonzero coefficients in the pseudo-Boolean polynomial $|B|$. The last column displays the achieved reduction that was computed as $reduction = (|C| - |B|)/|C| \times 100\%$.

The second group of reduction tests includes optimizational approaches that suggest (pre-)solving the problem. These are reductions based on comparison of estimated upper and (restricted or particular) lower bounds of the optimal solution. Such a reduction was used by Avella et al. ([3], reduction test *FIX2*) and is implemented in the given formulations MBpBMB and MBpBMB1 leading to even more compact models and, as will be showed in the next section, lower computing times.

The above can be summarized as follows: MBpBM in a natural way incorporates all the available in literature pure reductions and can be subjected to optimizational problem size reduction techniques.

4.5 Computational Results

In order to show the applicability of the compact MBpBM formulation, a number of computational experiments were held. The benchmark instances employed have been taken from two of the most widely used libraries: J. Beasley's OR-library (see <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html>) and randomly generated RW instances by Resende and Werneck (see, e.g., [23]). The common class of benchmark instances included in almost all publications devoted to the PMP itself is just the OR-Library instances. Since the main purpose of the experiments is to show that the new model for PMP is one of the best currently known models (see [19, 23]) which could be used to solve PMP instances to optimality based on general-purpose software, Xpress-MP and 15 largest instances (see [Tables 12](#) and [14](#)) have been used from OR-Library [7]. This problem library contains

Table 11 Effect of reduction tests for selected benchmark instances

Instance	m	p	C	B	Reduction (%)
pmed26	600	5	360,000	25,440	92.93
pmed27	600	10	360,000	24,117	93.30
pmed28	600	60	360,000	19,142	94.68
pmed29	600	120	360,000	17,724	95.08
pmed30	600	200	360,000	16,920	95.30
pmed31	700	5	490,000	25,940	94.71
pmed32	700	10	490,000	26,384	94.62
pmed33	700	70	490,000	21,030	95.71
pmed34	700	140	490,000	18,684	96.19
pmed35	800	5	640,000	27,788	95.66
pmed36	800	10	640,000	28,579	95.53
pmed37	800	80	640,000	23,324	96.36
pmed38	900	5	810,000	29,230	96.39
pmed39	900	10	810,000	27,638	96.59
pmed40	900	90	810,000	24,165	97.02
rw100	100	10	10,000	5,683	43.17
	100	20	10,000	5,045	49.55
	100	30	10,000	4,404	55.96
	100	40	10,000	3,771	62.29
	100	50	10,000	3,138	68.62
rw250	250	10	62,500		
	250	25	62,500	35,448	43.28
	250	50	62,500	31,497	49.60
	250	75	62,500	27,581	55.87
	250	100	62,500	23,592	62.25
	250	125	62,500	19,658	68.55
rw500	500	10	250,000		
	500	25	250,000		
	500	50	250,000		
	500	75	250,000	134,147	46.34
	500	100	250,000	126,281	49.49
	500	150	250,000	110,487	55.81
	500	200	250,000	94,795	62.08
	500	250	250,000	78,946	68.42
rw1000	1,000	50	1,000,000		
	1,000	100	1,000,000		
	1,000	200	1,000,000	505,207	49.48
	1,000	300	1,000,000	441,946	55.81
	1,000	400	1,000,000	378,732	62.13
	1,000	500	1,000,000	315,682	68.43

Table 12 Comparison of computing times for the new and Elloumi's NF formulations (15 largest OR-Library instances)

Instance	m	p	MBpBM	MBpBMB	MBpBMB1	Elloumi
pmmed26	600	5	163.84	194.08	111.81	180.31
pmmed27	600	10	27.59	41.00	21.31	43.73
pmmed28	600	60	2.48	8.63	2.13	3.61
pmmed29	600	120	1.78	6.50	1.31	2.91
pmmed30	600	200	1.50	5.56	0.78	4.81
pmmed31	700	5	153.22	132.91	57.05	90.95
pmmed32	700	10	33.13	53.17	43.39	37.64
pmmed33	700	70	3.09	10.11	2.69	4.73
pmmed34	700	140	3.72	8.03	1.97	7.11
pmmed35	800	5	70.30	233.66	154.41	514.72
pmmed36	800	10	2,256.83	2,014.70	4,252.13	6,737.25
pmmed37	800	80	3.91	12.61	3.08	7.00
pmmed38	900	5	1,328.34	368.73	2,041.28	307.00
pmmed39	900	10	572.81	713.59	444.08	473.95
pmmed40	900	90	5.39	15.53	4.02	8.42

40 different PMP instances, each representing a graph of n vertices and each being a client and a potential facility, and a specific value of p . Graphs are complete and range in size from 100 (with 10,000 arcs) to 900 (with 810,000 arcs) nodes. The distance c_{ij} between two nodes is the length of a shortest path linking them.

The experiments have been conducted on a personal computer with Intel 2.33-GHz processor 1.95-GB RAM and Xpress-MP as an ILP solver.

Tables 12 and 13 summarize the computational results obtained for the largest 15 OR instances and random RW instances, correspondingly. The first three columns contain the name of instance, the number of m nodes, and the number p of medians. The next three columns are related to the running times (in seconds) of the considered above variations of the new model: the initial MBpBM formulation and its modifications incorporating reductions based on bounds. The last column reflects computing times for Elloumi's NF model that were implemented and tested within the same environment as the proposed models so as to ensure consistent comparison of performance.

The computational experiments with OR and RW instances can be summarized as follows. The basic MBpBM formulation outperforms Elloumi's new formulation in most of the tested cases, especially for larger numbers of medians p . At the same time, the reduction based on bounds (see column MBpBMB) has comparatively poor performance in general. This can be explained by an increased number of nonzero coefficients in a constraint matrix. However, there exist instances (e.g., pmmed36 and pmmed38) for which MBpBMB performs better than other variations of the formulation based on a pseudo-Boolean polynomial and for which instance pmmed36 has three times smaller computing times compared to NF. Finally, the revised

Table 13 Comparison of computing times for the new and Elloumi's NF formulations (Resende and Werneck random instances)

Instance	m	p	MBpBM	MBpBMb	MBpBMb1	Elloumi
rw100	100	10	678.91	671.28	452.52	845.30
	100	20	4.00	6.44	2.22	5.25
	100	30	0.09	0.43	0.03	0.13
	100	40	0.08	0.34	0.02	0.14
	100	50	0.06	0.28	0.02	0.13
rw250	250	10	—	—	—	—
	250	25	—	—	—	—
	250	50	340.86	1,633.58	225.83	335.86
	250	75	1.09	8.50	0.48	2.08
	250	100	0.50	5.44	0.11	0.88
	250	125	0.66	5.02	0.27	1.38
rw500	500	10	—	—	—	—
	500	25	—	—	—	—
	500	50	—	—	—	—
	500	75	—	—	—	—
	500	100	—	—	—	—
	500	150	2.97	105.63	1.22	12.27
	500	200	2.25	54.78	0.28	4.11
	500	250	1.77	44.83	0.13	4.36
rw1000	1,000	10	*	*	*	*
	1,000	25	*	*	*	*
	1,000	50	*	*	*	*
	1,000	75	*	*	*	*
	1,000	100	*	*	*	*
	1,000	150	*	*	*	*
	1,000	200	*	*	*	*
	1,000	300	*		13.40	
	1,000	400	*		1.16	
	1,000	500	*		0.77	

—Not solved within 1 h

*Not enough memory for ILP solver

reduction based on bounds MBpBMb1 outperformed other considered models in almost all cases except pmed32, pmed36, and pmed38 (for pmed36, it is better than NF but is worse than MBpBM).

Trying to understand which of the considered, in Sect. 4.2, models is the best one, Elloumi [23] has compared the number of variables, constraints, and non-zero entries in the constraint matrix. For pmed33 instance, Elloumi's (our) model contains 31,852 (20,329) variables, 31,852 + 1 (20,329 + 1) constraints, and 553,004 (407,981) nonzero entries of constraint matrix. For the same instance pmed33, Church's BEAMR contains 9,940 variables and 9,241 constraints. The numbers of nonzero entries in constraint matrices are not indicated.

4.6 Summary and Future Research Directions

In this section, a new approach to formulate models for the p -median problem (PMP) is presented. First, the PMP is formulated using a pseudo-Boolean polynomial (pBp) as the objective function and with just one constraint related to the number of medians in a feasible solution. Then, the objective function is compacted by reducing similar monomials. After this, the nonlinear monomials in the objective function are linearized with additional variables and linear constraints. The resulting model belongs to the well-studied class of mixed Boolean linear programming models which is called mixed Boolean pseudo-Boolean model (MBpBM). The MBpBM allows to solve PMPs with much less execution times than are required by the best known models in the literature. It also allows us to solve much larger problems by general-purpose ILP software than is currently possible using previous model formulations. The MBpBM has been able to obtain an optimal solution to the fl1400 instance with $p = 400$, which remained unsolved in Avella et al. [4] by their state-of-the-art-algorithm for PMP as well as by Beltran et al.'s [9] advanced semi-Lagrangian approach based on proximal analytic center cutting plane method.

The main distinction between MBpBM and all well-known mixed integer PMP formulations is that the number of decision variables in MBpBM is automatically adjusted according to the number p of medians, that is, the number of decision variables as well as the number of constraints decrease linearly with increasing values of p . This feature of MBpBM implies that PMP instances with relatively large numbers of medians are easier to solve using standard MILP software applied with the MBpBM. Thus, the new model and the approach to create it not only extend the capability of general-purpose software to solve larger sized p -median problems but also make it possible to solve smaller problems more efficiently when using general-purpose software.

The MBpBM approach should also lead to reduced model sizes for other location models like the simple plant location problem and the capacitated simple plant location problem (SPLP), which is a generalization of PMP with fixed charges, as well as to improved data-correcting approach to the SPLP (see [46]). There are several interesting extensions to the work done in this chapter. One clear extension is in the development of heuristics based on either Hammer-Beresnev's functions (polynomials) or their linearized MILP.

A second important direction in this type of research is to exploit the most important ingredient of data correcting for Hammer-Beresnev functions (polynomials), namely, their truncations, by adding an artificial constraint on the cardinality of unknown optimal solution $p = |S^*|$ to the SPLP, for example, to an analogue of MBpBM, and hence to reduce the number of monomials in the corresponding Hammer-Beresnev polynomial. The correct number p of artificial medians might be found by solving a sequence of SPLPs based on the binary search as follows: $p = m/2, m/4, \dots$

Computational results reported in Table 14 show that the MBpBM is useful for p -median approach to large cell formation instances in group technology [43]. A third direction of research, and the one that is planned for the immediate future, is to find the optimal number of cells (see [42]).

Table 14 MBpBM for different numbers of medians in pmed39/pmed40

p	pmed39				pmed40			
	$f_C(S^*)$	M_{tr}	constraints	MBpBM	$f_C(S^*)$	M_{tr}	constraints	MBpBM
1	14,720	29,042	706,612	7.3	17,425	31,641	744,257	12.8
5	11,069	28,839	705,976	79.7	12,305	31,268	743,056	39.3
9	9,690	27,883	702,341	429.2	10,740	30,155	738,633	54.8
10	9,423	27,637	701,168	121.2	10,491	29,905	737,406	88.0
20	7,894	26,008	690,135	80.5	8,717	28,143	725,285	104.3
30	7,051	24,983	679,640	567.4	7,731	27,109	714,318	138.7
40	6,436	24,272	669,999	182.9	7,037	26,372	703,930	113.4
50	5,941	23,688	660,138	93.9	6,518	25,813	694,355	782.7
60	5,545	23,229	651,280	38.6	6,083	25,304	684,402	171.9
70	5,215	22,815	641,971	5.7	5,711	24,883	674,846	33.3
80	4,929	22,439	632,520	4.6	5,398	24,503	665,476	5.5
90	4,684	22,147	624,079	4.5	5,128	24,164	656,067	5.7
100	4,462	21,844	615,198	4.9	4,878	23,851	646,520	5.4
200	2,918	19,731	529,883	2.7	3,132	21,623	557,661	3.1
300	1,968	18,252	449,160	2.0	2,106	20,066	473,237	2.4
400	1,303	17,000	371,790	1.5	1,398	18,725	391,820	1.7
500	821	15,812	298,029	1.3	900	17,431	314,045	1.3
600	471	14,495	223,027	1.0	530	16,040	237,199	0.9
700	244	12,962	151,916	0.7	271	14,337	161,826	0.6
800	100	10,684	81,510	0.3	100	11,781	86,772	0.7

5 Maximization of General Submodular Functions

Let $N = \{1, 2, \dots, n\}$ and 2^N denote the set of all subsets of N . A function $z : 2^N \rightarrow \mathbb{R}$ is called *submodular* if for each $I, J \in 2^N$, $z(I) + z(J) \geq z(I \cup J) + z(I \cap J)$. The solution process of many classical combinatorial optimization problems, like the generalized transportation problem, quadratic cost partition (QCP) problem with nonnegative edge weights and set covering, can be formulated as the maximization of a submodular function (MSF), that is, the problem:

$$\max\{z(T) | T \subseteq N\}.$$

Although the general problem of the maximization of a submodular function is known to be NP-hard (see [60]), there has been a sustained research effort aimed at developing practical procedures for solving medium and large-scale problems in this class (see [39]). In the remainder of this section, two data-correcting algorithms for solving the problem are suggested. Note that [Theorem 1](#) assumes the following form for this problem.

Lemma 5 Consider two submodular functions $z_1(\bar{x}) = \sum_{i=1}^n \sum_{j=i}^n c_{ij}^1$ $\prod_{k=i}^j x_k$ and $z_2(\bar{x}) = \sum_{i=1}^n \sum_{j=i}^n c_{ij}^2 \prod_{k=i}^j x_k$. Let \bar{x}_1^* and \bar{x}_2^* be the maximum

points of $z_1(\bar{x})$ and $z_2(\bar{x})$ respectively. Then,

$$z_1(\bar{x}_1^*) - z_1(\bar{x}_2^*) \leq \sum_{i=1}^n \sum_{j=1}^n |c_{ij}^1 - c_{ij}^2|.$$

The algorithm described in Sect. 5.1 has been published in Goldengorin et al. [44], while that described in Sect. 5.2 has been published in Goldengorin and Ghosh [40]. For each of the two algorithms, a class of polynomially solvable instances for the submodular function maximization problems is described. Then, the data-correcting algorithms that use this class of polynomially solvable instances to solve a general submodular function maximization problem are described. The classes of polynomially solvable instances are algorithmically defined, that is, they are classes of instances that are solved to optimality using a prespecified polynomial algorithm.

5.1 A Simple Data-Correcting Algorithm

The class of polynomially solvable instances described here is defined using a polynomial time algorithm called the preliminary preservation (PP) algorithm. Normally, these algorithms terminate with a subgraph of the Hasse diagram of the original instance which is guaranteed to contain the maximum. However, for instances where PP returns a subgraph with a single node, that node is the maximum, and the instance is said to have been solved in polynomial time. Instances such as these make up the class of polynomially solvable instances that is considered here.

Let z be a real-valued function defined on the power set 2^N of $N = \{1, 2, \dots, n\}$; $n \geq 1$. For each $S, T \in 2^N$ with $S \subseteq T$ is defined

$$[S, T] = \{I \in 2^N \mid S \subseteq I \subseteq T\}.$$

Note that $[\emptyset, N] = 2^N$. Any interval $[S, T]$ is a subinterval of $[\emptyset, N]$ if $\emptyset \subseteq S \subseteq T \subseteq N$. This is denoted using the notation $[S, T] \subseteq [\emptyset, N]$. In this section, an interval is always a subinterval of $[\emptyset, N]$. It is assumed that z attains a finite maximum value on $[\emptyset, N]$ which is denoted by $z^*[\emptyset, N]$, and $z^*[S, T] = \max\{z(I) \mid I \in [S, T]\}$ for any $[S, T] \subseteq [\emptyset, N]$. $d_k^+(I) = z(I + k) - z(I)$ and $d_k^-(I) = z(I - k) - z(I)$ are also defined.

The following theorem and corollaries from Goldengorin et al. [44] act as a basis for the preliminary preservation (PP) algorithm described therein.

Theorem 5 Let z be a submodular function on $[S, T] \subseteq [\emptyset, N]$ and let $k \in T \setminus S$. Then the following assertions hold:

1. $z^*[S + k, T] - z^*[S, T - k] \leq z(S + k) - z(S) = d_k^+(S)$.
2. $z^*[S, T - k] - z^*[S + k, T] \leq z(T - k) - z(T) = d_k^-(T)$.

Corollary 3 ((Preservation rules of order zero)) *Let z be a submodular function on $[S, T] \subseteq [\emptyset, N]$, and let $k \in T \setminus S$. Then the following assertions hold:*

1. *First preservation rule: If $d_k^+(S) \leq 0$, then $z^*[S, T] = z^*[S, T - k] \geq z^*[S + k, T]$.*
2. *Second preservation rule: If $d_k^-(T) \leq 0$, then $z^*[S, T] = z^*[S + k, T] \geq z^*[S, T - k]$.*

The PP algorithm accepts an interval $[S, T]$, $S \subseteq T$ and tries to apply Corollary 3 repeatedly. It returns an interval $[X, Y]$, $S \subseteq X \subseteq Y \subseteq T$, such that $z^*[S, T] = z^*[X, Y]$. The pseudocode for this algorithm is given below.

Algorithm PP($[S, T]$)

Output: A subinterval of $[S, T]$ containing the maximum of z over $[S, T]$.

Code:

```

1. begin
2.     if  $T = S$  return  $[S, S]$ ;
3.     while  $T \neq S$  do begin
4.          $d_{max}^+ := \max\{d_k^+ | k \in T \setminus S\}$ ;
5.          $d_{max}^- := \max\{d_k^- | k \in T \setminus S\}$ ;
6.         if  $d_{max}^+ \leq 0$  then begin
7.              $k_{max}^+ := \arg \min\{k \in T \setminus S | d_k^+ = d_{max}^+\}$ ;
8.              $T := T - k_{max}^+$ ;
9.         end
10.        else if  $d_{max}^- \leq 0$  then begin
11.             $k_{max}^- := \arg \min\{k \in T \setminus S | d_k^- = d_{max}^-\}$ ;
12.             $S := S + k_{max}^-$ ;
13.        end
14.    end;
15. end;
16. end.

```

The PP algorithm is called repeatedly by the DCA-MSF to generate a solution to the MSF instance within the prescribed accuracy level α . The pseudocode for DCA-MSF is given below. As in the case of ATSP, a good problem-specific upper bound will improve the performance of the algorithm.

Algorithm DCA-MSF($[S, T], \alpha$)

Output: $x^\alpha \in [S, T]$ such that $z(x^\alpha) \geq z^*[S, T] - \alpha$.

Code:

```

1. begin
2.      $[S, T] := \text{PP}([S, T])$ ;
3.     if  $T = S$  return  $S$ ;
4.      $d_{max}^+ := \max\{d_k^+ | k \in T \setminus S\}$ ;
5.      $d_{max}^- := \max\{d_k^- | k \in T \setminus S\}$ ;
6.     if  $d_{max}^+ \leq d_{max}^-$  then begin

```

```

7.           if  $d_{max}^+ \leq \alpha$  then begin
8.                $k_{max}^+ := \arg \min\{k \in T \setminus S | d_k^+ = d_{max}^+\};$ 
9.               return DCA-MSF([S, T - k_{max}^+], \alpha - d_{max}^+) (* Correction *)
10.          end;
11.         else begin                                     (* Branch *)
12.              $x_1 := \text{DCA-MSF}([S + k_{max}^+, T], \alpha);$ 
13.              $x_2 := \text{DCA-MSF}([S, T - k_{max}^+], \alpha);$ 
14.             if  $z(x_1) \geq z(x_2)$  return  $x_1$ 
15.             else return  $x_2;$ 
16.         end;
17.     end
18.     else begin
19.         if  $d_{max}^- \leq \alpha$  then begin
20.              $k_{max}^- := \arg \min\{k \in T \setminus S | d_k^- = d_{max}^-\};$ 
21.             return DCA-MSF([S + k_{max}^-, T], \alpha - d_{max}^-) (* Correction*)
22.         end;
23.         else begin                                     (* Branch *)
24.              $x_1 := \text{DCA-MSF}([S + k_{max}^-, T], \alpha);$ 
25.              $x_2 := \text{DCA-MSF}([S, T - k_{max}^-], \alpha);$ 
26.             if  $z(x_1) \geq z(x_2)$  return  $x_1$ 
27.             else return  $x_2;$ 
28.         end;
29.     end;
30. end.

```

A version of this DCA is implemented in a toolbox for use in MATLAB or Octave (see [57]) with a correct note that “The Data Correcting Algorithm for maximizing general (not necessarily nondecreasing) submodular functions.”

5.2 A Data-Correcting Algorithm Based on Multi-level Search

The preservation rules mentioned in Corollary 3 look at a level which is exactly one level deeper in the Hasse diagram than the levels of S and T . However, instead of looking one level deep, it is possible to look r levels deep in order to determine whether an element can be included or excluded. Let

$$\begin{aligned} M_r^+[S, T] &= \{I \in [S, T] | |I \setminus S| \leq r\}, \\ M_r^-[S, T] &= \{I \in [S, T] | |T \setminus I| \leq r\}. \end{aligned}$$

The set $M_r^+[S, T]$ is a collection of all sets representing solutions containing more elements than S and which are no more than r levels deeper than S in the Hasse diagram. Similarly, the set $M_r^-[S, T]$ is a collection of all sets representing solutions containing less elements than T and which are no more than r levels deeper than T in the Hasse diagram. Let us further define the collections of sets.

$$N_r^+[S, T] = M_r^+[S, T] \setminus M_{r-1}^+[S, T],$$

$$N_r^-[S, T] = M_r^-[S, T] \setminus M_{r-1}^-[S, T].$$

The sets $N_r^+[S, T]$ and $N_r^-[S, T]$ are the collection of sets which are located exactly r levels above S and below T in the Hasse diagram, respectively.

Further, let $v_r^+[S, T] = \max\{z(I) | I \in M_r^+[S, T]\}$, $v_r^-[S, T] = \max\{z(I) | I \in M_r^-[S, T]\}$, $w_{rk}^+[S, T] = \max\{d_k^+(I) | I \in N_r^+[S+k, T]\}$ and $w_{rk}^-[S, T] = \max\{d_k^-(I) | I \in N_r^-[S, T-k]\}$.

Theorem 6 Let z be a submodular function on $[S, T] \subseteq [\emptyset, N]$ with $k \in T \setminus S$, and let r be a positive integer. Then the following assertions hold:

1. If $|N_r^+[S+k, T]| > 0$, then $z^*[S+k, T] - \max\{z^*[S, T-k], v_r^+[S, T]\} \leq \max\{w_{rk}^+[S, T], 0\}$.
2. If $|N_r^-[S, T-k]| > 0$, then $z^*[S, T-k] - \max\{z^*[S+k, T], v_r^-[S, T]\} \leq \max\{w_{rk}^-[S, T], 0\}$.

Proof Only part 1 is proved since the proof of part 2 is similar. The partition of interval $[S, T]$ may be represented as follows:

$$[S, T] = M_r^+[S, T] \cup \bigcup_{I \in N_r^+[S, T]} [I, T].$$

Using this representation on the interval $[S+k, T]$, we have $z^*[S+k, T] = \max\{v_r^+[S+k, T], \max\{z^*[I+k, T] | I \in N_r^+[S, T]\}\}$. Let $I(k) \in \arg \max\{z^*[I+k, T] | I \in N_r^+[S, T]\}$.

There are two cases to consider: $z^*[I(k)+k, T] \geq v_r^+[S+k, T]$ and $z^*[I(k)+k, T] < v_r^+[S+k, T]$.

In the first case, $z^*[S+k, T] = z^*[I(k)+k, T]$. For $I(k) \in N_r^+[S, T]$, [Theorem 5\(1\)](#) can be applied on the interval $[I(k), T]$ to obtain $z^*[I(k)+k, T] - z^*[I(k), T-k] \leq d_k^+(I(k))$, so that in this case, $z^*[S+k, T] - z^*[I(k), T-k] \leq d_k^+(I(k))$. Note that for $[I(k), T-k] \subseteq [S, T-k]$, we have $z^*[S, T-k] \geq z^*[I(k), T-k]$, which implies that $z^*[S+k, T] - z^*[S, T-k] \leq d_k^+(I(k))$. Adding two maximum operations, we get

$$z^*[S+k, T] - \max\{z^*[S, T-k], v_r^+[S+k, T]\} \leq \max\{d_k^+(I(k)), 0\}.$$

Since $w_{rk}^+[S, T]$ is the maximum of $d_k^+(I)$ for $I \in N_r^+[S+k, T]$, the required result is obtained.

In the second case, $z^*[S+k, T] = v_r^+[S+k, T]$ which implies that $z^*[S+k, T] - v_r^+[S+k, T] = 0$ or $z^*[S+k, T] - \max\{z^*[S, T-k], v_r^+[S+k, T]\} \leq 0$. Adding a maximum operation with $w_{rk}^+[S, T]$ completes the proof. ■

Corollary 4 (Preservation rules of order r) Let z be a submodular function on $[S, T] \subseteq [\emptyset, N]$ and let $k \in T \setminus S$. Then the following assertions hold:

1. *First Preservation Rule of Order r:* If $w_{rk}^+[S, T] \leq 0$, then $z^*[S, T] = \max\{z^*[S, T - k], v_r^+[S + k, T]\} \geq z^*[S + k, T]$.
2. *Second Preservation Rule of Order r:* If $w_{rk}^-[S, T] \leq 0$, then $z^*[S, T] = \max\{z^*[S + k, T], v_r^-[S, T - k]\} \geq z^*[S, T - k]$.

Notice that when [Corollary 3](#) is applied to an interval, a reduced interval is got; however, when [Corollary 4](#) is applied, a value v_r is got in addition to a reduced interval.

It can be proved by induction that the portion of the Hasse diagram eliminated by preservation rules of order $r - 1$ while searching for a maximum of the submodular function will certainly be eliminated by preservation rules of order r . In this sense, preservation rules of order r are not weaker than preservation rules of order $r - 1$ (for a detailed proof for the result that preservation rules of order 1 are not weaker than preservation rules of order 0, refer to Goldengorin [\[38\]](#)).

In order to apply [Corollary 4](#), functions that compute the value of $w_{rk}^+[S, T]$, $w_{rk}^-[S, T]$, $v_r^+[S + k, T]$, and $v_r^-[S, T - k]$ are needed. To that end, two recursive functions are defined, *PPArplus* to compute $w_{rk}^+[S, T]$ and $v_r^+[S + k, T]$ and *PPArminus* to compute $w_{rk}^-[S, T]$ and $v_r^-[S, T - k]$. The pseudocode for *PPArplus* is shown below. Its output is a three-tuple, containing, in order, $w_{rk}^+[S, T]$ and $v_r^+[S + k, T]$, and a solution in $M_r^+[S + k, T]$ whose objective function value is $v_r^+[S + k, T]$. The pseudocode for *PPArminus* can be constructed in a similar manner.

```

function PParplus([S, T], r, k)
1. begin
2.      $w := -\infty$ ;
3.      $v := -\infty$ ;
4.      $vset := \emptyset$ ;
5.      $(w, v, vset) := \text{IntPPArPlus}([S + k, T], r, w, v, vset)$ ;
6.     return  $(w, v, vset)$ ;
7. end;

function IntPPArplus([X, Y], r, w, v, vset)
1. begin
2.     for each  $t \in Y \setminus X$  do begin
3.         if  $z(X + t) > v$  then begin
4.              $v := z(X + t)$ ;
5.              $vset := (X + t)$ ;
6.         end;
7.         if  $d_t^+(X + t) > w$  then  $w := d_t^+(X + t)$ ;
8.         if  $d_t^+(X + t) > 0$  and  $r > 1$  then
9.              $(w, v, vset) := \text{IntPPArPlus}([X + t, Y], r - 1, w, v, vset)$ ;
10.    end;
11.    return  $(w, v, vset)$ ;
12. end;
```

Note that *PPArplus* and *PPArminus* are both $\mathcal{O}(n \binom{n}{r})$, that is, polynomial for a fixed value of r . However, in general, they are not polynomial in r .

PPArplus and *PPArminus* are used to describe the preliminary preservation algorithm of order r (*PPAr*(r)). Given a submodular function z on $[X, Y] \subseteq [\emptyset, N]$, *PPAr* outputs a subinterval $[S, T]$ of $[X, Y]$ and a set B such that $z^*[X, Y] = \max\{z^*[S, T], z(B)\}$ and $\min\{w_{rk}^+[S, T], w_{rk}^-[S, T]\} > 0$ for all $k \in T \setminus S$. At iteration i of the algorithm, when the search has been restricted to $[S_i, T_i]$, the algorithm starts by applying the PP algorithm (from Goldengorin et al. [44]) to this interval and reduces it to $[S'_i, T'_i]$. If $|T'_i \setminus S'_i| > 0$, an element $k \in T'_i \setminus S'_i$ is chosen, and the algorithm tries to apply [Corollary 4\(1\)](#) to decide whether it belongs to the set that maximizes $z(\cdot)$ over $[S_i, T_i]$ or not. If it does, then the search is restricted to the interval $[S'_i + k, T'_i]$. Otherwise, the search tries to apply [Corollary 4\(2\)](#) to decide whether the interval can be reduced to $[S'_i, T'_i - k]$.

Algorithm PPAr($[S, T], r$)

Output: $x^\alpha \in [S, T]$ such that $z(x^\alpha) \geq z^*[S, T] - \alpha$.

Code:

```

1. begin
2.    $X := S, Y := T; B := \arg \max\{z(S), z(T)\};$ 
3.   while  $Y \neq X$  do begin
4.      $[S_i, T_i] := \text{PP}([X, Y]);$ 
5.      $d^+ := \max\{d_k^+(S) | k \in T \setminus S\};$ 
6.      $d^- := \max\{d_k^-(T) | k \in T \setminus S\};$ 
7.     if  $d^+ > d^-$  then begin
8.        $k := \arg \max\{d_t^+(S) | t \in T \setminus S\};$ 
9.        $(w, v, vset) := \text{PPArplus}([S_i, T_i], r, k);$ 
10.      if  $v > z(B)$  then  $B := vset;$ 
11.      if  $w \leq 0$  then  $Y := T_i - k;$ 
12.      else return  $([S_i, T_i], B);$ 
13.    else begin
14.       $k := \arg \max\{d_t^-(S) | t \in T \setminus S\};$ 
15.       $(w, v, vset) := \text{PPArminus}([S_i, T_i], r, k);$ 
16.      if  $v > z(B)$  then  $B := vset;$ 
17.      if  $w \leq 0$  then  $X := S_i + k;$ 
18.      else return  $([S_i, T_i], \{w_{ri}^+[S_i, T_i]\}, \{w_{ri}^-[S_i, T_i]\}, B);$ 
19.    end;
20.  end;
21. end.
```

It is clear that if $r = |T \setminus S|$, then *PPAr* will always find an optimal solution to the given problem. However, *PPAr* is not a polynomial in r , and so *PPAr* with a large r is not practically useful.

PPAr can be embedded in a branch-and-bound framework to describe DCA-MSFr, a data-correcting algorithm based on *PPAr*. It is similar to the DCA-MSF proposed in Goldengorin et al. [44]. For DCA-MSFr, a submodular function z to be maximized over an interval $[S, T]$ and an accuracy parameter α are given, and it is needed to find a solution such that the difference between the objective function values of the solution output by DCA-MSFr and the optimal solution will not exceed α .

Notice that for a submodular function z , PPar with a fixed r may terminate with $T \neq S$ and $\min\{w_{ri}^+[S, T], w_{ri}^-[S, T] \mid i \in T \setminus S\} = \omega > 0$. The basic idea behind DCA-MSFr is that if this situation occurs, then the data of the current problem is corrected in such a way that ω is nonpositive for the corrected function and PPar can continue. Moreover, each correction of z needs to be carried out in such a way that the corrected function remains submodular. The attempted correction is carried out implicitly, in a manner similar to the one in Goldengorin et al. [44] but using Corollary 4 instead of Corollary 3. Thus, for example, if $w_{rj}^+[S, T] = \omega \leq \alpha$, then PPar is allowed to continue, but the accuracy parameter is reduced to $\alpha - \omega$.

If such a correction is not possible, that is, if ω exceeds the accuracy parameter, then it is branched on a variable $k \in \arg \max\{d_i^+(S), d_i^-(T) | i \in T \setminus S\}$ to partition the interval $[S, T]$ into two intervals $[S + k, T]$ and $[S, T - k]$. This branching rule was proposed in Goldengorin [33]. An upper bound for the value of z for each of the two intervals is then computed to see if either of the two can be pruned. The use of an upper bound due to Khachaturov [53] is described as follows. Let $d^+(S, T) = \{d_i^+(S) | d_i^+(S) > 0, i \in T \setminus S\}$ and $d^-(S, T) = \{d_i^-(T) | d_i^-(T) > 0, i \in T \setminus S\}$. Further, let $d^+[i]$ (respectively $d^-[i]$) denote the i -th largest element of $d^+(S, T)$ (respectively $d^-(S, T)$). Then ub described below is an upper bound to $z^*[S, T]$.

$$ub[S, T] = \max \left\{ \min_{i=1, \dots, |T \setminus S|} \left\{ z(S) + \sum_{j=1}^i d^+[j], z(T) + \sum_{j=1}^i d^-[j] \right\} \right\}.$$

The following pseudocode describes DCA-MSFr formally.

Algorithm DCA-MSFr($[S, T]$, α , r)

- ```

1. begin
2. $best_set := \arg \max\{z(S), z(T)\};$
3. $best := z(best_set);$
4. $(best_set, best) := \text{IntDCA-MSFr}([S, T], \alpha, r, best_set, best);$
5. return $best_set;$
6. end.

```

**function** IntDCA-MSFr( $[S, T]$ ,  $\alpha, r, best\_set, best$ )

- ```

1. begin
2.    $([S, T], \{w_{rk}^+\}, \{w_{rk}^-\}, B) := \text{PPAr}([S, T], r);$ 
3.   if  $z(B) > best$  then begin
4.      $best\_set := B;$ 
5.      $best := z(B);$ 
6.   end;
7.   if  $S = T$  return ( $best\_set, best$ );
8.    $\omega^+ := \max\{w_{rk}^+[S, T] | k \in T \setminus S\};$ 
9.   choose  $j^+$  from  $\min\{k | w_{rk}^+[S, T] = \omega^+, k \in T \setminus S\};$ 
10.   $\omega^- := \max\{w_{rk}^-[S, T] | k \in T \setminus S\};$ 
11.  choose  $j^-$  from  $j^- := \min\{k | w_{rk}^-[S, T] = \omega^-, k \in T \setminus S\};$ 
12.  if  $\omega^+ \leq \alpha$  then (* Correction *)
13.    IntDCA-MSFr( $[S + j^+, T], \alpha - \omega^+, r, best\_set, best$ );

```

```

14.      else if  $\omega^- \leq \alpha$  then                                (* Correction *)
15.          IntDCA-MSFr([S, T - j^-],  $\alpha - \omega^-$ , r, best_set, best);
16.      else begin                                         (* Branch [S, T] → [S + k, T], [S, T - k] *)
17.          choose k from arg max{ $d_i^+(S), d_i^-(T)$  |  $i \in T \setminus S$ };
18.          if ub[S + k, T] > best then begin                  (* Bound *)
19.              ( $bs_1, b_1$ ) := IntDCA-MSFr([S + k, T],  $\alpha, r, best\_set, best$ );
20.              if  $b_1 > best$  then begin
21.                  best_set :=  $bs_1$ ;
22.                  best :=  $b_1$ ;
23.              end;
24.          end;
25.          if ub[S, T - k] > best then begin                  (* Bound *)
26.              ( $bs_2, b_2$ ) := IntDCA-MSFr([S, T - k],  $\alpha, r, best\_set, best$ );
27.              if  $b_2 > best$  then begin
28.                  best_set :=  $bs_2$ ;
29.                  best :=  $b_2$ ;
30.              end;
31.          end;
32.      end;
33.  end;

```

5.3 Computational Experience with Quadratic Cost Partition Instances

In this section, the computational experience with DCA-MSFr is reported. The quadratic cost partition problem is chosen as a test bed. The quadratic cost partition (QCP) problem can be described as follows (see, e.g., [59]). Given nonnegative real numbers q_{ij} and real numbers p_i with $i, j \in N = \{1, 2, \dots, n\}$, the QCP is the problem of finding a subset $S \subseteq N$ such that the function $z(S) = \sum_{j \in S} p_j - \frac{1}{2} \sum_{i,j \in S} q_{ij}$ will be maximized. The density d of a QCP instance is the ratio of the number of finite q_{ij} values to $n(n - 1)/2$ and is expressed as a percentage. It is proved in Theorem 2.2 of Lee et al. [59] that $z(\cdot)$ is submodular.

In Goldengorin et al. [44], computational experiments with QCP have been restricted to instances of size not more than 80 because instances of that size have been considered in Lee et al. [59]. For these instances, it was shown that the average calculation times grow exponentially when the number of vertices increases and reduces exponentially with increasing density.

Herein, the performance of DCA-MSFr on QCP instances of varying size and densities is reported. The maximum time allowed for an instance is 10 CPU minutes on a personal computer running on a 300-MHz Pentium processor with 64-MB memory. The algorithms have been implemented in Delphi 3.

The instances on which the described algorithms are tested are statistically similar to the instances in Lee et al. [59]. Instances of size n and density $d\%$ are generated as follows. A graph with n nodes and $\frac{d}{100} \times \frac{n(n-1)}{2}$ random edges is generated. The edges are assigned costs from a $\mathcal{U}[1, 100]$ distribution. n edges connect each node to itself, and these edges are assigned costs from a $\mathcal{U}[0, 100]$ distribution. The distance matrix of this graph forms a QCP instance.

Fig. 8 Average number of subproblems generated against r for QCP instances with $n = 100$ and varying d values

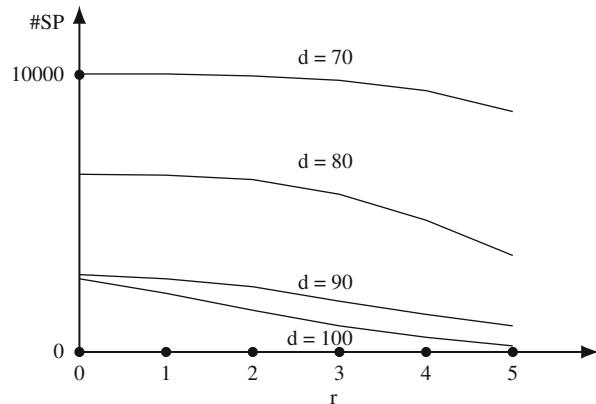
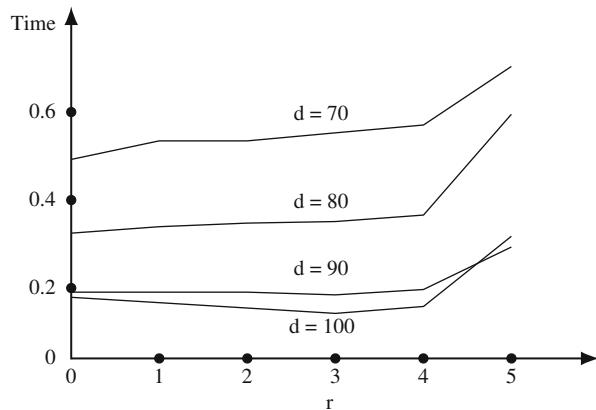


Fig. 9 Average execution time (in seconds) against r for QCP instances with $n = 100$ and varying d values



First, the effect of varying the value of r on the performance of DCA-MSFr(r) is reported. It is intuitive that DCA-MSFr(r) will require more execution times when the value of r increases. The computation experience with 10 QCP instances of size 100 and different densities is shown in Figs. 8–10. Figure 8 shows the number of subproblems generated when r is increased from a value of 0 (i.e., DCA-MSF) to 5. As is intuitive, the number of subproblems is reduced with increasing r for all density values. Figure 9 shows the execution times of DCA-MSFr(r) with varying d and r values. Recall that when the value of r increases, the time required at each subproblem increases, since PPAR requires more computations for larger r values. The decrease in the number of subproblems approximately balances the increase in the time at each subproblem for r values in the range 0 through 4. When $r = 5$, the computation times for DCA-MSFr(r) increase significantly for all density values. From Fig. 9, it seems that for dense graphs, r values of 3 or 4 are most favorable. This effect also holds for larger instances – Fig. 10 shows the execution times for instances with $n = 200$ and $d = 100$.

Fig. 10 Average execution time (in seconds) against r for QCP instances with $n = 200$ and $d = 100$

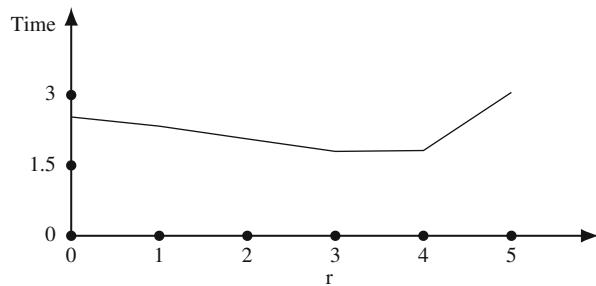


Table 15 Average execution times on QCP instances when $\alpha = 0\%$

Density(%)	Algorithm	Problem size (n)				
		100	200	300	400	500
100	DCA-MSFr(3)	0.098	2.63	18.316	85.827	229.408
	DCA-MSF	0.831	64.372	340.681	1,894.811	*
90	DCA-MSFr(3)	0.138	3.824	37.931	173.063	624.925
	DCA-MSF	1.027	78.614	794.037	3,505.892	*
80	DCA-MSFr(3)	0.28	9.506	98.69	679.914	*
	DCA-MSF	1.784	217.898	2,681.973	*	*
70	DCA-MSFr(3)	0.393	17.643	413.585	*	*
	DCA-MSF	2.498	631.492	*	*	*
60	DCA-MSFr(3)	0.731	86.33	*	*	*
	DCA-MSF	3.509	1,414.103	*	*	*
50	DCA-MSFr(3)	1.752	345.723	*	*	*
	DCA-MSF	9.382	*	*	*	*
40	DCA-MSFr(3)	3.457	*	*	*	*
	DCA-MSF	17.245	*	*	*	*
30	DCA-MSFr(3)	11.032	*	*	*	*
	DCA-MSF	48.013	*	*	*	*
20	DCA-MSFr(3)	47.162	*	*	*	*
	DCA-MSF	195.82	*	*	*	*
10	DCA-MSFr(3)	70.081	*	*	*	*
	DCA-MSF	446.293	*	*	*	*

Next, the results of the experiments to solve large-sized QCP instances with DCA-MSFr(r) are reported. Using the results obtained from the previous part of the study, DCA-MSFr(3) was chosen to be used as the algorithm of choice. For consideration, instances of the QCP with size n ranging from 100 to 500 and densities varying between 10 and 100 % have been chosen. These instances were tried to be solved exactly ($\alpha_0 = 0\%$), and with a prescribed accuracy $\alpha_0 = 5\%$ within 10 min. Tables 15 and 16 contain the average execution times in seconds for exact and approximate solutions with DCA-MSFr(3) and DCA-MSF. The entries marked “*” could not be solved within 10 min. From the table, it is possible to note that the execution times increase exponentially with increasing problem size

Table 16 Average execution times on QCP instances when $\alpha = 5\%$

Density(%)	Algorithm	Problem size (n)				
		100	200	300	400	500
100	DCA-MSFr(3)	0.094	2.444	17.179	85.096	222.883
	DCA-MSF	0.752	38.351	229.396	1,162.396	*
90	DCA-MSFr(3)	0.118	3.607	34.972	166.996	608.755
	DCA-MSF	0.916	49.926	583.754	1,996.544	*
80	DCA-MSFr(3)	0.228	8.186	89.685	580.789	*
	DCA-MSF	1.108	162.455	1,875.603	3,604.715	*
70	DCA-MSFr(3)	0.304	15.693	364.48	*	*
	DCA-MSF	1.593	376.629	3,165.384	*	*
60	DCA-MSFr(3)	0.517	72.931	*	*	*
	DCA-MSF	2.874	895.426	*	*	*
50	DCA-MSFr(3)	1.298	267.445	*	*	*
	DCA-MSF	5.931	1,937.673	*	*	*
40	DCA-MSFr(3)	2.179	*	*	*	*
	DCA-MSF	10.327	*	*	*	*
30	DCA-MSFr(3)	5.88	*	*	*	*
	DCA-MSF	22.209	*	*	*	*
20	DCA-MSFr(3)	17.477	*	*	*	*
	DCA-MSF	74.841	*	*	*	*
10	DCA-MSFr(3)	12.196	*	*	*	*
	DCA-MSF	95.122	*	*	*	*

and decreasing problem densities. Therefore, QCP instances with 500 vertices and densities between 90 and 100 % are the largest instances which can be solved by the DCA-MSFr(3) within 10 min on a standard personal computer. It is also seen that on average DCA-MSFr(3) takes roughly 11 % of the time taken by DCA-MSF for the exact solutions and roughly 13 % of the time taken by DCA-MSF for the approximate solutions. The reduction in time is more pronounced for problems with higher size and higher densities.

6 Conclusion

This chapter is concluded by the following fragment cited from Benati [10] “In (Goldengorin et al., [44]) a new branch and bound scheme to solve the submodular function maximization is proposed. It outperforms previous algorithms, such as branch and cut methods, in the max-cut problem solution. The algorithm is called data correcting method (DC-method), because it tries to obtain a polynomially solvable instance of the general problem by modifying the original data, and then the upper bound of the objective function of the original problem is calculated by the corrected instance. This method was developed by Russian scientists in the 70s. In this paper, the DC method is applied to the UCFLP with probabilistic customers.”

The first paper with successful application of data-correcting approach to solve a real-world assortment problem is published in Goldengorin [31].

Acknowledgements This work was carried out at the Laboratory of Algorithms and Technologies for Networks Analysis, National Research University Higher School of Economics, and supported by the Ministry of Education and Science of Russian Federation, Grant No. 11.G34.31.0057, and by the Scientific Foundation of the National Research University Higher School of Economics, project “Teachers–Students” No. 11-04-0008.

Cross-References

- ▶ [Algorithmic Aspects of Domination in Graphs](#)
- ▶ [Algorithms for the Satisfiability Problem](#)
- ▶ [Binary Unconstrained Quadratic Optimization Problem](#)
- ▶ [Combinatorial Optimization Algorithms](#)
- ▶ [Combinatorial Optimization in Data Mining](#)
- ▶ [Combinatorial Optimization Techniques for Network-Based Data Mining](#)
- ▶ [Combinatorial Optimization in Transportation and Logistics Networks](#)
- ▶ [Coverage Problems in Sensor Networks](#)
- ▶ [Graph Searching and Related Problems](#)
- ▶ [Graph Theoretic Clique Relaxations and Applications](#)
- ▶ [Max-Coloring](#)
- ▶ [Network Optimization](#)
- ▶ [On Coloring Problems](#)
- ▶ [Optimal Partitions](#)
- ▶ [Optimization Problems in Data Broadcasting](#)
- ▶ [Optimization Problems in Online Social Networks](#)
- ▶ [Optimizing Data Collection Capacity in Wireless Networks](#)
- ▶ [Packing Circles in Circles and Applications](#)
- ▶ [Quadratic Assignment Problems](#)
- ▶ [Reformulation-Linearization Techniques for Discrete Optimization Problems](#)
- ▶ [Resource Allocation Problems](#)
- ▶ [Rollout Algorithms for Discrete Optimization: A Survey](#)
- ▶ [Social Structure Detection](#)
- ▶ [Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work](#)
- ▶ [Steiner Minimum Trees in \$E^3\$: Theory, Algorithms, and Applications](#)
- ▶ [Variations of Dominating Set Problem](#)

Recommended Reading

1. B.F. AlBdaiwi, B. Goldengorin, G. Sierksma, Equivalent instances of the simple plant location problem. *Comput. Math. Appl.* **57**, 812–820 (2009)
2. B.F. AlBdaiwi, D. Ghosh, B. Goldengorin, Data aggregation for p-median problems. *J. Comb. Optimi.* **21**, 348–363 (2011)

3. P. Avella, A. Sforza, Logical reduction tests for the p -median problem. *Ann. Oper. Res.* **86**, 105–115 (1999)
4. P. Avella, A. Sassano, I. Vasil'ev, Computational study of large-scale p -median problems. *Math. Program. Ser. A* **109**, 89–114 (2007)
5. E. Balas, P. Toth, Branch and bound methods, Chapter 10 in Lawler et al. [58]
6. J.E. Beasley, Lagrangian heuristics for location problems, *Eur. J. Oper. Res.* **65**, 383–399 (1993)
7. J.E. Beasley, OR-Library, Available at the web address, <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/pmedinfo.html>
8. A.S. Belenky (ed.), Mathematical modeling of voting systems and elections: theory and applications. *Math. Comput. Model.* **48**(9–10), 1295–1676 (2008)
9. C. Beltran, C. Tadonki, J.-PH. Vial, Solving the p -median problem with a semi-Lagrangian relaxation. *Comput. Optim. Appl.* **35**, 239–260 (2006)
10. S. Benati, An improved branch & bound method for the uncapacitated competitive location problem. *Ann. Oper. Res.* **122**, 43–58 (2003)
11. V.L. Beresnev, On a problem of mathematical standardization theory. *Upravljajemye Sistemy* **11**, 43–54 (1973) (in Russian)
12. O. Bilde, J. Krarup, Bestemmelse af optimal beliggenhed af produktionssteder, Research report, IMSOR (1967)
13. O. Bilde, J. Krarup, Sharp lower bounds and efficient algorithms for the simple plant location problem. *Ann. Discret. Math.* **1**, 79–97 (1977)
14. E. Boros, P.L. Hammer, Pseudo-Boolean optimization. *Discret. Appl. Math.* **123**, 155–225 (2002)
15. O. Briant, D. Naddef, The optimal diversity management problem. *Oper. Res.* **52**, 515–526 (2004)
16. M.J. Brusco, H.-F. Köhn, Optimal partitioning of a data set based on the p -median problem. *Psychometrika* **73**(1), 89–105 (2008)
17. N. Christofides, *Graph Theory: An Algorithmic Approach* (Academic, London, 1975)
18. R.L. Church, COBRA: a new formulation of the classic p-median location problem. *Ann. Oper. Res.* **122**, 103–120 (2003)
19. R.L. Church, BEAMR: an exact and approximate model for the p-median problem. *Comput. Oper. Res.* **35**, 417–426 (2008)
20. G. Cornuejols, G. Nemhauser, L.A. Wolsey, A canonical representation of simple plant location problems and its applications. *SIAM J. Matrix Anal. Appl. (SIMAX)* **1**(3), 261–272 (1980)
21. G. Cornuejols, G.L. Nemhauser, L.A. Wolsey, The uncapacitated facility location problem, in *Discrete Location Theory*, ed. by P.B. Mirchandani, R.L. Francis (Wiley-Interscience, New York, 1990), pp. 119–171
22. H.A. Eiselt, V. Marianov (eds.), *Foundations of Location Analysis* (Springer, New York/London, 2011)
23. S. Elloumi, A tighter formulation of the p-median problem. *J. Comb. Optim.* **19**, 69–83 (2010)
24. D. Erlenkotter, A dual-based procedure for uncapacitated facility location. *Oper. Res.* **26**, 992–1009 (1978)
25. R.D. Galvão, L.A. Raggi, A method for solving to optimality uncapacitated location problems. *Ann. Oper. Res.* **18**, 225–244 (1989)
26. M.R. Garey, D.S. Johnson, *Computers and Intractability* (Freeman, San Francisco, 1979)
27. D. Ghosh, B. Goldengorin, G. Sierksma, Data correcting algorithms in combinatorial optimization, in *Handbook of Combinatorial Optimization*, vol. 5, ed. by D.-Z. Du, P.M. Pardalos (Springer, Berlin, 2005), pp. 1–53
28. D. Ghosh, B. Goldengorin, G. Sierksma, Data correcting: a methodology for obtaining near-optimal solutions, in *Operations Research with Economic and Industrial Applications: Emerging Trends*, ed. by S.R. Mohan, S.K. Neogy (Anamaya Publishers, New Delhi, 2005), pp. 119–127
29. P.C. Gilmore, E.L. Lawler, D.B. Shmoys, Well-solved special cases, Chapter 4 in Lawler et al. [58]

30. B.L. Golden, S. Raghavan, E.A. Wasil (eds.), *The Vehicle Routing Problem: Latest Advances and New Challenges* (Springer, New York, 2010)
31. B.I. Goldengorin, The design of optimal assortment for the vacuum diffusion welding sets. Standart i Kachestvo **2**, pp. 19–21 (1975) (in Russian)
32. B. Goldengorin, Methods of solving multidimensional unification problems. Upravljaemye Sistemy **16**, 63–72 (1977) (in Russian)
33. B. Goldengorin, A correcting algorithm for solving some discrete optimization problems. Sov. Math. Dokl. **27**, 620–623 (1983)
34. B. Goldengorin, A correcting algorithm for solving allocation type problems. Autom. Remote Control **45**, 590–598 (1984)
35. B. Goldengorin, Correcting algorithms for solving multivariate unification problems. Sov. J. Comput. Syst. Sci. **1**, 99–103 (1985)
36. B. Goldengorin, On the exact solution of problems of unification by correcting algorithms. Doklady Akademii Nauk, SSSR **294**, 803–807 (1987)
37. B. Goldengorin, *Requirements of Standards: Optimization Models and Algorithms* (Russian Operations Research, Hoogezand, 1995)
38. B. Goldengorin, *Data Correcting Algorithms in Combinatorial Optimization*, Ph.D. thesis, SOM Research Institute, University of Groningen, Groningen, The Netherlands, 2002
39. B. Goldengorin, Maximization of submodular functions: theory and enumeration algorithms. Eur. J. Oper. Res. **198**, 102–112 (2009)
40. B. Goldengorin, D. Ghosh, The multilevel search algorithm for the maximization of submodular functions applied to the quadratic cost partition problem. J. Glob. Optim. **32**, 65–82 (2005)
41. B. Goldengorin, D. Krushinsky, Complexity evaluation of benchmark instances for the p-median problem. Math. Comput. Model. **53**, 1719–1736 (2011)
42. B. Goldengorin, D. Krushinsky, A computational study of the Pseudo-Boolean approach to the p-median problem applied to cell formation, in *Network Optimization*. Lecture Notes in Computer Science, vol. 6701 (Springer, Berlin/Heidelberg, 2011), pp. 503–516
43. B. Goldengorin, D. Krushinsky, J. Slomp, Flexible PMP approach for large-size cell formation. Oper. Res. **60**, 1157–1166
44. B. Goldengorin, G. Sierksma, G.A. Tijssen, M. Tso, The data-correcting algorithm for minimization of supermodular functions. Manag. Sci. **45**, 1539–1551 (1999)
45. B. Goldengorin, D. Ghosh, G. Sierksma, Branch and peg algorithms for the simple plant location problem. Comput. Oper. Res. **30**, 967–981 (2003)
46. B. Goldengorin, G.A. Tijssen, D. Ghosh, G. Sierksma, Solving the simple plant location problem using a data correcting approach. J. Glob. Optim. **25**, 377–406 (2003)
47. G. Gutin, A.P. Punnen (eds.), *The Traveling Salesman Problem and its Variations* (Kluwer, Dordrecht, 2002)
48. S.L. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph. Oper. Res. **12**, 450–459 (1964)
49. S.L. Hakimi, Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Oper. Res. **13**, 462–475 (1965)
50. P.L. Hammer, Plant location – a Pseudo-Boolean approach. Isr. J. Technol. **6**, 330–332 (1968)
51. R.M. Karp, A patching algorithm for the nonsymmetric traveling salesman problem. SIAM J. Comput. **8**(4), 561–573 (1979)
52. V.R. Khachaturov, *Some Problems of the Consecutive Calculation Method and Its Applications to Location Problems*, Ph.D. thesis, Central Economics and Mathematics Institute, Russian Academy of Sciences, Moscow, 1968, (in Russian)
53. V.R. Khachaturov, *Mathematical Methods of Regional Programming* (Nauka, Moscow, 1989), (in Russian)
54. B.M. Khumawala, An efficient branch and bound algorithm for the warehouse location problem. Manag. Sci. **18**, B718–B731 (1975)
55. M. Körkel, On the exact solution of large-scale simple plant location problems. Eur. J. Oper. Res. **39**, 157–173 (1989)
56. Y.A. Koskosidis, W.B. Powell, Clustering algorithms for consolidation of customer orders into vehicle shipments. Transp. Res. **26B**, 365–379 (1992)

57. A. Krause, SFO: a toolbox for submodular function optimization. *J. Mach. Learn. Res.* **11**, 1141–1144 (2010)
58. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Wiley-Interscience, Chichester/New York, 1985)
59. H. Lee, G.L. Nemhauser, Y. Wang, Maximizing a submodular function by integer programming: polyhedral results for the quadratic case. *Eur. J. Oper. Res.* **94**, 154–166 (1996)
60. L. Lovasz, Submodular functions and convexity, in *Mathematical Programming: The State of the Art*, ed. by A. Bachem, M. Grötschel, B. Korte (Springer, Berlin, 1983), pp. 235–257
61. M. Minoux, Accelerated greedy algorithms for maximizing submodular set functions, in *Actes Congrès IFIP*, ed. by J. Stoer (Springer, Berlin, 1977), pp. 234–243
62. N. Mladenovic, J. Brimberg, P. Hansen, J.A. Moreno-Peréz, The p-median problem: a survey of metaheuristic approaches. *Eur. J. Oper. Res.* **179**, 927–939 (2007)
63. J.M. Mulvey, M.P. Beck, Solving capacitated clustering problems. *Eur. J. Oper. Res.* **18**, 339–348 (1984)
64. E.D. Nering, A.W. Tucker. *Linear Programs and Related Problems* (Academic, San Diego, 1993)
65. D.W. Pentico, The assortment problem: a survey. *Eur. J. Oper. Res.* **190**, 295–309 (2008)
66. H. Pirkul, Efficient algorithms for the capacitated concentrator location problem. *Comput. Oper. Res.* **14**(3), 197–208 (1987)
67. J. Reese, Solution methods for the p-median problem: an annotated bibliography. *Networks* **48**, 125–142 (2006)
68. G. Reinelt, TSPLIB 95 (1995), <http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html>
69. C.S. ReVelle, R. Swain, Central facilities location. *Geogr. Anal.* **2**, 30–42 (1970)
70. C.S. ReVelle, H.A. Eiselt, M.S. Daskin, A bibliography for some fundamental problem categories in discrete location science. *Eur. J. Oper. Res.* **184**, 817–848 (2008)
71. K.E. Rosing, C.S. ReVelle, H. Rosing-Vogelaar, The p-median and its linear programming relaxation: an approach to large problems. *J. Oper. Res. Soc.* **30**, 815–822 (1979)
72. E.L.F. Senne, L.A.N. Lorena, M.A. Pereira, A branch-and-price approach to p-median location problems. *Comput. Oper. Res.* **32**, 1655–1664 (2005)
73. TSP-library, Available at the web address <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>
74. L. Wolsey, Mixed integer programming, in *Wiley Encyclopedia of Computer Science and Engineering*, ed. by B. Wah (Wiley, Chichester, 2008)
75. Y. Won, K.C. Lee, Modified p-median approach for efficient GT cell formation. *Comput. Ind. Eng.* **46**, 495–510 (2004)

Dual Integrality in Combinatorial Optimization

Xujin Chen, Xiaodong Hu and Wenan Zang

Contents

1	Introduction	996
2	Preliminaries	998
2.1	Two Classical Theorems	998
2.2	Dual $1/d$ -Integrality	999
2.3	Hypergraphs	1001
2.4	Transformations Preserving Dual Integrality	1003
2.5	Graphs and Matroids	1006
3	Complexity	1008
3.1	Recognition of TDI Systems	1008
3.2	Maximum Feedback Vertex Set Packing in Tournaments	1010
4	A Unified Approach to Box-Mengerian Hypergraphs	1010
4.1	ESP Property	1010
4.2	Path Hypergraphs	1012
4.3	Cycle Hypergraphs	1013
4.4	Matroid Port Hypergraphs	1018
4.5	ESP Hypergraphs vs. Min–Max Theorems	1022
4.6	Blockers of ESP Hypergraphs	1023
5	Total Dual Integral Systems	1029
5.1	Edge Covers and Stars	1030
5.2	Feedback Vertex Sets and Cycles	1031
5.3	Stable Marriage Polytopes	1034
5.4	Weighted Restricted 2-Matching Polytopes	1035
5.5	Ordered Submodular Games	1037
5.6	Orientation Constrained Network Design	1039
5.7	Rooted k -Connections in Directed Graphs	1042
5.8	Cyclic Orders in Directed Graphs	1045

X. Chen (✉) • X. Hu

Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing,
People's Republic of China

e-mail: xchen@amss.ac.cn; xdhu@amss.ac.cn

W. Zang

Department of Mathematics, The University of Hong Kong, Hong Kong
e-mail: wzang@maths.hku.hk

6	The Dual Half-Integrality	1046
6.1	The Quasi-ESP Property	1047
6.2	Half-Integral Circuit Packing in Matroids	1052
7	Conclusion	1056
	Cross-References	1059
	Recommended Reading	1059

Abstract

The notion of total dual integrality and its variants are powerful tools to derive combinatorial min–max relation efficiently, yielding many fundamental results in combinatorial optimization. Following the pioneering work of Edmonds and Gales (A min–max relation for submodular functions on graphs, in *Annals of Discrete Mathematics*, vol. 1, North-Holland, Amsterdam, 1977, pp. 185–204), considerable research efforts have been devoted to the study of the dual integrality from theoretical and algorithmic points of view. This chapter reviews significant progresses that have been made since the publication of Schrijver’s celebrated monograph (*Combinatorial Optimization - Polyhedra and Efficiency*, Springer, Berlin, 2003), which contained a comprehensive and in-depth treatment of the state-of-the-art in dual integrality theory and its application, among many others.

1 Introduction

Many important combinatorial optimization problems, such as stable matching, facility location, and pattern recognition can be naturally formulated as integer linear programs, most of which are known to be NP-hard. One approach to getting around these problems is to consider corresponding linear programming (LP) relaxations and explore integrality properties satisfied by their constraints.

Definition 1 A rational system $A\mathbf{x} \geq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ of linear inequalities is *integral* or equivalently the polyhedron $P = \{\mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is *integral* if all extreme points of P are integral.

It is well known that polyhedron P is integral if and only if the minimum in the LP-duality equation

$$\min\{\mathbf{w}^T \mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\} = \max\{\mathbf{y}^T \mathbf{b} : \mathbf{y}^T A \leq \mathbf{w}^T, \mathbf{y} \geq \mathbf{0}\} \quad (1)$$

has an integral optimal solution, for every integral vector \mathbf{w} for which the optimum is finite. If, instead, the maximum in the equation enjoys this property, then the system $A\mathbf{x} \geq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ is associated with total dual integrality.

Definition 2 A rational system $Ax \geq b, x \geq \mathbf{0}$ of linear inequalities is *totally dual integral (TDI)* if the maximum in (1) has an integral solution y , for every integral vector w for which the maximum is finite. The system $Ax \geq b, x \geq \mathbf{0}$ is called *box-totally dual integral (box-TDI)* if $Ax \geq b, x \geq \mathbf{0}, u \geq x \geq l$ is TDI for all rational vectors u and l , where coordinates of u are allowed to be $+\infty$.

The model of TDI systems and box-TDI systems plays a crucial role in combinatorial optimization and serves as a general framework for establishing various min–max theorems because total dual integrality implies primal integrality (shown by Edmonds and Giles [41]) and box-total dual integrality implies total dual integrality (shown by Schrijver [101]).

Theorem 1 Let π denote the linear system $Ax \geq b, x \geq \mathbf{0}$. (i) If π is TDI and b is integral, then π is integral. (ii) If π is box-TDI, then π is TDI.

Under what conditions do these dual integrality and/or their variants hold? This question is of both great theoretical interest and practical value; it is also the major concern of polyhedral combinatorics (see, e.g., [102]). Many celebrated results and conjectures in combinatorial optimization can be rephrased by saying that certain systems are TDI or box-TDI. Since its introduction by Edmonds and Giles [41], extensive research on total dual integrality and its variants, referred to collectively in this chapter as *dual integrality*, has brought about a variety of exciting results, as well as challenges, on theory and applications. Many of the deepest results in this area come in “excluded minor” flavor, while their proofs often employ “decomposition” characterizations to decompose the combinatorial structures to building blocks and make use of “sum” operations to recover the whole pictures.

In 2003, Schrijver published a three-volume monograph [103] giving a comprehensive review of the state-of-the-art of those topics on dual integrality in combinatorial optimization. Since then ten years have passed, during which significant progresses have been made on both the theoretical side and the algorithmic side. The main purpose of this chapter is to give a brief overview of the recent progress on the dual integrality theory and its applications that were not included in [103], putting emphasis on the study of hypergraphs in the context of packing and covering.

Chapter Outline. Section 2 provides preliminaries for dual integrality, as well as graphs and matroids. Section 3 discusses complexity issues for recognizing (box)-TDI systems and packing feedback vertex sets. Section 4 presents a unified approach to establishing box-Mengerian hypergraphs and reviews box-TDI systems (not) falling within this general framework. Section 5 exhibits in separate subsections TDI systems established by publications over the last decade. Section 6 deals with dual half-integrality, which is a natural relaxation of dual integrality, and has interesting real-world applications. Section 7 concludes this chapter by commenting on some missing topics and pointing out several future research directions.

2 Preliminaries

The reader is assumed to be familiar with basic polyhedral theory; see Sect. 5 of [103] for an introduction to this subject. This chapter discusses mainly the (box-)total dual integrality for linear systems of form $Ax \geq b, x \geq 0$, in view that typical combinatorial optimization problems either minimize over this kind of systems or maximize over their duals (cf. the LP-duality equation (1)). Generalizing Definition 2, TDI and box-TDI systems are defined as follows.

Definition 3 A rational system $Ax \geq b$ is said to be *TDI* if for every integral vector w , the dual of minimizing $w^T x$ over $Ax \geq b$: $\max\{y^T b : y^T A = w^T, y \geq 0\}$ has an integral optimal solution y , provided the optimum is finite. A rational $Ax \geq b$ is *box-TDI* if $Ax \geq b, u \geq x \geq l$ is TDI for all rational vector u and l , where coordinates of u and l are allowed to be $+\infty$ or $-\infty$.

The present section is divided into five subsections. Section 2.1 presents two classical sufficient conditions, one of which is also necessary, for linear systems to be box-TDI. Section 2.2 generalizes the classical concepts of TDI and box-TDI systems to their fractional $\frac{1}{d}$ -counterparts. Implication relations among dual integral properties are also discussed. Section 2.3 rephrases dual integrality in the context of hypergraphs and clutters, defining the notion of (box-)Mengerian hypergraph which is naturally and closely related to min-max relation in combinatorial optimization. Section 2.4 discusses operations on linear systems that preserve (box-)total dual integrality. Section 2.5 gives preliminaries on graph theory and matroid theory.

2.1 Two Classical Theorems

Schrijver [100] analyzed proof techniques of a number of classical min–max theorems, such as the max-flow min-cut theorem, the Lucchesi–Younger theorem [82], and the Fulkerson’s optimal arborescence theorem [58], and observed that these proofs essentially proceed by showing that, first, the active constraints in the optimum of the LP relaxation of the problem in consideration can be chosen to be “nice,” say “cross-free” or “laminar”; second, these nice constraint sets are totally unimodular. Based on this observation, Schrijver proved the following general theorem (see Theorem 5.35 in [103]), which implies that the above-mentioned min–max theorems can all be further strengthened with box-TDI properties:

Theorem 2 *Let $Ax \geq b$ be a linear system. Suppose that for any rational vector c , the program $\min\{c^T x : Ax \geq b\}$ has (if finite) an optimal dual solution y such that the rows of A corresponding to positive components of y form a totally unimodular submatrix of A . Then $Ax \geq b$ is box-TDI.*

Let \mathbb{R} , \mathbb{Q} , and \mathbb{Z} denote the sets of real numbers, rational numbers, and integers, respectively; let \mathbb{R}_+ , \mathbb{Q}_+ , and \mathbb{Z}_+ denote the sets of nonnegative numbers in the

corresponding sets. For any two sets Ω and K , where Ω is always a set of numbers and K is always finite, symbol Ω^K denotes the set of vectors $\mathbf{x} = (x(k) : k \in K)$ whose coordinates are members of Ω . Given $\mathbf{x} \in \Omega^K$, the vectors $(\lfloor x_k \rfloor : k \in K)$ and $(\lceil x_k \rceil : k \in K)$ are written as $\lfloor \mathbf{x} \rfloor$ and $\lceil \mathbf{x} \rceil$, respectively. The next is a characterization of box-TDI systems due to Cook [25] (cf. Corollary 22.9a of [101]).

Theorem 3 *A rational system $A\mathbf{x} \geq \mathbf{b}$ defined on \mathbb{R}^n is box-TDI if and only if it is TDI and for every $\mathbf{w} \in \mathbb{Q}^n$ there exists $\tilde{\mathbf{w}} \in \mathbb{Z}^n$ such that $\lfloor \mathbf{w} \rfloor \leq \tilde{\mathbf{w}} \leq \lceil \mathbf{w} \rceil$ and such that each optimal solution of $\min\{\mathbf{w}^\top \mathbf{x} : A\mathbf{x} \geq \mathbf{b}\}$ is also an optimal solution of $\min\{\tilde{\mathbf{w}}^\top \mathbf{x} : A\mathbf{x} \geq \mathbf{b}\}$.*

Almost all known box-TDI systems can be verified via totally unimodular matrices (cf. [Theorem 2](#)) or the ESP property to be discussed in [Sect. 4](#) (cf. [Theorem 14](#)), but the box-TDI system associated with 2-edge-connected spanning subgraphs [19] is an exception (cf. [Theorems 45](#) and [46](#)), where the proof relied heavily on the above Cook's Theorem ([Theorem 3](#)).

2.2 Dual $1/d$ -Integrality

In this chapter, symbol “ d ” is reserved for representing a positive integer; adjectives “ $\frac{1}{d}$ -” and “half-” are used interchangeably. A real vector \mathbf{x} is $\frac{1}{d}$ -integral if all coordinates of $d\mathbf{x}$ are integers. This subsection discusses briefly moderate relaxations on primal and dual integrality (cf. [Definitions 1](#) and [2](#)), which give “primal” (box-) $\frac{1}{d}$ -integrality (see [Definitions 4](#) and [6](#)), and (box-)total dual $\frac{1}{d}$ -integrality (see [Definition 5](#)).

Definition 4 A rational system $A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ of linear inequalities is $\frac{1}{d}$ -integral or equivalently the polyhedron $P = \{\mathbf{x} : A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ is $\frac{1}{d}$ -integral if all extreme points of P are $\frac{1}{d}$ -integral.

Definition 5 A linear system $A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ is called totally dual $\frac{1}{d}$ -integral ($\frac{1}{d}$ -TDI) if the maximum in the LP-duality equation (1) has a $\frac{1}{d}$ -integral optimal solution, for every integral vector \mathbf{w} for which the optimum is finite. Furthermore, the system is called box-totally dual $\frac{1}{d}$ -integral (box- $\frac{1}{d}$ -TDI) if $A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}, \mathbf{u} \geq \mathbf{x} \geq \mathbf{l}$ is $\frac{1}{d}$ -TDI for all rational vectors \mathbf{u} and \mathbf{l} , where coordinates of \mathbf{u} are allowed to be $+\infty$.

Similar to the Edmonds–Giles theorem [41] (see [Theorem 1\(i\)](#)), from total dual $\frac{1}{d}$ -integrality, one can derive $\frac{1}{d}$ -integrality, provided the system has an integral right-hand-side vector.

Theorem 4 *If $A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ is a $\frac{1}{d}$ -TDI system and \mathbf{b} is integral, then $A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ is $\frac{1}{d}$ -integral.*

Proof It suffices to show that $\min\{w^T x : Ax \geq b, x \geq 0\}$ has a $\frac{1}{d}$ -integral optimal solution for any integral w such that the optimal value of $\min\{w^T x : Ax \geq b, x \geq 0\}$ is finite (cf. Sect. 5.15 of [103] or Sect. 1.1 of [28]). Note from the definition of the $\frac{1}{d}$ -TDI system that $\max\{y^T(\frac{1}{d}b) : y^T(\frac{1}{d}A) \leq w^T, y \geq 0\}$ has an integral optimal solution; this solution is obviously an optimal solution to $\max\{y^T b : y^T(\frac{1}{d}A) \leq w^T, y \geq 0\}$. It follows from [Definition 5](#) that $\frac{1}{d}Ax \geq b$, $x \geq 0$ is TDI and further from [Theorem 1\(i\)](#) that $\min\{w^T x : \frac{1}{d}Ax \geq b, x \geq 0\}$ has an integral optimal solution x^* . It is easy to see that $\frac{1}{d}x^*$ is a $\frac{1}{d}$ -integral optimal solution to $\min\{w^T x : Ax \geq b, x \geq 0\}$. \square

For brevity, given linear system $Ax \geq b$, $x \geq 0$, $u \geq x \geq l$ and vector w , let $\text{Min}(A, b, l, u, w)$ and $\text{Max}(A, b, l, u, w)$ denote the minimization problem and maximization problem, respectively, in the LP-duality equation

$$\begin{aligned} \min\{w^T x : Ax \geq b, x \geq 0, u \geq x \geq l\} &= \max\{\alpha^T \mathbf{1} + \beta^T l - \gamma^T u \\ &\quad : \alpha^T A + \beta^T - \gamma^T \leq w^T, \alpha, \beta, \gamma \geq 0\} \end{aligned}$$

Lemma 1 *If linear system $\frac{1}{d}Ax \geq b$, $x \geq 0$ is box-TDI, then the linear system $Ax \geq b$, $x \geq 0$ is box- $\frac{1}{d}$ -TDI.*

Proof For any integral w and rational u, l , the box-TDI-ness says that $\text{Max}(\frac{1}{d}A, b, dl, du, w)$ has an integral optimal solution (y^*, α^*, β^*) , which is obviously an integral optimal solution to $\text{Max}(\frac{1}{d}A, \frac{1}{d}b, l, u, w)$. It follows that $(\frac{1}{d}y^*, \alpha^*, \beta^*)$ is a $\frac{1}{d}$ -integral optimal solution to $\text{Max}(A, b, l, u, w)$ as desired. \square

Lemma 2 *If linear system $Ax \geq b$, $x \geq 0$ is box- $\frac{1}{d}$ -TDI, then $\frac{1}{d}Ax \geq b$, $u \geq x \geq l$ is integral for all rational u and l .*

Proof For any integral w and rational u, l , the box- $\frac{1}{d}$ -TDI-ness of $Ax \geq b$, $x \geq 0$ and [Theorem 4](#) imply that $\text{Min}(A, b, \frac{1}{d}l, \frac{1}{d}u, dw)$ has a $\frac{1}{d}$ -integral optimal solution x^* . It follows that dx^* is an integral solution to $\text{Min}(\frac{1}{d}A, b, l, u, w)$ as desired. \square

Gerards and Laurent [62] proposed the concept of box- $\frac{1}{d}$ -integrality of a linear system (and its associated polyhedron), which is evidently stronger than integrality, and can be derived from box-total dual integrality as shown by Chen et al. [20].

Definition 6 A linear system $Ax \geq b$, $x \geq 0$ or equivalently polyhedron $\{x : Ax \geq b, x \geq 0\}$ is called *box- $\frac{1}{d}$ -integral* if $Ax \geq b$, $x \geq 0$, $u \geq x \geq l$ is $\frac{1}{d}$ -integral for all $\frac{1}{d}$ -integral vectors u and l .

Lemma 3 *If $Ax \geq b$, $x \geq 0$ is a box-TDI system and b is integral, then $Ax \geq b$, $x \geq 0$ is box- $\frac{1}{d}$ -integral, for every integer $d > 0$.*

2.3 Hypergraphs

In this chapter, a *collection* is a synonym of a multiset in which elements may occur more than once. So if $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_m\}$ is a collection, then possibly $\lambda_i = \lambda_j$ for some distinct i, j . In contrast, in a *set* and in a *subset* (of a collection), all its elements are distinct. The *size* $|\Lambda|$ of Λ is defined to be m . The union of two collections Λ_1 and Λ_2 , written as $\Lambda_1 \cup \Lambda_2$, is the collection where the multiplicity of every element is the sum of its multiplicities in Λ_1 and Λ_2 . A *family of sets* is a collection of subsets of some given ground set.

A rich variety of combinatorial optimization problems falls within the general framework of *packing* and *covering* in hypergraphs for which total dual integrality is rephrased as the so-called Mengerian property [36, 62, 103].

Definition 7 A *hypergraph* is a pair $\mathcal{H} = (V, \mathcal{E})$, where V is a finite set and \mathcal{E} is a collection of subsets of V . Elements of V and \mathcal{E} are called the *nodes* and *hyperedges* of \mathcal{H} , respectively. A *clutter* is a hypergraph in which no hyperedge is contained in another one. A hypergraph is often referred to as the hypergraph of its hyperedges.

Definition 8 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph and let w be a nonnegative integral *weight* function (vector) defined on V . A collection \mathcal{F} of hyperedges in \mathcal{E} is called a w -*packing* of \mathcal{H} (or hyperedges of \mathcal{H}) if each $v \in V$ belongs to at most $w(v)$ members of \mathcal{F} . A node subset U of V is called a *node cover* of \mathcal{H} if U intersects all hyperedges in \mathcal{E} . Let $\nu(\mathcal{H}, w)$ denote the maximum size of a w -packing of \mathcal{H} , and let $\tau(\mathcal{H}, w)$ denote the minimum total weight of a node cover of \mathcal{H} . The integers $\nu(\mathcal{H}, w)$ and $\tau(\mathcal{H}, w)$ are referred to as *w-packing number* and *w-covering number* of \mathcal{H} , respectively.

Let A be the V - \mathcal{E} incidence matrix of hypergraph \mathcal{H} . Then a w -packing and a node cover of \mathcal{H} correspond to an integral solution of the maximization and that of the minimization in (1), respectively. So, the weak duality theorem of linear programming implies that

$$\nu(\mathcal{H}, w) \leq \tau(\mathcal{H}, w). \quad (2)$$

This inequality, however, needs not hold equality in general. As a matter of fact, the ratio of $\nu(\mathcal{H}, w)$ over $\tau(\mathcal{H}, w)$ can be arbitrarily close to 0 even when $w = \mathbf{1}$, the all one vector, as shown by Erdős and Pósa [43].

A hypergraph \mathcal{H} is called *Mengerian* if the min–max relation $\nu(\mathcal{H}, w) = \tau(\mathcal{H}, w)$ is satisfied by any nonnegative integral function w defined on V . (3)

It follows from [Theorem 1\(i\)](#) that \mathcal{H} is Mengerian if and only if $Ax \geq \mathbf{1}$, $x \geq \mathbf{0}$ is a TDI system. From the strong combinatorial feature of hypergraphs, one can see

that the total dual integrality is a common framework to prove a bunch of min–max relations in combinatorial optimization.

Definition 9 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph, and let A be the \mathcal{E} - V incidence matrix of \mathcal{H} .

- (i) \mathcal{H} is *ideal* if the system $Ax \geq \mathbf{1}, x \geq \mathbf{0}$ is integral.
- (ii) \mathcal{H} is *Mengerian* if the system $Ax \geq \mathbf{1}, x \geq \mathbf{0}$ is TDI
- (iii) \mathcal{H} is *box-Mengerian* if the system $Ax \geq \mathbf{1}, x \geq \mathbf{0}$ is box-TDI.

Mengerian hypergraphs [62, 103, 107] are also known as hypergraphs of the *max-flow min-cut property* [107]. They derive the “Mengerian” name from the following interpretation [20]. Given a graph $G = (V, E)$ together with two distinct vertices s and t in V , let \mathcal{H} be the hypergraph with node set E and hyperedge set consisting of the edge sets of all st -paths in G . Then the statement “ \mathcal{H} is Mengerian” is exactly the edge version of Menger’s Theorem (see Corollary 9.1b of [103]). The notion of idealness is also known as the *width-length property* (A. Lehman, 1981, On the width-length inequality and degenerate projective planes, unpublished manuscript), the *weak max-flow min-cut property* [107], or the \mathcal{Q}_+ -*MFMC property* [99]. The following corollary of [Theorem 1](#) says that box-Mengerian property implies Mengerian property, and Mengerian property implies idealness.

Corollary 1 (i) Every box-Mengerian hypergraph is Mengerian.
(ii) Every Mengerian hypergraph is ideal.

The following simple clutters (cf. [Definition 7](#)) will act as excluded minors (cf. [Definition 12](#)) in characterizing Mengerian and box-Mengerian hypergraphs; see for instance [Corollaries 4](#) and [5](#).

Example 1 None of the following clutters is box-Mengerian:

- (i) Clutter $C_3 = (\{1, 2, 3\}, \{\{1, 2\}, \{2, 3\}, \{1, 3\}\})$ is not ideal.
- (ii) Clutter $Q_6 = (\{1, 2, 3, 4, 5, 6\}, \{\{1, 3, 5\}, \{1, 2, 6\}, \{2, 3, 4\}, \{4, 5, 6\}\})$ = (the edge set of K_4 , {edge sets of triangles in K_4 }) is ideal but not Mengerian (see, e.g., [107]), where K_4 denotes the complete graph on four vertices.
- (iii) Clutter $Q_7 = (\{1, 2, 3, 4, 5, 6, 7\}, \{\{1, 4, 7\}, \{2, 5, 7\}, \{3, 6, 7\}, \{1, 2, 6, 7\}, \{1, 3, 5, 7\}, \{2, 3, 4, 7\}, \{4, 5, 6, 7\}\})$ is Mengerian but not box-Mengerian (see Lemma 2.1 of [20]).

Every hypergraph $\mathcal{H} = (V, \mathcal{E})$ is associated with a clutter $\mathcal{H}^{\min} = (V, \{F \in \mathcal{E} : \nexists F' \in \mathcal{E} \text{ with } F' \subset F\})$. The following easy observation legitimizes switching between “hypergraphs” and “clutters” whichever is more convenient for stating total dual integrality:

Observation 1 \mathcal{H} is Mengerian (resp. box-Mengerian) if and only if so is \mathcal{H}^{\min} .

Definition 10 The blocker $b(\mathcal{H})$ of a hypergraph \mathcal{H} is the hypergraph which has node set the same as \mathcal{H} and hyperedge set consist of all inclusionwise minimal node covers of \mathcal{H} (cf. Definition 8). A clutter and its blocker form a *blocking pair* in the sense that the blocker of the blocker of a clutter is the clutter itself.

More generally, one can easily show that $b(b(\mathcal{H})) = \mathcal{H}^{\min}$ holds for all hypergraphs \mathcal{H} [40]. The blocking pair Q_6 (see Example 1(ii)) and its blocker $b(Q_6)$ often occur in counterexamples of min–max relations on packing and covering (see, e.g., [107] and Sect. 4.4 of this chapter).

Example 2 Clutter $b(Q_6) = (E(K_4), \mathcal{E})$ has its node set the edge set of K_4 and its hyperedge set \mathcal{E} consist of the triangles and perfect matchings of K_4 . It has been well known that $b(Q_6)$ is Mengerian (see, e.g., [103, 107]).

It is clear from Definition 10 that the blocker of any hypergraph is a clutter. Let \mathcal{H} be a hypergraph with hyperedge-node incidence matrix A . Observe that the polytope

$$B(\mathcal{H}) = \{x : Ax \geq \mathbf{1}, \mathbf{1} \geq x \geq \mathbf{0}\} \quad (4)$$

contains all incidence vectors of node covers of \mathcal{H} . For this reason, the polytope $B(\mathcal{H})$ is often referred to as the *fractional node cover polytope* of \mathcal{H} and as the *node cover polytope* of \mathcal{H} if $B(\mathcal{H})$ is integral. It follows from Corollary 1 that if $\mathcal{H} = (V, \mathcal{E})$ is Mengerian, then $B(\mathcal{H})$ is the polytope of the node cover of \mathcal{H} , and, given any weight on V , a node cover of \mathcal{H} with minimum weight can be found in polynomial time provided the problem of finding a minimum weighted hyperedge in \mathcal{E} is polynomial-time solvable (cf. [65]). Corresponding to Definition 5, (box-)Mengerian hypergraphs have the following fractional $\frac{1}{d}$ - generalizations:

Definition 11 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph and let A be the \mathcal{E} - V incidence matrix. Then \mathcal{H} is called $\frac{1}{d}$ -Mengerian (resp. box- $\frac{1}{d}$ -Mengerian) if $Ax \geq \mathbf{1}$, $x \geq \mathbf{0}$ is a $\frac{1}{d}$ -TDI (resp. box- $\frac{1}{d}$ -TDI) system.

2.4 Transformations Preserving Dual Integrality

Operations on linear systems which preserve (box-)total dual integrality serve as tools for making inductive proof or directly discovering new (box-)TDI systems. This subsection presents some operations introduced recently that are useful in combinatorial applications. The interested reader is referred to Sect. 22.5 of [101] and Sect. 5.17 of [103] to see how the (box-)total dual integrality behaves under more transformations: scalar multiplications, “slack” variable addition and removal, variable repetition, and Fourier–Motzkin elimination, to name a few.

Theorem 5 Let A be an nonnegative integer matrix. If $Ax \geq \mathbf{1}$, $x \geq \mathbf{0}$ is a TDI system, then so is $Ax \geq \mathbf{1}$, $\mathbf{1} \geq x \geq \mathbf{0}$.

The above result by Schrijver (Theorem 5.23 of [103]) implies that the fractional polytope of the node cover (see [Definition 10](#)) of a Mengerian hypergraph (see [Definition 9](#)) is integral.

Corollary 2 Let \mathcal{H} be a Mengerian hypergraph with hyperedge-node incidence matrix A . Then the polytope of the node cover of \mathcal{H} is $\{x : Ax \geq \mathbf{1}, \mathbf{1} \geq x \geq \mathbf{0}\}$.

The following two lemmas [19] follow immediately from the definition of box-TDI systems ([Definition 2](#)). In particular the second lemma exhibits some dominance consistent with intuition.

Lemma 4 Let matrix A' be obtained from matrix A by adding a zero-column. If system $Ax \geq b$, $x \geq \mathbf{0}$ is box-TDI, then so is the system $A'x' \geq b$, $x' \geq \mathbf{0}$.

Lemma 5 Suppose a_1 and a_2 are rational vectors with $a_1 \leq a_2$, and b_1 and b_2 are rational numbers with $b_1 \geq b_2$. Then the system $Ax \geq b$, $a_1^T x \geq b_1$, $a_2^T x \geq b_2$, $x \geq \mathbf{0}$ is box-TDI if and only if the system $Ax \geq b$, $a_1^T x \geq b_1$, $x \geq \mathbf{0}$ is box-TDI.

Column Contraction and Deletion. Suppose the rows and columns of matrix A are indexed by disjoint sets R and S , respectively. For any $s \in S$, let A/s be the matrix obtained from A by deleting the column indexed by s ; let $A\backslash s$ be the matrix obtained from A by deleting column s and all rows r for which the (r, s) -th entry $A_{r,s}$ of A is not zero. As the notations indicate, A/s and $A\backslash s$ are viewed as the “contraction” and “deletion,” respectively, of column (coordinate) s from the system. The following two theorems show that the property of being a box-TDI system is preserved under the “contraction” and “deletion” of columns. The first fact was observed in [101], on page 323.

Theorem 6 If $Ax \geq b$, $x \geq \mathbf{0}$ is a box-TDI system and $A' = A/s$, then $A'x' \geq b$, $x' \geq \mathbf{0}$ is also box-TDI.

Similarly, the operation on A to produce $A\backslash s$ also preserves box-total dual integrality, as recently shown by Chen et al. [19]. In this chapter, for any vector $x \in \Omega^K$ and any $J \subseteq K$, the $|J|$ -dimensional vector $x|_J = (x(j) : j \in J)$ stands for the projection of x to Ω^J . In addition, $x(J)$ denotes the value $\sum_{j \in J} x(j)$.

Theorem 7 If $Ax \geq b$, $x \geq \mathbf{0}$ is a box-TDI system, $A' = A\backslash s$, and A is nonnegative, then $A'x' \geq b'$, $x' \geq \mathbf{0}$ is also box-TDI, where $b' = b|_{R'}$ with $R' = \{r : A_{r,s} = 0\}$.

Taking Minors. As the names indicate, column contraction and deletion on linear systems discussed above generalize contraction and deletion on hypergraphs defined below.

Definition 12 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph. For any two disjoint subsets X and Y of V (possibly X or Y is empty), let $V' = V - (X \cup Y)$ and let \mathcal{E}' be the set of all *minimal* members in $\{F - Y : F \cap X = \emptyset \text{ and } F \in \mathcal{E}\}$, where the adjective *minimal* is meant with respect to set-inclusion rather than size. Then $\mathcal{H}' = (V', \mathcal{E}')$ is called the *minor* of \mathcal{H} obtained by deleting X and contracting Y and written as $\mathcal{H}' = H \setminus X / Y$.

For brevity, the empty set \emptyset will be omitted from the notations by putting $\mathcal{H} \setminus X / \emptyset = \mathcal{H} \setminus X$ and $\mathcal{H} \setminus \emptyset / Y = \mathcal{H} / Y$. As shown by Seymour [107], Mengerian property is closed under taking minors, so is the idealness. The same is true for box-Mengerian property, which can be seen from Lemma 5 and Theorems 6 and 7.

Theorem 8 All minors of a box-Mengerian (resp. Mengerian, ideal) are box-Mengerian (resp. Mengerian, ideal).

Series Extension. Let A be a matrix and let A' be obtained from A by adding a new column equal to an existing one. Then A' is called a *series extension* of A . The next lemma follows easily from the definition of TDI systems (Definition 2).

Lemma 6 If the system $Ax \geq b$, $x \geq \mathbf{0}$ is TDI and A' is a series extension of A , then the system $A'x' \geq b$, $x' \geq \mathbf{0}$ is also TDI.

Chen et al. [19] proved the following analogous result for box-TDI systems.

Theorem 9 If the system $Ax \geq b$, $x \geq \mathbf{0}$ is box-TDI and A' is a series extension of A , then the system $A'x' \geq b$, $x' \geq \mathbf{0}$ is also box-TDI.

This theorem resembles a result of Edmonds and Giles [42] on repeating variables (cf. Theorem 22.10 of Schrijver [101]), but does not follow from Edmonds and Giles' result which could imply as follows: “if the system $Ax \geq b$, $x \geq \mathbf{0}$ is box-TDI and A' is a series extension of A , then the system $A'x' \geq b$, $Bx' \geq \mathbf{0}$ is also box-TDI, where matrix B is obtained from an identity matrix by adding a unit vector.” Interestingly, the proof of Theorem 9 in [19] and that of Edmonds and Giles' result in [101] used Cook's characterization of box-TDI systems (see Theorem 3).

Parallel Extension. Unlike series extension, the operation of parallel extension requires the original matrix A and its outcome A' the *parallel extension* of A to take the following form:

$$A = \begin{bmatrix} A_1 & \frac{1}{2}\mathbf{1} \\ A_2 & \mathbf{0} \end{bmatrix} \text{ and } A' = \begin{bmatrix} \mathbf{0} & \frac{1}{2} & \frac{1}{2} \\ A_1 & \frac{1}{2}\mathbf{1} & \mathbf{0} \\ A_1 & \mathbf{0} & \frac{1}{2}\mathbf{1} \\ A_2 & \mathbf{0} & \mathbf{0} \end{bmatrix}. \quad (5)$$

By applying Cook's sufficient and necessary condition ([Theorem 3](#)) for box-TDI systems to both systems $A\mathbf{x} \geq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$ and $A'\mathbf{x}' \geq \mathbf{1}$, $\mathbf{x}' \geq \mathbf{0}$ with various objective functions, Chen et al. [19] proved that the box-total dual integrality of $A\mathbf{x} \geq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$ is maintained under the parallel extension.

Theorem 10 *Let matrix A and its parallel extension be as expressed in (5). If $A\mathbf{x} \geq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$ is box-TDI, then so is $A'\mathbf{x}' \geq \mathbf{1}$, $\mathbf{x}' \geq \mathbf{0}$.*

2.5 Graphs and Matroids

Polyhedral combinatorics benefit greatly from results and techniques from graph theory and matroid theory as they provide structures on more realistic settings. This subsection gives basic concepts and notations on graphs and matroids, in particular those related to the topics of the present chapter.

For terminology on graphs, this chapter follows [6, 33]. A *graph* $G = (V, E)$ consists of a *vertex set* $V = V(G)$ of *vertices* and an *edge set* $E = E(G)$ of *edges*. An edge with end vertices (ends) u and v is written as uv . When G is a directed graph, elements in $E(G)$ are often called *arcs*. Given graph G , writing $G' \subseteq G$ means that G' is a subgraph of G . In this chapter, graphs are finite; a cycle in a directed graph is always understood as a directed one. This chapter often does not distinguish a one-element set $\{v\}$ (sometimes called a singleton) from its only element v .

- (i) A *triangle* in a graph is a cycle of length 3 in terms of number of vertices on the cycle.

Let F be a set of edges (arcs) of a graph G , the graph derived from G by deleting F and identifying each pair of ends of the edges (arcs) in F is said to be obtained by *contracting* S , and is denoted by G/F .

- (ii) A *minor* of a graph G is a graph obtained from a subgraph of G by contracting edges (arcs).

For any graph $G = (V, E)$ and $S \subseteq V \cup E$, notation $G \setminus S$ denotes the graph obtained from G by deleting S as well as edges (arcs) incident with any vertex in S . An undirected graph is *complete* if any two of vertices are adjacent.

- (iii) The simple complete graph on n vertices is denoted by K_n .

An undirected graph G is *bipartite* if $V(G)$ can be partitioned into nonempty disjoint subsets X and Y so that both X and Y are stable sets (also known as independent sets) of G ; such a partition (X, Y) is called a *bipartition* of G .

If, in addition, each vertex of X is adjacent to each vertex of Y , then G is called a *complete bipartite graph*.

- (iv) The simple complete bipartite graph with bipartition (X, Y) is denoted by $K_{m,n}$, where $|X| = m$ and $|Y| = n$.

It is well known that an undirected graph is bipartite if and only if it contains no cycle of odd length.

To *subdivide* an edge uv in an undirected graph is to replace uv by a path utv , where t is not a vertex in the original graph.

- (v) A *subdivision* of an undirected graph is any graph derivable from it by recursively subdividing edges.

Definition 13 Let $G = (V, E)$ be an undirected graph. A vertex subset $U \subseteq V$ is called a *vertex cover* of G if $G \setminus U$ contains no edge. An edge subset $F \subseteq E$ is called an *edge cover* of G if every vertex of G is incident with at least one edge in F .

Definition 14 Let $G = (V, E)$ be an undirected graph. For any $\emptyset \neq S \subset V$, let $\delta(S)$ denote the set of edges in E with one end in S and the other in $V - S$. In particular, if $v \in V$, then $\delta(v)$ denotes the set of edges in G incident with v . A set of edges in G is called a *cut* if it is $\delta(S)$ for some nonempty proper subset S of V .

Let $G = (V, E)$ be a directed graph. For any $S \subseteq V$, let $\delta^{\text{in}}(S)$ denote the set of arcs in E that *enter* S , that is, arcs with their heads in S and tails in $V - S$; and let $\delta^{\text{out}}(S)$ denote the set of arcs in E that *leave* S , that is, arcs with their tails in S and heads in $V - S$.

Definition 15 A set of arcs in a directed graph $G = (V, E)$ is called a (*directed*) *cut* if it is $\delta^{\text{in}}(S)$ for some nonempty proper subset S of V .

For terminology on matroids, this chapter follows [88]. Basically, a matroid M is a pair (E, \mathcal{C}) , where E is a finite set and \mathcal{C} is a set of subsets of E that satisfies certain axioms (see Sect. 1.1 of [88]). Members of E and \mathcal{C} are called *elements* and *circuits* of M , respectively. In this chapter, matroid $M = (E, \mathcal{C})$ is often considered as a hypergraph with node set E and hyperedge set \mathcal{C} . So:

- (vi) Minors of matroid $M = (E, \mathcal{C})$ are defined as in [Definition 12](#).

Of special interest is the fact that minors of matroids are matroids.

- (vii) The *uniform matroid* $U_{n,r} = (E, \mathcal{C})$ has $|E| = n$ elements, and its circuit set \mathcal{C} consists of all subsets of E of size $r + 1$.
- (viii) The *Fano matroid* $F_7 = (E, \mathcal{C})$, whose elements and circuits correspond to points and lines of the Fano plane, has element set $E = \{1, 2, 3, 4, 5, 6, 7\}$, and circuit set \mathcal{C} consist of $\{1, 2, 3\}$, $\{1, 4, 7\}$, $\{1, 5, 6\}$, $\{2, 4, 6\}$, $\{2, 5, 7\}$, $\{3, 4, 5\}$, $\{3, 6, 7\}$ (the lines of the Fano plane) and their complements.
- (ix) The *series extension* F_7^+ of the Fano matroid F_7 is the matroid obtained from F_7 by adding a new element 8 in series with, say, the element 7. Hence, F_7^+ is the matroid defined on $\{1, 2, 3, 4, 5, 6, 7, 8\}$ whose circuit set is $\{C : C \text{ is}$

a circuit of F_7 with $7 \notin C\} \cup \{C \cup \{8\} : C \text{ is a circuit of } F_7 \text{ with } 7 \in C\}$. The elements 7, 8 are called *series elements* of F_7^+ . Up to isomorphism, the series extension of F_7 is unique.

Every matroid $M = (E, \mathcal{C})$ is determined by a set $\mathcal{I}(M) \subseteq 2^E$ such that circuits of M are precisely the minimal elements of $2^E - \mathcal{I}(M)$. With each matroid $M = (E, \mathcal{C})$, a *dual* matroid M^* , on the same set E of elements, can be associated in such a way that $\mathcal{I}(M^*) = \{I \subseteq E : E - I \text{ contains a maximal element of } \mathcal{I}(M)\}$.

- (x) The *dual Fano matroid* F_7^* is the dual of F_7 ; its circuits are $\{4, 5, 6, 7\}$, $\{2, 3, 5, 6\}$, $\{2, 3, 4, 7\}$, $\{1, 3, 5, 7\}$, $\{1, 3, 4, 6\}$, $\{1, 2, 6, 7\}$, and $\{1, 2, 4, 5\}$ (the complements of the lines of the Fano plane).

Definition 16 Every undirected graph $G = (V, E)$ defines a matroid $M(G) = (E, \mathcal{C})$, where \mathcal{C} consists of edge sets of all cycles of G . Every matroid of this kind is called a *graphic matroid*. The dual of a graphic matroid $M(G)$ is the matroid $M^*(G) = (E, \mathcal{D})$, where \mathcal{D} consists of all nonempty minimal cuts of G . The dual of a graphic matroid is called a *cographic matroid*.

3 Complexity

For the background of computational complexity, the reader is referred to the classic book [60] by Garey and Johnson. Two subsections below mainly report negative results on the complexity of recognizing TDI systems and that of packing feedback vertex sets in tournaments, respectively.

3.1 Recognition of TDI Systems

A number of well-known results and difficult conjectures in combinatorial optimization can be rephrased by saying that certain polyhedra are integral or certain linear systems are TDI, for instance, the celebrated strong perfect graph theorem [23]. So the following recognition problems, all proposed in Schrijver [101], are of both great theoretical interest and practical value:

Problem 1 Given a rational linear system, does it determine an integral polyhedron?

Problem 2 Given a rational linear system, is it TDI (cf. Definition 3)?

Problem 3 Given a rational linear system, is it box-TDI (cf. Definition 3)?

Schrijver [101] showed that Problem 1 is in co-NP, and so are Problems 2 and 3 if the matrix involved in the linear system is integral. In general, however, the question whether Problems 2 and 3 are in NP or co-NP is still unanswered. It is worthwhile pointing out that, first, Problems 2 and 3 can all be solved in polynomial time if the

rank of the matrix involved in the linear system is a fixed constant, as shown by Cook et al. [26]; second, classical results of Chvátal [24], Fulkerson [57], and Lovász [81] imply that for any 0–1 matrix A , the system $Ax \leq \mathbf{1}, x \geq \mathbf{0}$ defines an integral polyhedron if and only if it is TDI (cf. [Definition 3](#)) if and only if A is the clique-vertex incidence matrix of a perfect graph. Since clique-vertex incidence matrices of graphs can be recognized in polynomial time, according to a result of Gilmore on conformal hypergraphs (see page 396 of Berge [4]), the recent breakthrough on perfect graphs [22, 23] implies that such a TDI system can be recognized in polynomial time.

In connection with [Problem 2](#), Edmonds and Giles [42] conjectured that it is co-NP-complete to decide whether a given rational linear system is TDI. This conjecture was recently confirmed by Ding et al. [35] who established the following result.

Theorem 11 *Let A be a 0–1 matrix with precisely two 1s in each column. Then the problems of deciding whether the linear system $Ax \geq \mathbf{1}, x \geq \mathbf{0}$ (i) defines an integral polyhedron, (ii) is TDI, and (iii) is box-TDI are all co-NP-complete.*

Ding et al. [35] showed that the problems addressed in (i)–(iii) are essentially equivalent to the problem of recognizing the so-called quasi-bipartite graphs.

Definition 17 A graph G is called *quasi-bipartite* if for any odd cycle C in G , the deletion of all vertices on C from G results in at least one isolated vertex.

They [35] verified that it is co-NP-complete to recognize quasi-bipartite graphs, thus proving [Theorem 11](#). It can be deduced from the theorem that [Problems 1–3](#) are all NP-hard. Since the matrix in [Theorem 11](#) has only 0,1 elements, the NP-hardness is strengthened by the following corollary.

Corollary 3 *It is NP-hard to recognize (box-) Mengerian hypergraphs.*

A polyhedron is called a 0–1 *Polyhedron* if all its extreme points are 0–1 vectors [12]. Due to the graphical construction of their proofs, the results of Ding et al. [35] imply that distinguishing whether a polyhedron, given by its facets, is either a 0–1 polyhedron or fractional is an NP-hard problem.

Recently, the co-NP-completeness of deciding whether $Ax \geq \mathbf{1}, x \geq \mathbf{0}$ is TDI has been extended to the conic system by Pap [89] as follows.

Theorem 12 *Let A be a 0–1 matrix with at most three 1s in each row. Then the problem of deciding whether the linear system $Ax \geq \mathbf{0}$ is TDI is co-NP-complete.*

For characterizing properties of a general TDI system (see [Definition 3](#)), the reader is referred to the recent paper on testing total dual integrality by O’Shea and Sebö [87], who showed that a system of linear inequalities is TDI if and only if

its coefficient vectors form a Hilbert basis, and there exists a test-set for the system's dual integer programs where all test vectors have positive entries equal to 1.

3.2 Maximum Feedback Vertex Set Packing in Tournaments

Given a directed graph $G = (V, E)$ with a nonnegative integral weight function w on V , a *feedback vertex set* (FVS) is a subset of V that meets all directed cycle in G . The *FVS packing problem* on G is to find a w -FVS packing in G of maximum size (cf. Definition 8). In connection with this problem, Frank [50] suggested the following question: Given a directed graph G , can one decide in polynomial time whether the vertices of G can be colored by red or blue so that every cycle contains at least one red vertex and at least one blue vertex? Or is this an NP-complete problem? The following theorem [21] states that Frank's problem is NP-complete even when G is restricted to a tournament (an orientation of a complete graph):

Theorem 13 *It is NP-complete to decide whether the vertex set of a given tournament can be partitioned into two feedback vertex sets.*

Chen et al. [21] proved the NP-completeness by reduction from the NOT-ALL-EQUAL 3-SATISFIABILITY problem [96]. Particularly, Theorem 13 says that the FVS packing problem on tournament is NP-hard even for the unweighted version with $w = \mathbf{1}$.

4 A Unified Approach to Box-Mengerian Hypergraphs

In this section, five subsections combine to make an exposition of a unified approach to studying box-Mengerian hypergraphs. Section 4.1 presents a sufficient condition for box-Mengerian hypergraphs: the so-called ESP property (see Definition 18). Sections 4.2–4.4 exhibit several classes of box-Mengerian hypergraphs discovered by using ESP property. Section 4.5 relates some well-known min–max relation in the literature with the ESP property. Section 4.6 discusses the blockers (cf. Definition 10) of ESP hypergraphs in the context of the ESP property.

4.1 ESP Property

Due to NP-hardness of recognizing box-Mengerian hypergraphs (see Corollary 3), a basic theme in combinatorial optimization is to identify such objects associated with various problems. By Cook's characterization of box-TDI systems [25], a hypergraph $\mathcal{H} = (V, \mathcal{E})$ is box-Mengerian if and only if $Ax \geq \mathbf{1}$, $x \geq \mathbf{0}$ is a TDI system and for any rational vector $c = (c_1, c_2, \dots, c_n)^T$, where $n = |V|$, there exists an integral vector $\tilde{c} = (\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_n)^T$ such that $\lfloor c_i \rfloor \leq \tilde{c}_i \leq \lceil c_i \rceil$, for all $1 \leq i \leq n$, and such that every optimal solution of $\min \{c^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\}$

is also an optimal solution of $\min \{\tilde{c}^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\}$. Nevertheless, this necessary and sufficient condition is not so ‘‘user-friendly’’ and can hardly be verified in practice. Recalling Schrijver’s sufficient condition for box-TDI systems [103] ([Theorem 2](#)), when matrix A over there is restricted to the hyperedge-node incidence matrix of a hypergraph and vector b over there is set to $\mathbf{1}$, the theorem yields a sufficient condition for box-Mengerian hypergraphs. For various classes of box-Mengerian hypergraphs resulted from this theorem, see [100, 103]. Owing to the demanding total unimodularity requirement, it is desirable to have some other powerful approaches for establishing box-Mengerian hypergraphs. Chen et al. [17] derived an analogue of [Theorem 2](#) to fulfill such a need.

Definition 18 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph and let Λ be a collection of its hyperedges. Let $d_\Lambda(v)$ denote the number of hyperedges in Λ that contain v . An *equitable subpartition* of Λ consists of two collections Λ_1 and Λ_2 of hyperedges in \mathcal{E} (which are not necessarily in Λ) such that:

- (i) $|\Lambda_1| + |\Lambda_2| \geq |\Lambda|$.
- (ii) $d_{\Lambda_1 \cup \Lambda_2}(v) \leq d_\Lambda(v)$ for all $v \in V$.
- (iii) $\max\{d_{\Lambda_1}(v), d_{\Lambda_2}(v)\} \leq \lceil d_\Lambda(v)/2 \rceil$ for all $v \in V$.

The hypergraph \mathcal{H} is called *equitably subpartitionable*, abbreviated ESP, if every collection of its hyperedges admits an equitable subpartition.

The above (i), (ii), and (iii) are referred to as *ESP property*, which was first introduced by Ding and Zang [36] in their characterization of all graphs with the min–max relation on packing and covering cycles, where they proved that every ESP hypergraph is Mengerian. Chen et al. [17] showed that ESP property turns out to be even sufficient for box-Mengerian hypergraphs.

Theorem 14 Every ESP hypergraph is box-Mengerian.

Let $Ax \geq \mathbf{1}, x \geq \mathbf{0}, u \geq x \geq l$ be a linear system. With a slight abuse of notation, let $\text{Max}(A, l, u, w)$ stand for both the linear program $\max\{\alpha^T \mathbf{1} + \beta^T l - \gamma^T u : \alpha^T A + \beta^T - \gamma^T \leq w^T, \alpha, \beta, \gamma \geq \mathbf{0}\}$ and its optimal value. When integrality is imposed on its solutions, let $\text{Max}(A, l, u, w; \mathbb{Z})$ stand for the corresponding integer program and optimal value. When half-integrality is imposed, let $\text{Max}(A, l, u, w; \mathbb{Z}/2)$ stand for the corresponding program and optimal value, where $\mathbb{Z}/2 = \{k/2 : k \in \mathbb{Z}\}$. Suppose A is the \mathcal{E} - V incidence matrix of a hypergraph $\mathcal{H} = (V, \mathcal{E})$, and suppose $(\alpha^*, \beta^*, \gamma^*)$ is an optimal solution to $\text{Max}(A, l, u, w; \mathbb{Z})$.

Definition 19 Let Λ be the hyperedge collection of \mathcal{H} such that each $U \in \mathcal{E}$ appears exactly $\alpha^*(U)$ times in Λ . The hyperedge collection Λ is said to be the one *corresponding* to α^* .

[Theorem 14](#) is straightforward from the following stronger statement [17].

Theorem 15 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph and let A be the \mathcal{E} - V incidence matrix. Suppose that for any $\mathbf{l}, \mathbf{u} \in \mathbb{Q}_+^V$ and $\mathbf{w} \in \mathbb{Z}^V$ with finite $\text{Max}(A, \mathbf{l}, \mathbf{u}, \mathbf{w})$, there exists an optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ to $\text{Max}(A, \mathbf{l}, \mathbf{u}, 2\mathbf{w}; \mathbb{Z})$ such that the hyperedge collection corresponding to $\boldsymbol{\alpha}^*$ admits an equitable subpartition. Then \mathcal{H} is box-Mengerian.

The proof of this theorem relies on the following fundamental result on total dual integrality, as shown by Schrijver and Seymour [104].

Lemma 7 The system $A\mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0}, \mathbf{u} \geq \mathbf{x} \geq \mathbf{l}$ is TDI if and only if

$$\text{Max}(A, \mathbf{l}, \mathbf{u}, 2\mathbf{w}; \mathbb{Z}) \leq 2 \text{Max}(A, \mathbf{l}, \mathbf{u}, \mathbf{w}; \mathbb{Z}) \quad (6)$$

for any integral vector \mathbf{w} for which $\text{Max}(A, \mathbf{l}, \mathbf{u}, \mathbf{w})$ is finite.

By the hypothesis of [Theorem 15](#), there exists an optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ to $\text{Max}(A, \mathbf{l}, \mathbf{u}, 2\mathbf{w}; \mathbb{Z})$ such that the hyperedge collection Λ corresponding to $\boldsymbol{\alpha}^*$ admits an equitable subpartition (Λ_1, Λ_2) . Suppose that for $i = 1, 2$, the hyperedge collection Λ_i corresponds to $\boldsymbol{\alpha}_i \in \mathbb{Z}_+^{\mathcal{E}}$. Based on the ESP property, $\boldsymbol{\beta}_i, \boldsymbol{\gamma}_i \in \mathbb{Z}_+^V, i = 1, 2$ can be defined such that both $(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_1)$ and $(\boldsymbol{\alpha}_2, \boldsymbol{\beta}_2, \boldsymbol{\gamma}_2)$ are feasible solutions to $\text{Max}(A, \mathbf{l}, \mathbf{u}, \mathbf{w}; \mathbb{Z})$, and $\sum_{i=1}^2 (\boldsymbol{\alpha}_i^T \mathbf{1} + \boldsymbol{\beta}_i^T \mathbf{l} - \boldsymbol{\gamma}_i^T \mathbf{u}) \geq (\boldsymbol{\alpha}^*)^T \mathbf{1} + (\boldsymbol{\beta}^*)^T \mathbf{l} - (\boldsymbol{\gamma}^*)^T \mathbf{u}$. It follows that the inequality $\boldsymbol{\alpha}_i^T \mathbf{1} + \boldsymbol{\beta}_i^T \mathbf{l} - \boldsymbol{\gamma}_i^T \mathbf{u} \geq [(\boldsymbol{\alpha}^*)^T \mathbf{1} + (\boldsymbol{\beta}^*)^T \mathbf{l} - (\boldsymbol{\gamma}^*)^T \mathbf{u}] / 2$ holds for $i = 1$ or 2 , establishing (6) and thus [Theorem 15](#).

Let \mathcal{H} be a hypergraph. The combination of [Theorem 14](#) and [Corollary 1](#) provides the following string of implications, which will be used explicitly or implicitly in the remaining discussions:

$$\mathcal{H} \text{ is ESP} \Rightarrow \mathcal{H} \text{ is box-Mengerian} \Rightarrow \mathcal{H} \text{ is Mengerian} \Rightarrow \mathcal{H} \text{ is ideal.} \quad (7)$$

[Sections 4.2–4.4](#) below exhibit some applications of [Theorem 14](#), where several classes of box-Mengerian hypergraphs (on paths, cycles, matroid ports) have been established. In comparison with [Theorem 2](#) from which none of these box-TDI-ness can be derived directly, the approach involving ESP property is of transparent combinatorial nature and hence is fairly easy to work with.

4.2 Path Hypergraphs

Let T be a tree with edge set V , and let \mathcal{E} be edge sets of some paths in T , such that each edge of T is contained in at least one of these paths. The hypergraph $\mathcal{H} = (V, \mathcal{E})$ is called the *edge path tree* (EPT) hypergraph supported by T , where T is known as a *supporting tree* of \mathcal{H} (note that a supporting tree may not be unique). A subset of hyperedges $\{P_1, P_2, \dots, P_k\}$ in \mathcal{H} is called a *pie*

if T contains a vertex u and k edges e_1, e_2, \dots, e_k all incident with u , such that $P_i \cap \{e_1, e_2, \dots, e_k\} = \{e_i, e_{i+1}\}$ for $i = 1, 2, \dots, k$, where $e_{k+1} = e_1$; the pie is called *odd* if k is odd. For any two disjoint subsets X and Y of V (possibly X or Y is empty), let $\mathcal{H}' = \mathcal{H} \setminus X / Y$ be the minor of \mathcal{H} obtained by deleting X and contracting Y (cf. [Definition 12](#)). Clearly, \mathcal{H}' is supported by T' , the tree obtained from T by contracting edges in $X \cup Y$. The hypergraph \mathcal{H} is said to be *odd-pie-free* if it contains no odd pies and is said to be *odd-M-pie-free* if none of its minors is an odd pie. As observed by Apollonio [1], an odd-M-pie-free hypergraph may contain an odd pie.

Theorem 16 *Let $\mathcal{H} = (V, \mathcal{E})$ be an EPT clutter. Then the following statements are equivalent: (i) \mathcal{H} is odd-M-pie-free; (ii) \mathcal{H} is ideal; (iii) \mathcal{H} is Mengerian; (iv) \mathcal{H} is box-Mengerian; and (v) \mathcal{H} is ESP.*

The equivalence of the first three statements was established by Apollonio [1]. Recently, it was shown that for the same combinatorial structures, the stronger statements (iv) and (v) hold. The proof [17] relies heavily on the following two results due to Apollonio [1]:

- The incidence matrix of every odd-pie-free EPT hypergraph is totally unimodular.
- If an odd-M-pie-free EPT clutter \mathcal{H} contains an odd pie \mathcal{P} , then there exist four hyperedges P_1, P_2, P_3, P_4 in \mathcal{H} , with $\{P_3, P_4\} \subseteq \mathcal{P}$, such that $P_1 \cap P_2 = \emptyset$, $P_3 \cap P_4 \neq \emptyset$, and $P_1 \cup P_2 \subseteq (P_3 - P_4) \cup (P_4 - P_3)$.

The first result, together with the Ghouila-Houri theorem (see Theorem 19.3 of [101]), implies that every odd-pie-free EPT hypergraph is ESP, while the second result provides a recursive method to transform any hyperedge collection Λ of an odd-M-pie-free EPT clutter to a hyperedge collection Λ' of an odd-pie-free EPT hypergraph such that every equitable subpartition of Λ' is an equitable subpartition of Λ . It follows that every odd-M-pie-free EPT clutter is ESP and thus box-Mengerian by [Theorem 14](#).

4.3 Cycle Hypergraphs

Let $G = (V, E)$ be a graph (undirected or directed) and let $w \in \mathbb{Z}_+^V$ be a weight function on V . Let $\mathcal{H} = (V, \mathcal{E})$ be *cycle hypergraph* of G , where \mathcal{E} consists of the vertex sets of all cycles in G .

Definition 20 A *feedback vertex set* (FVS) of G , or equivalently a node cover of cycle hypergraph \mathcal{H} (see [Definition 8](#)), is a vertex subset that intersects each cycle in G . A w -*cycle packing* of G , or equivalently a w -packing of \mathcal{H} (see [Definition 8](#)), is a collection \mathcal{C} of cycles (with repetition allowed) such that each vertex v is contained in at most $w(v)$ members of \mathcal{C} . The *feedback vertex set problem* is to find an FVS with minimum total weight, denoted as $\tau(G, w) = \tau(\mathcal{H}, w)$.

The *cycle packing problem* is to find a w -cycle packing with maximum size, denoted as $v(G, w) = v(\mathcal{H}, w)$.

Graph G is said to be *cycle Mengerian* if its cycle hypergraph is Mengerian. The problems of computing $\tau(G, w)$ and $v(G, w)$ arise in a variety of applications, and both problems are known to be NP-hard [60]. However, according to a powerful result of Grötschel et al. [65], in case of cycle Mengerian graph G , the problem of computing $\tau(G, w)$ and $v(G, w)$ is equivalent to finding a shortest cycle in a graph, which is solvable in polynomial time. Therefore, cycle Mengerian graphs form a class for which both $\tau(G, w)$ and $v(G, w)$ can be computed in polynomial time.

4.3.1 Cycle Mengerian Undirected Graphs

Ding and Zang [36] obtained a characterization of all cycle Mengerian undirected graphs. This characterization involves forbidden subgraphs depicted in Fig. 1. An *odd ring* is a graph obtained from an odd cycle by replacing each edge $e = xy$ with either a triangle containing e or two triangles xab, ycd together with two additional edges ac and bd . A *wheel* is obtained from a cycle by adding a new vertex and making it adjacent to all vertices of the cycle. As usual, $K_{2,3}$ denotes the complete bipartite graph whose bipartition consists of independent sets of sizes 2 and 3, respectively (cf. Sect. 2.5(iv)).

Theorem 17 *Let $G = (V, E)$ be a simple undirected graph and let $\mathcal{H} = (V, \mathcal{E})$ be its cycle hypergraph. Then the following statements are equivalent: (i) G contains no induced subgraph isomorphic to a subdivision of an odd ring, or a wheel, or $K_{2,3}$; (ii) \mathcal{H} is ideal; (iii) \mathcal{H} is Mengerian; (iv) \mathcal{H} is box-Mengerian; and (v) \mathcal{H} is ESP.*

The equivalence of statements (i)–(iii) and (v) was established by Ding and Zang [36]; the strengthening from the total dual integrality to the box-total dual integrality follows instantly from Theorem 14 [17]. The most difficult part of proving Theorem 17 is showing the implication from (i) to (v). Ding and Zang [36] accomplished this task by deriving a structural theorem, which asserts that if a graph has no forbidden structures, then it can be expressed as “sums” of some prime graphs. They showed that every prime graph possesses an ESP cycle hypergraph and that, when piecing together (according to the “sum” operations) graphs whose cycle hypergraphs are ESP, the resulting graph also has an ESP cycle hypergraph.

One may consider the edge version of the above cycle packing problem, where associated the simple undirected graph G is the hypergraph $\mathcal{H}' = (E, \mathcal{E}')$ whose hyperedge set \mathcal{E}' consists of the edge sets of all cycles in G . The characterization of the edge version, however, is quite easy as remarked by Ding and Zang [36].

Remark 1 Every block of graph G is either K_2 or a cycle, if and only if \mathcal{H}' is ideal, if and only if \mathcal{H}' is Mengerian, if and only if \mathcal{H}' is box-Mengerian, and if and only if \mathcal{H}' is ESP.

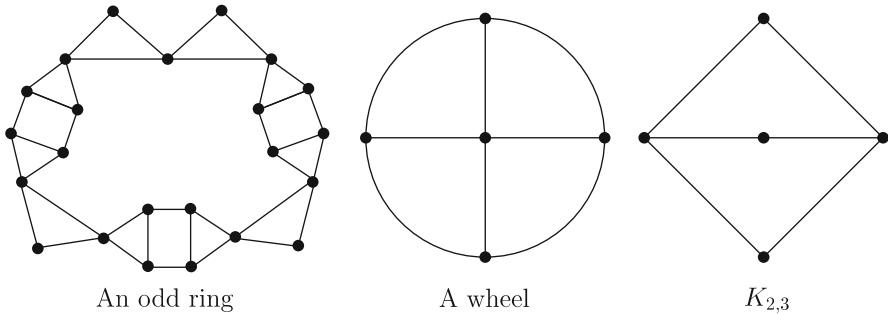


Fig. 1 Forbidden subgraphs for cycle Mengerian graphs

It is straightforward to check that \mathcal{H}' is ESP if G is as described. On the other hand, when G is not such a graph, it must have a subgraph H that is a subdivision of K_4 with one edge missing (cf. Sect. 2.5(v)). With the weight setting $w|_{E(H)} = \mathbf{1}$ and $w|_{E-E(H)} = \mathbf{0}$ [36], it is easy to see that $\min\{w^T \mathbf{x} : A\mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0}\}$ has an optimal value 1.5, which says that \mathcal{H}' is not ideal (cf. Definition 9). Now all equivalences follow from (7).

4.3.2 Cycle Mengerian Directed Graphs

As far as integral packing properties are concerned, directed cycles are not so manageable as their undirected counterparts. Due to the long list of forbidden structures, to find a good characterization of all cycle Mengerian directed graphs seems to be extremely difficult. While this characterization problem remains open in general, it was resolved completely on tournaments (orientations of undirected complete graphs) and bipartite tournaments (orientations of undirected complete bipartite graphs) by Cai et al. [13, 14].

Theorem 18 *Let $G = (V, E)$ be a tournament and let $\mathcal{H} = (V, \mathcal{E})$ be its cycle hypergraph. Then the following statements are equivalent: (i) G contains neither F_1 nor F_2 as a subgraph (see Fig. 2); (ii) \mathcal{H} is ideal; (iii) \mathcal{H} is Mengerian; (iv) \mathcal{H} is box-Mengerian; and (v) \mathcal{H} is ESP.*

Note that a vertex subset of a tournament is an FVS if and only if it intersects all triangles (see Sect. 2.5(i)) and that the cycle packing problem (see Definition 20) on tournaments actually reduces to the triangle packing problem.

The equivalence of statements (i) and (iii) of Theorem 18 was derived in [13], where Cai et al. gave a structural description of tournaments $G = (V, E)$ with no F_1 nor F_2 . The authors started with a vertex u with the maximum out-degree in G and partitioned the vertices of G according to their distances (in terms of number of arcs) from u , that is, $V = \bigcup_{i=1}^k V_i$ where a vertex $v \in V_i$ if and only if the distance from u to v is $i - 1$. The subtournament induced by V_i was shown to be acyclic if G contains

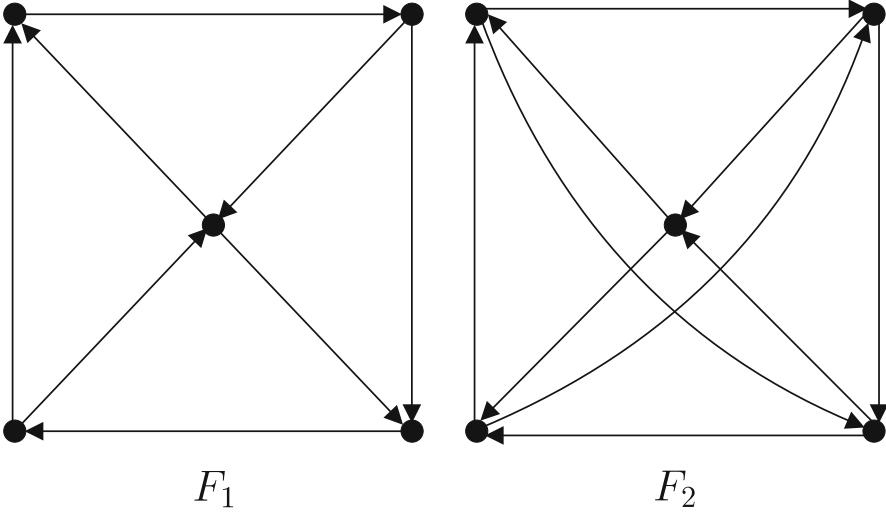


Fig. 2 Forbidden subtournaments F_1 and F_2 , where the two arcs not shown in F_1 may take any directions

no F_1 nor F_2 . This property leads to a natural order for vertices in V_i . Moreover, for every triangle in the tournament, its three vertices are in three consecutive subsets of the partition, that is, V_i, V_{i+1}, V_{i+2} for some i . Therefore, from the ordering of vertices, a lexicographical ordering \prec on all triangles in G free of F_1 and F_2 can be obtained in strongly polynomial time [13, 17]. To prove (i) \Rightarrow (iii), Cai et al. [13] showed the “smallest” triangle under \prec must be contained in a maximum w -cycle packing in G (see Definition 20), and this triangle contains a vertex such that decreasing its weight by 1 would reduce the cycle covering number $\tau(G, w)$ of G by 1 (cf. Definitions 20 and (2.3)). This fact leads to an inductive argument to verify statement (iii) and provides an efficient combinatorial algorithm for finding a maximum w -cycle packing in G as well as a minimum w -cycle covering in G . In addition, applying the local ratio technique (see, e.g., [3]) to F_1 and F_2 , Cai et al. [13] devised a 2.5-approximation algorithm for the feedback vertex set problem in any tournament, which is known to be NP-hard by reduction from the vertex cover problem [110] (see Definition 40).

Chen et al. [17] exhibited box-TDI and ESP properties associated with the same combinatorial structures. Roughly speaking, given tournament G with no F_1 nor F_2 , and a collection $\Lambda = \{V(T_1), V(T_2), \dots, V(T_m)\}$ of vertex sets of triangles T_1, T_2, \dots, T_m in G , where triangle T_i is “smaller than or equal to” T_{i+1} under \prec for $i = 1, 2, \dots, m - 1$, it can be shown that $\Lambda_1 = \{V(T_i) : i \text{ is odd and } 1 \leq i \leq m\}$ and $\Lambda_2 = \{V(T_i) : i \text{ is even and } 1 \leq i \leq m\}$ form an equitable subpartition (Λ_1, Λ_2) of Λ , thus giving the implication (i) \Rightarrow (v) in Theorem 18.

To verify the relatively easy implication from (ii) to (i) in Theorem 18, Chen et al. [17] considered tournament $G = (V, E)$ containing F_1 or F_2 . Let A be the

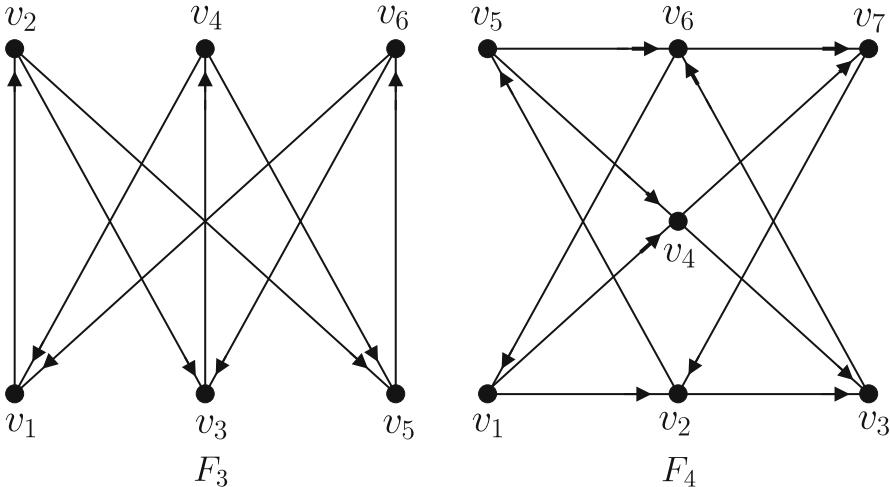


Fig. 3 Forbidden bipartite subtournaments F_3 and F_4

\mathcal{E} - V incidence matrix of $\mathcal{H} = (V, \mathcal{E})$. The authors defined integral weight $w \in \mathbb{Z}_+^V$ such that the optimal value of the linear program $\min\{w^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\}$ is within interval $[3/2, 5/3]$ and thus cannot be an integer, saying that \mathcal{H} is not ideal (cf. [Definition 9](#)).

Theorem 19 Let $G = (V, E)$ be a tournament and let $\mathcal{H} = (V, \mathcal{E})$ be its cycle hypergraph. Then the following statements are equivalent: (i) G contains neither F_3 nor F_4 as a subgraph (see [Fig. 3](#)); (ii) \mathcal{H} is ideal; (iii) \mathcal{H} is Mengerian; (iv) \mathcal{H} is box-Mengerian; and (v) \mathcal{H} is ESP.

The situation in bipartite tournaments is similar in spirit to that in tournaments. Strengthening the equivalence (i) \Leftrightarrow (iii) established by Cai et al. [14], one can show that the exclusion of F_3 and F_4 actually characterizes ESP cycle hypergraphs for bipartite tournaments.

A directed cycle of length 4 is called a 4-cycle. The 4-cycles in bipartite tournaments are counterparts of triangles in tournaments, as bipartite graphs contain no cycle of odd length. Let $G = (V, E)$ be a bipartite tournament with no bipartite subtournament isomorphic to F_3 or F_4 . Then V can be partitioned into V_1, V_2, \dots, V_k such that for every 4-cycle in G , its four vertices are in $V_i, V_{i+1}, V_{i+2}, V_{i+3}$ for some i , and a lexicographical ordering \prec can be defined on 4-cycles in G such that the lexicographically smallest 4-cycle is contained in some maximum w -cycle packing of G (see [14]). From the structural description, Cai et al. [14] obtained not only a min–max theorem—the w -cycle packing number $v(G, w)$ equals the w -cycle covering number $\tau(G, w)$ for any bipartite tournament G with no F_3 nor F_4 —but also strongly polynomial time algorithms for the cycle packing

problem and the feedback vertex set problem (cf. [Definition 20](#)). Cai et al. [14] also showed the NP-completeness of the feedback vertex set problem on general bipartite tournaments. A 4-approximation algorithm for this problem can be obtained by the primal-dual method of Goemans and Williamson [63]. Based on the min–max theorem, Cai et al. [14] improved the ratio from 4 to 3.5. This exhibits another example of applying the duality structures of linear programs to approximation algorithms.

The proof of sufficiency of (i) for \mathcal{H} to be ESP is almost identical to that of [Theorem 18](#). In view of (7), it remains to show the necessity of (i) for $\mathcal{H} = (V, \mathcal{E})$ to be ideal. To this end, suppose that G contains F_i as a subgraph for $i = 3$ or 4. Let vertices $v_1, v_2, \dots \in V(F_i)$ be as specified in [Fig. 3](#), let A be the \mathcal{E} - V incidence matrix of \mathcal{H} , and let integral weight $w \in \mathbb{Z}_+^V$ be defined by

$$w|_{V(F_i)} = \mathbf{1} \text{ and } w|_{V - V(F_i)} = \mathbf{0}.$$

It is easy to see that $x \in \mathbb{Q}_+^V$ with $x|_{\{v_2, v_4, v_6\}} = \frac{1}{2}\mathbf{1}$, $x|_{V(F_i) - \{v_2, v_4, v_6\}} = \mathbf{0}$ and $x|_{V - V(F_i)} = \mathbf{1}$ is a feasible solution to $\min\{w^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\}$ of value $w^T x = 3/2$, giving

$$\min\{w^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\} \leq 3/2. \quad (8)$$

In case of $F_3 \subseteq G$, let C_1 , C_2 , and C_3 be the three 4-cycles in F_3 with vertex sets $\{v_1, v_2, v_3, v_4\}$, $\{v_3, v_4, v_5, v_6\}$, and $\{v_5, v_6, v_1, v_2\}$, respectively. Then $\{C_1, C_2, C_3\}$ is a $2w$ -cycle packing in G of size 3, which, together with the primal and dual relation between packing and covering (cf. (1)), implies

$$\min\{w^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\} \geq 3/2. \quad (9)$$

In case of $F_4 \subseteq G$, let C_1 , C_2 , C_3 , and C_4 be the 4-cycles in F_4 with vertex sets $\{v_1, v_2, v_5, v_6\}$, $\{v_2, v_3, v_6, v_7\}$, $\{v_1, v_4, v_3, v_6\}$, and $\{v_2, v_5, v_4, v_7\}$, respectively. Then $\{C_1, C_2, C_3, C_4\}$ is a $3w$ -cycle packing in G of size 4, implying

$$\min\{w^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\} \geq 4/3. \quad (10)$$

It follows from (8) to (10) that $\min\{w^T x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\} \in [4/3, 3/2]$ is not an integer and \mathcal{H} is not ideal (cf. [Definition 9](#)).

4.4 Matroid Port Hypergraphs

The reader is referred to [88] for an in-depth account of matroid theory and undefined terms. Let M be a matroid on $E \cup \{\ell\}$, where $\ell \notin E$ is a distinguished element of M . A matroid obtained from M by deleting and contracting elements in E is called a *minor* of M using ℓ (cf. [Sect. 2.5\(vi\)](#)).

Definition 21 The ℓ -port of matroid M on $E \cup \{\ell\}$ is the hypergraph $\mathcal{P}_{M,\ell} = (E, \mathcal{E})$ with node set E and hyperedge set $\mathcal{E} = \{P : P \subseteq E \text{ with } P \cup \{\ell\} \text{ a circuit of } M\}$.

As usual, let $U_{2,4}$ be the uniform matroid on four elements of rank two, let F_7 be the Fano matroid, let F_7^* be the dual of F_7 , and let F_7^+ be the unique series extension of F_7 (see Sect. 2.5(vii)–(ix)). In 1977, Seymour [107] characterized all pairs (M, ℓ) for which $\mathcal{P}_{M,\ell}$ is Mengerian.

Theorem 20 Let M be a matroid and let ℓ be an element of M . Then $\mathcal{P}_{M,\ell}$ is Mengerian if and only if M has neither $U_{2,4}$ -minor using ℓ nor F_7^* -minor using ℓ .

This far-reaching theorem, which yields a number of important min–max relations in combinatorial optimization, has attracted tremendous research efforts in matroid optimization; see, for instance, Ding and Zang [37], Gerards and Laurent [62], Guenin [68], Seymour [108], Truemper [111], Tseng and Truemper [112]. Let $\mathcal{P}_{M,\ell} = (E, \mathcal{E})$ be the ℓ -port of matroid M on $E \cup \{\ell\}$, and let A be the \mathcal{E} - E incidence matrix of $\mathcal{P}_{M,\ell}$. A natural extension of the fractional node cover polytope $B(\mathcal{P}_{M,\ell})$ of $\mathcal{P}_{M,\ell}$ (see (4)) is given by dropping the upper constraint as follows:

$$\tilde{B}(\mathcal{P}_{M,\ell}) = \{x : Ax \geq 1, x \geq 0\}.$$

In 1995, Gerards and Laurent [62] obtained a structural characterization of all pairs (M, ℓ) for which $\tilde{B}(\mathcal{P}_{M,\ell})$ is box- $\frac{1}{d}$ -integral (see Definition 6) for all positive integers d ; this theorem can be found in several interesting applications [62, 78]. In 2008, Chen et al. [20] managed to characterize all pairs (M, ℓ) for which $\mathcal{P}_{M,\ell}$ is box-Mengerian; this characterization also yields a number of interesting results in combinatorial optimization. Recently, Chen et al. [17] strengthened this box-TDI property with ESP property (cf. Theorem 14) and presented a much shorter proof than the one given in [20]. The latter proof [17] curtails many technical parts of the original proof [20] and hence is much easier to follow, which exhibits the power of ESP property approach for packing and covering in hypergraphs.

Theorem 21 Let M be a matroid on $E \cup \{\ell\}$ with $\ell \notin E$. Then the following statements are equivalent: (i) M has no $U_{2,4}$ -minor using ℓ , no F_7^* -minor using ℓ , and no F_7^+ -minor using ℓ as a series element; (ii) $\tilde{B}(\mathcal{P}_{M,\ell})$ is box- $\frac{1}{d}$ -integral for all positive integers d ; (iii) $\mathcal{P}_{M,\ell}$ is box-Mengerian; and (iv) $\mathcal{P}_{M,\ell}$ is ESP.

The equivalence of (i) and (ii) was established by Gerards and Laurent [62], and that of (i) and (iii) was verified by Chen et al. [20], who particularly showed that box-total dual integrality of a linear system implies its box- $\frac{1}{d}$ -integrality for all positive integers d (see also Lemma 3). Therefore, (i) \Rightarrow (ii) turns out a corollary of (i) \Rightarrow (iii). The proof of (i) \Rightarrow (iii) [17] is in the same spirit as the proof of (i) \Rightarrow (v) in Theorem 17. For convenience, let \mathfrak{I} be the set of all pairs (M, ℓ) , where M is a matroid on at least two elements, including ℓ , such that M has no

$U_{2,4}$ -minor using ℓ , no F_7^* -minor using ℓ , and no F_7^+ -minor using ℓ as a series element. By the structural result of Gerards and Laurent [62], which says that, for every (M, ℓ) in \mathfrak{I} , either M is regular or M/ℓ is not 3-connected, matroids in \mathfrak{I} can be decomposed into regular matroids. Chen et al. [17] first proved that $\mathcal{P}_{M,\ell}$ is ESP if M is regular and then showed the preservation of ESP property when smaller matroids in \mathfrak{I} associated with ESP properties are composed to form a larger matroid in \mathfrak{I} .

By symmetry, all ports of $U_{2,4}$ (cf. Sect. 2.5(vii)) are isomorphic and so are all ports of F_7^* (cf. Sect. 2.5(x)). These two kinds of ports are precisely clutters C_3 and Q_6 , respectively, defined in Example 1. Similarly, if ℓ_1 and ℓ_2 are the two elements of F_7^+ (cf. Sect. 2.5(ix)) that are in series, then the ℓ_1 -port and ℓ_2 -port of F_7^+ are isomorphic, which are clutter Q_7 in Example 1(iii). The next two corollaries of Theorem 21 [20] are stated in terms of clutters.

Corollary 4 *A matroid port is ESP, if and only if it is box-Mengerian, if and only if none of C_3 , Q_6 , and Q_7 is its minor.*

Ports with no C_3 -minors are known as *binary* clutters, which are precisely ports of binary matroids [106]. Corollary 4 clearly implies the following, which is basically equivalent to Theorem 21.

Corollary 5 *A binary clutter is ESP, if and only if it is box-Mengerian, if and only if neither Q_6 nor Q_7 is its minor.*

By the above two corollaries, it is easy to see that *all bipartite graphs are ESP*, which strengthens the well-known statement that bipartite graphs are Mengerian (see, e.g., [103] page 1399). Moreover, all applications in Sect. 7 of [20] have automatic “strengthening” from “box-TDI” to “ESP.”

Definition 22 A *signed graph* is a pair (G, Σ) of an undirected graph G and a subset $\Sigma \subseteq E(G)$. A cycle C of G is *odd* if $|E(C) \cap \Sigma|$ is odd.

It is well known [62, 107] that the clutter of edge sets of odd cycles in a signed graph (G, Σ) (cf. Definition 7) is a binary clutter, written as $\mathcal{H}_{G,\Sigma}$. Note that Q_6 equals $\mathcal{H}_{K_4,E(K_4)}$ yet Q_7 does not equal any $\mathcal{H}_{G,\Sigma}$. More on signed graphs and odd cycles can be found in [28, 103]; in particular, the reader is referred to Corollary 7.1 of [20] for minor operations on signed graphs. When restricted to signed graphs, Corollary 5 combined with Theorem 20 gives the following.

Remark 2 The clutter of odd cycles in a signed graph (G, Σ) is ESP, if and only if it is box-Mengerian, if and only if it is Mengerian, if and only if (G, Σ) has no $(K_4, E(K_4))$ -minor.

Definition 23 A *graft* (G, T) is an undirected graph G together with a subset $T \subseteq V(G)$ with even $|T|$. A subset of $E(G)$ is a *T-join* if it induces a forest whose

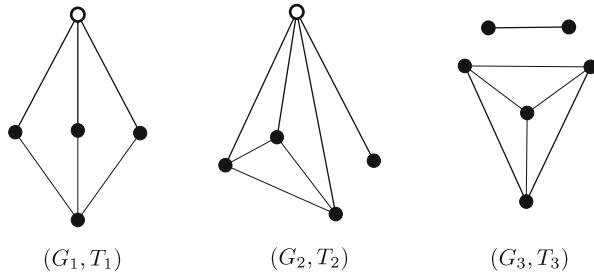


Fig. 4 Excluded grafts for ESP hypergraphs of T -joins

odd-degree vertices coincide with those in T . A subset C of $E(G)$ is called a T -cut if there exists subset $S \subseteq V$ with odd $|S \cap T|$ such that $C = \delta(S)$. cf. [Definition 14](#).

The set of all T -joins (resp. T -cuts) forms a binary clutter [20, 28, 103, 109]. More specifically, one can define a binary matroid M such that the ℓ -port of M is the T -join clutter of G , and the ℓ -port of M^* , the dual of M , is the T -cut clutter of G (see, e.g., [20]). A *minor* of a graft is obtained by performing a sequence of edge deletions and edge contractions (cf. [Sect. 2.5\(ii\)](#)), where in contracting edge uv into a vertex, the vertex is put in new T if exactly one of u and v is in old T . In the context of T -joins and T -cuts, [Corollary 5](#) and [Theorem 20](#) are rephrased as the following characterizations.

Remark 3 The clutter of T -cuts in graft (G, T) is ESP, if and only if it is box-Mengerian, if and only if it is Mengerian, if and only if (G, T) has no $(K_4, V(K_4))$ -minor.

Remark 4 The clutter of T -joins in graft (G, T) is ESP, if and only if it is box-Mengerian, if and only if (G, T) has no (G_i, T_i) -minor, $i = 1, 2, 3$, depicted in [Fig. 4](#), where solid vertices are those in T_i , $i = 1, 2, 3$. The clutter of T -joins in (G, T) is Mengerian if and only if (G, T) has no (G_1, T_1) -minor.

Let $G = (V, E)$ be an undirected graph, and let $\{s_1, t_1\}$ and $\{s_2, t_2\}$ be two pairs of vertices in V with $s_1 \neq t_1$ and $s_2 \neq t_2$.

Definition 24 A *two-commodity path* is an s_1t_1 -path or an s_2t_2 -path. A *two-commodity cut* is a minimal set of edges separating both s_1 from t_1 , and s_2 from t_2 .

As pointed out in [107], two-commodity paths (resp. two-commodity cuts) form a binary clutter; in fact, if M is the matroid on $E \cup \{\ell\}$ such that its ℓ -port is the two-commodity path clutter of G , then the ℓ -port of M^* , the dual of M , is the two-commodity cut clutter of G , and vice versa. [Corollary 5](#) and [Theorem 20](#) provide the following remarks on two-commodity paths and two-commodity cuts, where the

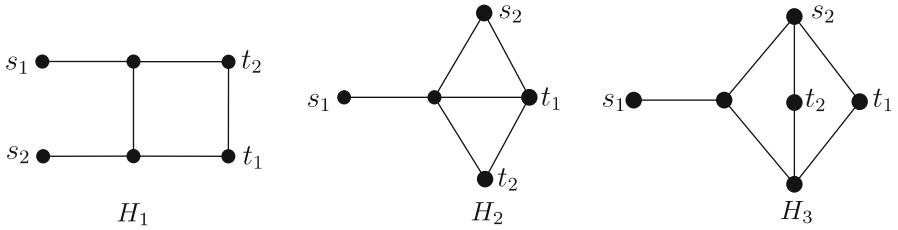


Fig. 5 Excluded graph minors for ESP hypergraphs of two-commodity paths/cuts

exclusion of graphs H_1 , H_2 , H_3 in Fig. 5 is up to permuting indices of s_1, s_2, t_1, t_2 and exchanging s_i with t_i , $i = 1, 2$.

Remark 5 The clutter of two-commodity paths in G is ESP, if and only if it is box-Mengerian, if and only if it is Mengerian, if and only if no subgraph of G can be contracted to H_1 .

Remark 6 The clutter of two-commodity cuts in G is ESP, if and only if it is box-Mengerian, if and only if no subgraph of G can be contracted to H_2 or H_3 . The clutter of two-commodity cuts in G is Mengerian if and only if no subgraph of G can be contracted to H_2 .

4.5 ESP Hypergraphs vs. Min–Max Theorems

A number of known min–max relations enjoy the ESP property (cf. Definition 18), due to their cross-free structures.

Definition 25 A collection Λ of subsets of a ground set U is *cross-free* if, for all $U_1, U_2 \in \Lambda$, one has $U_1 \subseteq U_2$ or $U_2 \subseteq U_1$ or $U_1 \cap U_2 = \emptyset$ or $U_1 \cup U_2 = U$.

Lemma 8 Let Λ be a collection of directed cuts in $D = (V, E)$ (see Definition 15) such that $\{S \subseteq V : C = \delta^{\text{in}}(S) \text{ and } C \in \Lambda\}$ is cross-free. Then Λ admits an equitable subpartition.

Proof Let A be the Λ - E incidence matrix. It can be seen from Corollary 13.21a of [103] that A is totally unimodular. Using (iv) of Corollary 19.3 in [101], collection Λ can be partitioned into two collections Λ_1 and Λ_2 such that the sum of rows of M indexed by Λ_1 minus the sum of rows of M index Λ_2 is a $\{0, \pm 1\}$ -vector. It is easy to see that (Λ_1, Λ_2) is an equitable subpartition of Λ as desired. \square

Definition 26 In a directed graph $D = (V, E)$, a *dicut* is a cut $\delta^{\text{in}}(U)$ with $U \subset V$ (cf. Definition 15) such that no arcs leaves U . Given $r \in V$, an r -*cut* is a cut $\delta^{\text{in}}(U)$ with $U \subseteq V - \{r\}$. Let V be partitioned into two sets R and S . An R - S *bicut* is a cut $\delta^{\text{in}}(U)$ in D with U a subset or a superset of S . Let $D_0 = (V, E_0)$ be a directed

graph such that for each arc rs of D , there exist vertices $r', s' \in V$, and directed rr' -path, $s'r'$ -path, and $s's$ -path in D_0 . A D_0 -cut is a cut $\delta^{\text{in}}(U)$ in D such that no arc in E_0 entering U .

Notice that for $r \in V$, an $\{r\}-(V - \{r\})$ bicut is simply an r -cut. The next corollary summarizes the cross-free essences in Lucchesi–Younger theorem [82], Fulkerson’s optimum arborescence theorem [58], Schrijver’s bibranching theorem, and Shrijver’s strong connector theorem [98].

Corollary 6 *The following hypergraphs are ESP, and so box-Mengerian: (i) the hypergraph of dicuts, (ii) the hypergraph of r -cuts, (iii) the hypergraph of R - S bicuts, and (iv) the hypergraph of D_0 -cuts.*

Proof Let Λ be a collection of hyperedges of the hypergraph in consideration. It was shown that there exists collection Ω of hyperedges of the hypergraph such that $|\Omega| \geq |\Lambda|$, $d_\Omega(v) \leq d_\Lambda(v)$ for all nodes v of the hypergraph, and the collection $\{U \subseteq V : C = \delta^{\text{in}}(U) \text{ and } C \in \Omega\}$ is cross-free (see, e.g., Theorems 55.2, 54.8 and 57.3 in [103]). It can be deduced from Lemma 8 that Ω admits an equitable subpartition, so does Λ . \square

4.6 Blockers of ESP Hypergraphs

This subsection investigates the blockers (see Definition 10) of ESP hypergraphs in the context of ESP property. Similar to Observation Observation 1, it is easy to see that a hypergraph \mathcal{H} is ESP if and only if so is \mathcal{H}^{\min} .

4.6.1 ESP Blockers

A blocking pair (see Definition 10) is said to be *ESP* if both clutters of the pair are ESP.

Corollary 7 *Each of the following form an ESP blocking pair:*

- (i) *The clutter of T -cuts and the clutter of T -joins in grafts with no $(K_4, V(K_4))$ -minor nor (G_i, T_i) -minor, $i = 1, 2, 3$ (see Definition 23 and Fig. 4)*
- (ii) *The clutter of st -paths and the clutter of st -cuts in an undirected graph*
- (iii) *The clutter of edges and the clutter of vertex cover (see Definition 13) in a bipartite graph*

Proof Statement (i) follows from Remarks 3 and 4. When $T = \{s, t\}$, Statement (i) specializes to Statement (ii). Both clutters in (iii) are known to be binary (cf. Chap. 80 of [103]). It is easy to check that neither clutter can have Q_6 or Q_7 as a minor. Hence (iii) follows from Corollary 5. \square

4.6.2 Non-ESP Blockers

Recall from Example 2 that $b(Q_6)$ is Mengerian [103]. It is actually box-Mengerian, as implied by Theorem 15 and the following ESP property.

Remark 7 $b(Q_6)$ is ESP, and so box-Mengerian.

An instant example for “non-ESP blockers of ESP hypergraphs” is the clutter Q_6 (see [Example 1\(ii\)](#)), which is the blocker of the ESP hypergraph $b(Q_6)$ (see [Remark 7](#)). Since Q_6 is non-Mengerian [103, 107], it is non-ESP by [Theorem 15](#). Another example comes from dijoins.

Definition 27 A *dijoin* of a directed graph D is a set of arcs in D that intersects every dicut of D (see [Definition 26](#)).

The clutter of dijoins in a directed graph D is the blocker of the hypergraph of dicuts in D . The former may be non-Mengerian (so non-ESP) as Shrijver’s example [97] shows (cf. [Corollary 8\(iv\)](#)), while the latter is always ESP as [Corollary 6\(i\)](#) asserts.

At this point, it might be more interesting to see if there exist Mengerian blockers of ESP hypergraphs that are not ESP. The examples to be presented for the non-ESP blockers of the ESP-hypergraphs have the common feature described in the next lemma.

Lemma 9 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph with node set $V = \{v_i : i = 1, 2, \dots, |V|\}$. Suppose that \mathcal{E} contains three hyperedges $S_1 = \{v_1, v_4, v_5\}$, $S_2 = \{v_2, v_5, v_6\}$, and $S_3 = \{v_3, v_6, v_4\}$. Then \mathcal{H} is not ESP. If $b(\mathcal{H})$ has four hyperedges whose intersections with $\{v_1, v_2, v_3, v_4, v_5, v_6\}$ are $\{v_1, v_2, v_3\}$, $\{v_1, v_6\}$, $\{v_2, v_4\}$, and $\{v_3, v_5\}$, respectively, then \mathcal{H} is not box-Mengerian.

Proof It is clear that \mathcal{H} is not ESP as hyperedge collection $\{S_1, S_2, S_3\}$ does not admit any equitable subpartition. Let A be the \mathcal{E} - V incidence matrix of \mathcal{H} . Define $w \in \mathbb{Z}_+^V$ and $\mathbf{l}, \mathbf{u} \in \mathbb{Q}_+^V$ by setting $w(v_i)$ to be 1 if $i \leq 6$ and to 0 otherwise, setting $\mathbf{l}(v_i)$ to be $\frac{1}{2}$ if $i \leq 3$ and to 0 otherwise, and setting $\mathbf{u} = \mathbf{1}$. Let $\mathbf{x}_* \in \mathbb{Q}_+^V$ be defined by

$$x_*(v_1) = x_*(v_2) = x_*(v_3) = \frac{1}{2}, \quad x_*(v_4) = x_*(v_5) = x_*(v_6) = \frac{1}{4},$$

$$\text{and } x_*(v_i) = 1 \text{ for all } 7 \leq i \leq |V|.$$

If the vector $A\mathbf{x}_*$ has some coordinate smaller than 1, then \mathcal{H} has a hyperedge $S \in \mathcal{E}$ such that $S \in \{v_4, v_5, v_6\}$ or $S \in \{v_i, v_j\}$ for some $i \in \{1, 2, 3\}$ and some $j \in \{4, 5, 6\}$. Neither case is possible since $\{v_4, v_5, v_6\}$ does not intersect the hyperedge $\{v_1, v_2, v_3\}$ of $b(\mathcal{H})$, and $\{v_i, v_j\}$ does not intersect one of the three hyperedges $\{v_1, v_6\}$, $\{v_2, v_4\}$, $\{v_3, v_5\}$ of $b(\mathcal{H})$. It follows that $A\mathbf{x}_* \geq \mathbf{1}$ and further that

$$\begin{aligned} \mathbf{x}_* \text{ is a feasible solution to } & \min\{\mathbf{w}^T \mathbf{x} : A\mathbf{x} \geq \mathbf{1}, \mathbf{u} \geq \mathbf{x} \geq \mathbf{l}, \mathbf{x} \geq 0\} \\ & \text{with objective value } \mathbf{w}^T \mathbf{x}_* = 2.25. \end{aligned} \tag{11}$$

Observe that the maximization problem $\max\{\boldsymbol{\alpha}^T \mathbf{1} + \boldsymbol{\beta}^T \mathbf{l} - \boldsymbol{\gamma}^T \mathbf{u} : \boldsymbol{\alpha}^T A + \boldsymbol{\beta}^T - \boldsymbol{\gamma}^T \leq \mathbf{w}^T, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma} \geq \mathbf{0}\}$ has a feasible solution $(\boldsymbol{\alpha}_*, \boldsymbol{\beta}_*, \boldsymbol{\gamma}_*)$ with $\alpha_*(S_i) = \frac{1}{2}$ for $i = 1, 2, 3$, $\alpha_*(S) = 0$ for all $S \in \mathcal{E} - \{S_1, S_2, S_3\}$, $\beta_*(v_i) = \frac{1}{2}$ for $i = 1, 2, 3$, $\beta_*(v_i) = 0$ for $4 \leq i \leq |V|$, and $\boldsymbol{\gamma}_* = \mathbf{0}$. By (11), it follows from $\boldsymbol{\alpha}_*^T \mathbf{1} + \boldsymbol{\beta}_*^T \mathbf{l} - \boldsymbol{\gamma}_*^T \mathbf{u} = 2.25 = \mathbf{w}^T \mathbf{x}_*$ that the maximization problem has optimal value 2.25 and therefore cannot have an integral optimal solution as \mathbf{l} and \mathbf{u} are both half-integral. \square

In a directed graph $D = (V, E)$ with $r, s \in V$, vertex r reaches vertex s , or equivalently s is reachable from r if there is an rs -directed path in D . The following definition gives the node covers (see Definition 8) of the r -cut hypergraph and the R - S bicut hypergraph.

Definition 28 Let $D = (V, E)$ be a directed graph with a specified vertex $r \in V$. An r -arborescence in D is a set E' of $|V| - 1$ arcs in E forming a spanning tree (V, E') in which r reaches every vertex of V . Let V be partitioned into two sets R and S . A subset $E' \subseteq E$ is an R - S bibranching in D if in the graph (V, E') , each vertex in S is reachable from a vertex in R , and each vertex in R reaches a vertex in S .

For every directed graph D as described in the above definition, the following (i) and (ii) hold:

- (i) The clutter \mathcal{A}_D of r -arborescences in D (see Definition 28) is the blocker of the hypergraph of r -cuts in D (see Definition 26). The clutter \mathcal{A}_D is Mengerian by Edmonds' disjoint arborescences theorem [39].

More generally,

- (ii) The clutter \mathcal{B}_D of R - S bibranchings in D (see Definition 28) is the blocker of the hypergraph of R - S bicuts in D (see Definition 26). The clutter \mathcal{B}_D is Mengerian by Schrijver's disjoint bibranchings theorem [98].

A directed graph is source-sink connected if for each strong component C not entered by any arc and each strong component C' not left by any arc in the directed graph, some vertex in C reaches some vertex in C' . So an acyclic directed graph is source-sink connected if each source reaches each sink.

Definition 29 Let directed graphs $D = (V, E)$ and $D_0 = (V, E_0)$ be as described in Definition 26. A strong connector for D_0 in D is a subset $E' \subseteq E$ such that the directed graph $(V, E_0 \cup E')$ is strongly connected.

For every pair of directed graphs D and D_0 as described in the above definition, the following holds:

- (iii) The clutter \mathcal{S}_{D_0} of strong connectors of D_0 in D (see Definition 29) is the blocker of the hypergraph of D_0 -cuts in D (see Definition 26). When D_0 is source-sink connected, the clutter \mathcal{S}_{D_0} is Mengerian [98].

A directed graph is *series-parallel* if its underlying undirected graph contains no subdivision of K_4 (cf. Sect. 2.5(iii) and (v)). Given a directed tree T , which is an orientation of an undirected tree, a directed graph $D = (V, E)$ is called a *transitive closure* of T if $V = V(T)$ and $xy \in E$ if and only if x reaches y in T . For every directed graph D , the following holds:

- (iv) The clutter \mathcal{J}_D of dijoins in D (see Definition 27) is the blocker of the hypergraph of dicuts in D (see Definition 26). When D is a series-parallel directed graph [79], or a source-sink connected directed graph [47, 98], or a transitive closure of a directed tree [103], the clutter \mathcal{J}_D is Mengerian.

For undirected graph G , the following holds:

- (v) The clutter \mathcal{F}_G of feedback vertex sets in G (see Definition 20) is the blocker of the hypergraph of cycles (vertex sets of cycles) in G . When G contains no induced subgraph isomorphic to a subdivision of an odd ring or a wheel or $K_{2,3}$ (see Fig. 1), the clutter \mathcal{F}_G is Mengerian [18].

The following corollary of Lemma 9 asserts that none of the above Mengerian blockers is ESP:

Corollary 8 *The following blockers of ESP hypergraphs are all Mengerian but not necessarily box-Mengerian. Nor are they ESP:*

- (i) *The clutter \mathcal{A}_D of r-arborescences in directed graph D*
- (ii) *The clutter \mathcal{B}_D of R-S bibranching in directed graph D*
- (iii) *The clutter \mathcal{S}_{D_0} of strong connectors for sink-source connected directed graph D_0*
- (iv) *The clutter \mathcal{J}_D of dijoins in series-parallel directed graphs D , source-sink connected directed graph D , or transitive closure D of a directed tree*
- (v) *The clutter \mathcal{F}_G of feedback vertex sets in undirected graphs G which contain no induced subgraph isomorphic to a subdivision of an odd ring or a wheel or $K_{2,3}$*

Proof The task is to exhibit examples satisfying Lemma 9. They are depicted in Fig. 6, where symbols v_1, v_2, \dots, v_6 are put close to the hypergraph nodes they represent. They represent arcs in directed graph D and vertices in undirected graph G .

(i) Figure 6(1) presents a directed graph D with root r for which \mathcal{A}_D is neither ESP nor box-Mengerian. (ii) The conclusion follows immediately from (i), since \mathcal{A}_D is a special case of \mathcal{B}_D as shown by setting $R = \{r\}$ and $S = V - \{r\}$ for directed graph $D = (V, E)$ with $r \in V$. With the setting, a minimal R-S bibranching is simply an r -arborescence. (iii) Consider clutter \mathcal{S}_{D_0} of strong connectors in D in Fig. 6(1) for D_0 in Fig. 6(2), where sink r of D_0 is connected to all other vertices (sources). Notice that S_1, S_2 , and S_3 in Lemma 9 are corresponding strong connectors, that is, hyperedges of \mathcal{S}_{D_0} . (iv) The directed graph D in Fig. 6(3) is both series-parallel and source-sink connected. A transitive closure D of directed tree T is shown in Fig. 6(4), where thick arcs are those of T . These directed graphs D give rise to clutters \mathcal{J}_D which satisfy all the conditions of Lemma 9 in place of \mathcal{H} . (v) Clearly the graph $G = (V, E)$ in Fig. 6(5) is free of the forbidden structures and gives the clutter \mathcal{F}_G to which Lemma 9 applies. \square

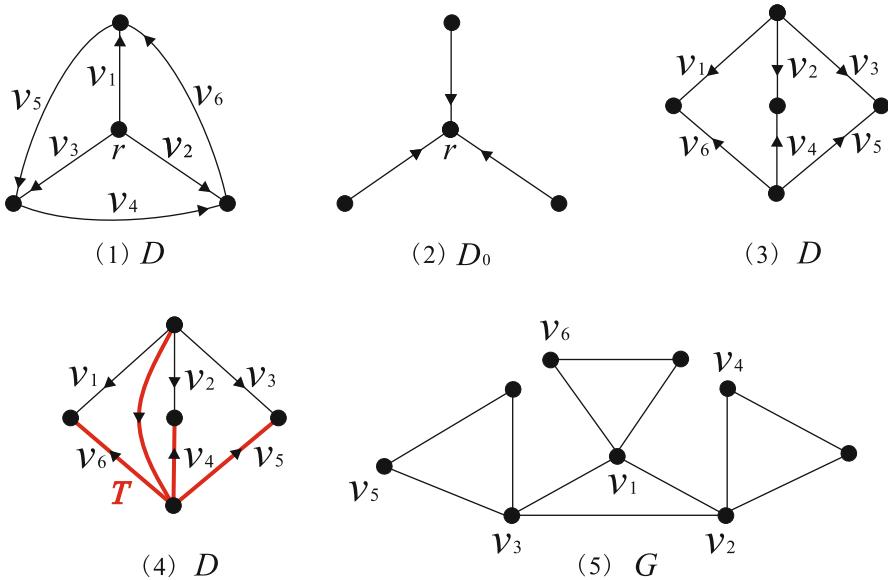


Fig. 6 Non-ESP blockers of ESP hypergraphs

A couple of blocking pairs possess Mengerian properties at the same time, for example, those in [Corollaries 6, 7, and 8](#), despite some exceptions, for example, dicuts vs. dijoins. In view that all hypergraphs in [Corollary 8](#) are Mengerian, it turns out much harder to preserve ESP property under taking blockers.

Remark 8 The ESP property is not closed under taking blockers nor is the box-Mengerian property.

It is even worse that in general no box- $\frac{1}{d}$ -Mengerian property with d bounded can be guaranteed for the blocker of an ESP hypergraph.

[Figure 7](#) gives extensions of the examples in [Fig. 6](#), which lead to the following negative statement on box- $\frac{1}{d}$ -Mengerian property for the Mengerian blockers of ESP hypergraphs.

Remark 9 Let \mathcal{H}_H denote the clutter in (i)–(v) of [Corollary 8](#). For any integer $d \geq 1$, there is graph H such that \mathcal{H}_H is not box- $\frac{1}{d}$ -Mengerian.

Proof In case of (i) and (ii), take H as the directed graph D in [Fig. 7\(1\)](#) with setting $R = \{r\}$ and $S = V(D) - \{r\}$ for case (ii). In case of (iii), take H as D_0 in [Fig. 7\(2\)](#) and consider \mathcal{H}_H the clutter of strong connectors for D_0 in the directed graph D in [Fig. 7\(1\)](#). In case of (iv), take H as D in [Fig. 7\(3\)](#) which is a transitive closure of the directed tree T in [Fig. 7\(4\)](#). In case of (v), consider H as the undirected graph G in [Fig. 7\(5\)](#).

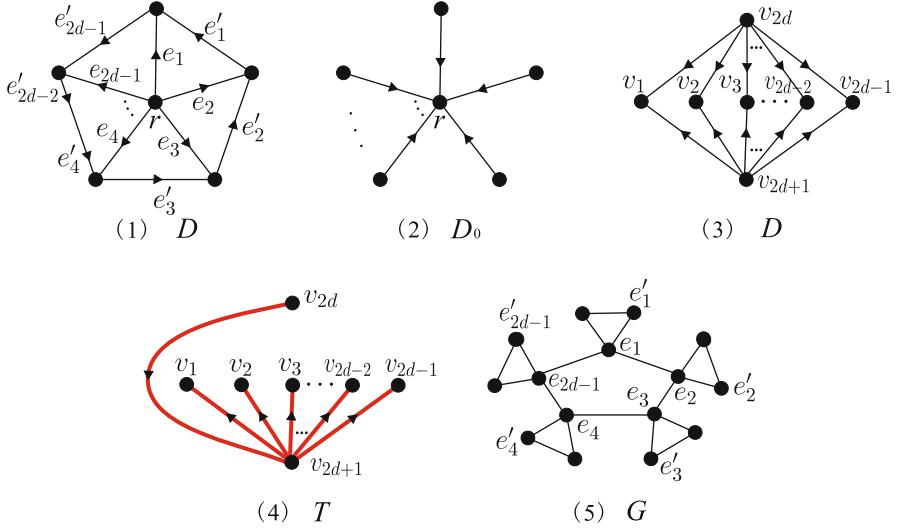


Fig. 7 Mengerian hypergraphs which are not box- $\frac{1}{d-1}$ -Mengerian

To verify the conclusion, investigate $\mathcal{H}_H = \mathcal{J}_D$ for the series-parallel directed graph D in Fig. 7(3). (The other cases of \mathcal{H}_H can be treated similarly). Observe that the directed graph D is an orientation of complete bipartite graph $K_{2,2d-1}$, $d \geq 2$, where $V(D)$ consists of bipartition $\{v_{2d}, v_{2d+1}\}$ of sources and $\{v_1, v_2, \dots, v_{2d-1}\}$ of sinks, and $E(D) = \{e_i, e'_i : i = 1, 2, \dots, 2d - 1\}$ is labeled in such a way that e_i and e'_i are incident with v_{2d} and v_{2d+1} , respectively, and have common head v_i , $i = 1, 2, \dots, 2d - 1$. Think of the clutter $\mathcal{J}_D = (E(D), \mathcal{E})$ of dijoints in D . Define $w, u \in \mathbb{Z}_+^{E(D)}$ and $l \in \mathbb{Q}_+^{E(D)}$ by setting

$$w = \mathbf{1}, u = \mathbf{1} \text{ and } l(e_i) = \frac{2d-3}{2d-2} \text{ and } l(e'_i) = 0 \text{ for } i = 1, 2, \dots, 2d-1.$$

Let A be the \mathcal{E} - $E(D)$ incidence matrix of \mathcal{J}_D . Note that the minimization problem $\min\{w^T x : Ax \geq \mathbf{1}, u \geq x \geq l, x \geq 0\}$ has a feasible solution x_* with $x_*(e_i) = \frac{2d-3}{2d-2}$ and $x_*(e'_i) = \frac{1}{(2d-2)^2}$, $i = 1, 2, \dots, 2d-1$, yielding objective value

$$w^T x_* = \frac{2d-3}{2d-2}(2d-1) + \frac{1}{(2d-2)^2}(2d-1) = \frac{(4d^2-10d+7)(2d-1)}{(2d-2)^2}.$$

On the other hand, D has dijoints $J_i := \{e_i\} \cup \{e'_1, e'_2, \dots, e'_{2d-1}\} - \{e'_i\} \in \mathcal{E}$ for $i = 1, 2, \dots, 2d-1$. It is easy to see that the maximization problem $\max\{\alpha^T \mathbf{1} + \beta^T l - \gamma^T u : \alpha^T A + \beta^T - \gamma^T \leq w^T, \alpha, \beta, \gamma \geq \mathbf{0}\}$ has a feasible solution $(\alpha_*, \beta_*, \gamma_*)$ given by $\alpha_*(J_i) = \frac{1}{2d-2}$ for $i = 1, 2, \dots, 2d-1$, $\alpha_*(J) = 0$ for all $J \in \mathcal{E} - \{J_i : i = 1, 2, \dots, 2d-1\}$; $\beta_*(e_i) = \frac{2d-3}{2d-2}$, $\beta_*(e'_i) = 0$ for $i = 1, 2, \dots, 2d-1$; and

$\gamma_* = \mathbf{0}$. From

$$\begin{aligned}\alpha_*^T \mathbf{1} + \beta_*^T \mathbf{l} - \gamma_*^T \mathbf{u} &= \frac{1}{2d-2}(2d-1) + \frac{2d-3}{2d-2} \cdot \frac{2d-3}{2d-2}(2d-1) \\ &= \frac{(4d^2 - 10d + 7)(2d-1)}{(2d-2)^2} = \mathbf{w}^T \mathbf{x}_*,\end{aligned}$$

it follows that the maximization problem has optimal value $\frac{(4d^2 - 10d + 7)(2d-1)}{(2d-2)^2}$, and hence cannot have any optimal solution that is $\frac{1}{d-1}$ -integral, as $\mathbf{u} = \mathbf{1}$ and \mathbf{l} is $\frac{1}{2d-2}$ -integral. \square

Definition 30 A clutter is *diadic* if each of its hyperedge intersects each of its minimal node cover in at most two nodes.

Consider the Mengerian clutter \mathcal{J}_D of dijoins (cf. Corollary 8(iv)) in the series-parallel directed graph D in Fig. 6(3). It is routine to check that \mathcal{J}_D is diadic. Particularly, \mathcal{J}_D is a circulant clutter [28] with node set $\{u_1, u_2, u_3, u_4, u_5, u_6\}$, where $u_1, u_2, u_3, u_4, u_5, u_6$ are arcs $v_1, v_2, v_3, v_6, v_4, v_5$ in Fig. 6(3), respectively, and hyperedge set $\{\{u_i, u_{i+1}, u_{i+2}\} : i = 1, 2, 3, 4\} \cup \{\{u_5, u_6, u_1\}, \{u_6, u_1, u_2\}\}$. Corollary 8(iv) says that \mathcal{J}_D is not box-Mengerian. This, together with a result of Cornuégols et al. [30], makes the following remark:

Remark 10 Ideal diadic clutters are Mengerian, but not necessarily box-Mengerian.

5 Total Dual Integral Systems

This section briefly reviews TDI systems established for edge covers and stars in Sect. 5.1, for feedback vertex sets and cycles in Sect. 5.2, for stable marriage in Sect. 5.3, for weighted restricted 2-matching in Sect. 5.4, for ordered submodular games in Sect. 5.5, for network design with orientation constraints in Sect. 5.6, for rooted k -connections in Sect. 5.7, and for cyclicly ordered directed graphs in Sect. 5.8. The results in the first two subsections are characterizations of both sufficiency and necessity for systems under investigation to be TDI, while the TDI-ness addressed in the third subsection with respect to stable marriage polytopes can be viewed as a sufficient condition of bipartiteness imposed on general stable matching polytopes to obtain total dual integrality. The fourth subsection provides a TDI description of the convex hull of $\{0, 1, 2\}$ -incidence vector of restricted perfect 2-matchings. The description involves complicate inequalities which have large coefficients. The TDI systems in the fifth subsection helps to prove the nonemptiness of cores of ordered submodular games. The total dual integrality established in the sixth and seventh subsections relies on positive intersecting supmodularity and intersecting supermodularity, where the latter property often yields the polynomial time solvability of the optimization problems over the TDI

systems despite their exponentially large dimension. The TDI systems presented in the last subsection concerns with cycle packing and covering in directed graphs whose vertices are given with cyclic orders, where so-called winding numbers are involved.

Every TDI system $Ax \geq b, x \geq \mathbf{0}$ is associated with one or more beautiful min–max theorems. When A is the hyperedge-node incidence matrix of a hypergraph \mathcal{H} , system $Ax \geq b, x \geq \mathbf{0}$ being TDI amounts to saying that $\tau(\mathcal{H}, w) = v(\mathcal{H}, w)$, that is, there holds the *min–max relation*:

“The minimum weight of a node cover of \mathcal{H} equals the maximum

$$\text{size of a } w\text{-packing of } \mathcal{H}.\quad (12)$$

One of the fundamental topics in combinatorial optimization is to identify scenarios under which the problems of such a covering-packing type enjoy the beautiful min–max relations. Results reviewed in Sects. 5.1 and 5.2 come in this flavor. In addition, both subsections state the characterizations for Mengerian hypergraphs and Mengerian blockers in the same theorem, with the aim of emphasizing the special features of the blocking pairs. Roughly speaking, edge covers and stars discussed in Sect. 5.1 share essentially (but not exactly) the same graphical characterization for Mengerian property, which stands in contrast with feedback vertex sets and cycles whose Mengerian hypergraphs exclude exactly the same forbidden structures, as described in Sect. 5.2.

5.1 Edge Covers and Stars

Two classical min–max theorems on edge cover and edge coloring of bipartite graphs have recently been extended to general graphs.

Definition 31 Let $G = (V, E)$ be a simple undirected graph. A *star* centered at a vertex $v \in V$ consists of all edges incident with v in G .

Let $w \in \mathbb{Z}_+^E$ be a nonnegative integral weight function defined on E . As defined in Schrijver [103], a w -stable set of G is a nonnegative integral vector $y \in \mathbb{Z}_+^V$ satisfying $y_u + y_v \leq w(e)$ for each edge $e = uv \in E$. Recalling Definition 8, it is easy to see that such an integral y corresponds to a w -star packing \mathcal{S} in G such that for each $v \in V$, packing \mathcal{S} contains exactly $y(v)$ copies of the star centered at v . The clutter of edge covers (see Definition 13) and the clutter of stars (see Definition 31) in an undirected graph form a blocking pair (cf. Definition 10). The weighted versions of Kőnig-Rado edge cover theorem [94] (see Theorem 19.4 of [103]) and Gupta’s edge cover packing theorem [70] (see Theorem 20.5 of [103]) state that the hypergraph of stars in a bipartite graph and the hypergraph of edge covers in a bipartite graph are Mengerian, respectively. The bipartiteness sufficient conditions for the total dual integrality (min–max relations (12)) have been recently

improved by Ding et al. [35] who showed that the same min–max relation holds on a graph if and only if the graph is essentially a quasi-bipartite graph (cf. [Definition 17](#)).

Theorem 22 *Let G be an undirected graph. The following statement holds:*

- (i) *For any nonnegative integral weight function $w \in \mathbb{Z}_+^E$, the minimum weight of an edge cover in G is equal to the maximum size of a w -star packing in G (i.e., the hypergraph of stars in G is Mengerian) if and only if G is quasi-bipartite, and no component of G contains K_4 as a spanning subgraph.*
- (ii) *For any nonnegative integral weight function $w \in \mathbb{Z}_+^E$, the minimum weight of a star in G is equal to the maximum size of a w -edge cover packing in G (i.e., the hypergraph of edge covers in G is Mengerian) if and only if G is quasi-bipartite.*

5.2 Feedback Vertex Sets and Cycles

It was proved by Lehman [80] that a clutter is ideal if and only if its blocker is ideal. Since the clutter of feedback vertex sets (abbreviated to FVSs, see [Definition 20](#)) and the clutter of cycles (more accurately, vertex sets of cycles) are blockers of each other, it follows from [Theorems 17](#) and [18](#) that both clutters of FVSs and cycles are ideal if and only if the underlying graphs free of the forbidden subgraphs specified in statements (i) of the two theorems. However, as far as the stronger property of being Mengerian is concerned, a blocking pair may fail to have a common characterization; see, for instance, the blocking pair of the edge cover clutter and the star clutter discussed in [Theorem 22](#) above. In other words, Mengerian property of hypergraphs is not closed under taking blockers (cf. [Theorem 8](#)).

Packing FVSs turns out not so amenable as its “blocker” counterpart—packing cycles whose associated min–max relation has been “characterized” in [Sect. 4.3](#). For instance, the FVS clutter in [Corollary 8\(v\)](#) is neither ESP nor box-Mengerian, while its cycle counterpart in [Theorem 17](#) is. Despite the gap between FVSs and cycles on box-Mengerian property, somewhat surprisingly, the same combinatorial structure assures the Mengerian property for both FVS clutter and cycle clutter, which gives the min–max relations of (12) type, as [Theorems 23](#) and [25](#) below state.

Theorem 23 *The following two statements are equivalent for every simple undirected graph $G = (V, E)$:*

- (i) *For any nonnegative integral weight function $w \in \mathbb{Z}_+^V$, the minimum weight of an FVS in G is equal to the maximum size of a w -cycle packing in G .*
- (ii) *For any nonnegative integral weight function $w \in \mathbb{Z}_+^V$, the minimum weight of a cycle in G is equal to the maximum size of a w -FVS packing in G .*
- (iii) *Graph G contains no induced subgraph isomorphic to a subdivision of an odd ring, or a where, or $K_{2,3}$ (see [Fig. 1](#)).*

For convenience, let \mathfrak{L} denote the class of simple undirected graphs G satisfying (iii). Ding and Zang [36] proved (i) \Leftrightarrow (iii), whose strengthening has been discussed

in [Theorem 17](#), and Chen et al. [18] proved (i) \Leftrightarrow (iii), whose proof relied heavily on the structural characterization of graphs in \mathfrak{L} [36] described as follows.

Let G_1 and G_2 be two undirected graphs. The *0-sum* of G_1 and G_2 is obtained by taking the disjoint union of these two graphs; the *1-sum* is obtained by identifying a vertex of G_1 with a vertex of G_2 . A *2-sum* of G_1 and G_2 is obtained by first choosing a triangle $r_i s_i t_i$ from G_i ($i = 1, 2$) such that t_i has degree two in G_i , then deleting t_i from G_i ($i = 1, 2$), and, finally, identifying $r_1 s_1$ with $r_2 s_2$. A triangle T of an undirected graph G is called *stable* if $G \setminus V(T)$ is connected and every vertex in T has degree at least three in G . A *3-sum* of G_1 and G_2 is obtained by identifying a stable triangle in G_1 with a stable triangle in G_2 .

A *rooted graph* consists of an undirected graph G and a specified set R of edges such that each edge in R belongs to a triangle and each triangle in G contains at most one edge from R . The operation of *adding pendent triangles* to the rooted graph G works as follows: For each edge uv in R , introduce a new vertex t_{uv} and two new edges ut_{uv} and vt_{uv} . The following reformulation of Theorem 3.1 in [36] gives the “decomposition” characterization of graphs in \mathfrak{L} .

Theorem 24 *For any graph $G \in \mathfrak{L}$, either (a) G is a k -sum of two smaller graphs, for $k = 0, 1, 2, 3$; or (b) G is obtained from a rooted 2-connected line graph by adding pendent triangles. (In the latter case (b), G is referred to as a prime graph).*

As mentioned at the beginning of [Sect. 5.2](#), condition (iii) $G \in \mathfrak{L}$ is necessary for the idealness of the FVS clutter of G , so by [Corollary 1](#) condition (iii) must be necessary for the min–max relation (i) which says that the clutter is Mengerian. To verify sufficiency of (iii) for (i), Chen et al. [18] resorted to [Theorem 24](#), which asserts that every graph in \mathfrak{L} can be constructed from *prime graphs* in (b) by “summing” operations in (a). In comparison with the proof for ESP cycle hypergraphs in [36] (cf. [Theorem 17](#)) which adopted the same decomposition-sum strategy by virtue of [Theorem 24](#), more complicated arguments and techniques were used in [18] to prove that FVS clutter being Mengerian is preserved under the “summing” operations and all prime graphs have Mengerian FVS clutters. A trade-off here is the algorithmic consequence.

- The nonconstructive proof by Ding and Zang [36] does not imply an algorithm for the cycle packing problem (see [Definition 20](#)) on graphs in \mathfrak{L} .
- The constructive proof by Chen et al. [19] can be converted into a polynomial time algorithm to find a maximum FVS w -packing for any graph $G = (V, E) \in \mathfrak{L}$ and any $w \in \{0, 1\}^V$. Roughly, the algorithm starts with an arbitrary collection \mathcal{F} of subsets of V such that $|\mathcal{F}|$ equals the minimum weight of a cycle G and every vertex v belongs to exactly $w(v)$ members of \mathcal{F} . Then the algorithm adjusts the collection to make it more efficient, meaning to make sets in, that is, members of, \mathcal{F} to meet as many induced cycles as possible while keeping the size $|\mathcal{F}|$ of the collection unchanged. The algorithm first makes members of \mathcal{F} meet all triangles and then other cycles in G . One of the key steps for making these adjustments is to find a better partition of the union of two members of \mathcal{F} (cf. Lemma 4.1 of [18]), where the structure stated in [Theorem 24\(b\)](#) plays

an important role. Finally the algorithm terminates when \mathcal{F} turns out to be a w -packing of FVSs in G and therefore a maximum one in view of the invariant $|\mathcal{F}|$ which equals the w -covering number (cf. inequality (2)).

Although in general the blocker of a Mengerian hypergraph does not have to be Mengerian (see, e.g., Remark 7 for Q_6), the famous max-flow min-cut theorem and a Fulkerson theorem [56] (see page 115 of [101]) assert that both the hypergraph of st -paths in a graph and its blocker are Mengerian (Corollary 7(ii) in this chapter strengthens the result), so are the hypergraph of r -arborescences and its blocker by Edmond's disjoint arborescence theorem [39] and Fullerson's optimum arborescence theorem [58] (cf. Corollary 8 in this chapter). These classes of hypergraphs \mathcal{H} certainly satisfy the condition:

$$\text{“}\mathcal{H}\text{ is Mengerian if and only if so is } b(\mathcal{H})\text{.”}$$

However, the satisfaction is trivial in the sense that no hypergraph in these classes are non-Mengerian. In contrast, Theorem 23 exhibits nontrivial satisfaction by the class of FVS hypergraphs and the class of cycle hypergraphs on undirected graphs, so does the following theorem on cycle packing and FVS packing in tournaments (cf. Sect. 4.3.2).

Theorem 25 *The following two statements are equivalent for every tournament $G = (V, E)$:*

- (i) *For any nonnegative integral weight function $w \in \mathbb{Z}_+^V$, the minimum weight of a FVS in G is equal to the maximum size of a w -cycle packing in G .*
- (ii) *For any nonnegative integral weight function $w \in \mathbb{Z}_+^V$, the minimum weight of a cycle in G is equal to the maximum size of a w -FVS packing in G .*
- (iii) *Tournament G has no subtournament isomorphic to F_1 nor F_2 (see Fig. 2).*

Note that the word “cycle” in this theorem can be replaced with “triangle,” meaning a directed cycle of length three in the tournament. The equivalence (i) \Leftrightarrow (iii) was established by Cai et al. [13] who obtained a structural description of tournaments satisfying (iii) (cf. Theorem 18 and the two paragraph succeeding it). Based on the structural description, Chen et al. [21] proved (i) \Leftrightarrow (iii).

Corollary 9 *Let \mathcal{H} be the FVS (resp. cycle) hypergraph on a undirected graph or a tournament. Then \mathcal{H} is Mengerian if and only if $b(\mathcal{H})$ is Mengerian.*

Complementary to the NP-hardness of the FVS packing problem on tournaments (cf. Theorem 13), Chen et al. [21] designed an exact algorithm running in quartic time for the FVS packing problem on tournaments with no F_1 nor F_2 . By exploiting the structural characterization given in Theorem 25 and using the exact algorithm as a subroutine, the authors provided a $\frac{2}{5}$ -approximation algorithm for the FVS packing problem on general tournaments based on the subgraph removal technique.

5.3 Stable Marriage Polytopes

Let $G = (V, E)$ be a graph. For each $v \in V$, let \prec_v be a strict linear order on $\delta(v)$ (cf. [Definition 14](#)). Let \prec be the collection of all these \prec_v for $v \in V$. The pair (G, \prec) is referred to as a *preference system*, and a *bipartite preference system* if G is a bipartite graph. An edge $e \in E$ is said to *dominate* an edge $f \in E$ if they have a common end v such that $e \preceq_v f$, meaning either $e = f$ or $e \prec_v f$. A matching of G is called *stable* if each edge of G is dominated by some edge in the matching. The *stable matching problem* is to determine if a given preference system admits a stable matching. The origin of this problem can be traced back to 1962 when Gale and Shapley [59] proposed the well-known *stable marriage problem* (which corresponds to the case of bipartite preference system); since then the stable matching problem and its variants have been subjects of extensive research. Of particular interest is the *minimum weight stable matching problem*, which takes a preference system (G, \prec) and an integral weight function w defined on $E(G)$ as input, and aims to find a stable matching of (G, \prec) with minimum total weight as output. This problem is NP-hard in general as shown by Feder [46] and is polynomial time solvable for bipartite preference system, since it was discovered by Rothblum [95] that the convex hull of stable matchings of a bipartite preference system can be described by a very simple system of linear inequalities, that is, [\(14\)](#)–[\(16\)](#) below. More specifically, the *minimum weight stable marriage problem* amounts to solving the following LP, where $\varphi(e)$ denotes the set of all edges of $G = (V, E)$ that dominate edge e :

$$\min \quad w^T x \tag{13}$$

$$\text{s.t.} \quad x(\delta(v)) \leq 1 \quad \forall v \in V \tag{14}$$

$$x(\varphi(e)) \geq 1 \quad \forall e \in E \tag{15}$$

$$x \geq \mathbf{0} \tag{16}$$

The integrality of Rothblum's system [\(14\)](#)–[\(16\)](#) with respect to bipartite preference system has recently been strengthened to total dual integrality (cf. [Theorem 1](#)) by Király and Pap [76], as the following theorem states:

Theorem 26 *If (G, \prec) is a bipartite system, then linear system [\(14\)](#)–[\(16\)](#) is TDI.*

Rothblum's system does not seem to fit easily into the known general framework for deriving total dual integrality, such as submodular flows and lattice polyhedra. Király and Pap [76] proved the TDI-ness by showing that an integral optimal solution to the LP-dual of the minimization [\(13\)](#)–[\(16\)](#) can be derived from the dual solutions of the associated rotation system, which has a totally unimodular matrix. Their constructive proof gives rise to a strongly polynomial algorithm for finding an integral optimal dual solution.

It should be noted that the system of Rothblum does not have the property that the two types of inequalities [\(14\)](#) and [\(15\)](#) separately define TDI systems (which holds, e.g., in the case of polymatroid intersection, or for any system described by

a totally unimodular matrix). In fact, neither the system of (14) and (16) nor the system of (15) and (16) is integral (cf. Theorem 1). This can be seen, respectively, from a triangle with all one weight and the example given by Király and Pap [76].

5.4 Weighted Restricted 2-Matching Polytopes

Let $G = (V, E)$ be an undirected graph, with possible loops or parallel edges. Fix an arbitrary collection \mathcal{C} of some (maybe none) cycles in G , each of which has an odd length in terms of the number of edges, and is called an *odd cycle* in G . A loop is regarded as an odd cycle.

Definition 32 A \mathcal{C} -*matching* of G is the edge set of a subgraph of G the components of which are single edges and cycles in \mathcal{C} . A \mathcal{C} -matching is *perfect* if it covers all vertices in V . The $\{0, 1, 2\}^E$ -*incidence vector* x of a \mathcal{C} -matching M is given by $x|_{E-M} = \mathbf{0}$, $x(e) = 1$ if edge $e \in E$ is in a cycle component of M , and $x(e) = 2$ if $\{e\}$ is a single edge component of M .

This subsection makes no difference between the \mathcal{C} -matching and its $\{0, 1, 2\}^E$ -incidence vector. Let P denote the convex hull of perfect \mathcal{C} -matchings, which is referred to as the *perfect \mathcal{C} -matching polytope* of G . When $\mathcal{C} = \emptyset$, the integral vectors multiplied by $1/2$ in P are precisely the perfect matchings of G , and $\frac{1}{2}P$ is known as the *perfect matching polytope* of G . When \mathcal{C} consists of all odd cycles in G , the integral vectors in P are precisely the perfect 2-matchings of G , and P is called the *perfect 2-matching polytope* of G . So, a *perfect 2-matching* of G is a vector assigning values 0, 1, or 2 to the edges in E such that the sum of values of edges incident with any vertex in V is equal to 2.

Pulleyblank and Edmonds [93] gave a TDI description of the perfect matching polytope (see, e.g., (25.20) and Corollary 25.4b of [103]), which is essential to solve the minimum weight perfect matching problem. This approach has been generalized to extensions of matching, one of which is a TDI description of the triangle-free perfect 2-matching polytope by Cornuéjols and Pulleyblank [32] (see also Theorem 30.14 of [103]), and a strongly polynomial time algorithm to find a minimum weight triangle-free perfect 2-matching. Thus, a natural question is whether one can also find a minimum weight pentagon-free perfect 2-matching, which was proposed by Schrijver [103] (see page 544), with the remark that the approach to the triangle-free problem does not extend to the pentagon-free problem. In 2009, Pap [90] solved the pentagon-free problem, and, more generally, solved a whole bunch of weighted restricted 2-matching problems.

A main result of [90] is a complete TDI description of restricted 2-matchings given Theorem 27 below. The polyhedral description is quite technical and involves so-called valid systems to be defined via a couple of definitions as follows.

An undirected graph $G = (V, E)$ is called *factor-critical* if the deletion of any vertex from G leaves a graph with a perfect matching. Given $F \subseteq E$, the set of ends of edges in F is written as $V(F)$; given $U \subseteq V$, the set of edges in E with both ends in U is written as $E(U)$.

Definition 33 A family of sets is called *laminar* if any two of the sets are disjoint or one of them is a subset of the other.

Definition 34 Let $G = (V, E)$ be an undirected graph. A set \mathcal{L} is called a *nice family* if the following conditions are satisfied:

- (i) $\mathcal{L} \subseteq 2^E$ is a laminar family.
- (ii) The subgraph $G(F) = (V(F), F)$ is factor-critical for all $F \in \mathcal{L}$.
- (iii) $\{V(F) : F \in \mathcal{L}\}$ is a laminar family.
- (iv) For every vertex $v \in V$, there exists a matching M in G such that v is exposed in M , and $|M \cap F| = (|V(F)| - 1)/2$ for all $F \in \mathcal{L}$.

A pair $(\mathcal{L}, \mathbf{m})$ of nice family \mathcal{L} and a nonnegative function $\mathbf{m} \in \mathbb{R}_+^{\mathcal{L}}$ is called a *nice system*. Given a set \mathcal{C} of some odd cycles in G , a nice system $(\mathcal{L}, \mathbf{m})$ is called a *valid* system if $\sum_{F \in \mathcal{L}} m(F)x(F) \leq \sum_{F \in \mathcal{L}} m(F)(|V(F)| - 1)$ holds for any \mathcal{C} -matching x of G (cf. [Definition 32](#)).

Theorem 27 Let $G = (V, E)$ be an undirected graph and \mathcal{C} be a set of odd cycles in G . The following is a TDI description of the perfect \mathcal{C} -matching polytope of G , which has integer coefficients:

$$\begin{aligned} x(\delta(v)) &= 2 && \forall v \in V \\ x(E(u)) &\leq |U| && \forall U \subseteq V \\ \sum_{F \in \mathcal{L}} m(F)x(F) &\leq \sum_{F \in \mathcal{L}} m(F)(|V(F)| - 1) && \forall \text{ valid system } (\mathcal{L}, \mathbf{m}) \text{ with } \mathbf{m} \in \mathbb{Z}_+^{\mathcal{L}} \end{aligned} \tag{17}$$

The TDI description also may be regarded as a generalization of its unweighted min–max formula [\[31\]](#) (see also Theorem 5 of [\[90\]](#)). While most known TDI descriptions only have 0, 1, 2 or 0 ± 1 coefficients, the last set of inequalities, called *valid inequalities*, in the above system (17) has arbitrarily large coefficients. To get a finite TDI description with integer coefficients, one may only put the valid inequalities for $(\mathcal{L}, \mathbf{m})$ corresponding to a Hilbert base of some $\{\mathbf{m} \geq \mathbf{0} : (\mathcal{L}, \mathbf{m}) \text{ is valid}\}$.

Besides the large coefficients, system (17) is not given explicitly; thus, polynomial time separation or optimization is not straightforward. In order to have such an algorithm, one has to specify explicitly the odd cycles in \mathcal{C} . On the other hand, the following algorithmic result of [\[90\]](#) shows the strong polynomial time solvability for a bunch of special cases.

Theorem 28 Let k be a fixed nonnegative integer. Given undirected graph G with edge-weights and a set \mathcal{C} of odd cycles in G , suppose that either:

- (i) \mathcal{C} contains all odd cycles longer than k and maybe some shorter cycles.
- (ii) \mathcal{C} can be enumerated in polynomial time (e.g., every member of \mathcal{C} has length less than k).

Then there is a strongly polynomial time algorithm for finding a minimum weight perfect \mathcal{C} -matching of G .

The solved special cases stated in [Theorem 28](#) include minimum weight perfect matching, minimum weight triangle-free and/or pentagon-free perfect 2-matching, and several other relaxations of the traveling salesman problem.

[Theorems 27](#) and [28](#) were proved in [90] by a primal-dual method, using the polynomial time algorithm of Cornuéjols and Hartvigsen [31] for the unweighted (i.e., unit weight) case as a subroutine. Specifically, the following principle was applied iteratively: maintain a dual solution, and either find a primal solution that satisfies the complementary slackness conditions (which is equivalent with an instance of the unweighted problem) or improve on the dual solution (which is done by using the unweighted min–max formula [31]). To obtain a polynomial time algorithm, Pap [90] showed how to implement oracles to test critical subgraphs in strongly polynomial time.

5.5 Ordered Submodular Games

Faigle and Kern [45] introduced a general order theoretic LP model, which combined with total dual integrality finds applications to the existence of the core in cooperative game theory, among others.

Let E be a finite set and let (E, \preceq) be a (partial) order on E . An element $e \in S$ is called an *upper neighbor* of the other element $f \in E$ if $f \prec e$ and there is no $g \in E$ with $f \prec g \prec e$. An *antichain* is a set of pairwise incomparable elements with respect to \prec . Let \mathcal{S} denote the set of antichains of (E, \preceq) . A distributive lattice $(\mathcal{S}, \vee, \wedge)$ can be defined by for all $S, T \in \mathcal{S}$ setting

$$\begin{aligned} S \vee T &= \text{the set of maximal elements of } \{e \in E : e \prec f \text{ for some } f \in S \cup T\}, \\ S \wedge T &= \text{the set of maximal elements of } \{e \in E : e \prec f \text{ and } e \prec g \text{ for} \\ &\quad \text{some } f \in S, g \in T\}. \end{aligned} \tag{18}$$

Let $b : \mathcal{S} \rightarrow \mathbb{R}$ or equivalently $\mathbf{b} \in \mathbb{R}^{\mathcal{S}}$ be a function on \mathcal{S} . The function \mathbf{b} is said to be *weakly increasing* relative to (E, \preceq) if for every $e \in E$ with at least two upper neighbors, the following property holds: $b(S) < b(S \cup \{e\})$ for every $S \in \mathcal{S}$ with $S \cup \{e\} \in \mathcal{S}$. The function \mathbf{b} is said to be *submodular* relative to (E, \preceq) if

$$b(S \vee T) + b(S \wedge T) \leq b(S) + b(T) \text{ for all } S, T \in \mathcal{S}.$$

Theorem 29 Let A be the \mathcal{S} - E incidence matrix of (E, \preceq) . If $\mathbf{b} \in \mathbb{R}^{\mathcal{S}}$ is a submodular and weakly increasing function on \mathcal{S} , then the following hold:

- (i) $Ax \leq b$ is a box-TDI system.
- (ii) For any function $c \in \mathbb{R}^E$, both $\max\{c^T x : Ax \leq b\}$ and the dual program $\max\{y^T b : y^T A = c, y \geq 0\}$ can be solved optimally in time polynomial in the dimension of A .

Proof It can be seen from the proof of Faigle and Kern [45, page 489–490] that the condition of [Theorem 2](#) has been satisfied, which establishes (i). The matroid-type greedy algorithm [44] (see also the (dual) greedy algorithm in [45]) finds the optimal solutions for both programs in (ii). \square

Similarly, Faigle and Kern [45] generalized the intersection theorem of Edmonds and Giles [41] from unordered ground sets to rooted forests. If an order (E, \preceq) is a rooted forest, then every function $b \in \mathbb{R}^S$ is trivially weakly increasing because every element of E has at most one upper neighbor relative to (E, \preceq) .

Theorem 30 Let A be the S - E incidence matrix of rooted forest (E, \preceq) . If $g, h \in \mathbb{R}^S$ are submodular and $b \in \mathbb{R}^S$ satisfies $b(S) = \min\{g(S), h(S)\}$ for all $S \in \mathcal{S}$, then $Ax \leq b$ is a box-TDI system.

The basic model of cooperative game theory comprises a set N of players. Subsets of N are called *coalitions*. There is a *characteristic function* $v : 2^N \rightarrow \mathbb{R}$ that assigns to each coalition S its value $v(S)$, where $v(\emptyset) = 0$. One may think of $v(S)$ as the *cost* generated by S . A solution concept is a method to divide the total cost $v(N)$ among the individual players in a “fair” way. The concept of the *core* [86] suggests to allocate a vector $x \in \mathbb{R}^N$ such that no coalition S is allocated more than its true cost $v(S)$. The *core* is defined as $Core(v) = \{x \in \mathbb{R}^N : x(S) \leq v(S) \text{ for all } S \subset N \text{ and } x(N) = v(N)\}$. Given a subfamily \mathcal{E} of essential coalitions, v induces the characteristic function $\tilde{v} : 2^N \rightarrow \mathbb{R}$ via

$$\tilde{v}(S) = \min\{\sum_j v(E_j) : E_j \in \mathcal{E} \text{ and } E_j \text{'s partition } S\} \text{ for every } S \subseteq N \quad (19)$$

with the understanding that $\tilde{v}(\emptyset) = 0$ and $\tilde{v}(S) = \infty$ if S cannot be partitioned into members of \mathcal{E} . The cooperative game (N, \tilde{v}) arising from (N, \mathcal{E}, v) is called a *partition game*. The following necessary and sufficient condition [45] for nonempty core states the existence condition of an integral optimal solution for a related LP.

Theorem 31 Let (N, \mathcal{E}, v) be a partition game, and let A be the \mathcal{E} - N incidence matrix. Then $Core(\tilde{v}) \neq \emptyset$ if and only if $\min\{(v|_{\mathcal{E}})^T y : y^T A = \mathbf{1}, y \geq 0\}$ has an integral optimal solution.

This theorem can be proved easily by using \tilde{v} ’s definition (19) and observing that nonempty $Core(\tilde{v})$ is exactly the set of optimal solution to the LP dual of $\min\{(v|_{\mathcal{E}})^T y : y^T A = \mathbf{1}, y \geq 0\}$. The following turns out a corollary of [Theorems 29, 1](#), and [31](#).

Corollary 10 Let $(N, \mathcal{E}, \mathbf{v})$ be a partition game. If the value function $\mathbf{v} \in \mathbb{R}^{\mathcal{E}}$ is submodular and weakly increasing relative to some (partial) order (N, \preceq) , then $\text{Core}(\tilde{\mathbf{v}}) \neq \emptyset$.

Along the same line, games whose value functions satisfy the conditions of [Theorem 30](#) also have nonempty cores.

Corollary 11 Let $(N, \mathcal{E}, \mathbf{v})$ be a partition game. If there exist rooted forest (N, \preceq) whose set of antichains is \mathcal{E} and submodular functions $\mathbf{g}, \mathbf{h} \in \mathbb{R}^{\mathcal{E}}$ such that the value function $\mathbf{v} \in \mathbb{R}^{\mathcal{E}}$ satisfies $v(S) = \min\{g(S), h(S)\}$ for all $S \in \mathcal{E}$, then $\text{Core}(\tilde{\mathbf{v}}) \neq \emptyset$.

The first prerequisite for applying [Corollaries 10](#) and [11](#) to cooperative games is to define appropriate partial orders (N, \preceq) which are seldom given in original game settings but constitutes basis where submodularity and weak increasing property can be discussed. Faigle and Kern [\[45\]](#) illustrated an application of [Corollary 11](#) to a generalization of the classical assignment games to three dimensions. Generalizations to higher dimensions are then straightforward to obtain.

Example 3 (3-Dimensional-Assignment Game). Let the player set N be the disjoint union of three sets I , J , and K , where $|I| = |J| = |K| = n$. The players are to form teams (i, j, k) with one member from each set. The “cost” incurred at forming such a team is $v(i, j, k) \geq 0$. How should the teams be formed and the total cost be allocated among the $3n$ players in the best possible way?

Assume that I , J , and K are geometrically represented as points on three parallel lines in \mathbb{R}^2 and the points are ordered in the same direction along the lines. Each team (i, j, k) is associated with a real number $\gamma(i, j, k) \in \mathbb{R}_+$ equal to the Euclidean circumference of the triangle on the three points corresponding to i, j, k , respectively. The number $\gamma(i, j, k)$ can be interpreted as a possible cost of forming team (i, j, k) .

Define partial order (N, \preceq) to be the union of three unrelated copies of the chain $\{1 < 2 < \dots < n\}$, one for each of I, J, K . Clearly, (N, \preceq) is a rooted forest, and the set $\mathcal{E} = \{(i, j, k) : 1 \leq i, j, k \leq n\}$ of all 3-element antichains is closed under the operations \vee and \wedge (cf. [\(18\)](#)) relative to (N, \preceq) . Moreover $\gamma(i, j, k)$ is a submodular function relative to (N, \preceq) . If the constant $\varsigma \geq 0$ offers an alternative cost per team (due to another cost scheme for which a team may opt), then the induced cost function $v(i, j, k) = \min\{\varsigma, \gamma(i, j, k)\}$, $(i, j, k) \in \mathcal{E}$ satisfies the conditions of [Corollary 11](#), which implies a nonempty core $\text{Core}(\tilde{\mathbf{v}})$ of the partition game $(N, \mathcal{E}, \mathbf{v})$.

5.6 Orientation Constrained Network Design

Khanna et al. [\[75\]](#) studied directed network design problems on mixed networks with orientation constraints in the following more general framework: Given a

directed graph $G = (V, E)$ with a *cost* function $\mathbf{w} \in \mathbb{Z}^E$, a *requirement* function $\mathbf{r} : 2^V \rightarrow \mathbb{Z}$ defined over all subsets of V , and \mathcal{R} a family of disjoint *constrained* arc pairs (e, e^-) , each of which induces a *digon*, that is, two arcs $e, e^- \in E$, arcs in E and pairs in \mathcal{R} are associated with nonnegative integral upper bounds given by $\mathbf{u}, \mathbf{l} \in \mathbb{Z}_+^{E \cup \mathcal{R}}$. The *orientation constrained network design* (OCND) problem consists of finding an optimal integral solution of the following LP, where $\delta^{\text{out}}(S)$ denotes the set of arcs in E that leave $S \subseteq V$:

$$\min \quad \mathbf{w}^T \mathbf{x} \quad (20)$$

$$\text{s.t.} \quad x(\delta^{\text{out}}(S)) \geq r(S) \quad \forall S \subset V \quad (21)$$

$$u((e, e^-)) \geq x(\{e, e^-\}) \geq l((e, e^-)) \quad \forall (e, e') \in \mathcal{R} \quad (22)$$

$$\mathbf{u}|_E \geq \mathbf{x} \geq \mathbf{l}|_E \quad (23)$$

A typical setting that falls within the general OCND framework was described in [75]: given a mixed graph $G = (V, F \cup E)$, where F consists of (undirected) edges and E consists of (directed) arcs. An orientation of an undirected edge is obtained by replacing it by one of two possible directed arcs parallel to it. Network designers often wish, at minimum cost, to fulfill a given connectivity requirement in a network by selecting a subgraph and orienting its undirected edges (i.e., satisfying orientation constraint $x(\{e, e'\}) = 1$). The typical case generalizes a couple of well-known NP-hard problems, including the problem of finding a minimum cost 2-edge-connected subgraph of an undirected graph and the problem of finding a minimum cost strongly-connected subgraph of a directed graph. In this subsection, the set function $r : 2^V \rightarrow \mathbb{Z}$ is always assumed to satisfy $r(\emptyset) = r(V) = 0$.

Definition 35 On a ground set V , two subsets $X, Y \subseteq V$ are called *intersecting* if none of $X - Y$, $Y - X$, and $X \cap Y$ is empty. A set function $r : 2^V \rightarrow \mathbb{R}$ is *intersecting supermodular*, if $r(X) + r(Y) \leq r(X \cap Y) + r(X \cup Y)$ for any intersecting pair $X, Y \subseteq V$; it is *positively intersecting supermodular* if $r(X) + r(Y) \leq r(X \cap Y) + r(X \cup Y)$ holds whenever $r(X) > 0, r(Y) > 0$ and $X, Y \subseteq V$ are intersecting.

Khanna et al. [75] proved that if the requirement function in the OCND problem is positively intersecting supermodular, then the polyhedron $\{\mathbf{x} \in \mathbb{R}^E : \mathbf{x}$ satisfies (21)–(23)\} is integral. Actually, by virtue of Theorem 2, their argument [75] suffices to prove the following stronger statement:

Theorem 32 *If the requirement function $r : 2^V \rightarrow \mathbb{Z}_+$ is positively interesting submodular, then the system (21)–(22) is box-TDI.*

The integrality of $\{\mathbf{x} \in \mathbb{R}^E : \mathbf{x}$ satisfies (21)–(23)\} naturally suggests the use of the ellipsoid method together with an oracle for \mathbf{r} to design a polynomial time algorithm for the OCND problem. As remarked in [75], the LP (20)–(23) needs

not be solvable in polynomial time for every positively intersecting supermodular function, although this is the case for most common applications. This is because usually such functions arise from intersecting supermodular functions by truncating them to be nonnegative. However, it was shown that this is not true for all such functions r .

In contrast to the “intractability” in connection with positively intersecting submodularity, intersecting submodular requirement functions have a stronger and nicer property, by which the OCND problem can be solved to optimality in polynomial time. Moreover, the results can be extended from graphs to directed hypergraphs, as shown by Frank et al. [53].

Definition 36 A *directed hypergraph* is a pair $\mathcal{H} = (V, \mathcal{E})$, where V is a finite set whose elements are called *nodes*, and \mathcal{E} a finite collection of so-called *hyperarcs* (possibly with repetition). A hyperarc $a \in \mathcal{E}$ consists of a subset $S_a \subseteq V$ and a designated head node $h_a \in S_a$.

Clearly, a directed graph is a directed hypergraph where every hyperarc has two nodes. Let $\mathcal{H} = (V, \mathcal{E})$ be a directed hypergraph and let $S \subseteq V$. It is convenient to view collection $\mathcal{E}|_V = \{S_a : a \in \mathcal{E}\}$ as the “restriction” of \mathcal{E} to V that contain underlying hyperedges, and consider $\delta^{\text{in}}(S) = \{S_a \in \mathcal{E}|_V : h_a \in S \text{ and } S_a \not\subseteq S\}$ consisting of hyperarcs that “enter” S . [Theorem 32](#) is clearly a corollary of the following more general result by Frank et al. [53], which also treats function property for tractability.

Theorem 33 Let $\mathcal{H} = (V, \mathcal{E})$ be a directed hypergraph for which $\mathcal{E}|_V$ can be partitioned into $\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_k$ such that sets (underlying hyperedges) in \mathcal{E}_i are the same for $i = 1, 2, \dots, k$. Given integers $u_i \geq l_i$, $i = 1, 2, \dots, k$, and set function $r : 2^V \rightarrow \mathbb{Z} \cup \{-\infty\}$, if r is positively intersecting submodular, then the linear system

$$\begin{aligned} x(\delta^{\text{in}}(S)) &\geq r(S) \quad \forall S \subseteq V \\ u_i &\geq x(\mathcal{E}_i) \geq l_i \quad \forall i = 1, 2, \dots, k \end{aligned} \tag{24}$$

is box-TDI. If r is intersecting submodular, then for any given $w, \tilde{u}, \tilde{l} \in \mathbb{Z}^{\mathcal{E}|_V}$ and the minimization problem

$$\min\{w^T x : \tilde{u} \geq x \geq \tilde{l} \text{ and } x \text{ satisfies (24)}\} \tag{25}$$

can be formulated as a submodular flow problem, solvable in polynomial time.

Frank et al. [53] showed that the so-called *orientation constraints* $u_i \geq x(\mathcal{E}_i) \geq l_i$, $\forall i = 1, 2, \dots, k$ can be incorporated into a construction of Schrijver [100] that transforms the minimization problem without orientation constraints

into a submodular flow problem which has efficient combinatorial algorithms (see, e.g., [55]).

Theorems 33 and 1 imply that the polyhedron $\{x : \tilde{u} \geq x \geq \tilde{l}$ and x satisfies (24)} is integral, and for every integral cost function w there exists an integral optimal solution to the dual of (25) where the family of the sets with positive dual variable is laminar (see [53] and page 313 of [101] for more details). This helps to formulate min–max relations for some graph problems. For example, what is the maximum number of undirected edges or the maximum number of arcs, that can be removed from a mixed graph such that the obtained subgraph has an orientation satisfying the requirements set by the requirement function r ? The following min–max formula due to Frank et al. [53] involves both of these problems. Let $G = (V, E)$ be an undirected graph, and let \mathcal{V} be a set of some subsets of V . Associate \mathcal{V} with the number

$$e_E(\mathcal{V}) = \max \left\{ \sum_{S \in \mathcal{V}} (\text{the number of arcs in orientation } \vec{G} \text{ that enter } S) : \right. \\ \left. \vec{G} \text{ is an orientation of } G \right\}.$$

Corollary 12 *Let $G = (V, F \cup E)$ be a mixed graph, where F is the set of undirected edges and E is the set of arcs. Let $w \in \{0, 1\}^{F \cup E}$ be a 0–1 cost function, and let $r : 2^V \rightarrow \mathbb{Z}_+$ be a positively intersecting supermodular set function. Then the minimum cost (with respect to w) of a subgraph that has an orientation satisfying requirement set by r equals*

$$\max \left\{ \sum_{S \in \mathcal{V}} r(S) - e_F(\mathcal{V}) - \sum_{S \in \mathcal{V}} |\delta_E^{\text{in}}(S)| + q(\mathcal{V}) : \mathcal{V} \subseteq 2^V \text{ is laminar} \right\}$$

where $\delta_E^{\text{in}}(S)$ is the set of arcs in E entering S , and $q(\mathcal{V})$ is the sum of the costs of the edges and arcs that enter at least one member of \mathcal{V} .

5.7 Rooted k -Connections in Directed Graphs

Let $G = (V, E)$ be a directed graph and $r_0 \in V$ be a specified vertex of G called the *root*. Graph G is called *rooted k -edge-connected* (resp. *rooted k -vertex-connected*) from r_0 if for every vertex $v \in V - \{r_0\}$, there exists at least k edge-disjoint (resp. openly disjoint) r_0v -paths in G . Suppose that G is endowed with a nonnegative arc cost function $w \in \mathbb{Z}_+^E$. Using polyhedral approaches, Frank [51] studied the *minimum cost rooted k -edge-connection* (resp. *minimum cost rooted k -vertex-connection*) problem which consists of finding a cheapest (with respect to w) subgraph $G' \subseteq G$ so that D' is rooted k -edge-connected (resp. rooted k -vertex-connected) from r_0 . Both problems have been solved to optimality in

polynomial time by Edmonds [38] and Frank and Tardos [54]. The contribution of [51] was highlighted by TDI descriptions of the rooted k -vertex-connected subgraphs and its generalizations, which were derived by using supermodular functions on bi-sets.

Definition 37 Let V be a ground set. A *bi-set* is a pair (S_O, S_I) with $S_I \subseteq S_O \subseteq V$. The set S_O is called the *outer member* of the bi-set, while S_I is called the *inner member*. A bi-set is *trivial* if $S_O = V$ or $S_I = \emptyset$, and is *void* if $S_I = \emptyset$.

Given a directed graph $G = (V, E)$, let \mathcal{B}_V denote the set of all bi-sets of V . An arc in E *enters* a bi-set (S_O, S_I) if its head belongs to S_O and its tail belongs to $V - S_O$. For any bi-set S , let $\delta^{\text{in}}(S)$ denote the set of arcs in E that enter S . Given two bi-sets $X = (X_O, X_I)$ and $Y = (Y_O, Y_I)$, their *intersection* and *union* are defined by $X \cap Y = (X_O \cap Y_O, X_I \cap Y_I)$ and $X \cup Y = (X_O \cup Y_O, X_I \cup Y_I)$, respectively.

Functions defined on \mathcal{B}_T are called *bi-set functions*. The bi-sets functions discussed in this subsection are assumed to be integer-valued and have zero value on void bi-sets. Analogous notions to those in [Definition 35](#) can be introduced for bi-set function as follows.

Definition 38 A bi-set function $r \in \mathbb{Z}^{\mathcal{B}_V}$ is *intersecting supermodular* if $r(X) + r(Y) \leq r(X \cap Y) + r(X \cup Y)$ holds for any two bi-sets X and Y with $X \cap Y$ nontrivial; it is *positively intersecting supermodular* if $r(X) + r(Y) \leq r(X \cap Y) + r(X \cup Y)$ holds whenever $r(X) > 0, r(Y) > 0$, and $X \cap Y$ nontrivial.

The combination of Frank’s “uncrossing” argument [51] and [Theorem 2](#) provides the following box-TDI system (26) under positively intersecting supermodularity. When further restricted to intersecting supermodular bi-set functions, the system defines a submodular flow polyhedron. A polyhedron P is said to be a *submodular flow polyhedron* if there exist directed graph $G = (V, E)$, an intersecting family \mathcal{V} of subsets of V (cf. [Definition 35](#)), an intersecting submodular function $s \in \mathbb{Z}^{\mathcal{V}}$, and vectors \mathbf{u} and \mathbf{l} in \mathbb{Z}^E (possibly having $-\infty$ coordinates and $+\infty$ coordinates, respectively) such that $P = \{\mathbf{x} : x(\delta^{\text{in}}(S)) - x(\delta^{\text{out}}(S)) \leq s(S) \text{ for all } S \in \mathcal{V}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}$.

Theorem 34 Let $G = (V, E)$ be a directed graph. If $r \in \mathbb{Z}^{\mathcal{B}_V}$ is a positively intersecting supermodular function, then the linear system

$$x(\delta^{\text{in}}(S)) \geq r(S) \text{ for every bi-set } S \in \mathcal{B}_V \quad (26)$$

is box-TDI. If r is intersecting supermodular, then $\{\mathbf{x} : \mathbf{u} \geq \mathbf{x} \geq \mathbf{l} \text{ and } \mathbf{x} \text{ satisfies (26)}\}$ is a submodular flow polyhedron for any $\mathbf{u}, \mathbf{l} \in \mathbb{Z}^E$, where coordinates of \mathbf{u} and \mathbf{l} are allowed to be $+\infty$ or $-\infty$.

Let $G = (V, E)$ be a directed graph with root $r_0 \in V$. Considering the intersecting supermodular bi-set function $s \in \{-\infty, k\}^{\mathcal{B}_V}$ defined as

$$s(S) = \begin{cases} k & \text{if } \emptyset \neq S_O = S_I \subseteq V - \{r_0\} \\ -\infty & \text{otherwise} \end{cases}$$

from the integrality implied by the box-TDI system in [Theorem 34](#), it is easy to see that the minimum cost rooted k -edge-connected problem amounts to solving the following LP:

$$\begin{aligned} & \min\{w^T x : \mathbf{1} \geq x \geq \mathbf{0} \text{ and } x(\delta^{\text{in}}(S)) \geq s(S) \text{ for every bi-set } S \in \mathcal{B}_V\} \\ &= \min\{w^T x : \mathbf{1} \geq x \geq \mathbf{0} \text{ and } x(\delta^{\text{in}}(S)) \geq k \text{ for every nonempty } S \subseteq V - \{r_0\}\} \end{aligned}$$

This fact has been extended by Frank [\[51\]](#) to the following corollary of [Theorem 34](#):

Corollary 13 *Let $G = (V, F \cup E)$ be a directed graph with root $r_0 \in V$ and terminal set $T \subseteq V - \{r_0\}$ so that the head of each edge in E is in T . Then the convex hull of incidence vectors of the arc set $E' \subseteq E$, such that the directed graph $(V, F \cup E')$ is rooted k -edge-connected from r_0 to T , is equal to the polyhedron defined by the following TDI system:*

$$\mathbf{1} \geq x \geq \mathbf{0} \text{ and } x(\delta^{\text{in}}(S)) \geq k - (\delta^{\text{in}}(S) \cap F) \text{ for every } S \subseteq V - \{r_0\} \text{ with } S \cap T \neq \emptyset.$$

Furthermore, the polyhedron is a submodular flow polyhedron.

For the minimum cost rooted k -vertex-connected problem, Frank [\[51\]](#) gave a common generalization on rooted k -connection for both vertex and edge, where the notion rooted (k, g) -connectivity was introduced. Let $G = (V, E)$ be a directed graph with root r_0 . For $\mathbf{g} \in \mathbb{Z}_+^V$, G is said to be rooted (k, \mathbf{g}) -connected from r_0 if for every $v \in V - \{r_0\}$ there are at least k openly disjoint r_0v -paths in G where each vertex $u \in V - \{r_0, v\}$ is used by at most $g(u)$ of these paths.

Corollary 14 *Let $G = (V, E)$ be a directed graph with root $r_0 \in V$. Then the convex hull of incidence vectors of the arc set $E' \subseteq E$, such that the directed graph (V, E') is rooted (k, \mathbf{g}) -connected from r_0 , is equal to the polyhedron defined by the following TDI system:*

$$\begin{aligned} & \mathbf{1} \geq x \geq \mathbf{0} \text{ and } x(\delta^{\text{in}}(S)) \geq k - g(S_O - S_I) \text{ for every bi-set } S \in \mathcal{B}_V \text{ with} \\ & \quad \emptyset \neq S_I \subseteq S_O \subseteq V - \{r_0\}. \end{aligned}$$

Furthermore, the polyhedron is a submodular flow polyhedron.

5.8 Cyclic Orders in Directed Graphs

Seb   [105] proved a range of min–max theorems about cycle packing and covering in directed graphs whose vertices are cyclically ordered.

Let $G = (V, E)$ be a directed graph with its n vertices ordered by a *cyclic order* on V which arises from a linear order v_1, \dots, v_n of V with the additional relation that v_n is followed by v_1 . Every arc $a = uv \in E$ can be considered going forward (or clockwise) in this order and can be associated with the corresponding interval $\langle u, v \rangle$ of the circle between its tail u and its head v in clockwise direction, which consists of vertices that follow u and precede v in the clockwise order, including u and v . The number of vertices in the interval $[u, v]$ minus one is defined as the *length* of a , written as $\text{length}(a) = |\langle u, v \rangle| - 1$. Let C be a cycle of G and Λ be a collection of cycles in G ; then the *winding numbers* $\text{ind}(C)$ of C and $\text{ind}(\Lambda)$ of Λ are, respectively, defined as

$$\text{ind}(C) = \frac{1}{n} \sum_{a \in E(C)} \text{length}(a), \text{ and } \text{ind}(\Lambda) = \sum_{C' \in \Lambda} \text{ind}(C').$$

A cyclic order of V has n *openings*, described by two consecutive vertices u, v in the cyclic order: It is the linear order whose first element is v , and it is followed by the elements of V in clockwise order with u as the last element. Given directed graph $G = (V, E)$, a linear order of V is said to be *compatible* with G if every arc in E that is contained in a cycle of G is also contained in a cycle with exactly one backward arc, and arcs not contained in any cycle of G are forward arcs. A cyclic order is said to be *compatible with* G , if it has an opening which is compatible with G . Two cyclic orders are *equivalent* if one arises from the other by a sequence of elementary changes, that is, interchanges between nonadjacent vertices that are consecutive in the cyclic order. Given a directed graph $G = (V, E)$ with a cyclic order, $S \subseteq V$ is called a *cyclic stable set* if S is a stable set of G , and it forms an interval in some equivalent cyclic order. Directed graphs with every vertex contained in some cycle are called *simplified*.

The results by Seb   [105] provide a bunch of min–max relations between cycle packing and covering, concerning with various objects in cyclic directed graphs such as stable sets, their unions, feedback vertex sets and feedback arc sets. Of particular interest was the establishment of two classes of box-TDI systems.

Theorem 35 *Let $G = (V, E)$ be a simplified strongly connected directed graph given with a compatible cyclic order. Then, for every positive integer d , the following linear system (27) and system (28) are both box-TDI:*

$$\text{System: } x(V(C)) \leq d \cdot \text{ind}(C) \text{ for every cycle } C \text{ of } G \quad (27)$$

$$\text{System: } x(V(C)) \geq d \cdot \text{ind}(C) \text{ for every cycle } C \text{ of } G \quad (28)$$

Sebö [105] showed that integral solutions to (27) with $d = 1$ are precisely the characteristic (incidence) vectors of acyclic stable sets. Thus, the box-TDI-ness of (27) implies the following min–max theorem (cf. [Theorem 1](#)).

Theorem 36 *Let $G = (V, E)$ be a strongly connected directed graph given with a compatible cyclic order, and let $w \in \mathbb{Z}_+^V$. A w -cover of G is a collection Λ of cycles in G such that for every $v \in V$ at least $w(v)$ cycles in Λ each contain v . Then $\max\{w(S) : S \text{ is a cyclic stable set of } G\} = \min\{\text{ind}(\Lambda) : \Lambda \text{ is a } w\text{-cover of } G\}$.*

An interesting application of this min–max theorem gives a simple proof [105] for Bessy and Thomassé’s result [5] which confirmed Gallai’s conjecture: The vertices of a strongly connected directed graph can be covered by at most as many cycles as the stability number of the graph.

All min–max relations presented in [105] were proved using cyclic orders and a unique elementary argument based on network flows. In this way, antiblocking and blocking relations were established, leading to a combination of “integer decomposition, integer rounding” and “dual integrality” that particularly contains the matroid partition theorem and Dilworth’s theorem (see, e.g., [103]). Sebö [105] provided a common reason for all these min–max equalities to hold and some possible other ones which satisfy the same abstract properties.

Using the duality theorem of linear programming, total unimodularity, Charbit and Sebö [15] obtained the following theorem “polar to” [Theorem 36](#).

Theorem 37 *Let $G = (V, E)$ be a simplified directed graph with a cyclic order. Then*

$$\begin{aligned} \min\{\text{ind}(\Lambda) : \Lambda \text{ is a 1-cover of } G\} &= \max\{w(V) : w \in \mathbb{Z}^V \text{ and } w(C) \\ &\leq \text{ind}(C) \text{ for every cycle } C \text{ of } G\}. \end{aligned}$$

6 The Dual Half-Integrality

Recall from [Definition 5](#) that a linear system $Ax \geq b, x \geq \mathbf{0}$ is $\frac{1}{2}$ -TDI if the maximum in the LP-duality equation (1) has a half-integral optimal solution, for every integral vector w for which the optimum is finite. The following statement is an instant corollary of [Lemma 7](#).

Lemma 10 *The system $Ax \geq \mathbf{1}, x \geq \mathbf{0}, u \geq x \geq I$ is $\frac{1}{2}$ -TDI if and only if $\text{Max}(A, I, u, 4w; \mathbb{Z}) \leq 2 \text{Max}(A, I, u, 2w; \mathbb{Z})$ for any integer vector w for which $\text{Max}(A, I, u, w)$ is finite.*

This section focuses the dual half-integrality in the context of hypergraphs, namely, (box-)half-Mengerian property (see [Definition 11](#)). Well-known half-Mengerian (but not necessarily Mengerian) hypergraphs include the hypergraph of

T -cuts (see [Definition 23](#)) in undirected graphs (see Corollary 29.2d of [\[103\]](#)), the hypergraph of two-commodity paths (see [Definition 24](#)) in undirected graphs (see Hu's 2-commodity flow theorem [\[72\]](#)), and the hypergraph of odd cycles in signed graphs (see [Definition 22](#)) that do not have odd K_5 -minors (see the Geelen-Guenin theorem on evenly bipartite signed graphs [\[61\]](#)). [Section 6.1](#) develops an analogue to the ESP property, so-called quasi-ESP property, which is sufficient for box-half-Mengerian property, and helps to identify box-half-Mengerian hypergraphs of graphical edges and stars. [Section 6.2](#) deals with matroids, where matroids are considered as hypergraphs of their circuits, and (box-)Mengerian matroids are characterized in terms of excluded minors.

6.1 The Quasi-ESP Property

The ESP property is sufficient for box-Mengerian hypergraphs (see [Theorem 14](#)). Based on this property, several classes of box-Mengerian hypergraphs have been established (see [Sects. 4.2–4.5](#)). This approach can be naturally extended to hypergraphs with certain half-integrality properties.

Definition 39 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph and let Λ be a hyperedge collection of \mathcal{H} . A *quasi equitable subpartition* of Λ consists of two collections Λ_1 and Λ_2 of hyperedges in \mathcal{E} (which are not necessarily in Λ) such that:

- (i) $|\Lambda_1| + |\Lambda_2| \geq |\Lambda|$.
- (ii) $d_{\Lambda_1 \cup \Lambda_2}(v) \leq d_\Lambda(v)$ for all $v \in V$.
- (iii') $\max\{d_{\Lambda_1}(v), d_{\Lambda_2}(v)\} \leq 2\lceil d_\Lambda(v)/4 \rceil$ for all $v \in V$.
- (iv') $|d_{\Lambda_1}(v) - d_{\Lambda_2}(v)| \leq 2$ for all $v \in V$ with $d_{\Lambda_1 \cup \Lambda_2}(v) = d_\Lambda(v)$.

The hypergraph \mathcal{H} is called *quasi equitably subpartitionable*, abbreviated quasi-ESP, if every collection of its hyperedges admits a quasi equitable subpartition. We refer to the above (i), (ii), (iii'), and (iv') as *quasi-ESP property*.

Note that condition (iii) specified in ESP property (see [Definition 18](#)) implies conditions (iii') and (iv') of the quasi-ESP property. In general, conditions (i), (ii), and (iii') are not sufficient for box-half-Mengerian hypergraphs, though they do imply half-Mengerian property (see [\[37\]](#) for a proof).

Theorem 38 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph. If for every hyperedge collection Λ of \mathcal{H} , there exist two collections Λ_1 and Λ_2 of hyperedges in \mathcal{E} satisfying (i), (ii), and (iii'), then \mathcal{H} is half-Mengerian.

A counterpart of [Theorem 14](#) reads as follows:

Theorem 39 Every quasi-ESP hypergraph is box-half-Mengerian.

Obviously, the theorem is a corollary of the following stronger result.

Theorem 40 Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph and let A be the $\mathcal{E} - V$ incidence matrix. Suppose that for any $\mathbf{l}, \mathbf{u} \in \mathbb{Q}^V$ and $\mathbf{w} \in \mathbb{Z}^V$ with finite $\text{Max}(A, \mathbf{l}, \mathbf{u}, \mathbf{w})$, there exists an optimal solution $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ to $\text{Max}(A, \mathbf{l}, \mathbf{u}, 4\mathbf{w}; \mathbb{Z})$ such that the hyperedge collection corresponding to $\boldsymbol{\alpha}^*$ (cf. Definition 19) admits a quasi equitable subpartition. Then \mathcal{H} is box-half-Mengerian.

Proof Let $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \boldsymbol{\gamma}^*)$ be an optimal solution to $\text{Max}(A, \mathbf{l}, \mathbf{u}, 4\mathbf{w}; \mathbb{Z})$ such that the hyperedge collection Λ corresponding to $\boldsymbol{\alpha}^*$ admits a quasi equitable subpartition (Λ_1, Λ_2) . By Definition 11 and Lemma 10, it suffices to find a feasible solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$ to $\text{Max}(A, \mathbf{l}, \mathbf{u}, 2\mathbf{w}; \mathbb{Z})$ such that $\boldsymbol{\alpha}^T \mathbf{1} + \boldsymbol{\beta}^T \mathbf{l} - \boldsymbol{\gamma}^T \mathbf{u} \geq [(\boldsymbol{\alpha}^*)^T \mathbf{1} + (\boldsymbol{\beta}^*)^T \mathbf{l} - (\boldsymbol{\gamma}^*)^T \mathbf{u}] / 2$. Without loss of generality, it can be assumed that:

- (1) $\beta^*(v)\gamma^*(v) = 0$ for all $v \in V$.
- (2) $\beta^*(v) = 0$ for all $v \in V$ with $l(v) < 0$.
- (3) $d_\Lambda(v) + \beta^*(v) - \gamma^*(v) = 4w(v)$ for all $v \in V$ with $\beta^*(v) + \gamma^*(v) > 0$.

For $i = 1, 2$, let $\boldsymbol{\alpha}_i \in \mathbb{Z}_+^{\mathcal{E}}$ correspond to Λ_i . It follows from (i) of the quasi-ESP property that $\boldsymbol{\alpha}_1^T \mathbf{1} + \boldsymbol{\alpha}_2^T \mathbf{1} \geq (\boldsymbol{\alpha}^*)^T \mathbf{1}$. Considering Λ of $\sum_{v \in V} d_\Lambda(v)$ as small as possible, it can be deduced that

- (4) $|\Lambda_1| + |\Lambda_2| = |\Lambda|$ (i.e., $\boldsymbol{\alpha}_1^T \mathbf{1} + \boldsymbol{\alpha}_2^T \mathbf{1} = \boldsymbol{\alpha}^T \mathbf{1}$) and $d_{\Lambda_1}(v) + d_{\Lambda_2}(v) = d_\Lambda(v)$ for all $v \in V$.

It remains to show that for each $v \in V$, there exist nonnegative integers $\beta_1(v), \beta_2(v), \gamma_1(v), \gamma_2(v)$ such that

- (5) $\beta_1(v) + \beta_2(v) = \beta^*(v), \gamma_1(v) + \gamma_2(v) = \gamma^*(v)$ and $d_{\Lambda_i}(v) + \beta_i(v) - \gamma_i(v) \leq 2w(v)$ for $i = 1, 2$.

Upon obtaining (5), for $i = 1, 2$, set $\boldsymbol{\beta}_i = (\beta_i(v) : v \in V)$ and $\boldsymbol{\gamma}_i = (\gamma_i(v) : v \in V)$. By (4), it is easy to see that either $(\boldsymbol{\alpha}_1, \boldsymbol{\beta}_1, \boldsymbol{\gamma}_1)$ or $(\boldsymbol{\alpha}_2, \boldsymbol{\beta}_2, \boldsymbol{\gamma}_2)$ is a solution to $\text{Max}(A, \mathbf{l}, \mathbf{u}, 2\mathbf{w}; \mathbb{Z})$ as desired.

To verify (5), suppose $d_{\Lambda_s}(v) \leq d_{\Lambda_t}(v)$, where $\{s, t\} = \{1, 2\}$. If $d_{\Lambda_t}(v) \leq \lceil d_\Lambda(v)/2 \rceil$, then by $d_\Lambda(v) + \beta^*(v) - \gamma^*(v) \leq 4w(v)$, it can be seen from (1) to (4) that (5) is satisfied by setting $\beta_s(v) = \lceil \beta^*(v)/2 \rceil, \gamma_s(v) = \lfloor \gamma^*(v)/2 \rfloor, \beta_t(v) = \lfloor \beta^*(v)/2 \rfloor, \gamma_t(v) = \lceil \gamma^*(v)/2 \rceil$. Therefore, the proof reduces to the case where

- (6) $d_{\Lambda_t}(v) > \lceil d_\Lambda(v)/2 \rceil$.

If $\beta^*(v) = \gamma^*(v) = 0$, then (5) follows from setting $\beta_1(v), \beta_2(v), \gamma_1(v)$, and $\gamma_2(v)$ all equal to zero, as $\max\{d_{\Lambda_1}(v), d_{\Lambda_2}(v)\} \leq 2\lceil d_\Lambda(v)/4 \rceil \leq 2w(v)$ by (iii') of the quasi-ESP property. Hence, it remains to consider

- (7) $\beta^*(v) + \gamma^*(v) > 0$.

From (6) and (iii') of the quasi-ESP property, it follows that $\lceil d_\Lambda(v)/2 \rceil + 1 \leq d_{\Lambda_t}(v) \leq 2\lceil d_\Lambda(v)/4 \rceil$. By considering possible congruence of $d_\Lambda(v)$ modulo four and comparing the lower and upper bounds of the preceding inequality, it is easy to see that

- (8) $\lceil d_\Lambda(v)/2 \rceil + 1 = d_{\Lambda_t}(v) = 2\lceil d_\Lambda(v)/4 \rceil$ and $d_\Lambda(v) \equiv 1$ or $2 \pmod{4}$.

Observe further that

- (9) If $\gamma^*(v) > 0$, then $\gamma^*(v) \geq 2$.

Otherwise, $\gamma^*(v) = 1$. Statements (1) and (3) imply $d_\Lambda(v) \equiv 1 \pmod{4}$. It follows from (8) that $d_{\Lambda_t}(v) = [d_\Lambda(v) + 3]/2$, and it turns from (ii) of the quasi-ESP property that $d_{\Lambda_s}(v) \leq d_\Lambda(v) - [d_\Lambda(v) + 3]/2 = [d_\Lambda(v) - 3]/2$. So $d_{\Lambda_t}(v) - d_{\Lambda_s}(v) \geq 3$, which together with (4), gives a contradiction to (iv') of the quasi-ESP property. Thus, (9) is established.

By (8), $d_\Lambda(v) \equiv k \pmod{4}$, where $k = 1$ or 2 , and $d_{\Lambda_t}(v) = [d_\Lambda(v) + 4 - k]/2$. Set $\beta_t(v) = 0$ if $\beta^*(v) = 0$ and $[\beta^*(v) - 4 + k]/2$ otherwise, and set $\gamma_t(v) = 0$ if $\gamma^*(v) = 0$ and $[\gamma^*(v) + 4 - k]/2$ otherwise. From (1), (7), (3), and (9), it can be deduced that

$$(10) \quad 0 \leq \beta_t(v) \leq \beta^*(v), \quad 0 \leq \gamma_t(v) \leq \gamma^*(v), \text{ and } d_{\Lambda_t}(v) + \beta_t(v) - \gamma_t(v) = 2w(v).$$

Set $\beta_s(v) = \beta^*(v) - \beta_t(v)$ and $\gamma_s(v) = \gamma^*(v) - \gamma_t(v)$. Then, $0 \leq \beta_s(v) \leq \beta^*(v)$ and $0 \leq \gamma_s(v) \leq \gamma^*(v)$. If (5) were violated by $i = s$, then $d_\Lambda(v) + \beta^*(v) - \gamma^*(v) \geq \sum_{i=1}^2 [d_{\Lambda_i}(v) + \beta_i(v) - \gamma_i(v)] > 4w(v)$ by (10), contradicting (3). So (5) holds for $i = 1$ and 2 .

In characterizing (box-)Mengerian hypergraphs, the Q_6 -minor plays an important role (see, e.g., Corollaries 4 and 5). Although Q_6 is not ESP, an easy case analysis shows that it is quasi-ESP. Thus by Theorem 39, clutter Q_6 is box- $\frac{1}{2}$ -Mengerian. Sections 6.1.1 and 6.1.2 below show applications of quasi-ESP property to obtaining box-half-total dual integrality on packing edges and stars of undirected graphs.

6.1.1 Undirected Graphs

Let $G = (V, E)$ be an undirected graph with a nonnegative integral weight function w defined on V .

Definition 40 The *vertex cover problem* is to find a vertex cover in G (see Definition 13) with minimum total weight with respect to w .

As is well known, this NP-hard problem can be approximated within a factor of two. Despite tremendous research effort, no $(2 - \epsilon)$ -approximation algorithm has been found to date, no matter how small the positive constant ϵ is. Actually it is a widespread belief that 2 is the best approximation ratio we can achieve. One 2-approximation algorithm for the vertex cover problem is based on the following Balinski theorem [2] (see, e.g., Theorem 64.11 in [103]): Let A be the E - V incidence matrix. Then the polytope $P = \{x : Ax \geq \mathbf{1}, \mathbf{1} \geq x \geq \mathbf{0}\}$ is half-integral (cf. Definition 4). The algorithm proceeds by finding a half-integral optimal solution x^* to the LP problem $\min\{w^T x : Ax \geq \mathbf{1}, \mathbf{1} \geq x \geq \mathbf{0}\}$. Set $U = \{v : x^*(v) \geq 1/2\}$. Then U is a vertex cover as desired. In the literature P is called the *fractional vertex cover polytope*.

In view that box- $\frac{1}{2}$ -TDI-ness implies $\frac{1}{2}$ -TDI-ness, which in turn implies half-integrality, the following theorem presents a strengthening of Balinski theorem [2].

Theorem 41 Every graph is quasi-ESP, so it is box-half-Mengerian (by Theorem 39).

Proof Let $G = (V, E)$ be a graph and let Λ be an edge collection of G . To show that Λ admits a pseudo-equitable subpartition, let U be the set of all vertices of G that are incident with some edges in Λ and let $H = (U, \Lambda)$ (possibly H contains multiple edges). It can be assumed without loss of generality that H is connected (otherwise, turn to considering the components of H). Let $H' = H$ if H is Eulerian and let H' be obtained from H by adding a new vertex v^* and then making it adjacent to all vertices of odd degree in H otherwise. Then the degree of each vertex of H' is even, so H' admits an Eulerian tour T . Let a be the starting vertex of T . Clearly it can be guaranteed that:

- Vertex a is precisely v^* , if any, and $d_\Lambda(a) \equiv 2 \pmod{4}$ if H is Eulerian and has an odd number of edges.

Let E_1 consist of all odd-numbered edges in T and let E_2 consist of all even-numbered edges in T . It is easy to see that $d_{E_1}(v) = d_{E_2}(v)$ for all vertices v of H' , except possibly vertex a when H' has an odd number of edges; in this case $d_{E_1}(a) - d_{E_2}(a) = 2$. Set $\Lambda_i = \Lambda \cap E_i$ for $i = 1, 2$. Then:

- $|\Lambda_1| + |\Lambda_2| = |\Lambda|$.
- $d_{\Lambda_1 \cup \Lambda_2}(v) = d_\Lambda(v)$ for all vertices v of H .
- $\max\{d_{\Lambda_1}(v), d_{\Lambda_2}(v)\} \leq \lceil d_\Lambda(v)/2 \rceil$ for all vertices v in H , except possibly a when H is Eulerian and has an odd number of edges; in this case $\max\{d_{\Lambda_1}(a), d_{\Lambda_2}(a)\} = 1 + d_\Lambda(a)/2 = 2\lceil d_\Lambda(a)/4 \rceil$.
- $|d_{\Lambda_1}(v) - d_{\Lambda_2}(v)| \leq 2$ for all vertices v of H .

So (i), (ii), (iii'), and (iv') of the quasi-ESP property (see Definition 39) are all satisfied by Λ_1 and Λ_2 , and hence, (Λ_1, Λ_2) is a quasi equitable subpartition of Λ .

Corollary 15 Let A be a 0–1 matrix with precisely two 1s in each row. Then the linear system $Ax \geq \mathbf{1}$, $x \geq \mathbf{0}$ is box- $\frac{1}{2}$ -TDI.

6.1.2 Star Hypergraphs

Let $G = (V, E)$ be a simple undirected graph. Let \mathcal{S} be the set of all stars in G (see Definition 31). The *star hypergraph* $\mathcal{H} = (E, \mathcal{S})$ of G has its \mathcal{S} - E incidence matrix A identical with the V - E incidence matrix of G . The blocker of \mathcal{H} is the clutter of edge covers of G (see Definition 13). Clearly, G has an edge cover if and only if G contains no isolated vertex. In the literature $\{x : Ax \geq \mathbf{1}, x \geq \mathbf{0}\}$ is called the *fractional edge cover polyhedron* of G , which is half-integral as shown by Balinski [2] (see Theorem 30.10 in [103]). A closely related result of Schrijver asserts that $Ax \geq \mathbf{2}$, $x \geq \mathbf{0}$ is $\frac{1}{2}$ -TDI (see Corollary 30.11a in [103]), where $\mathbf{2}$ is the all-two vector. The next theorem strengthens these two results.

Theorem 42 Every star hypergraph is quasi-ESP, so it is box-half-Mengerian (by Theorem 39).

Proof Let $\mathcal{H} = (E, \mathcal{S})$ be the star hypergraph of a graph $G = (V, E)$ and let Λ be a hyperedge collection of \mathcal{H} . To show that Λ admits a quasi equitable subpartition, for convenience, Λ is viewed as a star collection of G . Let $S(v)$ denote the star of G centered at v , let $m_\Lambda(v)$ denote the multiplicity of $S(v)$ in Λ , let X be the set of all vertices v of G with $m_\Lambda(v) \equiv 3 \pmod{4}$, and let $Y = V - X$. Now let Λ_1 be the star collection such that $S(v)$ appears $\lceil m_\Lambda(v)/2 \rceil$ times for any $v \in X$ and $\lfloor m_\Lambda(v)/2 \rfloor$ times for any $v \in Y$, and let Λ_2 be the star collection such that $S(v)$ appears $\lfloor m_\Lambda(v)/2 \rfloor$ times for any $v \in X$ and $\lceil m_\Lambda(v)/2 \rceil$ times for any $v \in Y$. Clearly:

- (1) $|\Lambda_1| + |\Lambda_2| = |\Lambda|$.
- (2) $d_{\Lambda_1 \cup \Lambda_2}(e) = d_\Lambda(e)$ for all edges e of G .
- (3) $|d_{\Lambda_1}(e) - d_{\Lambda_2}(e)| \leq 2$ for all edges e of G .

Moreover, for any edge $e = uv$ of G and $i = 1, 2$, the equation $d_{\Lambda_i}(e) = m_{\Lambda_i}(u) + m_{\Lambda_i}(v)$ holds. Furthermore,

- (4) $d_{\Lambda_i}(e) \leq 2\lceil d_\Lambda(e)/4 \rceil$ for all edges e of G .

Assume the contrary: $d_{\Lambda_i}(e) \geq 2\lceil d_\Lambda(e)/4 \rceil + 1$ for some edge $e = uv$ of G and $i = 1$ or 2 . Set $j = 3 - i$. By (2), we have $d_{\Lambda_j}(e) \leq 2\lceil d_\Lambda(e)/4 \rceil - 1$. Since $d_{\Lambda_i}(e) - d_{\Lambda_j}(e) \geq 2\lceil d_\Lambda(e)/4 \rceil + 1 - (2\lceil d_\Lambda(e)/4 \rceil - 1) = 2$, from (3) it follows that $d_{\Lambda_i}(e) = 2\lceil d_\Lambda(e)/4 \rceil + 1$ and $d_{\Lambda_j}(e) = 2\lceil d_\Lambda(e)/4 \rceil - 1$. Hence, both $d_{\Lambda_i}(e)$ and $d_{\Lambda_j}(e)$ are odd numbers.

On the other hand, $d_{\Lambda_i}(e) - d_{\Lambda_j}(e) = 2$ if and only if $m_{\Lambda_i}(u) - m_{\Lambda_j}(u) = 1$ and $m_{\Lambda_i}(v) - m_{\Lambda_j}(v) = 1$ if and only if $m_{\Lambda_i}(u) = \lceil m_\Lambda(u)/2 \rceil = \lfloor m_\Lambda(u)/2 \rfloor + 1 = m_{\Lambda_j}(u) + 1$ and $m_{\Lambda_i}(v) = \lceil m_\Lambda(v)/2 \rceil = \lfloor m_\Lambda(v)/2 \rfloor + 1 = m_{\Lambda_j}(v) + 1$. So both $m_\Lambda(u)$ and $m_\Lambda(v)$ are odd. From the definition of Λ_1 and Λ_2 , it follows that either $\{u, v\} \subseteq X$ or $\{u, v\} \subseteq Y$. Thus, $m_\Lambda(u) \equiv m_\Lambda(v) \equiv 3$ or $1 \pmod{4}$ and therefore $d_{\Lambda_i}(e) = m_{\Lambda_i}(u) + m_{\Lambda_i}(v)$ is even. This contradiction justifies (4).

By (1)–(4), it is clear that (i), (ii), (iii'), and (iv') of the quasi-ESP property (see Definition 39) are all satisfied by Λ_1 and Λ_2 , and hence, (Λ_1, Λ_2) is a quasi equitable subpartition of Λ . \square

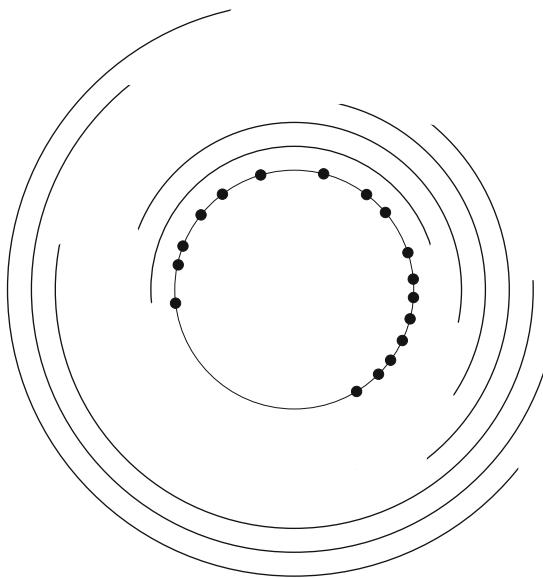
Corollary 16 *Let A be a 0–1 matrix with precisely two 1s in each column. Then the linear system $A\mathbf{x} \geq \mathbf{1}$, $\mathbf{x} \geq \mathbf{0}$ is box- $\frac{1}{2}$ -TDI.*

In the concluding part of Sect. 6.1, two examples show hypergraphs without quasi-ESP property.

Example 4 Circular arc clutters (clutters with hyperedges consecutive subsets of a set of nodes arranged in circle) are known to be diadic [34] (see Definition 30). Figure 8 presents such a clutter, where every arc represents an hyperedge in way of going through the nodes in the hyperedge. The example shows that diadic clutters are not necessarily quasi-ESP.

Example 5 A 0–1 matrix is a *hole matrix* if it contains exactly two 1s per row and per column and has no proper submatrix with this property. A *totally balanced matrix* is a 0–1 matrix in which every hole submatrix is the 2×2

Fig. 8 A circular arc clutter that is not quasi-ESP



submatrix of all 1s [27]. A hypergraph $\mathcal{H} = (V, \mathcal{E})$ is *totally balanced* if the \mathcal{E} - V incidence matrix is totally balanced [103]. For instance, a totally balanced hypergraph \mathcal{H} is given by setting $V = \{1, 2, \dots, 12\}$ and $\mathcal{E} = \{\{1, 5, 6\}, \{2, 5, 7\}, \{3, 5, 8\}, \{4, 5, 9\}, \{1, 2, 3, 4, 5, 10\}, \{1, 2, 3, 4, 5, 11\}, \{1, 2, 3, 4, 5, 12\}\}$. This hypergraph is not quasi-ESP since \mathcal{E} admits no quasi equitable subpartition.

6.2 Half-Integral Circuit Packing in Matroids

Let $M = (E, \mathcal{C})$ be a matroid (which is also viewed as a hypergraph with node set E and hyperedge set \mathcal{C}), and let $w \in \mathbb{Z}_+^E$ be a nonnegative integral weight function on E . It is not difficult to verify that hypergraph M of circuits is Mengerian (cf. Definition 9) if and only if M is the direct sum of circuits and coloops [37]. Let A be the $E - \mathcal{C}$ incidence matrix of M . It is nature to investigate the total dual half-integrality of the system $Ax \geq \mathbf{1}, x \geq \mathbf{0}$. Let

$$\begin{aligned} v_2(M, w) &= \max\{k : M \text{ has } k \text{ circuits (repetition allowed) such that each element } e \text{ of } M \text{ is used at most } 2w(e) \text{ times by these circuits}\} \\ &= \text{the maximum size of a } 2w\text{-circuit packing in } M; \\ \tau_2(M, w) &= \min\{w(X) : X \text{ is a collection of elements (repetition allowed) of } M \text{ such that every circuit in } M \text{ meets } X \text{ at least twice}\}. \end{aligned}$$

Then one may think of $v_2(M, \mathbf{w})$ as the maximum size of \mathbf{w} -“half-integral”-circuit packing in M .

6.2.1 Half-Mengerian Matroids

Recalling [Definition 11](#) and [Theorem 4](#), it is easy to obtain the following equivalence:

$$\begin{aligned} M \text{ is half-Mengerian} &\Leftrightarrow A\mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0} \text{ is } \frac{1}{2}\text{-TDI} \\ &\Leftrightarrow \frac{1}{2}A\mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0} \text{ is } \frac{1}{2}\text{-TDI} \\ &\Leftrightarrow v_2(M, \mathbf{w}) = \tau_2(M, \mathbf{w}) \text{ for all } \mathbf{w} \in \mathbb{Z}_+^E. \end{aligned} \quad (29)$$

Let $U_{2,4}$ be the uniform matroid of rank two on four elements. Let F_7 and F_7^* be the Fano matroid and its dual, respectively. Let K_n^- denote the graph obtained from K_n by deleting an edge, and let K be the first graph depicted in [Fig. 9](#). Ding and Zang [37] characterized all matroids that are half-Mengerian by proving the following theorem:

Theorem 43 *Let M be a matroid and let A be the circuit-element incidence matrix of M . Then the following statements are equivalent:*

- (i) *None of $U_{2,4}$, F_7 , F_7^* , $M(K_{3,3})$, $M(K_n^-)$ and $M(K)$ as a minor of M .*
- (ii) *The linear system $\frac{1}{2}A\mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0}$ is TDI.*
- (iii) *The polyhedron $\{\mathbf{x} : \frac{1}{2}A\mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0}\}$ is integral.*

Observe that if M is the cographic matroid of an undirected graph G , then circuits of M are precisely cuts of G and therefore A is the cut-edge incidence matrix of G . In this case, Ding and Zang’s [37] work is closely related to the *graphical traveling salesman problem* (GTSP), which, given a graph $G = (V, E)$ and a length function on E , is to find a tour of G with minimum total length, where a tour of G is an alternating sequence $T = v_0e_0v_1e_1\dots v_{k-1}e_{k-1}v_k$ of its vertices and edges such that $v_0 = v_k$, $e_i = v_iv_{i+1}$ for all $i < k$ and that each vertex of G appears in T at least once. GTSP is a relaxation of the classical TSP where a minimum Hamiltonian cycle is sought. The reader is referred to [29, 49] for a variety of motivations for investigating this graphical version. Due to its theoretical interest and practical value, the GTSP has been a subject of extensive research over the past two decades.

Let $G = (V, E)$ be an undirected graph. Each tour T of G is associate to its *incidence vector* $\mathbf{x} = (x(e) | e \in E)$, where $x(e)$ is the number of times that edge e occurs in the tour. Recalling the definition of cuts in G ([Definition 14](#)), it is easy to see that \mathbf{x} satisfies the following:

$$\begin{aligned} x(C) &\geq 2 \text{ for every cut } C \text{ of } G \\ x(e) &\in \mathbb{Z}_+ \text{ for all } e \in E. \end{aligned} \quad (30)$$

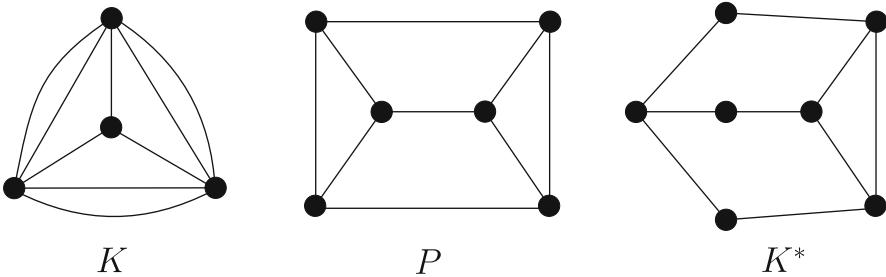


Fig. 9 Graphs K , P , and K^*

However, not every vector satisfying (30) is an incidence vector of a tour of G . In fact, it is easy to see that a vector \mathbf{x} satisfies (30) if and only if replacing each $e \in E$ with $x(e)$ parallel edges (e is actually deleted if $x(e) = 0$) results in a 2-edge-connected graph with vertex set V . (Such a graph $G_{\mathbf{x}}$ corresponds a tour of G only when $G_{\mathbf{x}}$ is Eulerian.)

Given a graph G with cut-edge incidence matrix A , let $P_0(G)$ be the convex hull of the incidence tours of G , let $P_1(G)$ be the convex hull of the incidence vectors of 2-edge-connected subgraphs of G where edges can be used several times (i.e., the convex hull of all vectors satisfying (30)), and let

$$P_2(G) = \{\mathbf{x} : \frac{1}{2}A, \mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0}\}. \quad (31)$$

The polyhedron $P_2(G)$ plays an important role in various polyhedral approaches to the GTSP; it is also closely related to several network design problems (see, for instance, [66, 67, 83–85]). Clearly, $P_0 \subseteq P_1(G) \subseteq P_2(G)$ and equalities need not hold in general. Observe that the classical traveling salesman polytope (the convex hull of the incidence vectors of all Hamiltonian cycles) is a subset of $P_0(G)$, and the integral vectors in $P_2(G)$ are precisely those satisfying (30).

Cornuéjols et al. [29] proposed the problem of characterizing all graphs G for which $P_1(G) = P_2(G)$ (equivalently $P_2(G)$ is integral). They also showed that all series-parallel graphs G enjoy this property, where a graph is called *series-parallel* if it does not contain K_4 as a minor (cf. Sect. 2.5(ii)). It is worthwhile pointing that, when restricted to cographic matroids, the equivalence of (i) and (iii) in Theorem 43 gives a complete solution to this research problem, which was independently solved by Vandenbussche and Nemhauser in [113]. Ding and Zang's [37] approach is different from that in [113]. They determined the complete structure of matroids that satisfy (i), and proved (ii) using this structure. Then they derived the equivalence of (i) and (iii) as a corollary. As is well known, (ii) is much stronger than (iii).

Let P and K^* , shown in Fig. 9, be the planar duals of K_5 and K , respectively. From Theorem 43 it can be seen that a graph G has an integral $P_2(G)$ if and only if it contains neither P nor K^* as a minor. With the same excluded minors, actually Ding and Zang [37] have drawn a much stronger statement as elaborated in the following

theorem, which establishes the implication (i) \Rightarrow (ii) of [Theorem 43](#) when M is cographic:

Theorem 44 *Let G be an undirected graph. Then the cographic matroid of G is half-Mengerian (cf. (29)) if and only if G contains neither P nor K^* as a minor.*

A one-page proof [37] showed that [Theorem 43](#) follows from [Theorem 44](#). To establish the latter theorem, Ding and Zang [37] proved that graphs without P and K^* minors can be expressed as sums of some prime graphs, which provides a structural characterization of half-Mengerian matroids. Then the authors proved that possessing the half-Mengerian property is preserved under summing operations on matroids. In turn, they introduced a packing property associated with cuts in graphs which is sufficient for the corresponding cographic matroids to be half-Mengerian. Finally they proved that each prime graph enjoys the packing property, from which [Theorem 44](#) follows. In order to verify the packing property on a few small graphs, the authors used computer to exhaust all the (about 2,700) possibilities.

6.2.2 Box-Half-Mengerian Matroids

Given undirected graph G with cut-edge incidence matrix A , integral vectors in $P_2(G)$ (see (31)) or equivalently vectors that satisfy (30) do not have to be 0–1 vectors, since they may have coordinates exceeding one. This could cause problems in certain applications. Clearly, a natural fix for this problem is to impose the box condition $\mathbf{1} \geq \mathbf{x} \geq \mathbf{0}$. This is what Mahjoub did in [83, 84], where he investigated the convex hull of incidence vectors of 2-edge-connected spanning subgraphs of G (with no edge repetition), and proved that if G is a series-parallel graph, then $P_3(G) = \{\mathbf{x} : \frac{1}{2}A\mathbf{x} \geq \mathbf{1}, \mathbf{1} \geq \mathbf{x} \geq \mathbf{0}\}$ is an integral polytope. Chen et al. [19] recently found the following common strengthening of the integrality of $P_2(G)$ [29] and the integrality of $P_3(G)$ [83] on series-parallel graphs G .

Theorem 45 *Let A be the cut-edge incidence matrix of a connected undirected graph G . Then the linear system $\frac{1}{2}A\mathbf{x} \geq \mathbf{1}, \mathbf{x} \geq \mathbf{0}$ is box-TDI if and only if G is a series-parallel graph.*

Since there are big differences between systems that are integral, that are TDI, and that are box-TDI (see, e.g., [103] and [Sects. 4.4–4.6](#) of this chapter), [Theorem 45](#) does discover much nicer feature of the series-parallel structure. One key to the proof [19] is the fact that every series-parallel graph is constructed from a forest by repeatedly adding loops and making series extension and/or parallel extensions (meaning that replacing an edge by two edges in series or in parallel). The other key consists of the operations under which box-total dual integrality is maintained, namely, column deletion, column addition, column series extension, and column parallel extension (see [Sect. 2.4](#)). The first key enables one to start with a forest and build up the whole picture step by step via series and parallel extensions, while the second key inductively guarantees the system at every step is box-TDI. A generalization of [Theorem 45](#) thinks of matrix A as circuit-element matrix of a

matroid. Chen et al. [19] characterized in terms of “excluded minor” the matroid M for which $\frac{1}{2}Ax \geq \mathbf{1}, x \geq \mathbf{0}$ is a box-TDI system.

Theorem 46 *Let M be a matroid and let A be the circuit-element incidence matrix of M . Then the following statements are equivalent:*

- (i) M is box-half-Mengerian.
- (ii) $\frac{1}{2}Ax \geq \mathbf{1}, x \geq \mathbf{0}$ is a box-TDI system.
- (iii) M has neither $U_{2,4}$ -minor nor $M(K_4)$ -minor.

Proof The equivalence (ii) \Leftrightarrow (iii) has been established in [19]. Definition 11 and Lemma 1 give (ii) \Rightarrow (i). If M has $U_{2,4}$ or $M(K_4)$ as a minor, then as argued in [19], the polytope $\{x : \frac{1}{2}Ax \geq \mathbf{1}, \mathbf{1} \geq x \geq \mathbf{0}\}$ is not integral, and (i) does not hold by Lemma 2. Hence, (i) \Rightarrow (iii). \square

7 Conclusion

In addition to the three-volume monograph by Schrijver [103], excellent treatments of the dual integrality with respect to combinatorial optimization problems include survey papers by Pulleyblank [92], Schrijver [99, 101, 102], and Frank and Király [52]. Aside from what has been reviewed in this chapter, many other recent works on dual integrality and its related topics have attracted considerable research attentions.

TDI Description. Pia and Zambelli [91] studied systems of the form $\mathbf{b} \leq Ax \leq \mathbf{d}$, $\mathbf{l} \leq x \leq \mathbf{u}$, where A is obtained from a totally unimodular matrix with two nonzero elements per row by multiplying by 2 some of its columns and where $\mathbf{b}, \mathbf{d}, \mathbf{l}, \mathbf{u}$ are integral vectors. They gave an explicit TDI description for the integer hull of the polyhedron P defined by the above inequalities. Since the inequalities of such a TDI system are Chvátal inequalities for P , their result implies that the matrix A has cut-rank 1. The authors also derived a strongly polynomial time algorithm to find an integral optimal solution for the dual of the problem of minimizing a linear function with integer coefficients over the TDI system.

Approximate min–max Relations. Samuel et al. [48] studied the ratio between the minimum size of an odd cycle vertex transversal and the maximum size of a collection of vertex-disjoint odd cycles in a planar graph. They showed that this ratio is at most 10. They also gave a shorter proof of the ratio 2 for the corresponding edge version of the problem, which was originally proved by Král and Voss [77].

Excluded Minor Characterizations. A 0–1 matrix is *balanced* if it contains no square submatrix of odd order with exactly two 1s per row and per column. Balanced matrices lead to ideal formulations for both set packing and set covering problems. Balanced graphs are those graphs whose clique-vertex incidence matrix is balanced. While forbidden induced subgraph characterization of balanced graphs was obtained by [7], there is no such characterization by minimal forbidden induced

subgraphs. Bonomo et al. [10] provided minimal forbidden induced subgraph characterizations of balanced graphs restricted to some graph classes which also lead to polynomial time or even linear time recognition algorithms within the corresponding subclasses.

A graph G is *clique-perfect* if the cardinality of a maximum clique-independent set of H equals the cardinality of a minimum clique-transversal of H , for every induced subgraph H of G . A graph G is *coordinated* if the minimum number of colors that can be assigned to the cliques of H in such a way that no two cliques with nonempty intersection receive the same color equals the maximum number of cliques of H with a common vertex, for every induced subgraph H of G . Coordinated graphs are a subclass of perfect graphs. Finding the complete lists of minimal forbidden induced subgraphs for the classes of clique-perfect and coordinated graphs turns out to be a difficult task. However, some (partial) characterizations have been obtained for several special classes of graphs [8, 9, 11].

Fractional Packing. Given an undirected graphs G , a k half-integral packing of an H -minor, where H is a fixed graph, is a collection of k H -minors of G such that each vertex of G is used at most twice. Ken-ichi [73] showed that the problem of deciding whether or not G has k half-integral packing of an H -minor for fixed k and graph H is polynomial time solvable when $H = K_6$ or $H = K_7$. The author also showed that Erdős–Pósa property holds for k half-integral packing of a K_6 -minor (resp. a K_7 -minor). Latter Ken-ichi and Bruce [74] studied the half integral k disjoint paths packing problem for which they provided an $O(n \log n)$ time algorithm for this problem for fixed k , improving upon Kleinberg’s $O(n^3)$ algorithm.

Consider an ideal clutter with a nonnegative capacity function on its vertices. It follows from idealness that for any nonnegative capacity the total multiplicity of an optimal fractional packing is equal to the minimum capacity of a node cover. Yuji [115] presented a polyhedral framework for fractional packing in ideal clutters. Given an n -node clutter, this framework finds an optimal hyperedge packing using at most n hyperedges with positive multiplicities, performing at most n times minimizations for the clutter and at most n^2 times minimizations for its blocker. Applied to the clutter of dijoints, the framework gave the first combinatorial polynomial time algorithm for fractional packing of dijoints.

Bounded Fractionality. Harai [71] gave a complete characterization of the class of weighted maximum multiflow problems whose dual polyhedra have bounded fractionality, which is a common generalization of two fundamental results of Karzanov for metric weight and 0–1 weight. Harai characterized not only the commodity graphs H for which the dual of maximum multiflow problem with respect to H has bounded fractionality but also the metrics on terminals for which the dual of metric-weighed maximum multiflow problem has bounded fractionality. Apart from the fractionality issues, the design of combinatorial or practical algorithms specialized to general multiflow problems is still a challenging

problem. The tight-span dual problem and the geometry of explored in [71] might give a basis against this challenge.

In concluding this chapter, it is worthwhile noting that there are many exciting and important problems on dual integrality yet to be solved. Among the other topics of interest are the following:

- (a) For the complexity of recognizing (box-)TDI systems, the question whether [Problems 2](#) and [3](#) are in NP or co-NP is still unanswered.

All tournaments G with Mengerian cycle hypergraphs \mathcal{H}_G and those with Mengerian FVS clutters $b(\mathcal{H}_G)$ have been characterized in terms of “forbidden subtournaments” (see [Theorem 25](#)). Coincidentally, $b(\mathcal{H}_G)$ is Mengerian if and only if \mathcal{H}_G is (see [Corollary 9](#)).

- (b) Major open problems in this research direction are to characterize all directed graphs G with Mengerian \mathcal{H}_G and those with Mengerian $b(\mathcal{H}_G)$. The arc versions of these problems are equally interesting.

While these problems are extremely hard in general, Guenin and Thomas [69] successfully characterized all directed graphs that pack, where a directed graph G *packs* if for any sub-directed graph H of G , the maximum number of vertex-disjoint cycles is equal to the minimum number of vertices in an FVS in H . Guenin strongly believes that the blocker version of their theorem holds on exactly the same class of directed graphs.

- (c) A directed graph G packs if and only if for any subgraph H of G , the maximum number of disjoint feedback vertex sets is equal to the minimum number of vertices in a cycle in H .

The comparison of statements (i) and (ii) of [Theorems 23](#) and [25](#) shows a preservation of the min–max relation under the interchange of the roles of feedback vertex sets and cycles. Similar interchange arises in the celebrated Lucchesi–Younger theorem [82] and Woodall’s conjecture [114]. The planar version of Lucchesi–Younger theorem asserts that in any planar directed graph, the minimum size of a transversal is equal to the maximum number of arc-disjoint cycles, where a *transversal* is a set of arcs which intersects every cycle in the directed graph.

- (d) Correspondingly, Woodall conjectured in 1978 that in a planar directed graph, the length of a shortest cycle is equal to the maximum cardinality of a collection of disjoint transversals. The conjecture has been a long-standing open problem.

Based on Ding and Zang’s [36] “excluded minor” characterization for cycle Mengerian undirected graphs (see [Theorem 17](#)), Chen and Chen [16] developed an efficient algorithm for finding a maximum w -cycle packing in cycle Mengerian undirected graphs. The algorithm, which employs the ellipsoid algorithm of Grötschel et al. [64] as a subroutine, is not strongly polynomial time.

- (e) It is challenging to design a combinatorial algorithm for the cycle packing problem (see [Definition 20](#)) on cycle Mengerian undirected graphs.

Acknowledgements This work is supported in part by the China 973 Project under Grant No. 2011CB80800, the National Natural Science Foundation of China under Grant No. 11222109, 11021161 and 10928102, the Chinese Academy of Sciences under Grant No. kjcx-yw-s7, the Research Grants Council of Hong Kong, and Seed Funding for Basic Research of the University of Hong Kong.

Cross-References

- ▶ [Fractional Combinatorial Optimization](#)
 - ▶ [Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs](#)
 - ▶ [Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work](#)
-

Recommended Reading

1. N. Apollonio, Integrality properties of edge path tree families. *Discrete Math.* **309**, 4181–4184 (2009)
2. M.L. Balinski, Integer programming: methods, uses, computation. *Manag. Sci. Ser. A* **12**, 253–313 (1965)
3. R. Bar-Yehuda, K. Bendel, A. Freund, D. Rawitz, Local ratio: A unified framework for approximation algorithms. *ACM Comput. Surv.* **36**, 422–463 (2004)
4. C. Berge, *Graphs and Hypergraphs* (North-Holland, Amsterdam, 1976)
5. S. Bessy, S. Thomassé, Spanning a strong digraph by α circuits: A proof of Gallai's conjecture. *Combinatorica* **27**, 659–667 (2007)
6. J.A. Bondy, U.S.R. Murty, *Graph Theory with Applications* (Macmillan, London, 1976)
7. F. Bonomo, G. Durán, M.C. Lin, J.L. Szwarcfiter, On balanced graphs. *Math. Program.* **105**, 233–250 (2006)
8. F. Bonomo, M. Chudnovsky, G. Durán, Partial characterizations of clique-perfect graphs I: subclasses of claw-free graphs. *Discrete Appl. Math.* **156**, 1058–1082 (2008)
9. F. Bonomo, M. Chudnovsky, G. Durán, Partial characterizations of clique-perfect graphs II: diamond-free and Helly circular-arc graphs. *Discrete Math.* **309**, 3485–3499 (2009)
10. F. Bonomo, G. Durán, M.C. M.D. Safe, A.K. Wagler, On minimal forbidden subgraph characterizations of balanced graphs. *Electron. Note Discrete Math.* **35**, 41–46 (2009)
11. F. Bonomo, G. Durán, F. Soulignac, G. Sueiro, Partial characterizations of coordinated graphs: Line graphs and complements of forests. *Math. Method Oper. Res.* **69**, 251–270 (2009)
12. E. Boros, K. Elbassioni, V. Gurvich, H.R. Tiwary, The negative cycles polyhedron and hardness of checking some polyhedral properties. *Ann. Oper. Res.* **188**, 63–76 (2011)
13. M. Cai, X. Deng, W. Zang, An approximation algorithm for feedback vertex sets in tournaments. *SIAM J. Comput.* **30**, 1993–2007 (2001)
14. M. Cai, X. Deng, W. Zang, A min–max theorem on feedback vertex sets. *Math. Oper. Res.* **27**, 361–371 (2002)
15. P. Charbit, A. Sebö, Cyclic orders: Equivalence and duality. *Combinatorica* **28**, 131–143 (2008)
16. Q. Chen, X. Chen, Packing cycles exactly in polynomial time. *J. Combin. Optim.* **23**, 167–188 (2012)
17. X. Chen, Z. Chen, W. Zang, A unified approach to box-Mengerian hypergraphs. *Math. Oper. Res.* **35**, 655–668 (2010)
18. X. Chen, G. Ding, X. Hu, W. Zang, A min–max relation on packing feedback vertex sets. *Math. Oper. Res.* **31**, 777–788 (2006)

19. X. Chen, G. Ding, W. Zang, The box-TDI system associated with 2-edge connected spanning subgraphs. *Discrete Appl. Math.* **157**, 118–125 (2009)
20. X. Chen, G. Ding, W. Zang, A characterization of box-Mengerian matroid ports. *Math. Oper. Res.* **33**, 497–512 (2008)
21. X. Chen, X. Hu, W. Zang, A min–max theorem on tournaments. *SIAM J. Comput.* **37**, 923–937 (2007)
22. M. Chudnovsky, G. Cornuéjols, X. Liu, P.D. Seymour, K. Vušković, Recognizing Berge graphs. *Combinatorica* **25**, 143–186 (2005)
23. M. Chudnovsky, N. Robertson, P.D. Seymour, R. Thomas, The strong perfect graph theorem. *Ann. Math.* **164**, 51–229 (2006)
24. V. Chvátal, On certain polytopes associated with graphs. *J. Combin. Theor Ser. B* **18**, 138–154 (1975)
25. W. Cook, On box totally dual integral polyhedra. *Math. Program.* **34**, 48–61 (1986)
26. W. Cook, L. Lovász, A. Schrijver, A polynomial-time test for total dual integrality in fixed dimension. *Math. Program. Study* **22**, 64–69 (1984)
27. M. Conforti, G. Cornuéjols, K. Vušković, Balanced matrices. *Discrete Math.* **306**, 2411–2437 (2006)
28. G. Cornuéjols, *Combinatorial Optimization: Packing and Covering* (SIAM, Philadelphia, 2001)
29. G. Cornuéjols, J. Fonlupt, D. Naddef, The traveling salesman problem on a graph and some related integer polyhedra. *Math. Program.* **33**, 1–27 (1985)
30. G. Cornuéjols, B. Guenin, F. Margot, The packing property. *Math. Program.* **89**, 113–126 (2000)
31. G. Cornuéjols, D. Hartvigsen, An extension of matching theory. *J. Combin. Theor Ser. B* **40**, 285–296 (1986)
32. G. Cornuéjols, W. Pulleyblank, A matching problem with side conditions. *Discrete Math.* **29**, 135–159 (1980)
33. R. Diestel, *Graph Theory*, 3rd edn. (Springer, New York, 2005)
34. G. Ding, Clutters with $\tau_2 = 2\tau$. *Discrete Math.* **115**, 141–152 (1993)
35. G. Ding, L. Feng, W. Zang, The complexity of recognizing linear systems with certain integrality properties. *Math. Program. Ser. A* **114**, 321–334 (2008)
36. G. Ding, W. Zang, Packing cycles in graphs. *J. Combin. Theor Ser. B* **86**, 381–407 (2002)
37. G. Ding, W. Zang, Packing circuits in matroids. *Math. Program. Ser. A* **119**, 137–168 (2009)
38. J. Edmonds, Submodular functions, matroids, and certain polyhedra, in *Combinatorial Structures and Their Applications* (Proceedings Calgary International Conference on Combinatorial Structures and Their Applications, Calgary, Alberta, 1969) ed. by R. Guy, H. Hanani, N. Sauer, J. Schönheim (Gordon and Breach, New York, 1970), pp. 69–87
39. J. Edmonds, Edge-disjoint branchings, in *Combinatorial Algorithmis* (Algorithmic Press, New York, 1973), pp. 91–96
40. J. Edmonds, D.R. Fulkerson, Bottleneck extrema. *J. Combin. Theor* **8**, 299–306 (1970)
41. J. Edmonds, R. Giles, A min–max relation for submodular functions on graphs, in *Annals of Discrete Mathematics*, vol. 1 (North-Holland, Amsterdam, 1977), pp. 185–204
42. J. Edmonds, R. Giles, Total dual integrality of linear inequality systems, in *Progress in Combinatorial Optimization*, ed. by W.R. Pulleyblank (Academic, Toronto, 1984), pp. 117–129
43. P. Erdős, L. Pósa, On the independent circuits contained in a graph. *Can. J. Math.* **17**, 347–352 (1965)
44. U. Faigle, W. Kern, Submodular linear programs on forests. *Math. Program.* **72**, 195–206 (1996)
45. U. Faigle, W. Kern, On the core of ordered submodular cost games. *Math. Program. Ser. A* **87**, 483–499 (2000)
46. T. Feder, A new fixed point approach for stable networks and stable marriages. *J. Comput. Syst. Sci.* **45**, 233–284 (1992)

47. P. Feofilloff, D.H. Younger, Directed cut transversal packing for source-sink connected graphs. *Combinatorica* **7**, 255–263 (1987)
48. S. Fiorini, N. Hardy, Nadia, B. Reed, A. Vetta, Approximate min–max relations for odd cycles in planar graphs. *Math. Program.* **110**, 71–91 (2007)
49. J. Fonlupt, D. Naddef, The traveling salesman problem in graphs with some excluded minors. *Math. Program.* **53**, 147–172 (1992)
50. A. Frank, A coloring question on digraphs, in *DMANET*, 1998
51. A. Frank, Rooted k-connections in digraphs. *Discrete Appl. Math.* **157**, 1242–1254 (2009)
52. A. Frank, T. Király, A Survey on covering supermodular functions, in *Research Trends in Combinatorial Optimization* (Boon, 2008) ed. by W. Cook, L. Lovász, J. Vygen, (Springer, Berlin, 2009), pp. 87–126
53. A. Frank, T. Király, Z. Király, On the orientation of graphs and hypergraphs. *Discrete Appl. Math.* **131**, 385–400 (2003)
54. A. Frank, É. Tardos, An application of submodular flows. *Lin. Algebra Appl.* **114–115**, 329–348 (1989)
55. S. Fujishige, *Submodular Functions and Optimization*, 2nd edn. *Annals of Discrete Mathematics*, vol. 58 (Elsevier, London, 2005)
56. D.R. Fulkerson, Networks, frames, and blocking systems, in *Mathematics of the Decision Sciences, Part I*, ed. by G.B. Dantzig, A.F. Veinott (American Mathematical Society, Providence, Rhode Island, 1968), pp. 303–334
57. D.R. Fulkerson, Blocking and antiblocking pairs of polyhedra. *Math. Program.* **1**, 168–194 (1971)
58. D.R. Fulkerson, Packing rooted directed cuts in a weighted directed graph. *Math. Program.* **6**, 1–13 (1974)
59. D. Gale, L. Shapley, College admissions and the stability of marriage. *Am. Math. Mon.* **69**, 9–15 (1962)
60. M.R. Garey, D.S. Johnson, *Computers and Intractability* (W.H. Freeman and Company, New York, 1979)
61. J. Geelen, B. Guenin, B, Packing odd circuits in Eulerian graphs. *J. Combin. Theor Ser. B* **86**, 280–295 (2002)
62. A.M.H. Gerards, M. Laurent, A characterization of box $\frac{1}{d}$ -integral binary clutters. *J. Combin. Theor Ser. B* **65**, 186–207 (1995)
63. M.X. Goemans, D.P. Williamson, The primal-dual method for approximation algorithms and its application to network design problems, in *Approximation Algorithms for NP-Hard Problems*, ed. by D.S. Hochbaum (PWS Publishing Company, Boston, MA), pp. 144–191
64. M. Grötschel, L. Lovász, A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1**, 169–197 (1981)
65. M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization* (Springer, Berlin, 1988)
66. M. Grötschel, C. Monma, M. Stoer, Facets for polyhedra arising in the design of communication networks with low-connectivity constraints. *SIAM J. Optim.* **2**, 474–504 (1992)
67. M. Grötschel, C. Monma, M. Stoer, Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints. *Oper. Res.* **40**, 309–330 (1992)
68. B. Guenin, A short proof of Seymour’s characterization of the matroids with the max-flow min-cut property. *J. Combin. Theor Ser. B* **86**, 273–279 (2002)
69. B. Guenin, R. Thomas, Packing directed circuits exactly. *Combinatorica* **31**, 397–421 (2011)
70. R.P. Gupta, An edge-coloration theorem for bipartite graphs with applications. *Discrete Math.* **23**, 229–233 (1978)
71. H. Hirai, Tight spans of distances and the dual fractionality of undirected multiflow problems. *J. Combin. Theor Ser. B* **99**, 843–868 (2009)
72. T.C. Hu, Multi-commodity network flows. *Oper. Res.* **11**, 344–360 (1963)
73. K. Ken-ichi, Half integral packing, Erdős-Pósa-property and graph minors, in *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1187–1196

74. K. Ken-ichi, R. Bruce, A nearly linear time algorithm for the half integral disjoint paths packing, in *Proceedings of the 19th annual ACM-SIAM Symposium on Discrete Algorithms*, 2008, pp. 446–454
75. S. Khanna, J. Naor, F.B. Shepherd, Directed network design with orientation constraints. *SIAM J. Discrete Math.* **19**, 245–257 (2005)
76. T. Király, J. Pap, Total dual integrality of Rothblum’s description of the stable-marriage polyhedron. *Math. Oper. Res.* **33**, 283–290 (2008)
77. D. Král, H. Voss, Edge-disjoint odd cycles in planar graphs. *J. Combin. Theor Ser. B* **90**, 107–120 (2004)
78. M. Laurent, S. Poljak, One-third-integrality in the max-cut problem. *Math. Program.* **71**, 29–50 (1995)
79. O. Lee, Y. Wakabayashi, Note on a min–max conjecture of Woodall. *J. Graph Theorem* **38**, 36–41 (2001)
80. A. Lehman, On the length-width inequality. *Math. Program.* **17**, 403–417 (1979)
81. L. Lovász, Normal hypergraphs and the perfect graph conjecture. *Discrete Math.* **2**, 253–267 (1972)
82. C.L. Lucchesi, D.H. Younger, A minimax relation for directed graphs. *J. Lond. Math. Soc.* **17**, 369–374 (1978)
83. A.R. Mahjoub, Two-edge connected spanning subgraphs and polyhedra. *Math. Program.* **64**, 199–208 (1994)
84. A.R. Mahjoub, On perfectly two-edge connected graphs. *Discrete Math.* **170**, 153–172 (1997)
85. C.L. Monma, B.S. Munson, W.R. Pulleyblank, Minimum weight 2-connected spanning networks. *Math. Program.* **46**, 153–172 (1990)
86. J. von Neumann, O. Morgenstern, *Theory of Games and Economic Behavior* (Princeton University Press, Princeton, 1944)
87. E. O’Shea, A. Sebö, Alternatives for testing total dual integrality. *Math. Program.* **132**, 57–78 (2012)
88. J. Oxley, *Matroid Theory* (Oxford University Press, Oxford, 1992)
89. J. Pap, Recognizing conic TDI systems is hard. *Math. Program. Ser. A* **128**, 43–48 (2011)
90. G. Pap, Weighted restricted 2-matching. *Math. Program. Ser. A* **119**, 305–329 (2009)
91. A.D. Pia, G. Zambelli, Half-integral vertex covers on bipartite bidirected braphs: total dual integrality and cut-rank. *SIAM J. Discrete Math.* **23**, 1281–1296 (2009)
92. W.R. Pulleyblank, Polyhedral combinatorics, in *Mathematical Programming – The State of the Art (Bonn, 1982)*, ed. by A. Bachem, M. Grötschel, B. Korte (Springer, Berlin, 1983), pp. 312–345
93. W. Pulleyblank, J. Edmonds, Facets of 1-matching polyhedra, in *Hypergraph Seminar (Proceedings Working Seminar on Hypergraphs, Columbus, Ohio, 1972)*, ed. by C. Berge, D. Ray-Chaudhuri (Springer, Berlin, 1974), pp. 214–242
94. R. Rado, Bemerkungen zur Kombinatorik im AnschluSS an Untersuchungen von Herrn D. König. *Sitzungsberichte der Berliner Mathematischen Gesellschaft* **32**, 60–75 (1933)
95. U. Rothblum, Characterization of stable matchings as extreme points of a polytope. *Math. Program.* **54**, 57–67 (1992)
96. T.J. Schaefer, The complexity of satisfiability problems, in *Proceedings of 10th ACM Symposium on Theory of Computing*, New York, 1986, pp. 216–226
97. A. Schrijver, A counterexample to a conjecture of Edmonds and Giles. *Discrete Math.* **32**, 213–214 (1980)
98. A. Schrijver, Min-max relations for directed graphs. *Ann. Discrete Math.* **16**, 127–146 (1982)
99. A. Schrijver, Min-max results in combinatorial optimization, in *Mathematical Programming: The State of the Art Bonn 1982*, ed. by A. Bachem, M. Grötschel, B. Korte (Springer, New York, 1983), pp. 439–500
100. A. Schrijver, Total dual integrality from directed graphs, crossing families, and sub- and supermodular functions, in *Progress in Combinatorial Optimization* (Academic Press, Toronto, 1984), pp. 315–361
101. A. Schrijver, *Theory of Linear and Integer Programming* (Wiley, New York, 1986)

102. A. Schrijver, Polyhedral combinatorics, in *Handbook of Combinatorics*, vol. 2, ed. by R.L. Graham, M. Grötschel, L. Lovász (Elsevier, Amsterdam, 1995), pp. 1649–1704
103. A. Schrijver, *Combinatorial Optimization - Polyhedra and Efficiency* (Springer, Berlin, 2003)
104. A. Schrijver, P.D. Seymour, A proof of total dual integrality of matching polyhedra, in *Mathematical Centre Report ZN 79/77* (Mathematical Centre, Amsterdam, 1977)
105. A. Sebö, Minmax relations for cyclically ordered digraphs. *J. Combin. Theor Ser. B* **97**, 518–552 (2007)
106. P.D. Seymour, The forbidden minors of binary clutters. *J. Lond. Math. Soc.* **12**, 356–360 (1976)
107. P.D. Seymour, The matroids with the max-flow min-cut property. *J. Combin. Theor Ser. B* **23**, 189–222 (1977)
108. P.D. Seymour, Decomposition of regular matroids. *J. Combin. Theor Ser. B* **28**, 305–359 (1980)
109. P.D. Seymour, On odd cuts and plane multicommodity flows. *Proc. Lond. Math. Soc.* **42**, 178–192 (1981)
110. E. Speckenmeyer, On feedback problems in digraphs, in *Graph-Theoretic Concepts in Computer Science, Lecture Notes in Computer Science*, vol. 411 (Springer, Berlin, 1989), pp. 218–231
111. K. Truemper, *Matroid Decomposition* (Academic Press, Toronto, 1992)
112. F.T. Tseng, K. Truemper, A decomposition of the matroids with the max-flow min-cut property. *Discrete Appl. Math.* **15**, 329–364 (1986)
113. D. Vandenbussche, G. Nemhauser, The 2-edge-connected subgraph polyhedron. *J. Combin. Optim.* **9**, 357–379 (2005)
114. D.R. Woodall, Menger and König systems, in *Theory and Applications of Graphs, Lecture Notes in Mathematics*, vol. 642 (1978), pp. 620–635
115. M. Yuji, Fractional packing in ideal clutters, in *Proceedings of the 18th annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1181–1186

Dynamical System Approaches to Combinatorial Optimization

Jens Starke

Contents

1	Introduction.....	1066
2	Assignment Problems.....	1068
3	Dynamical Systems.....	1071
4	Dynamical Systems Based on Penalty Methods.....	1072
4.1	Classical Approach.....	1072
4.2	Variants of Penalty Terms.....	1075
5	Statistical Approaches.....	1076
5.1	Metropolis Algorithm.....	1076
5.2	Simulated Annealing.....	1077
5.3	Evolutionary Strategies and Genetic Algorithms.....	1078
6	Neural Networks.....	1079
6.1	Hopfield Model.....	1079
6.2	Self-Organizing Maps and Kohonen Networks.....	1083
6.3	Adaptation to the Assignment Problem.....	1084
7	Double-Bracket Flows.....	1084
8	Replicator Equations.....	1085
9	Coupled Selection Equations.....	1085
9.1	Selection Equations.....	1086
9.2	How the Coupled Selection Equations Work.....	1087
9.3	Coupled Selection Equations of Pattern Formation.....	1088
9.4	Coupled, Piecewise Continuous Selection Equations.....	1095
10	Numerical Results.....	1100
10.1	Generation of Data Sets.....	1100

*The author would like to thank H. Haken, M.W. Hirsch, and M. Schanz for many fruitful discussions and in particular M. Schanz for the joint publication [122] on which this chapter is based on.

J. Starke (✉)

Department of Mathematics & Computer Science, Technical University of Denmark,
Kongens Lyngby, Denmark
e-mail: jsta@dtu.dk; jens@starke.me

10.2	Generation of Pseudo Random Initial Values.....	1102
10.3	Numerical Solution of Dynamical Systems.....	1103
10.4	Parameter Settings for the Numerical Simulations.....	1103
10.5	Comparison of Several Methods.....	1106
11	Assignments in Distributed Robotic Systems and Flexible Manufacturing.....	1111
11.1	Modeling Flexible Manufacturing Systems.....	1111
11.2	Ad Hoc Assignments with Coupled Selection Equations.....	1112
11.3	Navigation of the Robots.....	1113
11.4	Application Example: Assembling of a Space Station.....	1115
12	Conclusion.....	1117
	Cross-References.....	1118
	Recommended Reading.....	1118

Abstract

Several dynamical system approaches to combinatorial optimization problems are described and compared. These include dynamical systems derived from penalty methods; the approach of Hopfield and Tank; self-organizing maps, that is, Kohonen networks; coupled selection equations; and hybrid methods. Many of them are investigated analytically, and the costs of the solutions are compared numerically with those of solutions obtained by simulated annealing and the costs of a global optimal solution.

Using dynamical systems, a solution to the combinatorial optimization problem emerges in the limit of large times as an asymptotically stable point of the dynamics. The obtained solutions are often not globally optimal but good approximations of it. Dynamical system and neural network approaches are appropriate methods for distributed and parallel processing. Because of the parallelization, these techniques are able to compute a given task much faster than algorithms which are using a traditional sequentially working digital computer.

This chapter focuses on dynamical system approaches to the linear two-index assignment problem and the \mathcal{NP} -hard three-index assignment problem. These and extensions thereof can be used as models for many industrial problems like manufacturing planning and optimization of flexible manufacturing systems. This is illustrated for an example in distributed robotic systems.

1 Introduction

The key idea of a dynamical system approach to combinatorial optimization problems is that the reached asymptotically stable point of a dynamical system is identified with a solution to the combinatorial optimization problem. Clearly, this stable point has to be a feasible solution to the combinatorial optimization problem, that is, all constraints have to be respected in order to get a global optimal solution or at least a good approximation of it. The discrete positions of the asymptotically stable points allow the use of continuous dynamical systems for discrete optimization problems. Here, the question arises of how to define a dynamical system with appropriate properties.

There are many different dynamical system approaches to combinatorial optimization. The most fundamental differences are, first, the initial conditions; second, the set of stable points and the shape of the basins of attraction; and, third, the deterministic and stochastic properties, that is, the presence or absence of additional stochastic terms in the dynamical system.

To initialize the dynamical system, one can use random or calculated initial values. The detailed information about the specific optimization problem like costs of a single assignment can be used as initial values and/or for the definition of the dynamics, that is, the equation of motion. Examples of approaches with random initial values are penalty methods [66], the neural network approach of Hopfield and Tank [73], and variations of it. In [21], initial values with identical elements for different alternatives are used. In contrast to this, the approaches of coupled selection equations in [63, 118, 124] use calculated initial values which contain problem information.

The second difference, which deals with the stability properties of the dynamical system, is the most crucial point. In order to make dynamical systems without cost terms in the equation of motion applicable to combinatorial optimization problems, there must be a one-to-one mapping from the set of asymptotically stable points of the dynamical system to the set of feasible solutions to the combinatorial optimization problem. The reason for this is that they have to be universal for all problems of the same size. Methods which contain cost terms in the equation of motion do not require this one-to-one mapping, that is, not every feasible solution has to be represented as a stable point. This is due to the fact that the cost terms define for each problem a specific dynamical system. Nevertheless, both approaches, with or without cost terms in the equation of motion, must not have stable states which are no feasible solution. These important stability conditions of the dynamical system that are necessary to obtain always feasible solutions are fulfilled only in some approaches (see, e.g., [63, 118, 124]). Others, like [73], may produce spurious states.

The third difference is about the use of additional stochastic terms to prevent the system from getting stuck in a local minimum of the objective function. For this purpose, stochastic terms and simulated annealing techniques are used in [133, 134] to improve the solutions of [73].

The resulting solutions are usually not global optimal but nonetheless good approximations. Compared to other heuristics (see, e.g., [55] in [53]), the advantage of dynamical system approaches which include neural networks is that they are appropriate for distributed and parallel processing. The reason for this is that the dynamical system can be distributed among many coupled simple processing units. Because the communication effort is low, the parallelization is very effective. The use of parallel hardware for dynamical system approaches results therefore in very fast problem solvers. Another advantage of many dynamical system and neural network approaches is their error resistivity, that is, the process of finding a solution can be continued even after disturbances which often appear, for example, in robotics (see, e.g., [94, 123, 125]). This will be explained in Sect. 11.

To be able to compare different approaches, the following analysis focuses on two- and three-index assignment problems (also called two- and three-dimensional assignment problems). Nevertheless, most of the methods described below can be adapted to other combinatorial optimization problems. For solving maximum clique problems with so-called replicator equations, it is referred to [21] and [22]. Furthermore, it is worth mentioning that some of the approaches like penalty methods have their origin in continuous optimization. To apply these methods to combinatorial optimization problems, the set of solutions is restricted to discrete variables by using specific constraints.

2 Assignment Problems

In this section, two important special cases of assignment problems [29], the linear two-index assignment problem and the linear \mathcal{NP} -hard three-index assignment problem, are considered. Assignment problems are simple models for industrial optimization problems from the simple assignment of jobs to machines up to the organization of complex flexible manufacturing systems. This practical relevance justifies and motivates the investigation of assignment problems.

The linear two-index assignment problem allows comparison of the results with the optimal solution even for large problem sizes by using polynomial time algorithms, for example, the dual forest algorithm [2] or the Hungarian method [28, 40, 100]. As a second test example, the three-index assignment problem is investigated as representative of \mathcal{NP} -hard problems.

Definition 1 For given costs $(c_{ij}) \in \mathbb{R}^{n \times n}$, the task of the linear two-index assignment problem (two-dimensional assignment problem) is to find Boolean variables $x_{ij} \in \{0, 1\}$ with $i, j \in N := \{1, \dots, n\}$ so that the total costs

$$c := \sum_{i,j} c_{ij} \cdot x_{ij} \quad (1)$$

are minimal with respect to the constraints

$$\sum_i x_{ij} = 1 \quad \forall j \in N \quad \text{and} \quad (2)$$

$$\sum_j x_{ij} = 1 \quad \forall i \in N. \quad (3)$$

The constraints (2) and (3) imply that (x_{ij}) has to be a permutation matrix.

In graph theory, the assignment problem is known as the weighted bipartite matching problem. The standard example is the assignment of jobs to machines.

Here, n jobs are given which can be worked on each of n machines. The real number c_{ij} defines the costs which are incurred by the execution of job i on machine j :

$$\begin{array}{ccc} \text{job } i & \xrightarrow{\quad \text{machine } j \quad} & \\ \downarrow & \left(\begin{array}{c} c_{11} \cdots c_{1n} \\ \vdots \qquad \vdots \\ c_{n1} \cdots c_{nn} \end{array} \right) & \end{array}$$

The Boolean matrix

$$x_{ij} := \begin{cases} 1 & \text{if job } i \text{ is worked on machine } j \\ 0 & \text{otherwise} \end{cases}$$

is a permutation matrix, that is, every job has to be done once and only once and every machine has to be used once and only once at the same time.

Lemma 1 *The assignment problem (1)–(3) is invariant by adding a constant to each element of a row or column.*

Proof The proof is straightforward: Suppose the constant α_i is added to all elements of row i and the constant β_j is added to all elements of column j . Using the Kronecker symbol

$$\delta_{ii'} := \begin{cases} 1 & \text{for } i = i' \\ 0 & \text{for } i \neq i' \end{cases}$$

and $G := \{x^{(1)}, \dots, x^{(n!)}\}$ which is the set of all $n!$ permutation matrices, it follows with (2) and (3) that

$$\begin{aligned} (x_{kl})^{\text{opt}} &= \arg \min_{x^{(r)} \in G} \left(\sum_{i,j} (c_{ij} + \alpha_{i'} \delta_{ii'} + \beta_{j'} \delta_{jj'}) \cdot x_{ij}^{(r)} \right) \\ &= \arg \min_{x^{(r)} \in G} \left(\sum_{i,j} c_{ij} \cdot x_{ij}^{(r)} + \sum_{i,j} \alpha_{i'} \delta_{ii'} \cdot x_{ij}^{(r)} + \sum_{i,j} \beta_{j'} \delta_{jj'} \cdot x_{ij}^{(r)} \right) \\ &= \arg \min_{x^{(r)} \in G} \left(\sum_{i,j} c_{ij} \cdot x_{ij}^{(r)} + \alpha_i \sum_j x_{ij}^{(r)} + \beta_j \sum_i x_{ij}^{(r)} \right) \end{aligned}$$

$$\begin{aligned}
&= \arg \min_{x^{(r)} \in G} \left(\sum_{i,j} c_{ij} \cdot x_{ij}^{(r)} + \alpha_i + \beta_j \right) \\
&= \arg \min_{x^{(r)} \in G} \left(\sum_{i,j} c_{ij} \cdot x_{ij}^{(r)} \right). \quad \square
\end{aligned}$$

This property is used for the solution of the assignment problem by the Hungarian method [28, 40, 100] to transform the cost matrix into an appropriate form to be used during further steps. Without loss of generality, it can be assumed that $c_{ij} \geq 0 \quad \forall i, j$. Using the above-described transformation, one can always obtain a cost matrix with nonnegative entries so that a permutation matrix can be selected out of the vanishing entries. The necessary computation time of the Hungarian method is of order $O(n^4)$ instead of the exponential dependence of n by comparing all $n!$ feasible solutions using exhaustive search.

Alternative to the minimization problem (1), it is sometimes useful to work with the dual maximization problem, where

$$w := \sum_{i,j} w_{ij} \cdot x_{ij} \quad (4)$$

has to be maximized with winnings $w_{ij} \geq 0$ and the constraints (2) and (3) for $x_{ij} \in \{0, 1\}$. One gets (4) from (1) with the linear transformation

$$w_{ij} = -a \cdot c_{ij} + b \quad \text{with } a > 0 \quad \forall i, j. \quad (5)$$

The condition $w_{ij} \geq 0$ is no restriction, because of Lemma 1: The optimization problems (1) and (4) are invariant by adding a constant to every element of a specific row or column, so that $w_{ij} \geq 0$ can always be fulfilled.

Definition 2 In the (axial) three-index assignment problem (three-dimensional assignment problem), the Boolean variables $x_{ijk} \in \{0, 1\}$ with $i, j, k \in N := \{1, \dots, n\}$ have to be determined such that the total costs

$$c := \sum_{i,j,k} c_{ijk} \cdot x_{ijk} \quad (6)$$

are minimal with respect to the constraints

$$\sum_{i,j} x_{ijk} = 1 \quad \forall k \in N, \quad (7)$$

$$\sum_{i,k} x_{ijk} = 1 \quad \forall j \in N, \quad (8)$$

$$\sum_{j,k} x_{ijk} = 1 \quad \forall i \in N. \quad (9)$$

In a variant of this, the so-called planar three-index assignment problem, the constraints (7)–(9) are replaced with

$$\sum_i x_{ijk} = 1 \quad \forall j, k \in N, \quad (10)$$

$$\sum_j x_{ijk} = 1 \quad \forall i, k \in N, \quad (11)$$

$$\sum_k x_{ijk} = 1 \quad \forall i, j \in N. \quad (12)$$

Both variants of the three-index assignment problem are \mathcal{NP} -hard [29, 47, 115].

The standard example to the axial three-index assignment problem is the assignment of jobs, machines, and workers. A generalization of three-index assignment problems leads to multi-index assignment problems. Assignment problems of different types can be used to model some problems in flexible manufacturing systems, distributed autonomous robotic systems, and cellular robotic systems [118, 121, 123]. Additional details of such applications are given in Sect. 11.

3 Dynamical Systems

Dynamical systems are defined by the change of a state $x(t) \in \mathbb{R}^n$ with respect to the time t . For example, in physical systems, the state vector $x(t) \in \mathbb{R}^6$ may indicate the time-dependent position in space and velocity of a mass point.

In the case of continuous time, the time evolution of the state $x(t)$ is given by the equation of motion

$$\dot{x} = f(x) \quad (13)$$

with $t \in \mathbb{R}$. Here, \dot{x} denotes the time derivative $\frac{d}{dt}x$ of the state x , and $f : \mathbb{R}^n \mapsto \mathbb{R}^n$ determines this change in dependence of the state x . In the case of discrete time, the equation of motion is given by

$$x(t+1) = f(x(t)) \quad (14)$$

with $t \in \mathbb{Z}$.

In this chapter, mainly continuous time systems are considered. Starting with an initial condition $x(0)$, the equation of motion leads to a solution curve $x(t)$ in the state space which is called trajectory; the set of all solution curves is called phase flow of the dynamical system. For large times, the trajectory is often restricted to a

low-dimensional attractor. There exist several kinds of attractors, like limit cycles or strange attractors. The most simple but nevertheless important case of an attractor is an asymptotically stable point. Various techniques for the analysis of the asymptotic behavior are developed. For details see, for example, [7, 9–11, 68, 136], or [110].

In the dynamical system approach to combinatorial optimization, one can restrict to systems with asymptotically stable points x^* . This asymptotic behavior is defined as the limit of large times, that is, $x^* = \lim_{t \rightarrow \infty} x(t)$. These asymptotically stable points are taken as candidates of solutions of the combinatorial optimization problems. As it was already pointed out in the introduction, for some dynamical system approaches, there exists a one-to-one mapping of the asymptotically stable points to the solution of the combinatorial optimization problem (see, e.g., [118]). Others like the replicator dynamics [21] are metastable with respect to the space of feasible solutions (i.e., that the solution is lying on a simplex) or may produce spurious states which are no feasible solution (see, e.g., [73]).

4 Dynamical Systems Based on Penalty Methods

Nonlinear optimization problems are defined by a cost function which has to be minimized and by separately given constraints which have to be respected. By adding the cost function and further terms which contain the constraints, an optimization problem without separately given constraints is obtained. Clearly, by regarding a minimization problem, these terms have to vanish if the constraints are respected and have to be large, that is, have to cause costs or penalties, if the constraints are violated. Therefore, this procedure is called penalty method. Related to penalty methods are barrier methods (see, e.g., [89] and [75]) and interior-point methods (see, e.g., [78, 96]).

4.1 Classical Approach

Penalty methods can be used to solve optimization problems with constraints which are given as equalities or inequalities. Here, only problems with equality constraints are considered. Furthermore, this section focuses on nonlinear optimization problems, that is, either the objective function or cost function itself or the constraints are nonlinear. For further readings, see, for example, [13, 54, 66, 75, 114].

Suppose, the function which has to be minimized is given by $f : \mathbb{R}^n \mapsto \mathbb{R}$, and the equality constraints are given by $h : \mathbb{R}^n \mapsto \mathbb{R}^m$. Thus, the set of feasible points is defined as $G := \{x \in \mathbb{R}^n : h(x) = 0\}$. By using the penalty method, the nonlinear optimization problem

$$x^{\text{opt}} = \arg \min_{x \in G} f(x) \quad (15)$$

where the constraints are given by G is transformed in the unrestricted optimization problem

$$x^{\text{opt}} = \arg \min_{x \in \mathbb{R}^n} \tilde{f}(x) \quad (16)$$

with

$$\tilde{f}(x) = f(x) + \tilde{h}(x). \quad (17)$$

The function $\tilde{h}(x)$ contains the constraints and has to fulfill the condition:

$$\tilde{h}(x) = \begin{cases} 0 & \text{for } h(x) = 0 \\ \text{large positive value} & \text{for } h(x) \neq 0. \end{cases} \quad (18)$$

In order to fulfill the conditions (18), often the square of $h(x)$ multiplied by a large constant p is used:

$$\tilde{h}(x) = p \cdot (h(x))^2. \quad (19)$$

The following theorem states that the penalty method converges globally to a minimum with respect to the constraints (see, e.g., [66]):

Theorem 1 *Let $x(p)$ be a minimum point of (17) with (19), that is,*

$$\tilde{f}(x, p) = f(x) + p \cdot (h(x))^2 \quad (20)$$

on a compact set. Then

$$\lim_{p \rightarrow \infty} x(p) = x_0 \quad \text{and} \quad (21)$$

$$\lim_{p \rightarrow \infty} p \cdot h(x(p)) = 0, \quad (22)$$

where x_0 is the minimum point of f on G . From $0 \leq p < p'$, it follows that the inequalities

$$\tilde{f}(x(p), p) \leq \tilde{f}(x(p'), p') \leq f(x_0), \quad (23)$$

$$f(x(p)) \leq f(x(p')) \leq f(x_0), \quad \text{and} \quad (24)$$

$$h(x(p)) \geq h(x(p')) \geq 0 \quad (25)$$

are fulfilled.

See [66] for a proof.

In spite of this theorem, for the numerical handling of the penalty method, the constant p has to be chosen appropriately in dependence of the finite step size of numerical schemes like the gradient descent method. This is the weak point of these methods.

The large positive value in (18), which has to be fulfilled by the function \tilde{h} , causes a fast relaxation of the initial point $x(0)$ to the set of feasible solutions which respect the constraint $h(x) = 0$. This initial point $x(0)$ contains usually no problem information and is chosen, for example, randomly. See Sect. 9.3.1 for an exception of this. The penalty term $\tilde{h}(x)$ has to be chosen smooth in order to be able to apply gradient descent methods

$$\dot{x} = -\frac{\partial \tilde{f}}{\partial x} \quad (26)$$

to find the minimum. By using a digital computer, a finite step size is necessary, which makes the use of gradient descent methods problematic [75, 89], and one often has to accept low convergence rates because of the unfavorable eigenvalue structure of the problem. To improve the convergence rate conjugate gradient methods, the Newton method or variants of it can be used (see, e.g., [89]).

4.1.1 Constraints of Assignment Problems

The constraints of two-index assignment problems are considered in this section in detail. Other constraints can be treated similarly.

The constraint of Boolean variables $x \in \{0, 1\}$ can be achieved by using

$$x_{ij} \cdot (x_{ij} - 1) = 0 \Leftrightarrow x_{ij} = 0 \text{ or } x_{ij} = 1. \quad (27)$$

To satisfy the condition (18) for all values of x_{ij} , instead of (27), the condition

$$\tilde{h}_{ij}^{(1)} = x_{ij}^2 \cdot (x_{ij} - 1)^2 = 0 \quad (28)$$

can be used. Certainly, this is in some sense just arbitrary; there are many more ways to respect (18) than just to take the square. For an alternative, see the next section.

Dealing with the two-index assignment problem, the constraints (2) and (3) are rewritten as

$$\tilde{h}_j^{(2)} = \left(\sum_i x_{ij} - 1 \right)^2 = 0 \quad \text{and} \quad (29)$$

$$\tilde{h}_i^{(3)} = \left(\sum_j x_{ij} - 1 \right)^2 = 0. \quad (30)$$

Again, for $\tilde{h}_j^{(2)}$ and $\tilde{h}_i^{(3)}$, the square can be used so that the constraints of the two-index assignment problem are respected with the penalty term

$$\tilde{h}(x) = p_1 \sum_{i,j} \tilde{h}_{ij}^{(1)} + p_2 \sum_j \tilde{h}_j^{(2)} + p_3 \sum_i \tilde{h}_i^{(3)}. \quad (31)$$

Because of the symmetry, the constraints $\tilde{h}_j^{(2)}$ and $\tilde{h}_i^{(3)}$ have to be weighted equally. Therefore, the same constant p_2 is chosen for both terms.

The cost function (1) of the two-index assignment problem is treated as function

$$f(x) = \sum_{ij} c_{ij} x_{ij} \quad (32)$$

which has to be minimized. Using a gradient descent method to minimize (17), the resulting equation of motion is

$$\begin{aligned} \dot{x}_{ij} &= -c_{ij} - p_1 \left(4x_{ij}^3 - 6x_{ij}^2 + 2x_{ij} \right) \\ &\quad - 2p_2 \left(\sum_{i'} x_{i'j} - 1 \right) - 2p_2 \left(\sum_{j'} x_{ij'} - 1 \right). \end{aligned} \quad (33)$$

4.2 Variants of Penalty Terms

As mentioned before, there are many ways to respect (18). Instead of just taking the square of the constraints h , one can use several kinds of functions \tilde{h} (see [54, 114], or [132] for examples).

Furthermore, one can use the fact that (27) implies $x_{ij} = x_{ij}^2$ to obtain the penalty term, resulting in

$$\begin{aligned} \tilde{h}(x) &= p_1 \sum_j \left(\sum_i x_{ij}^2 - 1 \right)^2 + p_1 \sum_i \left(\sum_j x_{ij}^2 - 1 \right)^2 \\ &\quad + p_2 \sum_{i,j} \sum_{i' \neq i} x_{ij}^2 \cdot x_{i'j}^2 + p_2 \sum_{i,j} \sum_{j' \neq j} x_{ij}^2 \cdot x_{ij'}^2. \end{aligned} \quad (34)$$

Similarly, because of $x_{ij} = x_{ij}^2$, the original cost function can be changed into

$$f(x) = \sum_{ij} c_{ij} x_{ij}^2. \quad (35)$$

The resulting equation of motion by using a gradient descent method to minimize (17) with \tilde{h} defined in (34) is

$$\dot{x}_{ij} = (8p_1 - 2c_{ij})x_{ij} + 8p_2 x_{ij}^3 - 4(p_1 + p_2)x_{ij} \left(\sum_{j'} x_{ij'}^2 + \sum_{i'} x_{i'j}^2 \right). \quad (36)$$

The penalty term (34) is used in [62] to assign buildings to appropriate sites in regional planning. See in this respect also [31, 32] for describing settlement and occupation processes.

By using a gradient descent method of \tilde{h} in (34), that is, minimizing just the penalty terms without the original objective function, one obtains a dynamical system which is identical to the coupled selection equations described in Sect. 9 where the data of the original optimization problem is contained in the initial values. Utilizing methods from dynamical systems theory such as investigations of stability, attractors and their basins of attraction allow further analysis of the effect of the penalty terms, pointed out in detail in Sect. 9.3.1.

5 Statistical Approaches

The following statistical approaches are usually not regarded as dynamical systems but can be interpreted as very special cases of stochastic dynamical systems with discrete time dynamics. Furthermore, stochastic methods are often used in addition to deterministic methods to avoid stagnation of the dynamics in spurious states. For this reason, their inclusion is regarded in this chapter as justified. The Metropolis algorithm, simulated annealing, evolutionary strategies, and genetic algorithms are considered as a small selection of statistical approaches. Further examples are the greedy randomized adaptive search procedure for the three-index assignment problem proposed in Lidstrom et al. (1996, An approximation algorithm for the three-index assignment problem, unpublished) and the so-called tabu search (see, e.g., [30, 49–51]). See [56] for an overview of several stochastic methods applied to optimization.

5.1 Metropolis Algorithm

The Metropolis algorithm [92] is a modified Monte Carlo method. By the so-called importance sampling, the number of random points in promising areas of the state space is increased. The cost function which has to be minimized is called energy E because of the physical problem considered in the original article [92]. An ensemble of states is considered of which each state is assigned to an energy value. Starting from an initial state x , a randomly chosen test state x' in its local neighborhood is considered. If the energy $E(x')$ of the test state x' is less or equal than the energy $E(x)$ of the old state x , the test state will be used as new state. In the other case, that is, if the energy of the test state is larger, this state is taken with probability proportional to $e^{-\Delta E/T}$. Herein, $\Delta E = E(x') - E(x)$ is the energy difference between the test state and the current state. The parameter $T \in \mathbb{R}_+$ is called temperature. This behavior can be summarized by using the conditional probability

$$P(x \rightarrow x') = \begin{cases} 1 & \text{for } \Delta E = E(x') - E(x) \leq 0 \\ e^{-\Delta E/T} & \text{otherwise} \end{cases} \quad (37)$$

which gives the probability for changing from state x to state x' .

The fact that states with higher energy values can be taken gives the possibility for escaping from a spurious state. By repetition of this procedure, an (thermodynamic) equilibrium is reached. If a number of elements are considered in this energy landscape, this ensemble of elements reaches a Boltzmann distribution $n_r \propto e^{-E_r/T}$ [92]. Herein, n_r is the number of elements of the ensemble in the state r .

5.2 Simulated Annealing

The method of simulated annealing is a special case of a Metropolis algorithm. The idea is taken from material sciences. There, defects in crystals are eliminated by heating and a slow cooling process afterwards. A state of a crystal with defects has higher energy than a state without defects. Fast cooling processes result in the movement of the current state into a state with defects which correspond to a local energy minimum.

The basic idea is identical with the idea of the Metropolis algorithm. But in contrast to the Metropolis algorithm, in simulated annealing, the temperature T is decreased slowly. During this process of decreasing the temperature T , the acceptance of energetically unfavorable states becomes more and more improbable until $T = 0$, where only energetically better states can be accepted. See Fig. 1 for the energy E in dependence of the time t . One can observe that the height of the jumps to larger energy values is decreasing with the time t which is due to the decrease of the temperature T .

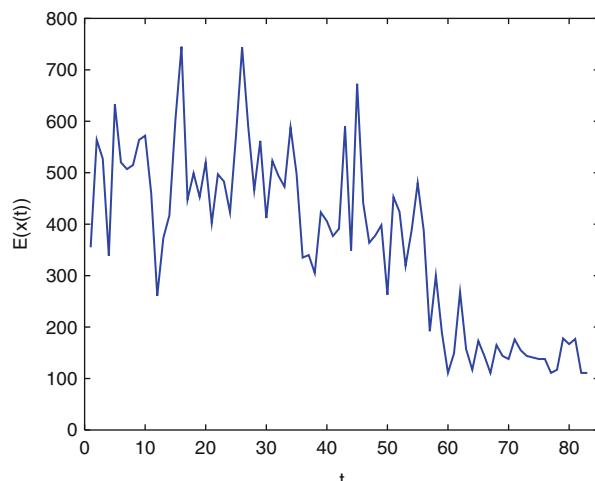


Fig. 1 Energy E over time t of an example of simulated annealing

The method of simulated annealing was used for the traveling salesman problem by Kirkpatrick [81–83]. It is possible to get good results with little computing power. The monotonically decreasing function of the temperature T has to be adapted to the optimization problem and the problem size. This has to be done empirically.

It can be proven that infinitely slow cooling results in a global optimal solution. This and further detailed analysis of simulated annealing can be found in [135]. Clearly, this fact does not help very much in practical applications because only finite computing time is available.

5.3 Evolutionary Strategies and Genetic Algorithms

Nature shows many different kinds of adaptation of several species to specific environmental conditions in biological evolution. Keeping this in mind, one can interpret evolution as optimization method. The basic principles of biological evolution are used by so-called evolutionary strategies (ES) and genetic algorithms (GA) for optimization problems. Rechenberg suggested evolutionary strategies to optimize technical systems like the shape of a jet [109]. The systems which have to be optimized are represented by a set of parameters. These have to be adapted in order to optimize some function like the thrust of the jet. The parameters are represented as real numbers which define some vector. This vector is copied and mutated with a given probability distribution like the deoxyribonucleic acid (DNA) in biological systems. The best of these variants are selected and copied and mutated again. The selection corresponds to Darwin's "survival of the fittest." This procedure results in a more detailed examination of the area in search space around successful variants. This method is used with a different number of parallel existing individuals, descendants, and selected individuals. These evolutionary strategies were developed further and compared with other methods by Schwefel [111].

Genetic algorithms are due to Holland [70]. In contrast to evolutionary strategies, they use a binary coding of the parameters which have to be optimized. The kind of coding is essential for the success of the approach of genetic algorithms. The so-called genetic operator is defined by point mutation, crossover, and inversion. This genetic operator is used to mutate the individuals. The composition of the genetic operator is in some sense arbitrary and requires a lot of experience in applications of these methods. No strict rules are known to construct this operator. Except of the parameter representation, the two approaches are quite similar. They are not distinguished precisely in literature.

In [36, 37], aspects of information theory and entropy concerning evolution and optimization methods are considered. Applications of evolutionary strategies and genetic algorithms to the traveling salesman problem can be found, for example, in [16, 99]. Further applications are published, for example, in [33, 52], and [93].

6 Neural Networks

Systems consisting of a large number of connected simple units are called distributed systems or (artificial) neural networks. The units are able to perform only simple operations. One fundamental advantage of these distributed systems is their error resistivity. An overview can be found in [95] or [65]. Many historically important papers on this topic are collected in [4] and [5]. A short discussion of the use of neural networks for combinatorial optimization is presented in [105, 106, 126, 139], and [113].

6.1 Hopfield Model

In [71], Hopfield points out the analogy between pattern recognition or correcting associative memories and the phase flow in state space of physical dynamical systems with several stable states. In particular, Hopfield suggests an associative memory similar to the Ising model of spin glass theory [19]. The two-state elements are called neurons, and the whole system is called an artificial neural network.

A similar model with continuous states is introduced in [72]. This model is the basis for the approach of Hopfield and Tank for combinatorial optimization in the example of the traveling salesman problem (TSP), wherein one has to find the shortest tour such that each of n given cities is visited once and only once and the tour starts and ends in the same city [73, 74].

The Hopfield model proposed in [72] can be realized in hardware using analog electronic circuits. These consist of a set of electronic nonlinear amplifiers which are interconnected by resistors. Using such a hardware, a parallel processing is natural. The nonlinear amplifiers transform an input (voltage) signal u_i in an output (voltage) signal V_i by

$$V_i = g_i(u_i). \quad (38)$$

The transfer function g_i which increases strictly monotonically with u_i has range $[0, 1]$ for $u_i \in [-\infty, \infty]$, and a well-defined inverse

$$u_i = g_i^{-1}(V_i). \quad (39)$$

Usually, the sigmoidal function $g_i(u_i) = (1 + \tanh(u_i))/2$ is used. For details of the electronic circuit, see [72] or [95]. The equation of motion of the electronic circuit and respectively of the Hopfield model can be written as

$$\tau_i \cdot \dot{u}_i = -u_i + \sum_j w_{ij} \cdot g_j(u_j) \quad (40)$$

with local time constants $\tau_i = C_i R_i$ and synaptic strengths $w_{ij} = R_i / R_{ij}$. The C_i represent capacitors, and R_i and R_{ij} are resistors.

To be able to apply the Hopfield model to optimization problems, the stability properties of the output V_i are important. This is discussed in the following theorem [72] before applications to combinatorial optimization problems are treated in the next sections:

Theorem 2 *For symmetric synaptic strengths $w_{ij} = w_{ji}$, the equation of motion (40) of the Hopfield network will tend to an asymptotically stable point for any initial value.*

Proof The proof is based on a construction of a so-called Lyapunov function E . In this context, this function is often called energy function even if it does not represent the physical energy of the electronic circuit which is modeled by the equation of motion (40). The Lyapunov function

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \frac{1}{R_{ij}} \cdot V_i \cdot V_j + \sum_{i=1}^n \frac{1}{R_i} \int_0^{V_i} g_i^{-1}(V) dV$$

is decreased monotonically by (40) because of

$$\begin{aligned} \dot{E} &= - \sum_i \dot{V}_i \left(\sum_j \frac{1}{R_{ij}} \cdot V_j - \frac{1}{R_i} g_i^{-1}(V_i) \right) \\ &= - \sum_i \dot{V}_i C_i \dot{u}_i \\ &= - \sum_i C_i \frac{dg_i^{-1}(V_i)}{dV_i} \cdot (\dot{V}_i)^2. \end{aligned}$$

The function $g_i^{-1}(V_i)$ increases strictly monotonically which implies that $dg_i^{-1}(V_i)/dV_i$ is positive. The constants C_i are positive for all i . Because of that, $\dot{E} \leq 0$ holds. Furthermore, E is bounded from below if the sigmoidal function is used for $g_i(u_i)$. Therefore, $\dot{E} = 0$ is equivalent to $\dot{V}_i = 0$ and $\dot{u}_i = 0$. This proves the theorem. \square

In the following, all g_i are assumed to be identical, that is, $g_i = g \forall i$. The high-gain limit of the amplifier, that is, an approximation of a step function for $g(u_i)$, gives the quadratic Lyapunov function or energy function

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_{ij} \cdot V_i \cdot V_j - \sum_{i=1}^n \beta_i \cdot V_i \quad (41)$$

with $\alpha_{ij}, \beta_i \in \mathbb{R}$ which depend on the resistors and capacitors of the electronic circuit. If they are chosen appropriate, the Hopfield model minimizes a

quadratic function which may be defined in dependence of a given Boolean, that is, 0/1-combinatorial optimization problem. Because of the approximation of a step function for $g(u_i)$ in the high-gain limit, the outputs agree only approximately with the values 0 and 1.

6.1.1 The Hopfield Model for the Traveling Salesman Problem

In [73], the energy function (41) for the traveling salesman problem (TSP) with distances $c_{ii'} \in \mathbb{R}_+$ between city i and city i' is given by

$$\begin{aligned} E = & \frac{A}{2} \sum_i \sum_j \sum_{j' \neq j} V_{ij} V_{ij'} + \frac{B}{2} \sum_i \sum_j \sum_{i' \neq i} V_{ij} V_{i'j} + \frac{C}{2} \left(\sum_i \sum_j V_{ij} - n \right)^2 \\ & + \frac{D}{2} \sum_i \sum_{i' \neq i} \sum_j c_{ii'} \cdot V_{ij} (V_{i',1+j \bmod n} + V_{i',1+(j-2+n) \bmod n}). \end{aligned} \quad (42)$$

The modulo formulation $1 + j \bmod n$ and $1 + (j - 2 + n) \bmod n$ is used instead of $j + 1$ and $j - 1$ to obtain properly defined indices in the set $N = \{1, \dots, n\}$. The constants $A, B, C, D \in \mathbb{R}_+$ are positive real numbers. The condition $V_{ij} = 1$ indicates the visit to the i th city at position j of the path in the total tour. To describe a tour which starts and ends at the same city and each city is visited once and only once, V has to be a permutation matrix. The first term in (42) causes high energy values for two nonvanishing elements in the same row i , the second term for two nonvanishing elements in the same column j , while the third term causes high energy values for a number of nonvanishing elements larger or smaller than n . The last term consists of the costs or distances c_{ij} . Therefore, this energy function can be regarded as cost function with penalty terms of a penalty method.

The terms in E result in so-called frustrated states. This expression is common in spin glass theory and designates several competing couplings between variables in the sense of low energetic states which cannot all be satisfied simultaneously. This results in several local minima where the system gets trapped in and does not reach a global minimum. As consequence, the approach does not only result in tours which are not the shortest ones but also in unfeasible solutions which do not represent a tour.

The equation of motion of the Hopfield network for the TSP [72, 73] is defined by

$$\begin{aligned} \dot{u}_{ij} = & -\frac{u_{ij}}{\tau} - A \sum_{j' \neq j} V_{ij'} - B \sum_{i' \neq i} V_{i'j} - C \left(\sum_{i'} \sum_{j'} V_{i'j'} - \tilde{n} \right) \\ & - D \sum_{i' \neq i} c_{ii'} (V_{i',1+j \bmod n} + V_{i',1+(j-2+n) \bmod n}) \end{aligned} \quad (43)$$

with $\tau \in \mathbb{R}_+$. The functions $V_{ij} = V_{ij}(u_{ij})$ are sigmoid functions of the form

$$V_{ij}(u_{ij}) = \frac{1}{2} \left(1 + \tanh \left(\frac{u_{ij}}{u_0} \right) \right) \quad (44)$$

where u_0 is a scaling factor. The constant \tilde{n} is used instead of n to adjust the operating point of the amplifiers [73]. The initial values $u_{ij}(0)$ are randomly chosen by

$$u_{ij}(0) = u_{00} + u_0 \left(\Theta - \frac{1}{2} \right) \quad (45)$$

around some center value u_{00} where Θ is a random number in the interval $[0, 1]$. The equation of motion (43) drives the system to the local minima of the energy function E [72]. The spurious states of E cause solutions which do not fulfill the given constraints. In other words, $\lim_{t \rightarrow \infty} V$ may not be a valid tour in the traveling salesman problem.

The problems of the above-mentioned local minima of the energy function E are reported in detail in [137] and [77], among others. The stagnation of the neural network dynamics in local minima of the energy function E can be avoided by using some additional stochastic terms. This was done in [133, 134] with annealing and normalization of the output. Theoretical investigations of stability of the Eq. (43) depending on the parameters A, B, C , and D and supplementary remarks are given in [77] and [90]. In [131], the objective function is written in normal form and is investigated. An analysis of the linearized dynamic of the Hopfield network is done in [48]. Further adaptations of the Hopfield network to special problems are made in [91]. The influence of the initial values is reported in [137] and [131]. The approach of Hopfield and Tank to the TSP is extended to an A/D converter, a signal decision, circuit and a linear programming circuit in [128].

The above-mentioned disadvantages of randomly chosen initial values, parameter sensitivity, and spurious states are avoided in other methods [26, 118] which are reported below. As already mentioned, another possible approach to avoid the stagnation of the dynamics in spurious states is the additional use of statistical methods.

6.1.2 Adaptation to the Assignment Problem

The adaptation of (41) to the two-index assignment problem can also be done easily. This results in the energy function

$$\begin{aligned} E = & \frac{A}{2} \sum_i \sum_j \sum_{j' \neq j} V_{ij} V_{ij'} + \frac{B}{2} \sum_i \sum_j \sum_{i' \neq i} V_{ij} V_{i'j} + \frac{C}{2} \left(\sum_i \sum_j V_{ij} - n \right)^2 \\ & + \frac{D}{2} \sum_i \sum_j c_{ij} \cdot V_{ij}^2, \end{aligned} \quad (46)$$

and the following equation of motion:

$$\dot{u}_{ij} = -\frac{u_{ij}}{\tau} - A \sum_{j' \neq j} V_{ij'} - B \sum_{i' \neq i} V_{i'j} - C \left(\sum_{i'} \sum_{j'} V_{i'j'} - \tilde{n} \right) - D \cdot c_{ij} \cdot V_{ij}. \quad (47)$$

Again, \tilde{n} is used instead of n . The output, which has to be a permutation matrix, is given by the $n \times n$ matrix (V_{ij}).

The choice of the energy function E for the two-index assignment problem is not unique, but the presented one is most similar to the energy function for the TSP presented by Hopfield and Tank.

6.2 Self-Organizing Maps and Kohonen Networks

Another neural network approach to combinatorial optimization is done in [35] and [6] for the TSP by using Kohonen networks, that is, self-organizing maps [84, 85]. This approach is also called elastic net or elastic snake method because of the fit of an initial elastic ring to the cities in the city space.

6.2.1 Self-Organizing Maps for the Traveling Salesman Problem

The elastic ring contains marked points which are iteratively moved to the cities. The initial ring is located in the center of the city positions. The marked points are finally mapped to the locations of the cities. Each marked point of the ring is mainly shifted in the direction of the nearest city. The other cities have a smaller influence which is decreasing with larger distance to the marked point. Eventually each city is assigned to one and only one marked point.

The description of the elastic net method in [35] is quite similar to a penalty method where the objective function

$$E = -\alpha K^2 \sum_i \ln \sum_j \phi(\|x_i - y_j\|, K) + \beta \sum_j \|y_{j+1} - y_j\|^2 \quad (48)$$

with

$$\phi(d, K) = e^{\frac{-d^2}{2K^2}} \quad (49)$$

and parameters α , β , and K has to be minimized. Using the gradient descent method

$$\dot{y}_j = -\frac{\partial E}{\partial y_j} \quad (50)$$

to minimize E , one obtains for the change of the space positions y_j of the marked points

$$\dot{y}_j = \alpha \sum_i \frac{\phi(\|x_i - y_j\|, K)}{\sum_{j'} \phi(\|x_i - y_{j'}\|, K)} (x_i - y_j) + \beta K(y_{j+1} - 2y_j + y_{j-1}). \quad (51)$$

The second term in (51) can be interpreted as elastic force which explains the name “elastic net method.” In [6], one can find a more algorithmic formulation which is also based on the idea of self-organizing maps. These approaches intuitively lead to a short tour of the TSP. The application of the self-organizing maps to the TSP produces also good results for large problems with up to 1,000 cities [6]. There are variants of this approach where the equation of motion cannot be written as gradient descent system (see, e.g., [42]).

6.3 Adaptation to the Assignment Problem

To our best knowledge, there is nothing published about the application of self-organizing maps or Kohonen networks to assignment problems. In principle, it should work to use the above-described strategy with vectors of a canonical basis for x_i , that is,

$$x_{ii'} := \delta_{ii'} \quad (52)$$

with the Kronecker symbol $\delta_{ii'}$ and $i, i' \in \{1, \dots, n\}$. The initial values for the vectors $y_j \in \mathbb{R}^n$ can either be chosen randomly or all equal.

Using an adapted cost term $\beta K \sum_{i,j} c_{ij} y_{ij}$ in (48), the resulting equation of motion is

$$\dot{y}_j = \alpha \sum_i \frac{\phi(\|x_i - y_j\|, K)}{\sum_{j'} \phi(\|x_i - y_{j'}\|, K)} (x_i - y_j) - \beta K(c_{1j}, c_{2j}, \dots, c_{nj})^T. \quad (53)$$

Like in the case of the TSP, where the marked points y_j tend to the cities x_i , the marked points y_j have to tend to the canonical basis vectors x_i . All vectors x_i together define a permutation matrix which is a feasible solution to the two-index assignment problem.

7 Double-Bracket Flows

Brockett introduces in [25] dynamical systems, so-called double-bracket flows, of the form

$$\dot{X} = [X; [X; Y]]$$

to diagonalize symmetric matrices, sort lists, and solve linear programming problems. Here, X and Y are symmetric $n \times n$ matrices, and $[X; Y] := XY - YX$ denotes the commutator. Such a double-bracket flow is used in [26] to solve the two-index assignment problem and is applied in [138] to \mathcal{NP} -hard combinatorial optimization problems. In these references, no numerical results are presented nor compared with other methods. See [20, 25, 26, 64, 127, 138, 140], and references there for further details.

8 Replicator Equations

A biological motivated dynamical system approach was used as heuristics for finding a maximum clique in a graph [22]. The so-called replicator equation

$$\dot{x}_i = x_i ((Wx)_i - x^T W x) \quad (54)$$

is used there to select from a graph the maximum clique [21]. The elements of the vector (x_1, \dots, x_N) describe the species. The weights $W = A + I$ are computed with the adjacency matrix A of the considered graph G and the identity matrix I .

The replicator dynamics is used in theoretical biology to describe an evolution process or a selection behavior of different species over time. The differential equation (54) has the nice property that the standard simplex $S_N = \{x \in \mathbb{R}^N : x_i \geq 0 \quad \forall i \text{ and } \sum_i x_i = 1\}$ is an invariant set with respect to its flow, that is, that its solutions stay on S_N . This fact simplifies certain types of analysis. Numerical results and extensions can be found, for example, in [23, 24, 101–104].

To our best knowledge, there are no results available about the application of replicator equations (54) to solve assignment problems. However, in [129] and [130], a synthesis of the replicator approach and the coupled selection equations described in Sect. 9 is used in a deterministic annealing procedure increasing iteratively the coupling strength between row and column elements and is applied to quadratic assignment problems.

Even though there is also a selection process involved, the approach should not to be mixed with the one of the coupled selection equations described in Sect. 9. The approach of the replicator equations does contain the detailed problem information only in the right-hand side of the equation of motion, and the vector $(1/N, \dots, 1/N)$ is used as initial value, that is, in contrast to the coupled selection equations, no problem information is used for the initial value.

9 Coupled Selection Equations

Using the method of coupled selection equations for combinatorial optimization problems, the initial conditions are calculated from the specific data of the problem, that is, the costs. They determine together with the basins of attraction the final solution. In contrast to standard penalty methods, the approach of Hopfield and

Tank, and the replicator dynamics approach, there are not necessarily cost terms, that is, specific problem data, in the equation of motion. The solution is obtained as output of the dynamical system in the limit of large times, when the state variables have reached the asymptotic stable points. Selection equations can be found in models of many biological, chemical, and physical systems [58, 59]. Below, the selection equations of pattern formation, which can be used for pattern recognition, are described first before going on to outline an approach to combinatorial optimization problems using coupled selection equations.

9.1 Selection Equations

The neural network approaches to associative memories which are described in [72, 80] and are motivated by physical spin glasses show many spurious states, that is, the dynamic ends often in patterns which are not stored. This unwanted behavior led to investigations about the storage capacity of these networks by means of statistical analysis (see, e.g., [3, 65, 95]).

To avoid the spurious states, Haken introduced a dynamical system where the stored patterns are the only asymptotically stable points of the dynamics. This dynamical system is called synergetic computer (see [57, 60, 61], and references there). The patterns are stored as vectors $\mu_i = (\mu_{i1}, \dots, \mu_{in})^T \in \mathbb{R}^n$ of the pixels for all $i \in M = \{1, \dots, m\}$. The vector component μ_{ij} represents the gray value of the corresponding pixel j of pattern i .

It is assumed that the vectors of the patterns are linear independent and that the number m of patterns is at most n , the number of pixels. The dynamics can be described with so-called order parameters ξ_i which are coefficients of the expansion of the test pattern $v \in \mathbb{R}^n$ with respect to the stored patterns μ_i and a rest vector ϱ

$$v = \sum_{i=1}^m \xi_i \mu_i + \varrho. \quad (55)$$

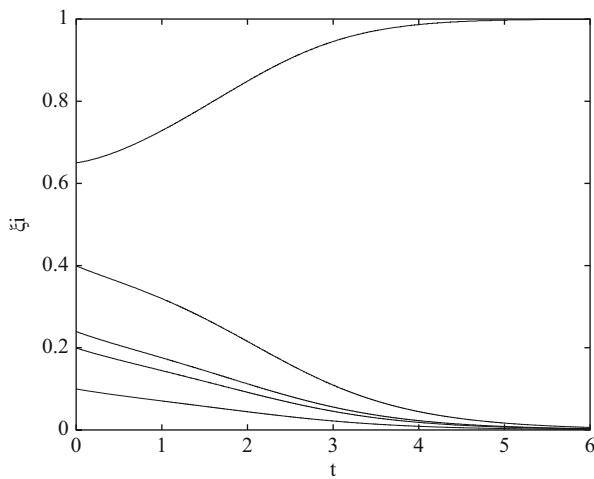
The equation of motion of the order parameters is defined by

$$\dot{\xi}_i = \xi_i - \xi_i^3 - \beta \cdot \xi_i \sum_{i' \neq i} \xi_{i'}^2 \quad \forall i \in M \quad (56)$$

with $\beta > 1$. To avoid the stagnation at some stationary but instable point, some noise is added to (56). To get a visual impression of the selection behavior of (56), see Fig. 2 for an example of the solution $\xi(t)$ over time. The selection behavior is formulated in the following theorem:

Theorem 3 *The dynamical system (56) is a selection equation, that is, one and only one mode survives, namely, the mode with the largest corresponding initial value ($\xi_{i'} \rightarrow 1$ for $t \rightarrow \infty$), and all others decay to zero ($\xi_i \rightarrow 0$ with $i \neq i'$ for $t \rightarrow \infty$).*

Fig. 2 Presentation of the components of the solution $\xi(t)$ over time t showing the selection behavior for $\beta = 2$. The component with the largest initial value wins the competition and converges to one, while all others decay to zero



Proof The proof is based on linear stability analysis and the fact that from $\xi_i > \xi_j$ it follows that $\dot{\xi}_i > \dot{\xi}_j$. See [60] for details. Equation (56) can be formulated as gradient flow $\dot{x}_i = -\frac{\partial \phi}{\partial \xi_i}$ with the potential ϕ which is plotted in Fig. 4. Further explanations can be found in Sect. 9.3 about the coupled selection equations. \square

A corresponding dynamical system can be found in the Bénard problem of fluid dynamics. There, it describes the selection behavior of roll patterns. The similarity between pattern formation and pattern recognition was pointed out by Haken [18, 57]. In [14, 67], a similar approach to an associative memory and related systems can be found which uses a linear mapping to a dynamical system with well-known asymptotic properties. Further selection equations can be found in models of biological selection of autocatalytic macromolecules. See the work of Eigen and Schuster in [38, 39] and work based on these in [37, 69].

Concerning coupled selection equations which set of asymptotically stable points is identical with the set of permutation matrices, a similar result to Theorem 3 will be stated below.

9.2 How the Coupled Selection Equations Work

The coupled selection equation approach (see, e.g., [63, 116–118, 124]) to combinatorial optimization is based on the idea of the competition between several modes or variables with respect to the constraints. By regarding dynamical systems which allow not only one but several modes to survive, it is possible to get complex solutions by composing several single modes. This competition of parts of the whole solution simultaneously operating in continuous space by the coupled selection equations leads to a significant reduction of the computational complexity. In contrast to this, a

comparison of all feasible solutions is impossible for problem sizes $n > 15$ because of the exponential growth of the number of all feasible solutions.

In the following, the maximization problems (4) which are transformations of two-index assignment problems are treated as well as the corresponding three-index problems. The maximization of the total winnings, that is, the minimization of the total costs, is done by a selection of the largest single winnings. The single modes or variables are initialized by the single winnings, that is, transformed costs. Therefore, the largest initial values have to win the competition. The total procedure can be regarded as a heuristic method to maximize the total winnings, that is, the sum of the single winnings.

This approach is similar to the intuitive approach of a human problem solver. Several alternatives which are parts of the total solution, are compared during the process of decision making, that is, the decisions for the alternatives are competing against each other.

9.2.1 Problem Information as Initial Values

The detailed problem information can be found in the valuation $w \in \mathbb{R}_+^n$ and in the constraints. The valuations are given by initial values to the dynamical system and are not included in the equation of motion itself. The constraints are respected by specific coupling terms of the dynamical system. The solution to the combinatorial optimization problem emerges as limit of large times $t \rightarrow \infty$. The ω -limit set of the dynamical systems considered here contains stable and unstable points only. There are no limit cycles or more complicated attractors. The position of the asymptotically stable points guarantees the emergence of feasible solutions only. The other way around, every feasible solution corresponds to one and only one asymptotically stable point.

The dynamical process changes the single continuous, that is, real-valued, winnings in Boolean values. This describes the selection of suitable, that is, as large as possible, single winnings as part of the total solution. The Boolean value 1 marks the choice of the corresponding single decision with corresponding winnings, while the Boolean value 0 marks that this alternative was not chosen.

In the following sections, two different types of coupled selection equations are considered each for the two- and three-index assignment problems as well as a hybrid method between coupled selection equations and a penalty approach. These are the coupled selection equations of pattern formation and the coupled, piecewise continuous selection equations. For an adaptation of the coupled selection equations to other optimization problems, appropriate coupling terms have to be used which lead to asymptotically stable points which respect the constraints.

9.3 Coupled Selection Equations of Pattern Formation

By numbering the variables with two indices and using specific coupling terms, the selection equation (56) can be adapted such that they can be used for two-index assignment problems.

9.3.1 Two-Index Assignment Problem

The given matrix (w_{ij}) of nonnegative winnings of the dual two-index assignment problem, that is, the maximization problem (4), is used to initialize the coupled selection equations with variables $(\xi_{ij}) \in \mathbb{R}_+^{n \times n}$, as it is described before, that is,

$$\xi_{ij}(0) := w_{ij} \quad \forall i, j. \quad (57)$$

For calculating w_{ij} , the transformation (5) is used so that w_{ij} lies on the interval $[0, 1]$.

To summarize the procedure of the coupled selection equation approach to combinatorial optimization again, the final solution of the optimization problem emerges in the limit ξ^* for $t \rightarrow \infty$ as a Boolean variable $x_{ij} \in \{0, 1\}$ for each element from the problem information $c_{ij} \in \mathbb{R}$, that is, the given costs:

$$c_{ij} \mapsto w_{ij} = \xi_{ij}(0) \longrightarrow \xi_{ij}(t) \longrightarrow \xi_{ij}^* = x_{ij}. \quad (58)$$

Extending the selection equations of pattern formation (56) with further coupling terms, the dynamical system

$$\dot{\xi}_{ij} = \xi_{ij} - \xi_{ij}^3 - \beta \cdot \xi_{ij} \sum_{i' \neq i} \xi_{i'j}^2 - \beta \cdot \xi_{ij} \sum_{j' \neq j} \xi_{ij'}^2, \quad (59)$$

results. All variables ξ_{ij} in the same row and same column are coupled with each other. For references, see [63, 116–118, 124]. Even though these equations were developed by the author of this chapter from scratch and independently, similar equations appeared already before in the context of an algorithmic procedure for the traveling salesman problem [15] but without detailed analytical investigations.

Theorem 4 *The set of asymptotically stable points of the dynamical system (59) is identical with the set of permutation matrices for nonnegative initial values and $\beta > 1/2$.*

Proof The proof is based on the examination of the Hesse matrix of the corresponding potential function. See Sect. 9.3.2 for the potential function and [118] for further details. \square

This theorem states that there always emerges a feasible solution in the limit of large times if a little bit noise is assumed to be present which prevents the system to stay at saddle points. Figure 3 shows this behavior in four steps of the evolution for the problem sizes $n^2 = 5^2$ and $\beta = 2$.

As described above, the specific problem definition is represented by the initial values and not by the dynamical system itself. That is, in contrast to penalty methods or the approach of Hopfield and Tank, for all transformed linear assignment problems (4), the same dynamical system (59) is used.

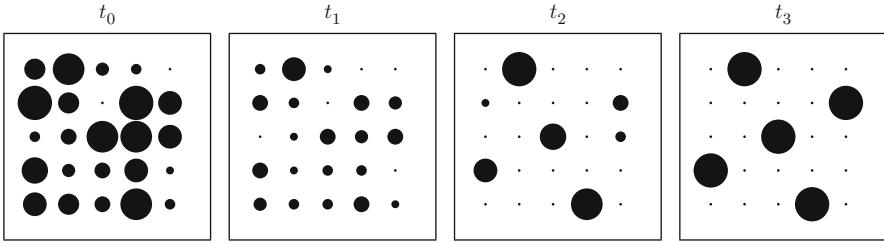


Fig. 3 Four time steps of a simulation of the coupled selection equations (59) solving a two-index assignment problem. The *dots* are arranged as a matrix (ξ_{ij}). The size of the *dots* is proportional to the values ξ_{ij} . The time t_0 indicates the initial state. At time t_3 , a stable point emerges which corresponds to a permutation matrix

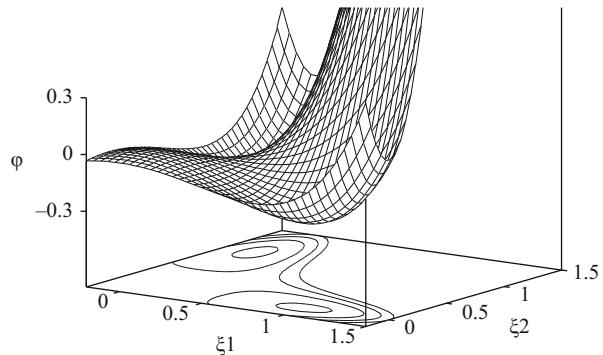


Fig. 4 Potential function φ for two variables $\xi_1, \xi_2 \geq 0$ with $\beta = 4$

9.3.2 Corresponding Gradient Flow and Potential Function

To get a visual impression of how the stability properties work, one can rewrite the equation of motion (59) as gradient flow

$$\dot{\xi}_{ij} = -\frac{\partial \varphi}{\partial \xi_{ij}} \quad \forall i, j \quad (60)$$

with the corresponding potential function

$$\varphi = -\frac{1}{2} \sum_{i,j} \xi_{ij}^2 + \frac{1}{4} \sum_{i,j} \xi_{ij}^4 + \frac{\beta}{4} \sum_{i,j} \sum_{i' \neq i} \xi_{ij}^2 \cdot \xi_{i'j}^2 + \frac{\beta}{4} \sum_{i,j} \sum_{j' \neq j} \xi_{ij}^2 \cdot \xi_{ij'}^2. \quad (61)$$

Figure 4 shows a plot of φ for two variables in one row (or column) which are denoted with ξ_1 and ξ_2 . This corresponds to the selection equation (56) discussed before. The manner in which the coupled selection equations work can be illustrated for the two variables ξ_1 and ξ_2 by a decomposition of the potential function φ in two parts φ_1 and φ_2 with $\varphi = \varphi_1 + \varphi_2$, where

Fig. 5 Potential function φ_1 for two variables $\xi_1, \xi_2 \geq 0$

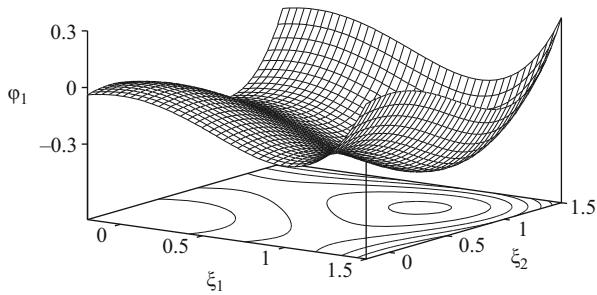
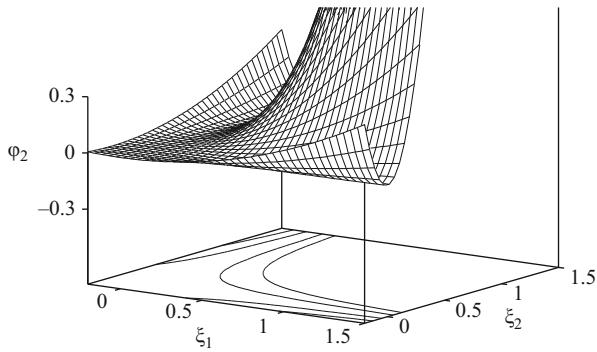


Fig. 6 Potential function φ_2 for two variables $\xi_1, \xi_2 \geq 0$ with $\beta = 4$



$$\varphi_1 = -\frac{1}{2} \sum_{i=1}^2 \xi_i^2 + \frac{1}{4} \sum_{i=1}^2 \xi_i^4 \quad (62)$$

and

$$\varphi_2 = \frac{\beta}{4} \sum_{i=1}^2 \sum_{i' \neq i} \xi_i^2 \cdot \xi_{i'}^2. \quad (63)$$

The potential function φ_1 is shown in Fig. 5. As one can see in Fig. 6, the potential function φ_2 prevents both of the variables ξ_1 and ξ_2 from being nonvanishing in a stable state by putting a “hill” in the corresponding area. This function φ_2 describes the coupling terms in the coupled selection equations. The superposition of φ_1 and φ_2 causes the minimum of φ_1 to be replaced by two minima in the total potential function φ . They are placed at the two saddle points of φ_1 . See Figs. 4–6 for illustration.

9.3.3 Interpretation as Penalty Terms

There is an amazing correlation of the coupled selection equations of pattern formation (59) to a penalty method with specific penalty terms. By adding $\frac{1}{4}$ to the potential (61), it can be written as

$$\tilde{\varphi} = \varphi + \frac{1}{4} \quad (64)$$

$$\begin{aligned} &= p_1 \sum_j \left(\sum_i \xi_{ij}^2 - 1 \right)^2 + p_1 \sum_i \left(\sum_j \xi_{ij}^2 - 1 \right)^2 \\ &\quad + p_2 \sum_{i,j} \sum_{i' \neq i} \xi_{ij}^2 \cdot \xi_{i'j}^2 + p_2 \sum_{i,j} \sum_{j' \neq j} \xi_{ij}^2 \cdot \xi_{ij'}^2 \end{aligned} \quad (65)$$

with $p_1 = \frac{1}{8}$ and $p_2 = \beta/4 - p_1$. Therefore, the potential (65) can be interpreted as penalty terms (18) of a penalty method and is identical to (34).

9.3.4 Hybrid Method: Coupled Selection Equations and Penalty Method

By using the objective function (35) together with these penalty terms and a gradient descent method, one obtains the equation of motion (36) as described in Sect. 4.2. The specific choice for the penalty terms which lead to Eq. (36) was certainly made to obtain this correspondence and to take advantage of mathematical tools from dynamical systems theory in particular the stability properties of (59). See [63] and [124] for further details.

Due to the cost term in (36), their stability properties change compared to those reported in Theorem 4 of the dynamical system (59). Because of the cost term in the equation of motion, the stability properties depend on the cost values. Also the basins of attraction of the dynamical system (18) are deformed compared with those of the coupled selection equations (59). A rough calculation can be done to demonstrate this fact. Assuming a stationary point has one and only one nonvanishing element in each row and each column, it follows from (36) that

$$0 = 2c_{ij} - 8p_1 + 8p_1\xi_{ij}^2 \quad \Leftrightarrow \quad \xi_{ij} = \sqrt{1 - \frac{c_{ij}}{4p_1}} \quad (66)$$

for these nonvanishing elements of the stationary points. This gives the deviation of the non-vanishing elements from 1 in dependence of p_1 and the costs c_{ij} . In [44], the stability investigations are extended for the case of larger costs c_{ij} shifting the position of the asymptotically stable positions.

Using the equation of motion (36), that is, coupled selection equations of pattern formation with an additional cost term, and the calculated initial values (57), one obtains a hybrid method between the coupled selection equations and the penalty method. As mentioned above, the basins of attraction are deformed compared to those of the coupled selection equations of pattern formation. This implies an advantageous change in the solution behavior which is discussed in Sect. 10.

9.3.5 Smooth Exact Penalty Function

The downside of having a dynamical system like (36) with asymptotically stable states which are just approximative permutation matrices can be overcome by

choosing yet another formulation of the penalty method or the dynamical system, respectively. A fourth-order objective function is used for this purpose in the following which is identical to the linear one in (1) or (32) at the positions of the Boolean variables, that is, for $x_{ij} \in \{0, 1\}$:

$$f(x) = \sum_{i,j} c_{ij} x_{ij} = \sum_{i,j} c_{ij} (2x_{ij}^2 - x_{ij}^4). \quad (67)$$

It can be observed that for $x_{ij} = 0$ and $x_{ij} = 1$, it holds that $(2x_{ij}^2 - x_{ij}^4) = x_{ij}$. The change of the originally linear objective function (32) to a nonlinear one with second- and fourth-order terms in (67) would usually be considered as disadvantageous, and it would be expected that this results in additional unwanted local minima corresponding to unfeasible solutions. Nevertheless, in the special case which is considered here, it is of big advantage, and it can be proven that there are only minima at feasible points.

The effective objective function of the penalty method can with (67) be written as

$$\begin{aligned} \tilde{\varphi} = & p_0 \sum_{i,j} c_{ij} (2\xi_{ij}^2 - \xi_{ij}^4) \\ & + p_1 \sum_i \left(\sum_j \xi_{ij}^2 - 1 \right)^2 + p_1 \sum_j \left(\sum_i \xi_{ij}^2 - 1 \right)^2 \\ & + p_2 \sum_{i,j} \left(\sum_{i' \neq i} \xi_{ij}^2 \xi_{i'j}^2 + \sum_{j' \neq j} \xi_{ij}^2 \xi_{ij'}^2 \right). \end{aligned}$$

Using as before a gradient descent method to minimize $\tilde{\varphi}$ results in the equation of motion

$$\frac{d}{dt} \xi_{ij} = (1 - \alpha c_{ij}) (\xi_{ij} - \xi_{ij}^3) - \beta \xi_{ij} \left(\sum_{j' \neq j} \xi_{ij'}^2 + \sum_{i' \neq i} \xi_{i'j}^2 \right), \quad (68)$$

where $\alpha > 0$ is small and the condition $\beta > 1/2(1 - \alpha \min_{i,j} c_{ij})$ has to be satisfied to be able to guarantee permutation matrices as asymptotically stable points. The initial values are chosen as before:

$$\xi_{ij}(0) = w_{ij} = -a \cdot c_{ij} + b \quad \forall i, j \quad \text{with } a > 0. \quad (69)$$

The fact that the inhomogeneity of the growth rates α_{ij} affects in (68) not only the linear term but also the cubic one leads in contrast to [43, 63, 124], and [44] to the wanted behavior. The ratio of the coefficients of these two terms is one, and the stable points are therefore located exactly at permutation matrices with $\xi_{ij} = 0$ or

$\xi_{ij} = 1$ (suppose the coupling strength is strong enough) and not at intermediate values. This is not only mathematically more elegant but simplifies also practical applications. This is formulated in the following:

Theorem 5 *The set of asymptotically stable points of the gradient flow (68) for the smooth exact penalty function with nonnegative initial values and $\beta > 1/2(1 - \alpha \min_{i,j} c_{ij})$ is identical to the set of feasible solutions of the two-index assignment problem (permutation matrices).*

Proof The proof is based on examination of the Hesse matrix $\left(\frac{\partial^2 \tilde{\varphi}}{\partial \xi_{ij} \partial \xi_{i'j'}}\right)$ for the gradient flow $\frac{d}{dt} \xi_{ij} = \frac{-\partial V}{\partial \xi_{ij}}$. The key point here is the scaling of $(\xi_{ij} - \xi_{ij}^3)$ with $(1 - \alpha c_{ij})$ as the ratio of the coefficients determines the positions of the asymptotically stable points. \square

9.3.6 Three-Index Assignment Problem

To treat the axial three-index assignment problem (6)–(9) or the corresponding dual problem, that is, the maximization problem, a three-index variable $(\xi_{ijk}) \in \mathbb{R}_+^{n \times n \times n}$ is necessary. Apart from that, the line of proceeding is identical with the introduced approach of coupled selection equations for the two-index assignment problem. As in the two-index case, the initial values are calculated by

$$\xi_{ijk}(0) := w_{ijk} \quad \forall i, j, k \quad (70)$$

using the winnings w_{ijk} . The equation of motion is defined by

$$\begin{aligned} \dot{\xi}_{ijk} &= \xi_{ijk} + (3\beta - 1)\xi_{ijk}^3 \\ &- \beta \cdot \xi_{ijk} \left(\sum_{i',j'} \xi_{i'j'k}^2 + \sum_{i',k'} \xi_{i'jk'}^2 + \sum_{j',k'} \xi_{ij'k'}^2 \right) \end{aligned} \quad (71)$$

such that the coupling terms ensure asymptotically stable points which respect the constraints of the axial three-index assignment problem. These couplings lead to a competition between all elements in the same plane. The stability properties can be formulated by:

Theorem 6 *The asymptotically stable points of (71) respect the constraints $\xi_{ijk} \in \{0, 1\} \forall i, j, k$, and (7)–(9). In addition, all points which respect these constraints are asymptotically stable points of (71).*

A proof can be found in [118].

The decision variables (x_{ijk}) can therefore be identified with the asymptotically stable points (ξ_{ijk}^*) of the dynamical system (71) as it is done in the two-index case.

The approach works similar for the planar three-index assignment problem with constraints (10)–(12), but in contrast to (71), the coupling terms are reduced from

index planes which are characterized by two varying indices to pillars with only one varying index:

$$\begin{aligned}\dot{\xi}_{ijk} &= \xi_{ijk} + (3\beta - 1)\xi_{ijk}^3 \\ &\quad - \beta \cdot \xi_{ijk} \left(\sum_{i'} \xi_{i'jk}^2 + \sum_{j'} \xi_{ij'k}^2 + \sum_{k'} \xi_{ijk'}^2 \right).\end{aligned}\quad (72)$$

The following comparison of numerical results in Sect. 10 includes the axial three-index assignment problem but not the planar one.

Coupled selection equations for multi-index assignment problems can be defined similarly to the extension of the two-index assignment problem to the (axial and planar) three-index one.

9.4 Coupled, Piecewise Continuous Selection Equations

Using coupled, piecewise continuous selection equations, more far-reaching analytical results can be obtained than with the smooth dynamical systems (59) and (71) with nonlinear right-hand sides consisting of polynomials up to third order.

A criterion is given which shows whether a solution obtained by the coupled selection equations is global optimal (Starke and Hirsch, 1996, Combinatorial optimization based on coupled piecewise continuous selection equations, unpublished). See also [118] for further details. Such a criterion is very useful, since it tells whether the solution can or cannot be improved by further trials with different optimization methods. Again, there are no spurious states in the coupled, piecewise continuous selection equations introduced in the following.

9.4.1 Two-Index Assignment Problem

The dynamical system for handling the transformed two-index assignment problem (4), that is, the coupled, piecewise continuous selection equations, is defined for all i, j as

$$\dot{\xi}_{ij} = \begin{cases} 1 - \xi_{ij} - \beta \sum_{i' \neq i} \xi_{i'j} - \beta \sum_{j' \neq j} \xi_{ij'} & \text{for } \xi_{ij} > 0 \\ 0 & \text{for } \xi_{ij} = 0 \\ \chi & \text{for } \xi_{ij} < 0 \end{cases}\quad (73)$$

with the constant $\chi > 0$.

Theorem 7 *The set of asymptotically stable points of the coupled, piecewise continuous selection equations (73) is identical with the set of permutation matrices for nonnegative initial values and $\beta > 1/2$.*

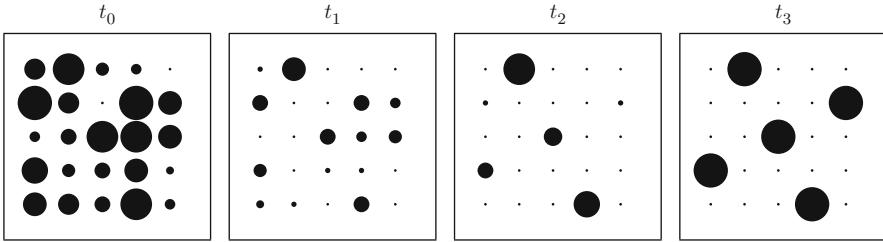


Fig. 7 Four time steps of a simulation of the coupled, piecewise continuous dynamical system (73) solving a two-index assignment problem. The *dots* are arranged as a matrix (ξ_{ij}). The size of the *dots* is proportional to the values ξ_{ij} . The time t_0 indicates the initial state. At time t_3 , a stable point emerges which corresponds to a permutation matrix

The proof is based on examining the neighborhood of the stationary points. The piecewise continuous dynamical system does not allow standard stability analysis (Starke and Hirsch, 1996, Combinatorial optimization based on coupled piecewise continuous selection equations, unpublished). See [118] for details.

The initial values are chosen as in the case of the coupled selection equations of pattern formation by (57). The emergence of a permutation matrix by using the dynamical system (73) is shown in Fig. 7. Some minor differences to Fig. 3 can be observed in the time evolution. Due to the jump in the equation of motion (73) of the coupled, piecewise continuous selection equations, some elements decay to zero after a finite time. In the case of coupled selection equations of pattern formation (59), the decay to zero is exponential.

It should be remarked that for all $\xi_{ij} > 0$, due to the linearity of the system in the form $\dot{x} = Ax$ with a matrix $A \in \mathbb{R}^{n \times n}$, it is possible to give the analytical solution as $x(t) = \exp(At)x(0)$. After an element ξ_{ij} reaches zero, one has to reduce the system with respect to this variable, and one can repeat the procedure. Nevertheless, the numerical comparison in the following did not take advantage of this.

The system (73) can be adapted to three- or multi-index assignment problems. Likewise, the following analysis of (73) can be extended to these problems, that is, also to the \mathcal{NP} -hard three-index assignment problem.

9.4.2 Analysis of the Basins of Attraction

The ability of the piecewise continuous dynamical system to solve assignment problems is determined by the position of the stable points and the basins of attraction like in all other dynamical system approaches. Therefore, the analysis of the basins of attraction is important.

Theorem 8 *For the dynamical system (73), the following relations are true with two permutation matrices $x^{(r)}$ and $x^{(s)}$:*

$$\sum_{i,j} \xi_{ij} x_{ij}^{(r)} \leqq \sum_{i,j} \xi_{ij} x_{ij}^{(s)} \quad \Rightarrow \quad \sum_{i,j} \dot{\xi}_{ij} x_{ij}^{(r)} \leqq \sum_{i,j} \dot{\xi}_{ij} x_{ij}^{(s)}, \quad (74)$$

if $\beta > 1/2$ and for all i, j $(x_{ij}^{(r)} = 1 \vee x_{ij}^{(s)} = 1) \Rightarrow \xi_{ij} \neq 0$. This means that the hyperplane

$$\sum_{i,j} \xi_{ij} x_{ij}^{(r)} = \sum_{i,j} \xi_{ij} x_{ij}^{(s)} \quad (75)$$

is repelling and separates the two basins of attraction of the asymptotically stable points $\xi^* = x^{(r)}$ and $\xi^* = x^{(s)}$.

The following definition is used in the proof:

Definition 3 The index sets N^2 , I^2 , and R^2 are defined by:

$$\begin{aligned} N^2 &:= N \times N. \\ I^2 &:= \{(i, j) \in N \times N : \xi_{ij} > 0\}. \\ R^2 &:= \{(i, j) \in N \times N : \xi_{ij} = 0\}. \end{aligned}$$

Proof of Theorem 8. Using the dynamical system (73) with $\beta > 1/2$ and the constraints of permutation matrices (2) and (3), it follows that

$$\begin{aligned} \sum_{i,j} \dot{\xi}_{ij} x_{ij}^{(r)} &= \sum_{(i,j) \in I^2} \left(1 + (2\beta - 1)\xi_{ij} - \beta \sum_{i' \in N} \xi_{i'j} - \beta \sum_{j' \in N} \xi_{ij'} \right) x_{ij}^{(r)} \\ &= |I^2| + (2\beta - 1) \sum_{(i,j) \in I^2} \xi_{ij} x_{ij}^{(r)} - \beta \sum_{(i,j) \in I^2} \sum_{i' \in N} \xi_{i'j} x_{ij}^{(r)} \\ &\quad - \beta \sum_{(i,j) \in I^2} \sum_{j' \in N} \xi_{ij'} x_{ij}^{(r)} \\ &= |I^2| + (2\beta - 1) \sum_{(i,j) \in N^2} \xi_{ij} x_{ij}^{(r)} - \beta \sum_{i',j} \xi_{i'j} \sum_i x_{ij}^{(r)} \\ &\quad - \beta \sum_{i,j'} \xi_{ij'} \sum_j x_{ij}^{(r)} \\ &= |I^2| + (2\beta - 1) \sum_{i,j} \xi_{ij} x_{ij}^{(r)} - 2\beta \sum_{i,j} \xi_{ij} \\ &\stackrel{\leq}{\geq} |I^2| + (2\beta - 1) \sum_{i,j} \xi_{ij} x_{ij}^{(s)} - 2\beta \sum_{i,j} \xi_{ij} \\ &= \sum_{i,j} \dot{\xi}_{ij} x_{ij}^{(s)}. \end{aligned}$$

□

By using this theorem, a criterion can be given to check if a solution to the two-index assignment problem obtained by the coupled, piecewise continuous selection equations is optimal or not.

9.4.3 Criterion for Global Optimal Solutions

The quality of the solutions of dynamical systems approaches is decisively given by the shape of the basins of attraction. Suppose the selection described by the coupled, piecewise continuous selection equations changes variables permanently to zero one after another, that is, variables are permanently eliminated as candidates for nonvanishing elements in the solution of the optimization problem. Then a recursive application of [Theorem 8](#) to the remaining nonvanishing elements guarantees an optimal solution. This fact is described in the following theorem:

Theorem 9 *The solution, that is, the reached asymptotically stable point of the dynamical system (73), is optimal in the sense of the two-index assignment problem (4) if for all t' it follows that*

$$\xi_{ij}(t') = 0 \quad \Rightarrow \quad \forall t > t' \quad \xi_{ij}(t) = 0 \quad (76)$$

is true for all ξ_{ij} , that is, if an element becomes zero and is staying at zero.

Proof The proof follows directly from [Theorem 8](#) concerning the basins of attraction and [Theorem 7](#) concerning the asymptotically stable points of the coupled, piecewise continuous selection equations. \square

The set of all linear, two-index assignment problems where an optimal solution can be guaranteed by using the coupled, piecewise continuous selection equations is given as

$$L^{\text{opt}} = \left\{ \xi(0) \in \mathbb{R}_+^{n \times n} : \forall t' \quad \xi_{ij}(t') = 0 \quad \Rightarrow \quad \forall t > t' \quad \xi_{ij}(t) = 0 \right\}$$

where $\xi_{ij}(t)$ is the solution of the dynamical system (73). In future work, a criterion for the initial values $\xi(0)$ must be sought which allows one to determine L^{opt} without simulation of the coupled, piecewise continuous selection equations.

9.4.4 Three-Index Assignment Problem

The extension of (73) to the axial three-index assignment problem results in

$$\dot{\xi}_{ijk} = \begin{cases} 1 + (3\beta - 1)\xi_{ijk} - \beta \left(\sum_{i',j'} \xi_{i'j'k} + \sum_{i',k'} \xi_{i'jk'} + \sum_{j',k'} \xi_{ij'k'} \right) & \text{for } \xi_{ijk} > 0 \\ 0 & \text{for } \xi_{ijk} = 0 \\ \chi & \text{for } \xi_{ijk} < 0 \end{cases} \quad (77)$$

with the constant $\chi > 0$.

Theorem 10 Let $\beta > 1/3$. The asymptotically stable points of (77) respect the constraints $\xi_{ijk} \in \{0, 1\}$ $\forall i, j, k$, and (7)–(9). In addition, all points which respect these constraints are asymptotically stable points of (77).

See [118] for a proof.

The decision variables (x_{ijk}) correspond to the asymptotically stable points (ξ_{ijk}^*) of the dynamical system (77).

9.4.5 Criterion for Global Optimal Solutions

The above-described investigations of the two-index assignment problem can be adapted to the three-index assignment problem.

Theorem 11 For the dynamical system (77), the following relation is true with two feasible solutions $x^{(r)}$ and $x^{(s)}$ which respect the constraints (7)–(9):

$$\sum_{i,j,k} \xi_{ijk} x_{ijk}^{(r)} \leqq \sum_{i,j} \xi_{ijk} x_{ijk}^{(s)} \Rightarrow \sum_{i,j} \dot{\xi}_{ijk} x_{ijk}^{(r)} \geqq \sum_{i,j} \dot{\xi}_{ijk} x_{ijk}^{(s)}, \quad (78)$$

if $\beta > 1/3$ and $\forall i, j, k \quad (x_{ijk}^{(r)} = 1 \vee x_{ijk}^{(s)} = 1) \Rightarrow \xi_{ijk} \neq 0$. This means that the hyperplane

$$\sum_{i,j,k} \xi_{ijk} x_{ijk}^{(r)} = \sum_{i,j,k} \xi_{ijk} x_{ijk}^{(s)} \quad (79)$$

is repelling and separates the two basins of attraction of the asymptotically stable points $\xi^* = x^{(r)}$ and $\xi^* = x^{(s)}$.

The proof is similar to the proof of [Theorem 8](#).

As in the case of the two-index assignment problem, a criterion can be given to check if a solution to the \mathcal{NP} -hard three-index assignment problem obtained by the coupled, piecewise continuous selection equations is optimal or not, that is, the piecewise linear structure of (77) allows the following theorem.

Theorem 12 The dynamical system (77) solves the corresponding maximization problem of the linear assignment problem (6) optimally if the condition

$$\xi_{ijk}(t') = 0 \Rightarrow \forall t > t' \quad \xi_{ijk}(t) = 0 \quad (80)$$

holds for all t' and for all i, j , and k .

Proof The proof is based on the examination of the basins of attraction described in [Theorem 11](#) and the position of the asymptotically stable points of the coupled, piecewise continuous selection equations (77) described in [Theorem 10](#). \square

By this theorem, one obtains a criterion for the \mathcal{NP} -hard three-index assignment problem (6) if the solution obtained by using the coupled, piecewise continuous selection equations is globally optimal.

10 Numerical Results

Numerical results are very important, in that they enable us to compare different methods for combinatorial optimization. Clearly, these results depend on the data sets, the kind of the combinatorial optimization problem, and the parameters of the method. There is no way to get rid of this drawback. Furthermore, the presentation of numerical results causes many problems with regard to the scientific demands. To make the simulations reproducible, all data sets used should be published as well. There are many cases in the history of science where the disregard of this rule has caused disputes. As an example, see the attempts in [137] to reproduce the results published in [73]. Because of the large amount of data, it is not possible to present them all in printed form. Electronic publishing of the data sets is one way of getting around this problem, but often, the further support of the provided material on the Internet is not guaranteed for a long time. Therefore, a pseudorandom number generator with integer arithmetic is used here in order to get reproducible data sets for the simulations.

10.1 Generation of Data Sets

In many algorithms running on a computer, random numbers are needed. However, the deterministic working principle of computers allow the production of only pseudorandom numbers. Algorithms producing these pseudo-random numbers can be found, for example, in [107]. For probabilities in the interval $[0, 1]$, real numbers or floating point numbers in computers have to be used. Because of rounding errors, these algorithms are not reproducible. In contrast, reproducibility can be achieved by using integer arithmetic. This is used in the following to obtain pseudorandom cost matrices (c_{ij}) and pseudorandom cost tensors (c_{ijk}).

The discrete time dynamic

$$x_{n+1} = (a \cdot x_n) \bmod m \quad (81)$$

with $x_n, a, m \in \mathbb{N}$ is applied. To produce the data sets, the initial value $x_0 = 111,111$ and the parameters $a = 1,233$ and $m = 260,669$ are used. The parameters a and m are chosen so that their product $a \cdot m$ is smaller than the largest 4-byte signed integer (`long int` in the programming language C), that is, $a \cdot m < 2^{31} - 1$. By using these parameters, a period of length 129,822 is obtained. This means that after 129,822 pseudorandom numbers, the sequence is repeating. The first pseudorandom number used for the data set is x_1 . Starting with x_0 , the time discrete dynamical system (81) produces the following sequence: 111,111, 148,638, 20,347, 63,627, 251,391, 29,662, 79,586, 117,994, 33,300, 133,867, 54,534, 248,489, 100,862, 23,733, 67,861, ...

The elements of the cost matrix and cost tensor are chosen in the set $\{0, 1, \dots, 99\}$ of integer values. To obtain this set of integer values, the function

$$\tilde{x}_n = \left\lfloor \frac{x_n}{m} \cdot 100 \right\rfloor \quad (82)$$

is used, where the Gauß bracket $\lfloor \cdot \rfloor$ cuts off the decimals. This derives the sequence 57, 7, 24, 96, 11, 30, 45, 12, 51, 20, 95, 38, 9, 26, 99, 89, 29, ... of pseudorandom numbers.

For the purpose of using this pseudorandom sequence for the elements of the cost matrices c_{ij} and cost tensors c_{ijk} , the uniform distribution of these elements in every length scale of the data set is favored. To illustrate that the distributions produced by the pseudorandom numbers are uniform, the relative frequency $r(\tilde{x}_n)$ is plotted in Fig. 8 for the first 100, 1,000, 10,000, and 100,000 pseudorandom numbers. In these plots, the relative frequency $r(\tilde{x}_n)$ is defined by the product of the number of pseudorandom numbers falling at intervals of a given length which contain \tilde{x} and the number of elements in the set $\{0, \dots, 99\}$, that is, 100, divided by the product of the total number of pseudorandom numbers and the length of the intervals. The plots show a good uniform distribution for every length scale of the data set.

The elements of the cost matrix $c_{ij}^{(l)}$ with $i, j \in N = \{1, \dots, n\}$ of data set l are defined by

$$c_{ij}^{(l)} := \tilde{x}_{((l-1) \cdot n^2 + (i-1) \cdot n + j)} \quad (83)$$

using the produced sequence of pseudorandom numbers. Similarly, the elements of the cost tensor $c_{ijk}^{(l)}$ with $i, j, k \in N = \{1, \dots, n\}$ of data set l are defined by

$$c_{ijk}^{(l)} := \tilde{x}_{((l-1) \cdot n^3 + (i-1) \cdot n^2 + (j-1) \cdot n + k)}. \quad (84)$$

The entries of the first cost matrices or cost tensors of each data set for a given problem size n start with the element \tilde{x}_1 of the pseudorandom sequence. The subsequent cost matrices or cost tensors use consecutively the elements of the pseudorandom number sequence.

As an example, the first 10×10 cost matrix

$$\left(c_{ij}^{(1)} \right) = \begin{pmatrix} 57 & 7 & 24 & 96 & 11 & 30 & 45 & 12 & 51 & 20 \\ 95 & 38 & 9 & 26 & 99 & 89 & 29 & 68 & 46 & 67 \\ 23 & 54 & 13 & 26 & 40 & 30 & 50 & 79 & 45 & 97 \\ 53 & 42 & 28 & 5 & 89 & 98 & 1 & 61 & 72 & 97 \\ 92 & 59 & 64 & 43 & 78 & 14 & 87 & 82 & 1 & 24 \\ 67 & 90 & 59 & 16 & 75 & 71 & 73 & 86 & 90 & 81 \\ 32 & 89 & 63 & 86 & 73 & 11 & 71 & 81 & 40 & 0 \\ 67 & 66 & 79 & 5 & 67 & 72 & 35 & 64 & 36 & 16 \\ 74 & 56 & 74 & 37 & 85 & 85 & 65 & 5 & 25 & 44 \\ 5 & 40 & 81 & 95 & 51 & 91 & 52 & 37 & 87 & 9 \end{pmatrix} \quad (85)$$

is given.

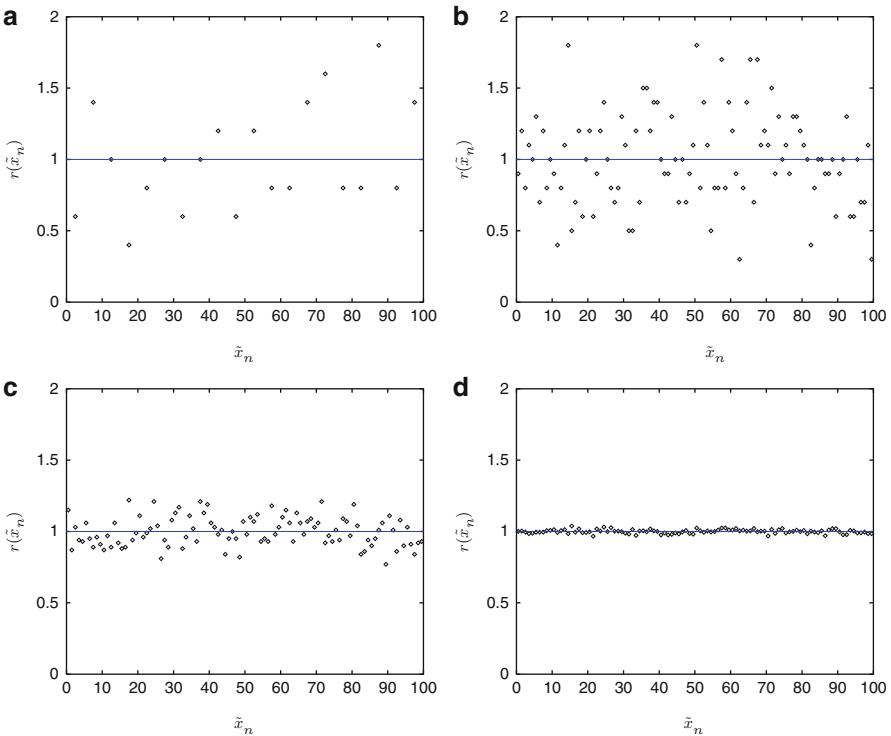


Fig. 8 Distribution of the pseudorandom numbers produced by the described algorithm in the interval $[0, 99]$. The relative frequency $r(\tilde{x}_n)$ of the first k pseudorandom numbers, where the pseudorandom numbers falling at intervals of length l are counted, is plotted over \tilde{x}_n and compared with the horizontal line $r = 1$. (a) With $k = 100$ and $l = 5$. (b) With $k = 1,000$ and $l = 1$. (c) With $k = 10,000$ and $l = 1$. (d) With $k = 100,000$ and $l = 1$

10.2 Generation of Pseudo Random Initial Values

For those methods which have to be initialized with random initial values, the same procedure as for the generation of the data sets is used to generate pseudorandom initial values. The reason for this effort is again the reproducibility of the simulations. This time, the discrete dynamics (81) is used with the initial value $x_0 = 72,664$ as well as the parameters $a = 1,233$ and $m = 260,669$. By using these parameters, a period of length 129,822 is obtained. Again, the first pseudorandom number used for the data set is x_1 . Starting with x_0 , the time discrete dynamical system (81) produces the following sequence: 72,664, 185,245, 61,041, 190,881, 232,835, 88,986, 238,758, 93,313, 99,900, 140,932, 163,602, 224,129, 41,917, 71,199, 203,583, 254,261, ...

The pseudorandom initial values have to lie in the interval $[0, 1]$. The accuracy of these values is chosen as 5 decimals. Therefore, the function

$$\tilde{x}_n = \left\lfloor \frac{x_n}{m} \cdot 100,000 \right\rfloor \frac{1}{100,000} \quad (86)$$

is used. This produces the sequence 0.71065, 0.23417, 0.73227, 0.89322, 0.34138, 0.91594, 0.35798, 0.38324, 0.54066, 0.62762, 0.85982, 0.16081, 0.27314, 0.78100, 0.97542, 0.68928, ... as pseudorandom initial values.

The elements of the initial $n \times n$ matrix and the initial $n \times n \times n$ tensor are defined equivalent to (83) and (84), respectively. As in the case of cost matrices and cost tensors, the entries are chosen consecutively from the sequence of pseudorandom numbers.

10.3 Numerical Solution of Dynamical Systems

The dynamical systems are solved numerically by using the explicit (forward) Euler method

$$\xi(t_{k+1}) = \xi(t_k) + \sigma \cdot \dot{\xi}(t_k)$$

with discrete time steps t_k and step size σ . This simple numerical method is sufficient because the dynamical systems considered are defined as gradient flows and their attractors are asymptotically stable points only and no limit cycles or even more complicated attractors occur. Furthermore, the method allows also for a simple implementation of the piecewise continuous dynamical systems. To avoid the stagnation at stationary but unstable points, some noise is added if the maximal component $\max_{i,j} |\dot{\xi}_{ij}|$ in the case of two-index assignment problems and $\max_{i,j,k} |\dot{\xi}_{ijk}|$ in the case of three-index assignment problems is smaller than 0.00001.

The integration was aborted if the constraints are fulfilled with accuracy of Δ which is given below, that is, elements which fall at interval $[-\Delta, \Delta]$ are taken as 0 and elements which fall at interval $[1 - \Delta, 1 + \Delta]$ are taken as 1. If this condition was not fulfilled after a given number of steps, the computation was stopped as well.

For parameter values, see the detailed description in the next section.

10.4 Parameter Settings for the Numerical Simulations

In order to allow a verification of our numerical results, the parameter settings for the investigated methods are specified below. Nevertheless, even by using reproducible data sets, there may occur some deviation of the obtained results to those computed on other machines due to numerical rounding errors, but these are negligible in the total comparison.

10.4.1 Dual Forest Algorithm

To evaluate the solutions of the methods used, the results are compared with global optimal solutions. In the case of the two-index assignment problem, the global optimal solutions are obtained by a dual forest algorithm [2], for which the C code of P. Kleinschmidt, A. Hefner, and H. Achatz has been used. Another possibility is the well-known Hungarian method [28, 100].

10.4.2 Branch and Bound

In the case of the \mathcal{NP} -hard axial three-index assignment problem, a branch and bound method has been used. Because of bounding conditions, the entire search tree is truncated which reduces the total number of computations compared with exhaustive search. The so far best solution found is used as bound for the subsequent branches. For unfavorable data sets, not enough bounding conditions can be found, which results in long computing times (see, e.g., [55]).

10.4.3 Penalty Methods

For the numerical integration of the equations of motion (33) and (36) which are derived from a penalty method, the explicit (forward) Euler scheme has been used.

For the classical penalty method (33), the step size $\sigma = 0.00001$ and the parameters $p_1 = 10$ and $p_2 = 10$ have been used, which seem to be appropriate for the two-index assignment problems of size 10×10 . In order to get parameter settings which are independent of the range of the data sets, the given costs are normalized on the interval $[0, 1]$. The accuracy of $\Delta = 0.2$ was chosen to stop the integration.

The parameters $p_1 = 1$ and $p_2 = 0.1$ and the step size $\sigma = 0.001$ have been used for the variant with squared variables (36) for the problem sizes 10×10 , 20×20 , and 30×30 .

10.4.4 Simulated Annealing

For the method of simulated annealing, an adapted routine of [107] has been used. The original pseudorandom generator included in this routine has been kept for the simulations. In the two-index case, the assignment, that is, the permutation matrix, is represented by an ordered list of the integer numbers $1, \dots, n$. This ordering is rearranged by reversing the order of a part of the list or replacing a part of the list between two arbitrary elements of the list in the simulated annealing procedure. For the three-index assignment problem, two of these lists have been used.

The cooling function $T(m) = 0.95^m \cdot T(0)$, where m denotes a step in the cooling procedure, has been chosen with the initial temperature $T(0) = 500$. The temperature is decreased if either the maximum number of reconfigurations of $1,000 \cdot n$ or maximum number of successful reconfigurations $100 \cdot n$ is reached. Further details can be found in [107].

10.4.5 Hopfield and Tank Approach

The costs of the data sets are normalized also for the Hopfield and Tank approach on the interval $[0, 1]$ for the same reason as mentioned in Sect. 10.4.3. For the equation

of motion (47) of the approach of Hopfield and Tank adapted to the two-index assignment problem, the parameter settings $A = B = 10$, $C = 2$, $D = 20$, $u_0 = 0.02$, $\tau = 1$, and $\tilde{n} = 15$ are used. These parameters have been found by successive determination of $A = B$ and C to obtain permutation matrices as output. After that, D is increased as long as possible so that a permutation matrix is maintained as output. Furthermore, $u_{00} = u_0 \cdot \operatorname{atanh}\left(\frac{2}{n} - 1\right)$ has been used which gives initial values where the sum over the elements of each row and column is equal to 1. These settings result in better solutions to the two-index assignment problem than the settings given in [73]. For the integration of the dynamical system, the step size $\sigma = 0.001$ has been used.

10.4.6 Coupled Selection Equations

The following description applies for both the coupled selection equations of pattern formation and the coupled, piecewise continuous selection equations.

For the two-index assignment problem, the initial values are calculated by using the transformation (5) with a and b so that the winnings w_{ij} are bounded by zero from below and by one from above. For this, it is possible to use the transformation $w_{ij} = 1 - \frac{c_{ij} - \min_{i,j} c_{ij}}{\max_{i,j} c_{ij}}$ or $w_{ij} = 1 - \frac{c_{ij} - \min_{i,j} c_{ij}}{\max_{i,j} c_{ij} - \min_{i,j} c_{ij}}$. The first is confined on the interval $[\min_{i,j} c_{ij} / \max_{i,j} c_{ij}, 1]$ and the latter on $[0, 1]$. For both, the coupled selection equations of pattern formation and the coupled, piecewise continuous selection equations, the coupling parameter $\beta = 1$ and the step size $\sigma = 0.001$ are chosen.

For the three-index assignment problem, the initial values are determined similarly to these of the two-index case. Again, the coupling parameter $\beta = 1$ has been chosen. For problems of size $10 \times 10 \times 10$, the fixed step size $\sigma = 0.002$ has been used for both methods, the coupled selection equations of pattern formation and the coupled, piecewise continuous selection equations.

Due to gradients of large Euclidean norm in the case of the $20 \times 20 \times 20$ problems using the coupled selection equations of pattern formation, the absolute values of the components of the gradient were confined to 0.01, while the used step size was $\sigma = 0.01$. This was not necessary for the coupled, piecewise continuous selection equations because of the piecewise linear equation of motion instead of the terms of order three in the previous case. Therefore, the fixed step size $\sigma = 0.001$ has been used.

10.4.7 Coupled Selection Equations with Cost Terms: Hybrid Method

In the case of the hybrid method between the coupled selection equations and the penalty method, that is, selection equations with cost terms, the cost values are used for the equation of motion as well as for the initial values. For the equation of motion, they are used in the same manner as for the penalty methods. For the initial values, the same normalization on the interval $[0, 1]$ as for the above-described selection equations has been done.

For the numerical simulation, the evolution equation (36) is used in the form

$$\dot{x}_{ij} = x_{ij} \left(\alpha_{ij} + (2\beta - 1)x_{ij}^2 - \beta \left(\sum_{j'} x_{ij'}^2 + \sum_{i'} x_{i'j}^2 \right) \right) \quad (87)$$

with the cost-dependent individual growth rates

$$\alpha_{ij} = \frac{3}{4} + \frac{1}{4} \left(1 - \frac{c_{ij} - \min_{i,j} c_{ij}}{\max_{i,j} c_{ij}} \right), \quad (88)$$

the parameter $\beta = 0.55$, and the step size $\sigma = 0.001$.

10.5 Comparison of Several Methods

The following tables show and compare the results of a dual forest algorithm; branch and bound; classical penalty method; a variation of the penalty method; simulated annealing; the approach of Hopfield and Tank; coupled selection equations of pattern formation; coupled, piecewise continuous selection equations; and a hybrid method that is a coupled selection equations with an additional cost term.

Several simulation results are presented in the following for the two- and axial three-index assignment problem with different problem sizes and various optimization methods. The data sets used for the numerical analysis are those described in Sect. 10.1. The corresponding pseudorandom initial values described in Sect. 10.2 are used for the methods which need randomly chosen initial values. These are the classical penalty method (33), the penalty method (36) with modified terms and the approach (47) of Hopfield and Tank. The presented results are the total costs $c^{(k)}$ of each used data set k in Tables 1 and 4 and the mean of the total costs \bar{c} of m investigated data sets in Tables 2, 3, and 5. The mean value is defined by

$$\bar{c} = \frac{1}{m} \sum_{k=1}^m c^{(k)}. \quad (89)$$

Furthermore, the standard deviation

$$s = \sqrt{\frac{1}{m} \sum_{k=1}^m (c^{(k)} - \bar{c})^2} = \sqrt{\frac{1}{m} \sum_{k=1}^m (c^{(k)})^2 - \left(\frac{1}{m} \sum_{k=1}^m c^{(k)} \right)^2} \quad (90)$$

is shown to give a measure for the spread of the obtained solutions compared to the spread of the global optimal solution. The ratio of these standard deviations indicates whether the quality of the obtained solutions is the same for all data sets

or whether it is varying strongly. The mean and the standard deviation are rounded with an accuracy up to the first decimal.

It has to be remarked that the computation time of the investigated optimization methods is not compared. There are several reasons for this. First, our main interest is not to find the fastest algorithm if the total amount of time is within a reasonable range. Instead, we focus upon the investigation of the method's optimization principle. Second, the comparison of CPU times of computer codes is problematic. This depends strongly on the art of programming, the compiler and compiler options used, and the hardware.

10.5.1 Two-Index Assignment Problem

[Table 1](#) shows numerical results for data set 1–10 obtained by the dual forest algorithm; the classical penalty methods; a penalty method using modified terms with squared variables; simulated annealing; the approach of Hopfield and Tank; coupled selection equations of pattern formation; coupled, piecewise continuous selection equations; and coupled selection equations of pattern formation with an additional cost term. A global optimal solution, which is obtained by the dual forest algorithm, gives a comparative value to the solutions obtained by the approximation methods. The symbol “–” in [Table 1](#) indicates that no feasible solution was found for this data set.

A summary of the results for data set 1–100 in the case of the problem sizes $n^2 = 10^2$ and $n^2 = 20^2$ and for data set 1–50 in the case $n^2 = 30^2$, respectively, can be found in the [Tables 2](#) and [3](#). The mean of the solutions and the standard deviation are given for every method. Furthermore, the number of feasible solutions obtained by the used methods is indicated.

In [Tables 2](#) and [3](#), the classical penalty method and the approach of Hopfield and Tank show obvious bad results. The quality of the solutions of the coupled selection equations of pattern formation and of the coupled, piecewise continuous selection equations is comparable with the quality of the solutions of simulated annealing. The hybrid method, that is, the coupled selection equations of pattern formation with calculated initial values and with cost terms in the equation of motion, produces remarkable good results. Compared with the coupled selection equations of pattern formation, this is due to the deformation of the basins of attraction caused by the cost term in the equation of motion. It is also much better than the penalty method with the same equation of motion but with randomly chosen initial values because the calculated initial values start the dynamics in a promising region. It should be remarked that the results of the coupled, piecewise continuous selection equations improve considerably if the initial state is shifted with the saddle point of the dynamics.

A comparison of the hybrid coupled selection equation approach with simulated annealing is done in [\[63\]](#). The results are summarized in [Fig. 9](#) where the scaling of the relative deviation of the optimization results compared to the optimal solutions is plotted against the problem size. The hybrid method scales with respect to the problem size remarkably better than simulated annealing.

Table 1 Results of the simulations for 10 data sets of the two-index assignment problems of size $n^2 = 10^2$ where the objective function (1) has to be minimized. The global optimal solution obtained by the dual forest algorithm serves as comparative value to the solutions obtained by the approximation methods

Method	Dual forest algorithm	Penalty methods	Squared variables	Simulated annealing	Neural networks	Selection equations	Piecewise continuous	Hybrid method
Variants of method		Classical		Hopfield and Tank	Pattern formation			Patt. form. and cost term
Costs for data set 1	111	337	158	111	353	129	111	111
Costs for data set 2	95	195	179	95	263	95	95	95
Costs for data set 3	138	414	182	157	355	139	214	138
Costs for data set 4	154	—	216	167	278	155	171	154
Costs for data set 5	137	218	165	165	—	137	166	138
Costs for data set 6	154	281	214	158	272	166	162	154
Costs for data set 7	117	383	166	117	—	117	117	117
Costs for data set 8	96	396	151	100	383	96	96	96
Costs for data set 9	90	309	129	90	329	111	143	90
Costs for data set 10	104	326	214	104	340	153	104	104
Mean of costs	119.6	317.7	177.4	126.4	321.6	129.8	137.9	119.7
Standard deviation	23.2	71.9	28.1	29.8	41.9	23.4	37.7	23.3

Table 2 Summary of the results of two-index assignment problems for $n^2 = 10^2$

Method	Dual forest algorithm	Penalty methods	Simulated annealing	Neural networks	Selection equations	Hybrid method
Variants of method		Squared variables		Hopfield and Tank	Pattern formation	Piecewise continuous
Problem size $n^2 = 10^2$, number of data sets = 100						
Feasible solutions	100	88	100	100	85	100
Mean of costs	129.5	300.5	189.2	138.0	307.8	136.7
Standard deviation	31.8	84.7	35.9	32.9	42.7	34.7
						42.6
						31.9

Table 3 Summary of the results of two-index assignment problems for $n^2 = 20^2$ and $n^2 = 30^2$

Method	Dual forest algorithm	Penalty methods	Simulated annealing	Neural networks	Selection equations	Hybrid method
Variants of method		Squared variables		Hopfield and Tank	Pattern formation	Piecewise continuous
Problem size $n^2 = 20^2$, number of data sets = 100						
Feasible solutions	100	100	100	92	100	100
Mean of costs	141.0	240.3	210.9	574.6	175.8	185.8
Standard deviation	22.4	30.2	33.4	43.9	33.1	49.6
Problem size $n^2 = 30^2$, number of data sets = 50						
Feasible solutions	50	50	50	44	50	50
Mean of costs	138.7	259.6	296.5	831.8	181.5	218.0
Standard deviation	21.3	25.2	40.3	58.2	35.6	56.5
						20.3

10.5.2 Three-Index Assignment Problem

For the \mathcal{NP} -hard axial three-index assignment problem, only the most promising approaches of Sect. 10.5.1 have been used. The solutions of the penalty methods and the approach of Hopfield and Tank are either not feasible or result in very high total costs. The hybrid method is not included here in the comparison, but detailed investigations can be found in [124].

Table 4 shows the results for the first 10 data sets for branch and bound, simulated annealing, coupled selection equations of pattern formation, and coupled, piecewise continuous selection equations. A summary of the results of data set 1–100 in the case of problem size $n^3 = 10^3$ and data set 1–10 in the case of problem size $n^3 = 20^3$ is presented in Table 5. The global optimal solution of the \mathcal{NP} -hard

Fig. 9 Scaling behavior of the relative deviation r of the obtained solutions compared to the optimal solutions by using simulated annealing (rhombs) and the hybrid method of coupled selection equations (triangles) over the problem size n . The hybrid method shows a remarkable better scaling than simulated annealing. Each point represents the mean of results of 10 data sets (Reprinted from [63])

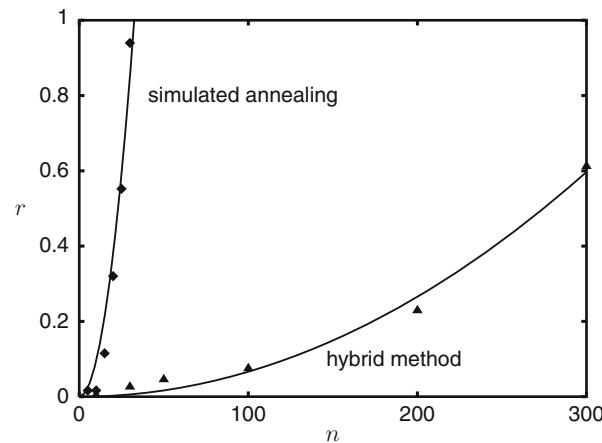


Table 4 Results of the simulations for 10 data sets of three-index assignment problems of size $n^3 = 10^3$. The global optimal solution obtained by a branch and bound method serves as comparative value to the solutions obtained by the approximation methods

Method	Branch and bound	Simulated annealing	Selection equations	
			Pattern formation	Piecewise continuous
Variants of method				
Costs for data set 1	17	25	103	64
Costs for data set 2	18	88	21	58
Costs for data set 3	18	61	23	142
Costs for data set 4	14	52	109	80
Costs for data set 5	23	102	141	49
Costs for data set 6	21	67	52	195
Costs for data set 7	33	78	90	72
Costs for data set 8	17	66	97	168
Costs for data set 9	26	97	123	81
Costs for data set 10	24	52	74	79
Mean of costs	21.1	68.8	83.3	98.8
Standard deviation	5.3	22.1	38.4	48.0

three-index assignment problem was obtained by branch and bound which is still practicable for the considered problem sizes. This gives a comparative value to the costs of the other solutions. The results obtained by simulated annealing and the coupled, piecewise continuous selection equations presented in Table 5 show a bad scaling behavior with respect to the problem size. The reason for the bad results of the coupled, piecewise continuous selection equations is due to the violation of condition (80) which is unfortunately the case in many numerical simulations. Further investigations under which conditions there exist elements which do not fulfill (80) and appropriate transformations to avoid these elements could improve the results of the coupled, piecewise continuous selection equations [118]. In opposite, the coupled selection equations of pattern formation show acceptable results.

Table 5 Summary of the results of three-index assignment problems for $n^3 = 10^3$ and $n^3 = 20^3$

Method	Branch and bound	Simulated annealing	Selection equations	
			Pattern formation	Piecewise continuous
<u>Variants of method</u>				
Problem size $n^3 = 10^3$, number of data sets = 100				
Feasible solutions	100	100	100	100
Mean of costs	21.0	66.6	69.1	95.8
Standard deviation	5.1	15.8	30.8	45.6
Problem size $n^3 = 20^3$, number of data sets = 10				
Feasible solutions	10	10	10	10
Mean of costs	5.4	164.6	61.1	162.1
Standard deviation	1.7	25.7	28.7	72.4

11 Assignments in Distributed Robotic Systems and Flexible Manufacturing

Traditional manufacturing processes are typically based on specialized machines and factories which are built and optimized to produce sometimes only one specific kind of product. In contrast to this, the demand for a production with high variety, small batch size, and short delivery periods requires flexible and modular manufacturing processes [87]. This modularity requires to optimize the combination of many subsystems and leads naturally to assignment problems. Together with flexibility due to modularity increases also the complexity and by that the error rate which exceeds the capabilities of traditional control methods [41, 112] and demands alternative approaches. A promising strategy is to adapt robust and flexible concepts from nature and to use self-organization and pattern formation principles [58, 98] in engineering [27, 34, 86]. By using selection processes as one of the main principles of pattern formation, namely, using different variants of coupled selection equations for assignment problems as described in Sect. 9, it is possible to control distributed robotic systems to perform certain tasks at manufacturing targets in a very robust way. The concept and first basic ideas are described in [123] and are further developed in [94]. An extension to situations with underlying three-index assignments are presented in [121] and [17]. Dynamic environments and time-dependent assignments are considered in [119] and [120] where sequential manufacturing tasks in logical order are fully considered with guaranteed feasibility of the assignment solutions. Practical and experimental tests with real robots are shown in [76] and [125].

11.1 Modeling Flexible Manufacturing Systems

In the area of flexible manufacturing systems and distributed robotic systems, the key of the problem is often to assign robots to manufacturing targets. The idea

to use cooperative autonomous robots (see, e.g., [12, 45, 46]) is discussed for a while with different viewing angles but often lacks in this area a formalization and general mathematical framework and description. On the other hand, the formalization by job shop problems results usually in too complicated setups and is therefore typically not used. One option for a relatively simple description with a mathematical model is to use assignment problems as basic building blocks. In particular, time-dependent robot-target assignments appear in the production processes of flexible manufacturing systems where sequences of tasks have to be machined. To simplify the situation compared to general job shop problems, the problem will be divided in several subproblems which are considered in not specified time intervals, that is, a treatment is taken which works local in time. This can be used to model a large class of application problems.

In extension to the two-index assignment problem (1) with constraints (2) and (3), it is also possible to consider situations with unequal number of targets and robots so that after the assignment there are spare targets or robots. Furthermore, at a given time, targets can be set active or inactive to capture sequential task processing in a logical order. Thus, time-dependent robot-target assignment problems are obtained, that is, in a given time interval, each active target has to be served by one robot.

11.2 Ad Hoc Assignments with Coupled Selection Equations

The term ad hoc assignments refers to the fact that there is no preexisting structure of robots and manufacturing targets available at a given time. This structure evolves in time depending on the current situation like sudden changes in demands, breakdowns of single robots, or obstacles in the way of maneuvering robots. Self-organization approaches like the one of coupled selection equations are in particular suited for this class of problems where flexibility and robustness are of importance.

To assign the robots to the manufacturing targets while considering time-dependent sequential manufacturing tasks in logical order with guaranteed feasibility of the assignment solutions, the coupled selection equations (59) can be extended with time-dependent parameters $\alpha_{ij}(t)$ and $\lambda_{ij}(t)$. The dynamical system results then to

$$\frac{d}{dt} \xi_{ij} = \kappa \xi_{ij} \left(\alpha_{ij} (\lambda_{ij} - \xi_{ij}^2) - \beta \sum_{i' \neq i} \xi_{i'j}^2 - \beta \sum_{j' \neq j} \xi_{ij'}^2 \right) \quad (91)$$

which defines the time evolution of preferences $\xi_{ij}(t)$ for robot i to manufacturing target j . Details can be found in [120]. Please note the similarity to Eq. (68) which was derived as penalty approach for assignment problems, where $(1 - \alpha c_{ij})$ corresponds to α_{ij} in the present equation and λ_{ij} was identical to one for all i, j . The initial conditions are chosen as the specific winnings w_{ij} which was described already in Sect. 9, that is, as $\xi_{ij}(0) = w_{ij}$ for all i, j . The w_{ij} can all be chosen nonnegative without loss of generality so that nonnegative initial values are obtained.

The extended coupled selection equations (91) serve as robust heuristic optimization strategy for the introduced time-dependent assignment problems. Similarly as described for Eq. (59), the dynamical system approach optimizes the assignment problem by choosing the largest single manufacturing profits for each robot-target assignment from the set of available robots and active targets so that the total profits are maximized. The solution $\xi_{ij}(t)$ tends to the unknown decision variables x_{ij} . For equal numbers of robots and targets, the system converges to a permutation matrix; for unequal numbers of targets and robots, the column or row elements of spare targets or robots tend to zero. This requires the assumption that there is a little bit noise present so that there is no stagnation at stationary but unstable points.

In addition to (59), the extended coupled selection equations (91) use a number of parameters: κ adjusts the time scales of the selection equation and its decision-making process with those of the velocities of the robots. The weight parameter $\alpha_{ij}(t) > 0$ captures updates of sudden problem changes by deviations from the default value $\alpha_{ij}(t) = 1$ for all i, j, t . This favors or penalizes the individual selection for the assignment preferences ξ_{ij} . Finally, changing the sign of the activation parameter $\lambda_{ij}(t) \in \{-10, 1\}$ switches over active and inactive targets: Positive $\lambda_{ij}(t)$ stabilizes the selection for active targets, and negative $\lambda_{ij}(t)$ destabilizes it for inactive targets. This allows for the modeling of sequential manufacturing steps and changes in demands of time-dependent assignment problems.

Similarly to the coupled selection equations (59), the feasibility of solutions can be guaranteed also for Eq. (91) if the coupling strength is large enough, that is, if $\beta > \frac{1}{2} \max_{i,j,t} \alpha_{ij}(t)$. This can be proven under the assumption that $\alpha_{ij}(t)$ and $\lambda_{ij}(t)$ tend to constants in the course of time until a target is served. These assumptions are necessary because otherwise, for example, fast oscillating demands would lead to situations where the tasks cannot be finished. In contrast to [43, 63], and [44], the inhomogeneity of the growth rates α_{ij} takes effect in (91) not only on the linear but also the cubic term. Therefore, the ratio of the coefficients of these two terms is one, and for large-enough couplings, the stable points are located at $\xi_{ij} = 0$ or $\xi_{ij} = 1$.

A distributed instead of a centralized control is possible by exchanging information of the row and column elements of (ξ_{ij}) , that is, $\sum_i \xi_{ij}^2$ and $\sum_j \xi_{ij}^2$ from time to time between the cooperative robots. By this, additional robustness against hardware failures is achieved [125].

The flexibility and fault tolerance of the approach are due to the continuous decision process which works locally in time of the manufacturing sequence. This means that sudden changes in the demands can still be considered in the selection process without the need of a restart and more profitable upcoming active targets can replace already assigned ones.

11.3 Navigation of the Robots

The navigation of autonomous robots in space is implemented in the following by combining a specifically constructed behavioral force model [8] with the extended coupled selection equations (91). The equation of motion

$$\frac{d}{dt} \mathbf{r}_i = v_i^0 \mathbf{e}_i^0(\mathbf{r}_i, \xi) + \mathbf{f}_i(\mathbf{r}_i) \quad (92)$$

describes the positions $\mathbf{r}_i \in \mathbb{R}^3$ of the robots i in space with velocity constant v_i^0 and the destination vector $\mathbf{e}_i^0(\mathbf{r}_i, \xi)$ [120]. The direction of the destination vector \mathbf{e}_i^0 tends towards the selected target as the preference matrix $\xi = (\xi_{ij}(t))$ converges to a decision matrix with elements equal to zero or one.

The definition of the destination vector

$$\mathbf{e}_i^0(\mathbf{r}_i, \xi) = \mathbf{N}_{\gamma\delta} \left(\sum_j \xi_{ij}(t) \mathbf{N}_{\gamma'\delta'} (\mathbf{g}_j - \mathbf{r}_i(t)) \right) \quad (93)$$

is based on the normed linear combination of the normed difference vectors of the position $\mathbf{r}_i(t)$ to all target positions \mathbf{g}_j weighted with $\xi_{ij}(t)$. Due to this property, its direction tends in the course of time towards the selected target. The operator $\mathbf{N}_{\gamma\delta}(\mathbf{y}) = \frac{1}{\|\mathbf{y}\|+1/(\gamma\|\mathbf{y}\|+\delta)} \cdot \mathbf{y}$ with $\gamma, \delta > 0$, where $\|\cdot\|$ is the Euclidean norm, mainly normalizes the vector \mathbf{y} and avoids singularities for the numerical integration at $\mathbf{y} = \mathbf{0}$.

The collision avoidance term

$$\mathbf{f}_i(\mathbf{r}_i) = \sum_{i' \neq i} \mathbf{f}_{ii'}^r(\mathbf{d}_{ii'}^r) + \sum_k \mathbf{f}_{ik}^o(\mathbf{d}_{ik}^o) \quad (94)$$

consists of a part for collision avoidance between robot i and other robots $i' \neq i$ and one between robot i and obstacles k . The terms $\mathbf{f}_{ii'}^r(\mathbf{d}_{ii'}^r) \in \mathbb{R}^3$ depend on distance vectors $\mathbf{d}_{ii'}^r$ between the surfaces of the objects obtained by the robots' sensors. They tend to infinity for small distances and have finite range, that is, they are identical zero for distances over a given threshold $\sigma^{r,o}$. This avoids that for large distances, small residuals of the collision avoidance terms could add up to unwanted spurious states, that is, unfeasible stable points of the dynamical system [94] at which the robots would be trapped. This is obtained by the choice

$$\mathbf{f}_{ii'}^r(\mathbf{d}) = \begin{cases} s^{r,o} (\tan \psi(\|\mathbf{d}\|) - \psi(\|\mathbf{d}\|)) \frac{\mathbf{d}}{\|\mathbf{d}\|}, & \|\mathbf{d}\| \leq \sigma^{r,o} \\ 0, & \|\mathbf{d}\| > \sigma^{r,o}, \end{cases} \quad (95)$$

where $s^{r,o} \in \mathbb{R}$ is the strength of the collision avoidance term, $\psi(\|\mathbf{d}\|) = \frac{\pi}{2} \left(\frac{\|\mathbf{d}\|}{\sigma^{r,o}} - 1 \right)$, and $\|\cdot\|$ is the Euclidean norm.

Alternatively to the behavioral force approach (92), also other path-planning methods (see, e.g., [79, 88, 125]) can be combined with the coupled selection equations to take advantage of the robust and fault-tolerant assignment process.

11.4 Application Example: Assembling of a Space Station

The example of assembling a space station with autonomous robots demonstrates the capabilities of the proposed method (cf. Fig. 10) and intends to illuminate possible future concepts in various fields with complex requirements. The robot's task is to collect different components from a space shuttle and transport them to certain positions at the space station where bars and solar panels have to be assembled. Each of the 8 identical robots is equipped with distance sensors in order to avoid collisions and can detect the relative direction towards the manufacturing targets to maneuver using Eqs. (92)–(95). As initial values $\xi_{ij}(0)$, large profits w_{ij} were used for small Euclidean distances $d_{ij}(t) = \|\mathbf{g}_j - \mathbf{r}_i(t)\|$ between robots i and manufacturing targets j so that the total travel distance is used as objective function which has to be minimized and the result can easily be evaluated by visual inspection. The same choice was done for the weight parameter $\alpha_{ij}(t)$ which respects by that time delays from maneuvering around obstacles in the selection process. Therefore, the linear transformation $W_{ij}^{a,b}(t) = a - b \frac{d_{ij}(t) - \min_{i',j'} d_{i'j'}(t)}{\max_{i',j'} d_{i'j'}(t) - \min_{i',j'} d_{i'j'}(t)}$ with $a, b > 0$ and $a \geq b$ such that $W_{ij}^{a,b}(t) \in [a - b, a]$ is used for the initial values $\xi_{ij}(0) = w_{ij} = W_{ij}^{a,b}(0)$ and weight parameters $\alpha_{ij}(t) = W_{ij}^{a',b'}(t)$.

The activation parameter $\lambda_{ij}(t)$ is chosen +1 for active and -10 for inactive manufacturing targets at each time t so that the decision variables are finally either zero or one and that there is a 10 times faster destabilization than stabilization of selections. To capture the manufacturing in the logical order of the sequential processing, each of the 44 manufacturing targets, either at the space station or at the material depot of the space shuttle, becomes active for the period where the requirements and prerequisites for the corresponding task are given and until the task is done. To exclude certain configurations, for example, to model breakdowns of robots, the corresponding activation parameters λ_{ij} can be switched to negative values. This results in the growth of preferences of an alternatively available robot and a subsequent replacement by the usual selection process of the equations.

Figure 10 shows the simulation results for the assembling of a space station. Active targets are green, and inactive targets are red boxes (see the online version for colors). Short histories of the robots' trajectories are plotted to visualize their movements in space. Between subfigures (a) and (b), the first assignment has been developed in the preference matrix. In Fig. 10d–f, the completion of the mission is shown including an automatic and self-organized replacement of a robot after a triggered breakdown which demonstrates the robustness. Subfigure (e) shows how collisions with obstacles are avoided, that is, two robots maneuver around the space station. The last figure (f) shows the completely assembled space station. The numerical solutions of (91) and (92) were calculated with the parameters $\kappa = 15$, $\beta = 4$, $a = 0.9$, $b = 0.8$, $a' = 1.5$, $b' = 1$ and $v_i^0 = 0.3$, $\sigma^r = 0.1$, $\sigma^o = 0.02$, $s^{r,o} = 1$, $\gamma = 10$, $\delta = 1$, $\gamma' = 10^6$, and $\delta' = 10^6$.

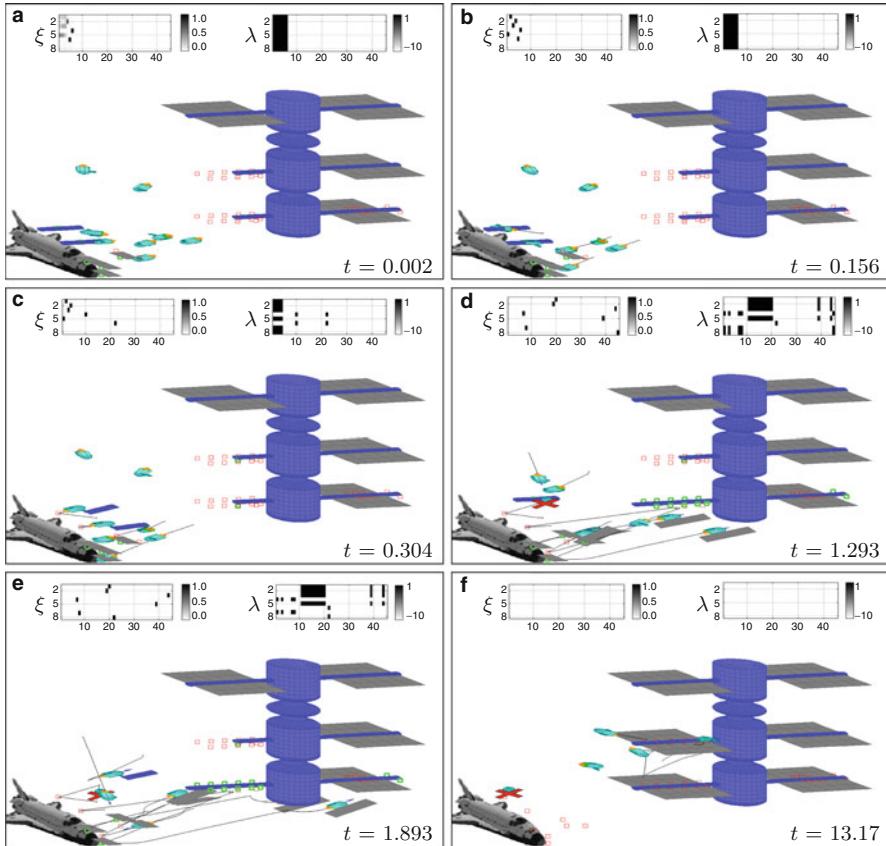


Fig. 10 Application example of the coupled selection equations (91) for a complex time-dependent manufacturing process with ad hoc assignments of robots to manufacturing targets. The presented snapshots show a simulation of assembling solar panels of a space station. The robustness of the approach is demonstrated by a triggered breakdown of a robot in (d) and the following transfer of the task to another robot while the whole process continues. The preference matrix ξ and the activation matrix λ are given, respectively, in the upper left and right corner of each snapshot (Reprinted from [120])

The simulation results show the successful application of the coupled selection equation approach to a complex manufacturing problem where flexible, modular, and decentralized structures are required. The extended selection equations (91) can be tailored to a wide range of applications by adapting the activation parameter $\lambda_{ij}(t)$ to the respective engineering problem. The approach is therefore very promising in a large number of problems in flexible manufacturing as well as distributed robotics, microrobotics, and nanorobotics, in particular where it is important that the total process continues even if the risk of partial failures is high.

12 Conclusion

This chapter about dynamical system approaches to combinatorial optimization problems focuses on the two- and on the \mathcal{NP} -hard three-index assignment problem. In addition to many other application examples, they can be used as simple models for manufacturing planning and distributed robotic systems. Therefore, there is an important practical relevance of assignment problems in general. The dynamical system approaches provide approximate solutions to the combinatorial optimization problems. These approximate solutions to the optimization problems emerge in the limit of large times as asymptotically stable states of the dynamical systems.

The classical penalty method and the approach of Hopfield and Tank cause problems in fulfilling the constraints, that is, even for the two-index assignment problem of size 10×10 , feasible solutions are not obtained for all investigated data sets. Furthermore, these methods are very parameter sensitive. Therefore, they cannot be used in practical applications. An adaptation of the parameters to new optimization problems is very time consuming, and there is neither a straightforward way nor a guarantee to find parameter settings which work well. Because these methods use random initial values, one can get good and bad solutions for the same specific combinatorial optimization problem. In contrast to real statistical approaches like simulated annealing, this does not at all seem to be a necessary ingredient of the approach but is used only because of the lack of a formula to calculate initial values. Calculated initial values allow the dynamics to start in a promising, problem corresponding basin of attraction instead of a randomly chosen one.

The results and comparisons presented in Sect. 10.5 show clearly that if there exists a polynomial time algorithm based on discrete operations, this is the best choice. The dynamical system approaches cannot compete against them except for special applications where error resistivity is relevant during the solution process. For cases where there are no polynomial time algorithms, the dynamical system approaches can be a good alternative to other approaches such as statistical ones. The comparison in Sect. 10.5 showed that the coupled selection equation approach outperforms the other tested dynamical system approaches. Remarkable good results are obtained by the hybrid method, that is, the coupled selection equations of pattern formation with calculated initial conditions and an additional cost term in the equation of motion. In the case of the \mathcal{NP} -hard three-index assignment problem, the results of the coupled selection equations of pattern formation are better than the results obtained by simulated annealing. For the coupled, piecewise continuous selection equations, a criterion for the global optimality of the obtained solutions is given.

An implementation in hardware, that is, an analog computer concept, could provide a big advantage concerning the computing time. One of the most promising hardware implementations is an optical computer. See [1, 108] for a recurrent laser system which is able to reconstruct a pattern out of a given set of patterns which are stored in a hologram. The nonlinearity there is based on a photorefractive crystal

which causes a selection process and favors the most similar stored pattern. In [97], an alternative optical system showing pattern formation is introduced using liquid-crystal light valves as nonlinear elements. These might be promising concepts for the construction of an optical computer for coupled selection equations.

Furthermore, dynamical system approaches are suitable for the adaptation to self-organizing structures like cellular and distributed robotic systems and flexible manufacturing systems which are error resistant and flexible with respect to changing demands. The robustness and capabilities with respect to flexibility to sudden problem changes are demonstrated for the coupled selection equation approach applied to control a distributed robotic system in a manufacturing process. The results are promising and include even recovery from breakdowns of parts of the system and a continuous functioning of the whole system.

This leads to the conclusion that the approaches of the coupled selection equations, in particular the coupled, piecewise continuous selection equations and the hybrid method, are promising approaches and worth for future investigations.

Cross-References

- [Advances in Scheduling Problems](#)
- [Combinatorial Optimization in Transportation and Logistics Networks](#)
- [Connections Between Continuous and Discrete Extremum Problems, Generalized Systems, and Variational Inequalities](#)
- [Fault-Tolerant Facility Allocation](#)
- [Greedy Approximation Algorithms](#)
- [Neural Network Models in Combinatorial Optimization](#)
- [Online and Semi-online Scheduling](#)
- [Quadratic Assignment Problems](#)

Recommended Reading

1. Y. Abu-Mostafa, D. Psaltis, Optical neural computers. *Sci. Am.* **256**(3), 66–73 (1987)
2. H. Achatz, P. Kleinschmidt, K. Paparrizos, A dual forest algorithm for the assignment problem, in *The Victor Klee Festschrift*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 4 (American Mathematical Society, Providence, 1991), pp. 1–10
3. S. Amari, Mathematical foundations of neurocomputing, in *Proceedings of the IEEE*, vol. 78 (IEEE, New York, 1990), pp. 1443–1463
4. J. Anderson, E. Rosenfeld, *Neurocomputing, Foundations of Research* (MIT, Cambridge, 1988)
5. J. Anderson, A. Pellionisz, E. Rosenfeld, *Neurocomputing 2, Directions for Research* (MIT, Cambridge, 1990)
6. B. Angèniol, G. De la Croix Vaubois, J.-Y. Le Texier, Self-organizing feature maps and the travelling salesman problem. *Neural Netw.* **1**, 289–293 (1988)

7. D. Anosov, I. Bronshtein, S. Aranson, V. Grines, Smooth dynamical systems, in *Dynamical Systems I*. Encyclopaedia of Mathematical Sciences, vol. 1 (Springer, Heidelberg/Berlin/New York, 1988), pp. 149–233
8. R. Arkin, *Behavior-Based Robotics* (MIT, Cambridge/London, 1998)
9. V.I. Arnol'd, *Gewöhnliche Differentialgleichungen* (Deutscher Verlag der Wissenschaften, Berlin, 1979/1991)
10. V.I. Arnol'd, *Geometrische Methoden in der Theorie der gewöhnlichen Differentialgleichungen* (Deutscher Verlag der Wissenschaften, Berlin, 1987)
11. V.I. Arnol'd, Yu. S. Il'yashenko, Ordinary differential equations, in *Dynamical Systems I*, ed. by D. Anosov, V. Arnol'd. Encyclopaedia of Mathematical Sciences, vol. 1 (Springer, Berlin/Heidelberg/New York, 1988), pp. 1–148
12. H. Asama, T. Arai, T. Fukuda, T. Hasegawa (eds.), *Distributed Autonomous Robotic System (DARS 5)* (Springer, Heidelberg/Berlin/New York, 2002)
13. M. Avriel, *Nonlinear Programming – Analysis and Methods* (Prentice-Hall, Englewood Cliffs, 1976)
14. B. Baird, Bifurcation and category learning in network models of oscillating cortex. *Physica D* **42**, 365–384 (1990)
15. W. Banzhaf, A new dynamical approach to the travelling salesman problem. *Phys. Lett. A* **136**(1, 2), 45–51 (1989)
16. W. Banzhaf, The molecular traveling salesman. *Biol. Cybern.* **64**, 7–14 (1990)
17. M. Becht, T. Buchheim, P. Burger, G. Hetzel, G. Kindermann, R. Lafrenz, N. Oswald, M. Schanz, M. Schulé, P. Molnar, J. Starke, P. Levi, Three-index assignment of robots to targets: an experimental verification, in *Proceedings of the 6th International Conference on Intelligent Autonomous Systems (IAS-6)*, ed. by E. Pagello et al. (IOS, Amsterdam/Washington, DC, 2000), pp. 156–163
18. M. Bestehorn, H. Haken, Associative memory of a dynamical system: the example of the convection instability. *Z. Phys. B* **82**, 305–308 (1991)
19. K. Binder, A. Young, Spin glasses: experimental facts, theoretical concepts, and open questions. *Rev. Mod. Phys.* **58**(4), 801–963 (1986)
20. A.M. Bloch, A. Iserles, On the optimality of double-bracket flows. *Int. J. Math. Math. Sci.* **2004**(61–64), 3301–3319 (2004)
21. I. Bomze, Evolution towards the maximum clique. *J. Glob. Optim.* **10**, 143–164 (1997)
22. I. Bomze, M. Budinich, P. Pardalos, M. Pelillo, The maximum clique problem, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos (Kluwer, Dordrecht/Boston/London, 1999)
23. I. Bomze, M. Pelillo, V. Stix, Approximating the maximum weight clique using replicator dynamics. *IEEE Trans. Neural Netw.* **11**(6), 1228–1241 (2000)
24. I. Bomze, M. Budinich, M. Pelillo, C. Rossi, Annealed replication: a new heuristic for the maximum clique problem. *Discret. Appl. Math.* **121**, 27–49 (2002)
25. R. Brockett, Dynamical systems that sort lists, diagonalize matrices and solve linear programming problems, in *Proceedings of the 27th Conference on Decision and Control* (IEEE, New York, 1988), pp. 799–803
26. R. Brockett, W. Wong, A gradient flow for the assignment problem, in *New Trends in Systems Theory*, ed. by G. Conte, A. Perdon, B. Wyman (Birkhäuser, Boston/Basel/Berlin, 1991), pp. 170–177
27. R. Brooks, New approaches to robotics. *Science* **253**, 1227–1232 (1991)
28. R. Burkard, *Methoden der Ganzzahligen Optimierung* (Springer, Wien/New York, 1972)
29. R. Burkard, M. Dell'Amico, S. Martello, *Assignment Problems* (Society for Industrial and Applied Mathematics, Philadelphia, 2009)
30. D. Cvijović, J. Klinowski, Taboo search: an approach to the multiple minima problem. *Science* **267**(3), 664–666 (1995)
31. A. Daffertshofer, How do ensembles occupy space? *Eur. Phys. J. Spec. Top.* **157**, 79–91 (2008)

32. A. Daffertshofer, H. Haken, J. Portugali, Self-organized settlements. *Environ. Plan. B Plan. Des.* **28**(1), 89–102 (2001)
33. L. Davis, *Handbook of Genetic Algorithms* (Van Nostrand Reinhold, New York, 1991)
34. G. Di Marzo Serugendo, A. Karageorgos, O.F. Rana, F. Zambonelli (eds.), *Engineering Self-Organising Systems – Nature-Inspired Approaches to Software Engineering* (Springer, Heidelberg/Berlin/New York, 2004)
35. R. Durbin, D. Willshaw, An analogue approach to the travelling salesman problem using an elastic net method. *Nature* **326**, 689–691 (1987)
36. W. Ebeling, Self-organization, valuation and optimization, in *On Self-Organization*, ed. by R. Mishra, D. Maaß, E. Zwierlein. Springer Series in Synergetics, vol. 61 (Springer, Berlin/Heidelberg, 1994), pp. 185–196
37. W. Ebeling, A. Engel, R. Feistel, *Physik der Evolutionsprozesse* (Akademie, Berlin, 1990)
38. M. Eigen, P. Schuster, The hypercycle – Part A: emergence of the hypercycle. *Die Naturwissenschaften* **64**, 541–565 (1977)
39. M. Eigen, P. Schuster, The hypercycle – Part B: the abstract hypercycle. *Die Naturwissenschaften* **65**, 7–41 (1978)
40. H. Eiselt, G. Pederzoli, C.-L. Sandblom, *Continuous Optimization Models – Operations Research* (Walter de Gruyter, Berlin/New York, 1987)
41. F. Tay, Contingency management in flexible manufacturing systems using modal state logic. *J. Manuf. Syst.* **18**(5), 345–357 (1999)
42. J. Fort, Solving a combinatorial problem via self-organizing process: an application of the kohonen algorithm to the traveling salesman problem. *Biol. Cybern.* **59**, 33–40 (1988)
43. T.D. Frank, On a multistable competitive network model in the case of an inhomogeneous growth rate spectrum: with an application to priming. *Phys. Lett. A* **373**(45), 4127–4133 (2009)
44. T.D. Frank, Multistable selection equations of pattern formation type in the case of inhomogeneous growth rates: with applications to two-dimensional assignment problems. *Phys. Lett. A* **375**(12), 1465–1469 (2011)
45. T. Fukuda, S. Nakagawa, Approach to the dynamically reconfigurable robotic system. *J. Intell. Robot. Syst.* **1**(1), 55–72 (1988)
46. T. Fukuda, T. Ueyama, *Cellular Robotics and Micro Robotic Systems*. World Scientific Series in Robotics and Automated Systems, vol. 10 (World Scientific, Singapore/New Jersey/Hong Kong, 1994)
47. M. Garey, D. Johnson, *Computers and Intractability* (Freeman and Company, San Francisco, 1979)
48. A. Gee, S. Aiyer, R. Prager, An analytical framework for optimizing neural networks. *Neural Netw.* **6**, 79–97 (1993)
49. F. Glover, Tabu search – Part I. *ORSA J. Comput.* **1**, 190–206 (1989)
50. F. Glover, Tabu search – Part II. *ORSA J. Comput.* **2**, 4–32 (1989)
51. F. Glover, E. Taillard, D. de Werra, *Tabu Search*. Annals of Operations Research, vol. 41 (J.C. Baltzer, Basel, 1993), pp. 3–28
52. D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, 1989)
53. R. Graham, M. Grötschel, L. Lovász, *Handbook of Combinatorics* (Elsevier Science B. V., Amsterdam/Lausanne/New York, 1995)
54. C. Großmann, J. Terno, *Numerik der Optimierung*. Teubner Studienbücher: Mathematik (Teubner, Stuttgart, 1993)
55. M. Grötschel, L. Lovász, Combinatorial optimization, in *Handbook of Combinatorics* [53], chapter 28, pp. 1541–1597
56. C. Guus, E. Boender, H. Edwin Romeijn, Stochastic methods, in *Handbook of Global Optimization*, ed. by R. Horst, P. Pardalos (Kluwer, Dordrecht/Boston/London, 1995), pp. 829–869

57. H. Haken, Pattern formation and pattern recognition – an attempt at a synthesis, in *Pattern Formation by Dynamic Systems and Pattern Recognition*, ed. by H. Haken. Springer Series in Synergetics, vol. 5 (Springer, Heidelberg/Berlin/New York, 1979), pp. 2–13
58. H. Haken, *Advanced Synergetics*. Springer Series in Synergetics (Springer, Heidelberg/Berlin/New York, 1983)
59. H. Haken, *Synergetics, An Introduction*. Springer Series in Synergetics (Springer, Heidelberg/Berlin/New York, 1983)
60. H. Haken, *Synergetic Computers and Cognition – A Top-Down Approach to Neural Nets*. Springer Series in Synergetics (Springer, Heidelberg/Berlin/New York, 1991)
61. H. Haken, *Principles of Brain Functioning – A Synergetic Approach to Brain Activity, Behavior and Cognition*. Springer Series in Synergetics (Springer, Berlin/Heidelberg/New York, 1996)
62. H. Haken, Decision making and optimization in regional planning, in *Knowledge and Networks in a Dynamic Economy*, ed. by M. Beckmann, B. Johansson, F. Snickars, R. Thord (Springer, Berlin/Heidelberg/New York, 1998)
63. H. Haken, M. Schanz, J. Starke, Treatment of combinatorial optimization problems using selection equations with cost terms – Part I: two-dimensional assignment problems. *Physica D* **134**, 227–241 (1999)
64. U. Helmke, J.B. Moore, *Optimization and Dynamical Systems* (Springer, London/Berlin/Heidelberg, 1994)
65. J. Hertz, A. Krogh, R. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley Publishing Company, Redwood City, 1991)
66. M. Hestenes, *Optimization Theory* (Wiley, New York/London, 1975)
67. M. Hirsch, B. Baird, Computing with dynamic attractors in neural networks. *BioSystems* **34**, 173–195 (1995)
68. M. Hirsch, S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra* (Academic, New York, 1974)
69. J. Hofbauer, K. Sigmund, *The Theory of Evolution and Dynamical Systems*. London Mathematical Society Student Texts, vol. 7 (Cambridge University Press, Cambridge/New York, 1988)
70. J. Holland, *Adaption in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, 1975)
71. J. Hopfield, Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci.* **79**, 2554–2558 (1982)
72. J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.* **81**, 3088–3092 (1984)
73. J. Hopfield, D. Tank, Neural computation of decisions in optimization problems. *Biol. Cybern.* **52**, 141–152 (1985)
74. J. Hopfield, D. Tank, Computing with neural circuits: a model. *Science* **233**, 625–633 (1986)
75. R. Horst, *Nichtlineare Optimierung* (Carl Hanser, München/Wien, 1979)
76. T. Kaga, J. Starke, P. Molnár, M. Schanz, T. Fukuda, Dynamic robot-target assignment – dependence of recovering from breakdowns on the speed of the selection process, in *Distributed Autonomous Robotic Systems (DARS 4)*, ed. by L.E. Parker, G. Bekey, J. Barhen (Springer, Heidelberg/New York/Tokyo, 2000), pp. 325–334
77. B. Kamgar-Parsi, B. Kamgar-Parsi, On problem solving with Hopfield neural networks. *Biol. Cybern.* **62**, 415–423 (1990)
78. N. Karmarkar, A new polynomial-time algorithm for linear programming. *Combinatorica* **4**, 373–395 (1984)
79. L.E. Kavraki, J.-C. Latombe, Probabilistic roadmaps for robot path planning, in *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, ed. by K. Gupta, A.P. del Pobil (Wiley, Chichester/New York, 1998), pp. 33–53
80. W. Kinzel, Spin glasses and memory. *Phys. Scr.* **35**, 398–401 (1987)
81. S. Kirkpatrick, Optimization by simulated annealing: quantitative studies. *J. Stat. Phys.* **34**(5/6), 975–986 (1984)

82. S. Kirkpatrick, G. Toulouse, Configuration space analysis of travelling salesman problems. *J. Phys.* **46**, 1277–1292 (1985)
83. S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing. *Science* **220**(4598), 671–680 (1983)
84. T. Kohonen, *Self-Organization and Associative Memory* (Springer, Berlin/Heidelberg/New York, 1984)
85. T. Kohonen, *Self-Organizing Maps* (Springer, Berlin/Heidelberg/New York, 1995)
86. M.J.B. Krieger, J.-B. Billeter, L. Keller, Ant-like task allocation and recruitment in cooperative robots. *Nature* **406**, 992–995 (2000)
87. A. Kusiak, Flexible manufacturing systems: a structural approach. *Int. J. Prod. Res.* **23**(6), 1057–1073 (1985)
88. J.-C. Latombe, *Robot Motion Planning*, 3rd edn. (Kluwer, Dordrecht/Boston/London, 1993)
89. D. Luenberger, *Introduction to Linear and Nonlinear Programming* (Addison-Wesley Publishing Company, New York/London, 1973)
90. S. Matsuda, Stability of solutions in Hopfield neural network. *Syst. Comput. Jpn* **26**(5), 67–78 (1995) (Translated from Vol. J77-D-II, No. 7, July 1994, pp. 1366–1374)
91. S. Matsuda, Theoretical considerations on the capabilities of crossbar switching by Hopfield networks, in *Proceedings of the 1995 IEEE International Conference on Neural Networks* (IEEE, 1995), pp. 1107–1110
92. N. Metropolis, M. Rosenbluth, A. Teller, E. Teller, Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**(6), 1087–1092 (1953)
93. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, Berlin/Heidelberg/New York, 1992)
94. P. Molnár, J. Starke, Control of distributed autonomous robotic systems using principles of pattern formation in nature and pedestrian behaviour. *IEEE Trans. Syst. Man Cybern. Part B* **31**(3), 433–436 (2001)
95. B. Müller, J. Reinhardt, *Neural Networks – An Introduction* (Springer, Berlin/Heidelberg/New York, 1991)
96. Y. Nesterov, Interior-point methods: an old and new approach to nonlinear programming. *Math. Program.* **79**, 285–297 (1997)
97. R. Neubecker, G.-L. Oppo, B. Thüring, T. Tschudi, Pattern formation in a liquid-crystal light valve with feedback, including polarization, saturation, and internal threshold effects. *Phys. Rev. A* **52**(1), 791–808 (1995)
98. G. Nicolis, I. Prigogine, *Self-Organization in Non-Equilibrium Systems* (Wiley, New York, 1977)
99. K. Pál, Genetic algorithms for the traveling salesman problem based on a heuristic crossover operation. *Biol. Cybern.* **69**, 539–546 (1993)
100. C. Papadimitriou, K. Steiglitz, *Combinatorial Optimization – Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, 1982)
101. M. Pelillo, Replicator equations, maximal cliques, and graph isomorphism. *Neural Comput.* **11**, 1933–1955 (1999)
102. M. Pelillo, Evolutionary game dynamics in combinatorial optimization: an overview, in *Proceedings of the EvoWorkshops on Applications of Evolutionary Computing*, ed. by E.J.W. Boers, J. Gottlieb, P.L. Lanzi, R.E. Smith, S. Cagnoni, E. Hart, G.R. Raidl, H. Tijink (Springer, Heidelberg/New York/Tokyo, 2001), pp. 182–192
103. M. Pelillo, Evolutionary game dynamics in combinatorial optimization: an overview, in *Applications of Evolutionary Computing*, ed. by E. Boers. Lecture Notes in Computer Science, vol. 2037 (Springer, Berlin/Heidelberg, 2001), pp. 182–192
104. M. Pelillo, K. Siddiqi, S.W. Zucker, Matching hierarchical structures using association graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **21**(11), 1105–1120 (1999)
105. P. Peretto, Neural networks and combinatorial optimization, in *Proceedings of the International Conference “Les Entretiens de Lyon”* (Springer, Paris, 1990), pp. 127–134
106. C. Peterson, B. Söderberg, Neural optimization, in *Brain Theory and Neural Networks*, ed. by M. Arbib (MIT, Cambridge/London, 1995), pp. 617–621

107. W. Press, S. Teukolsky, W. Vetterling, B. Flannery, *Numerical Recipes in C* (Cambridge University Press, Cambridge/New York/1992)
108. D. Psaltis, D. Brady, X. Gu, S. Lin, Holography and artificial neural networks. *Nature* **343**, 325–330 (1990)
109. I. Rechenberg, *Evolutionsstrategie* (Friedrich Frommann, Stuttgart Bad Cannstatt, 1973)
110. C. Robinson, *Dynamical Systems – Stability, Symbolic Dynamics, and Chaos* (CRC, Boca Raton/Ann Arbor/London, 1995)
111. H.-P. Schwefel, *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie* (Birkhäuser, Basel/Stuttgart, 1977)
112. Z. Simeu-Abazi, C. Sassine, Maintenance integration in manufacturing systems: from the modeling tool to evaluation. *Int. J. Flex. Manuf. Syst.* **13**(3), 267–285 (2001)
113. K. Smith, Neural networks for combinatorial optimization: a review of more than a decade of research. *INFORMS J. Comput.* **11**(1), 15–34 (1999)
114. P. Spellucci, *Numerische Verfahren der nichtlinearen Optimierung* (Birkhäuser, Basel/Boston/Berlin, 1993)
115. F.C.R. Spieksma, Multi-index assignment problems: complexity, approximation, applications, in *Nonlinear Assignment Problems: Algorithms and Applications*, ed. by L. Pitsoulis, P. Pardalos (Kluwer, Amsterdam, 2000), pp. 1–12
116. J. Starke, Cost oriented competing processes – a new handling of assignment problems, in *System Modelling and Optimization*, ed. by J. Doležal, J. Fidler (Chapman & Hall, London/Glasgow, 1996), pp. 551–558
117. J. Starke, Combinatorial optimization based on the principles of competing processes, in *Self-Organization of Complex Structures: From Individual to Collective Dynamics. Part I: Evolution of Complexity and Evolutionary Optimization*, ed. by F. Schweitzer (Gordon and Breach, London, 1997), pp. 165–178
118. J. Starke, *Kombinatorische Optimierung auf der Basis gekoppelter Selektionsgleichungen*. Ph.D. thesis, Universität Stuttgart, Verlag Shaker, Aachen, 1997
119. J. Starke, Dynamical assignments of distributed autonomous robotic systems to manufacturing targets considering environmental feedbacks, in *Proceedings of the 17th IEEE International Symposium on Intelligent Control (ISIC'02)*, Vancouver, 2002, pp. 678–683
120. J. Starke, C. Ellsässer, T. Fukuda, Self-organized control in cooperative robots using a pattern formation principle. *Phys. Lett. A* **375**, 2094–2098 (2011)
121. J. Starke, P. Molnár, Dynamic control of distributed autonomous robotic systems with underlying three-index assignments, in *Proceedings of the IECON 2000* (IEEE, New York, 2000), pp. 2093–2098
122. J. Starke, M. Schanz, Dynamical system approaches to combinatorial optimization, in *Handbook of Combinatorial Optimization*, vol. 2, ed. by D.-Z. Du, P. Pardalos (Kluwer, Dordrecht/Boston/London, 1998), pp. 471–524
123. J. Starke, M. Schanz, H. Haken, Self-organized behaviour of distributed autonomous mobile robotic systems by pattern formation principles, in *Distributed Autonomous Robotic Systems (DARS 3)*, ed. by T. Lueth, R. Dillmann, P. Dario, H. Wörn (Springer, Heidelberg/Berlin/New York, 1998), pp. 89–100
124. J. Starke, M. Schanz, H. Haken, Treatment of combinatorial optimization problems using selection equations with cost terms – Part II: three-dimensional assignment problems. *Physica D* **134**, 242–252 (1999)
125. J. Starke, T. Kaga, M. Schanz, T. Fukuda, Experimental study on self-organized and error resistant control of distributed autonomous robotic systems. *Int. J. Robot. Res.* **24**, 465–486 (2005)
126. G.A. Tagliarini, J.F. Christ, E.W. Page, Optimization using neural networks. *IEEE Trans. Comput.* **40**(12), 1347–1358 (1991)
127. T.-Y. Tam, Gradient flows and double bracket equations. *Differ. Geom. Appl.* **20**, 209–224 (2004)
128. D. Tank, J. Hopfield, Simple neural optimization networks: an A/D converter, signal decision circuit and a linear programming circuit. *IEEE Trans. Circuits Syst.* **CAS-33**(5), 533–541 (1986)

129. K. Tsuchiya, T. Nishiyama, K. Tsujita, A deterministic annealing algorithm for a combinatorial optimization problem by the use of replicator equations, in *IEEE International Conference on Systems, Man, and Cybernetics, 1999. Conference Proceedings*, Tokyo, vol. 1, 1999, pp. 256–261
130. K. Tsuchiya, T. Nishiyama, K. Tsujita, A deterministic annealing algorithm for a combinatorial optimization problem using replicator equations. *Physica D* **149**, 161–173 (2001)
131. Y. Uesaka, Mathematical aspects of neuro-dynamics for combinatorial optimization. *IEICE Trans. E* **74**(6), 1368–1372 (1991)
132. K. Urahama, Analog circuit for solving assignment problems. *IEEE Trans. Circuits Syst.* **41**(5), 426–429 (1994)
133. D. Van den Bout, T. Miller, A traveling salesman objective function that works, in *Proceedings of the IEEE International Conference on Neural Networks 1988*, vol. II (IEEE, San Diego, 1988), pp. II–299–II–303
134. D. Van den Bout, T. Miller III, Improving the performance of the Hopfield-Tank neural network through normalization and annealing. *Biol. Cybern.* **62**, 129–139 (1989)
135. P. van Laarhoven, E. Aarts, *Simulated Annealing: Theory and Applications* (Reidel Publishing Company, Dordrecht/Boston/Lancaster/Tokyo, 1987)
136. S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos* (Springer, Berlin/Heidelberg/New York, 1990)
137. G. Wilson, G. Pawley, On the stability of the travelling salesman problem algorithm of Hopfield and Tank. *Biol. Cybern.* **58**, 63–70 (1988)
138. W. Wong, Matrix representation and gradient flows for NP-hard problems. *J. Optim. Theory Appl.* **87**(1), 197–220 (1995)
139. A. Yuille, Constrained optimization and the elastic net, in *Brain Theory and Neural Networks*, ed. by M. Arbib (MIT, Cambridge/London, 1995), pp. 250–255
140. M.M. Zavlanos, G.J. Pappas, A dynamical system approach to weighted graph matching. *Automatica* **44**, 2817–2824 (2008)

Efficient Algorithms for Geometric Shortest Path Query Problems

Danny Z. Chen

Contents

1	Introduction	1126
2	The Gateway Paradigm	1129
3	Exact Shortest Path Queries	1129
3.1	The Visibility-Sensitive Approach	1130
3.2	The Reduced Visibility Graph Approach	1131
3.3	Other Related Path Problems	1137
4	Approximate Shortest Path Queries	1138
4.1	Main Ideas	1139
4.2	Planar Spanners	1140
4.3	Short Paths in Planar Graphs	1142
4.4	Short Path Queries Amid Obstacles in the Plane	1145
4.5	A Special Case	1148
5	Some Other Geometric Shortest Path Query Results	1148
6	Conclusion	1149
	Recommended Reading	1151

Abstract

Computing shortest paths in a geometric environment is a fundamental topic in computational geometry and finds applications in many other areas. The problem of processing geometric shortest path queries is concerned with constructing an efficient data structure for quickly answering on-line queries for shortest paths connecting any two query points in a geometric setting. This problem is a generalization of the well-studied problem of computing a geometric shortest path connecting two specified points. This chapter covers several effective

D.Z. Chen

Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN,
USA
e-mail: dchen@cse.nd.edu

algorithmic paradigms for processing geometric shortest path queries and related problems. These general paradigms have led to efficient techniques for designing algorithms and data structures for processing a variety of queries on exact and approximate shortest paths in a number of geometric and graph-theoretic settings. Some open problems and promising directions for future research are also discussed.

1 Introduction

The problems of computing optimal or near-optimal paths in a certain environment arise in many disciplines and in fact are one of the several most powerful tools for modeling combinatorial optimization problems. A well-known example of path-planning problem is to compute shortest paths in a graph [45, 53]. A geometric version of this shortest path problem is, given a d -dimensional space scattered with geometric obstacles, to compute a path connecting two specified locations such that the path does not intersect the interior of any obstacle and such that the total length of the path (based on a certain metric such as the Euclidean or L_p for a positive value p) is minimized. There are many variations and generalizations of this geometric shortest path problem. Figure 1 gives an example of a shortest obstacle-avoiding path in the plane.

Computing geometric shortest paths is a very fundamental topic in computational geometry, a field that studies the development of algorithms and analysis of complexity for problems with useful geometric structures [47, 49, 80, 84]. Geometric path-planning problems play an important role in solving other geometric problems and in many practical applications, such as computer-aided design, geographical information systems, intelligent transportation systems, operations research, pattern matching, plant and facility layout, robotics, and VLSI. Furthermore, geometric path-planning problems have significant connections with other fundamental topics in computational geometry (e.g., convexity, geometric graphs, geometric optimization, triangulation, visibility, and Voronoi diagrams) and with other disciplines (e.g., graph theory, combinatorial optimization, and networks). Problems in these areas or geometric topics often appear as subproblems in solving some geometric shortest path problems and vice versa. For example, geometric path-planning problems address quite naturally the combinatorial and algorithmic aspects of robot motion and navigation.

Consequently, a great deal of work has been done on solving various geometric shortest path problems with respect to different types of environment specifications, path constraints, obstacle natures, and optimization criteria, especially for planar and 3-dimensional (3-D) settings [74, 75, 77]. In particular, for computing shortest paths in the plane with polygonal obstacles, Hershberger and Suri [63] obtained an optimal $O(n \log n)$ -time algorithm (where n is the total number of obstacle vertices). For computing shortest paths in the 3-D space with polyhedral obstacles, Canny and Reif [23] showed that the problem is NP-hard, and several efficient approximation algorithms have been discovered [40, 41, 81].

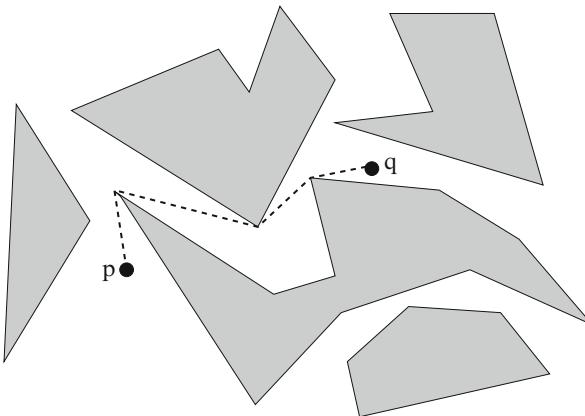


Fig. 1 A shortest obstacle-avoiding path connecting points p and q

This chapter mainly considers the following version of *geometric shortest path query problem*:

Given a set of (possibly weighted) obstacles in a geometric space S , construct an efficient data structure so that on-line queries on shortest paths (and their lengths) connecting any two query points in S can be quickly answered (e.g., in polylogarithmic time for a length query). If one of the two points for a path query is fixed, then the queries are called *single-source* path queries.

Despite the fact that there are numerous geometric algorithms known for computing a shortest path between two specified points and for computing single-source shortest paths, few results were known, until recently, for solving geometric shortest path *query* problems, even in the plane. In fact, the only known geometric path query algorithms before the results discussed in this chapter are for several cases of shortest paths inside a *simple polygon* [12, 39, 46, 54, 56, 57, 62, 87] (the optimal algorithms for these cases take $O(n)$ time and space to build a data structure for a simple polygon of n vertices; using such a data structure, a length query for any two query points can be answered in $O(\log n)$ time, and an actual shortest path connecting the two query points can be reported in $O(\log n + k)$ time, where k is the number of vertices on the output path). Clearly, the general problems which are concerned with geometric shortest path queries among *many* obstacles are of much more theoretical and practical interest and are certainly much more challenging than the simple polygon case.

It is obvious that geometric shortest path *query* problems are natural generalizations of the well-studied problem of computing a geometric shortest path connecting two specified points. Geometric shortest path queries are useful in databases for several application areas such as geographical information systems and intelligent transportation systems. These problems model the practical situations in which good quality routes (under certain criteria) between many pairs of different locations or between two constantly changing locations in a geometric environment are sought. For example, it is very common for a police car patrolling an area to quickly get to

a spot where an accident has occurred. In this situation, it is natural to treat both the locations of the police car and the accident as on-line input data to a path query. In graph theory, a common solution to the shortest path query problem is nothing more than computing and storing the all-pairs shortest paths in a graph. However, in a geometric environment, a solution based straightforward on computing and storing all-pairs geometric shortest paths will not work well because normally there are uncountably infinitely many points in a geometric space. Furthermore, unlike the graph version of the problems, it was not immediately clear until recently how to use geometric single-source shortest path query algorithms and data structures to obtain reasonably efficient solutions for processing geometric shortest path queries between arbitrary points.

This chapter discusses several effective algorithmic paradigms for processing geometric shortest path queries and related problems. (The author of this chapter was involved in many of the known results on geometric shortest path query problems.) These general paradigms have led to very efficient techniques for designing algorithms and data structures for a variety of queries on exact and approximate geometric shortest paths in planar settings. More importantly, these frameworks offer the promise of achieving new efficient algorithmic techniques for geometric shortest path queries and for other related problems. This chapter also discusses some open problems and promising directions for future research on geometric shortest path problems.

The focus of discussion in this chapter will be on the planar settings that contain polygonal obstacles. Let n denote the number of obstacle vertices in the plane.

It should be pointed out that it is possible to solve some geometric shortest path query problems in the plane by using known computational geometry techniques (e.g., by reducing such a problem to higher-dimensional point location). However, such an approach tends to yield data structures with a storage space and construction time of polynomials of rather high degrees, which would hardly be usable to practical applications. In contrast, the algorithmic solutions presented in this chapter all take either near-quadratic or even subquadratic space and construction time, and support in most cases polylogarithmic query time. Hence, in addition to their theoretical merit, the solutions in this chapter are likely to provide an impact to practical applications.

This chapter will focus on discussing *length queries* (i.e., reporting the length of a shortest or approximate shortest path). The length of a geometric path is often determined by the L_p metric for some positive value p (p is usually a positive integer). Recall that for two points a and b in the plane, the L_p distance $D_p(a, b)$ between a and b is defined as $D_p(a, b) = (|a_x - b_x|^p + |a_y - b_y|^p)^{1/p}$, where q_x and q_y denote the x - and y -coordinates of a point q . When an actual shortest path is desired, the algorithms discussed in this paper, after finding its length, can also report such an actual path P in an additional $O(|P|)$ time, where $|P|$ is the number of vertices of P .

The rest of the chapter is organized as follows. [Section 2](#) presents a general paradigm, called the *gateway paradigm*, for processing geometric shortest path queries. [Section 3](#) applies this paradigm to several problems of processing *exact*

geometric shortest path queries. [Section 4](#) applies this paradigm to problems of processing *approximate* geometric shortest path queries. [Section 5](#) sketches some recent developments on solving geometric shortest path query problems. [Section 6](#) discusses some open problems and promising directions for future research.

This chapter is a revised and updated version of the paper [28].

2 The Gateway Paradigm

The following paradigm, which was introduced in [30, 32], is a key to the algorithmic techniques for processing geometric shortest path queries presented in this chapter. Let s and t be two query points between which a shortest path in a geometric environment S is sought.

The Gateway Paradigm: For a query point s , identify a point set W_s in the geometric space S such that a shortest path in S connecting s and the other query point t passes through some point in W_s . The points in W_s are called the *gateways* of s . Once W_s is available, a shortest path between s and t can be obtained from the single-source path query data structures for the gateways of s (with each point in W_s being the source of such a data structure).

For the gateway paradigm to work well, several difficulties must be resolved:

1. W_s must have a reasonably small size because this will give rise to an $\Omega(|W_s| \times f(n))$ query time, where $f(n)$ is the time for a single-source path query and $|W_s|$ is the size of W_s .
2. It must be possible to quickly identify the gateway points of W_s for any query point s (this is part of the cost for the query time).
3. It must be known how to build efficient single-source data structures for the desired shortest paths (this is a key factor to the overall complexity bounds of the shortest path query data structure).
4. Let $W = \cup_{s \in S} W_s$, where S is the geometric space (i.e., W is the union of gateways for all possible query points in S). The size of W must be finite and, furthermore, cannot be too large (this is because for each point of W , it needs to build a single-source path query data structure).

Several different forms of the above general gateway paradigm arise in the solutions below to various geometric shortest path query problems, due to the specific constraints and geometric structures of each of these problems. These solutions will be discussed in the next two sections.

3 Exact Shortest Path Queries

This section discusses efficient algorithmic techniques for processing various length queries on *exact* geometric shortest paths among polygonal obstacles in the plane. The lengths of the paths considered here are determined according to the L_1 and L_2 (i.e., the Euclidean), the link (i.e., the number of edges) metric, or some combinations of these metrics. The algorithms in this chapter are crucially based on the gateway paradigm.

One thing needs to be mentioned before the discussion proceeds: For any two query points s and t , if the line segment \overline{st} whose end points are s and t does not cross any obstacle boundary edge, then the shortest path between s and t in the L_2 metric is simply \overline{st} . In this case, it says that s and t are *visible* to each other. It is easy to detect whether s and t are visible to each other by using the ray-shooting technique [1–3] in computational geometry. A *ray-shooting* operation is that given a point p in an obstacle scene and a direction d , “shoot” a ray $Ray(p, d)$ from p in the direction d , until $Ray(p, d)$ hits for the first time an obstacle boundary. To find out whether two query points s and t are visible to each other, one only needs to shoot a ray from s in the direction of t . It is known [1–3] that for a given polygonal obstacle scene, one can build a data structure in $O(n^2)$ space and $O(n^2 \log n)$ time that supports any ray-shooting operation in logarithmic time. Hence, it is easy to handle the case in which s and t are visible to each other by using a ray-shooting data structure. In the rest of this section, it assumes without loss of generality (WLOG) that the two query points s and t are *not* visible to each other.

3.1 The Visibility-Sensitive Approach

A general technique, called the *visibility-sensitive approach* [30], is first discussed for various geometric path queries. This approach is based on the computation of visibility polygons of query points s and t .

The *visibility polygon* Q_s of a point s is the (possibly unbounded) polygonal region in the plane that is visible from s , with the obstacle boundaries being the “opaque” objects (see Fig. 2 for an example). The polygon Q_s is represented as a counterclockwise sequence of vertices of the obstacle scene that are visible from s when scanning around s a ray that originates at s .

The simple observation used here is that when the query points s and t are not visible to each other, various optimal paths between s and t pass through one of the vertices visible from s (resp., t). Therefore, it can use the vertices of Q_s (resp., Q_t) as the gateways of s (resp., t). In this way, the set of gateways for the whole obstacle scene consists of all the n obstacle vertices.

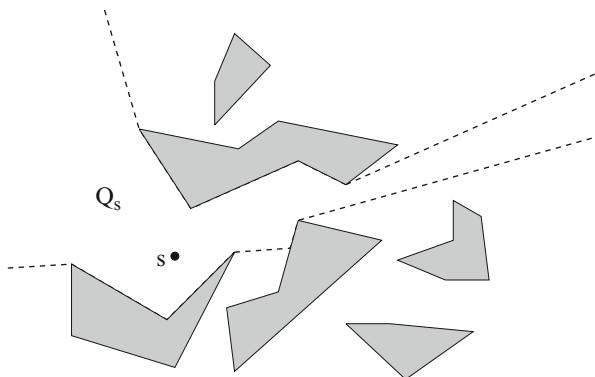
Based on the *visibility complex* of Pocchiola and Vegter [82, 83], it was developed in [30] the following *visibility-sensitive approach*:

Construct a geometric path data structure that supports, for any two query points s and t in the plane, an $O(\min\{|Q_s|, |Q_t|\} \times \log n)$ query time on path lengths.

This approach enables one to identify and use only $O(\min\{|Q_s|, |Q_t|\})$ instead of $O(|Q_s| + |Q_t|)$ gateway points, without having to compute both Q_s and Q_t completely. Note that $\min\{|Q_s|, |Q_t|\}$ can be much smaller than n and can be as small as $O(1)$. The visibility complex of a polygonal obstacle scene with n vertices can be constructed in $O(n^2)$ time and space [83].

Using this approach, data structures for length queries on various geometric paths were achieved that support a query time of $O(\min\{|Q_s|, |Q_t|\} \times \log n)$. In [30], this approach has been applied for solving three path query problems among polygonal

Fig. 2 The visibility polygon Q_s of a point s



obstacles: (1) Euclidean shortest paths, (2) approximate minimum-link paths, and (3) monotone paths (among convex polygonal obstacles).

- Euclidean shortest obstacle-avoiding paths among polygonal obstacles. The data structure uses $O(n^2 \log n)$ time and $O(n^2)$ space. This solution makes use of Hershberger and Suri's single-source path data structure [63].
- Approximate minimum-link paths among polygonal obstacles. The problem is to compute an obstacle-avoiding approximate minimum-link path between any two query points. The path obtained has at most two more links than the exact optimal path. The data structure takes $O(n((|E| + \ln)^{2/3} n^{2/3} l^{1/3} \log^{3.11} n + |E| \log^3 n))$ time and $O(n^2)$ space to build, where E is the edge set of the visibility graph of the obstacle scene and l is the longest link length among all the minimum-link paths between any two obstacle vertices. This solution makes use of Mitchell, Rote, and Woeginger's single-source path data structure [78].
- Monotone paths among convex polygonal obstacles. The problem is to find a direction d such that there is an obstacle-avoiding path between two query points that is monotone to d (it was shown in [11] that for convex polygonal obstacles, such a direction always exists). The data structure uses $O(n^2 \log n)$ time and $O(n^2)$ space to build. This solution makes use of Arkin, Connelly, and Mitchell's single-source path data structure [11].

3.2 The Reduced Visibility Graph Approach

The visibility-sensitive approach discussed in the previous subsection is quite general and gives rise to several reasonably efficient path query data structures. But, the query time of these data structures depends on the size of the smaller visibility polygon of the query points, which can still be as big as n in the worst case. In an attempt to reduce this query time while still maintaining the efficiency of the path query data structures, a different approach, called *reduced visibility graph approach*, was developed [30, 32]. This approach is particularly useful to queries on various rectilinear optimal paths and has yielded data structures that use near-quadratic construction time and space and support polylogarithmic query time.

A *rectilinear* geometric object is one whose boundary edges are parallel to a Cartesian coordinate axis (i.e., horizontal or vertical). Note that in rectilinear path problems, the polygonal obstacles need not be rectilinear. Since rectilinear shortest paths have many applications, especially in VLSI design, considerable work has been done on designing a variety of rectilinear shortest path algorithms [68]. Notably, Mitchell [71, 72] obtained an optimal $O(n \log n)$ -time algorithm for building an $O(n)$ -space data structure for planar rectilinear single-source shortest path queries among general (nonrectilinear) polygonal obstacles.

The reduced visibility graph approach is quite powerful and has been applied to the following optimal path query problems [30, 32]:

- L_1 shortest paths amid general polygonal obstacles.
- Shortest A -distance paths amid general polygonal obstacles. A -distance paths are paths whose edges can be in arbitrary directions but whose edge lengths are measured based on $|A|$ given orientations, where A is a finite set of allowed orientations [89].
- Rectilinear shortest paths amid *weighted* rectilinear polygonal obstacles such that the paths are allowed to penetrate “obstacles” with “weight factors” by paying higher costs for the path lengths (more on this later).
- A rectilinear path with the minimum number of links among all the rectilinear shortest paths between any two query points s and t , amid rectilinear polygonal obstacles.
- A rectilinear shortest path among all the rectilinear minimum-link paths between s and t , amid rectilinear polygonal obstacles.
- The rectilinear minimum-cost path between s and t amid rectilinear polygonal obstacles, where the cost is a nondecreasing function of the number of links and the length of a path.

The above paths whose “lengths” are determined according to a combination of the L_1 and link metrics are called *bicriteria* paths.

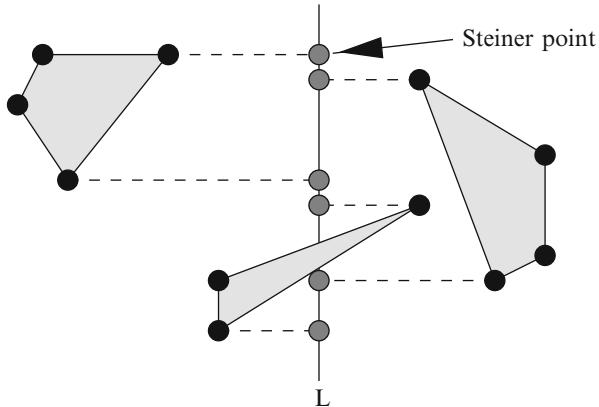
In a plane with *weighted* obstacles, each obstacle is associated with a nonnegative *weight* factor so that a path in the plane, if it intersects the interior of an obstacle, is charged extra cost based on the weight of that obstacle in addition to the cost of the L_1 length of the path. Specifically, the “length” of a path in the interior of an obstacle with a weight factor of W is $(1 + W)D_1$, where D_1 is the length of the path in the L_1 metric. Weighted obstacles often appear in path-planning problems in real applications. The weight factors, for example, can represent speed limits or traffic conditions of streets in a city. Note that shortest paths amid weighted obstacles are in fact a generalization of the shortest paths that completely avoid the interior of obstacles because shortest paths become obstacle-avoiding if one allows the weight of each obstacle be $+\infty$.

The rest of this subsection discusses the reduced visibility graph approach and the resulting optimal path query algorithms.

3.2.1 Reduced Visibility Graphs

A key component of the rectilinear path query data structures is the *reduced visibility graph* $G = (V(G), E(G))$, which captures the necessary information about shortest paths among the n obstacle vertices. This graph was introduced by Clarkson

Fig. 3 Creating Steiner points on the cut line L



et al. [42, 43] for computing L_1 shortest paths amid general polygonal obstacles and was generalized by Lee et al. [67] to computing rectilinear shortest paths amid weighted rectilinear polygonal obstacles. Yang et al. [90] also used this graph to compute rectilinear optimal bicriteria paths amid rectilinear polygonal obstacles.

The vertex set $V(G)$ of the reduced visibility graph G can be partitioned into the set V_O of obstacle vertices and the set V_S of *Steiner points*. The following recursive procedure is used to generate Steiner points for the graph G :

1. Draw a vertical (resp., horizontal) line L , which is called a *cut line*, at the median of the x -(resp., y -)coordinates of all the vertices in V_O .
2. Project all the vertices in V_O that are visible in the horizontal (resp., vertical) direction from the cut line L onto L . The projection points of V_O on L are the Steiner points of V_S on L . [Figure 3](#) gives an example of this step.
3. Use the cut line L to partition V_O into two subsets S_1 and S_2 , one on each side of L .
4. Perform this procedure recursively on the vertex sets S_1 and S_2 , respectively, until the size of each vertex set becomes 1.

Because the recursive procedure above has $O(\log n)$ recursion levels, it clearly generates $O(n \log n)$ Steiner points in V_S . Since $|V_O| = O(n)$, there are totally $O(n \log n)$ vertices in $V(G) = V_O \cup V_S$. Each cut line L is associated with a *level number*, which is the number of the recursion level at which L is used in the above procedure (with the root level being level 1).

The edge set $E(G)$ of G consists of the set E_V of line segments between every Steiner point in V_S and its corresponding vertex in V_O and the set E_L of line segments between consecutive Steiner points on every cut line if the two consecutive Steiner points are visible to each other. Clearly, $|E(G)| = O(n \log n)$.

The algorithms in [42, 67] run Dijkstra's shortest path algorithm on G [53] to find a shortest path between two points s and t in the plane (with both s and t being included in $V(G)$), in $O(n \log^2 n)$ time and $O(n \log n)$ space. A modified version G' of the graph G was then used in [43, 67] such that G' consists of $O(n \sqrt{\log n})$ vertices and $O(n \log^{1.5} n)$ edges. Consequently, running Dijkstra's shortest path algorithm on G' [53] takes $O(n \log^{1.5} n)$ time and space.

As will be seen in the next subsubsection, the vertex set $V(G)$ of G is used as the gateways for the whole plane. Hence, it needs to compute shortest paths in the reduced visibility graph G (with $O(n \log n)$ vertices and edges). In [30, 32], single-source and all-pairs shortest paths in G are computed without having to straightforwardly apply the shortest path algorithm in [53] to the graph G . Actually, by exploiting the special structures of G , it computes single-source shortest paths in G in $O(n \log^{1.5} n)$ time and $O(n \log n)$ space and all-pairs shortest paths in G in $O(n^2 \log^2 n)$ time and space, both of which are improvements over the previous algorithms in [42, 43, 67].

3.2.2 Characterization of Gateways

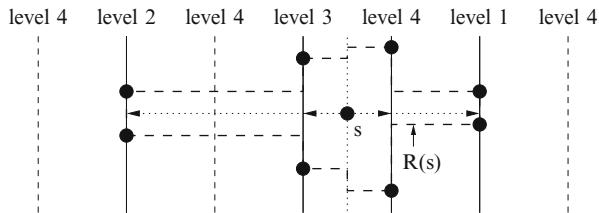
The idea for computing a set W_s of gateways for a query point s is that of “inserting” s into the reduced visibility graph G . That is, s is treated as if it were one of the obstacle vertices, and compute the edges in G adjacent to s that would have resulted if the graph construction procedure in the previous subsubsection for constructing G were applied to s . Note that such an “insertion” of a point into G does not essentially change the shortest path information contained in G (i.e., s is not a new obstacle); furthermore, the resulting “graph” (after the “insertions”) contains shortest path information among the inserted points and the vertices of G , if a shortest path between the inserted points does intersect some vertices of G . Also, note that the “insertion” process does not actually modify the graph G because the points are never truly inserted into G .

To “insert” a point s into the reduced visibility graph G , it is necessary to project the point s onto the relevant cut lines based on the graph construction procedure. A fixed set of $O(n)$ cut lines are used in the construction of G . These cut lines subdivide the plane recursively, and each cut line is associated with a particular recursion level. In consequence, all points in each region of the resulting planar subdivision can be projected onto the same subset of cut lines. If, according to the graph construction procedure, s would have been projected onto a cut line L , then it says L is a *projection cut line* of s . Note that if a cut line L is a projection cut line of s , then s is visible in the horizontal or vertical direction from L . But, not every cut line visible from s is a projection cut line of s . It is sufficient for the discussion to use vertical cut lines only.

The lemmas and observations in this subsection are proved in [30, 32].

The set W_s of gateways for a point s is determined as follows: For each projection cut line L of s , if z is the vertex of G on L that is immediately above (resp., below) the projection point of s on L , then $z \in W_s$ (see Fig. 4). Observe that if s were “inserted” into G , then the only edges adjacent to s in the graph would be those connecting s with its projection points. Because each projection point of s is adjacent to at most two neighboring vertices of G on its cut line, W_s controls every path in G from s to any other vertex of G . Since, at each recursion level of the graph construction procedure, s can be projected onto at most one cut line, there are $O(\log n)$ projection cut lines for s . Along each projection cut line of s , there can be at most two neighboring vertices of G . Therefore, the following two lemmas follow.

Fig. 4 The gateways and gateway region $\mathcal{R}(s)$ of a point s



Lemma 1 For each point s in the plane, $|W_s| = O(\log n)$.

Lemma 2 For each point s and each vertex v of G , there is a shortest s -to- v path in the plane that goes through a vertex of W_s .

For every vertex $z \in W_s$, an “edge” (s, z) in the graph is defined as follows: Let z be on a projection cut line L of s ; then, the edge (s, z) consists of the segments $sp(s)$ and $p(s)z$, where $p(s)$ is the projection point of s on L .

A gateway region $\mathcal{R}(s)$ (Fig. 4) of a point s is the area defined by an interconnection of the vertices in W_s . WLOG, it is only shown how the gateways of W_s that are in the first quadrant of s are connected together:

1. Let a vertical “pseudo” cut line pass through s .
2. For each vertex v of W_s in the first quadrant of s , project v horizontally to the projection cut line L of s that is immediately to the left of v (it can be shown that such a projection is always possible even among weighted obstacles). Note that L can be the “pseudo” cut line passing s .
3. Let the projection point of v on L be $p(v)$ and the vertex of W_s that is on L and in the first quadrant of s be u (if u exists).
4. Connect u and v by line segments $up(v)$ and $p(v)v$.
5. If v is the rightmost gateway of s , then connect v with the point (v_x, s_y) by a segment.
6. If L is the pseudo cut line passing through s (and hence u does not exist on L), then connect v and $p(v)$ by a segment.

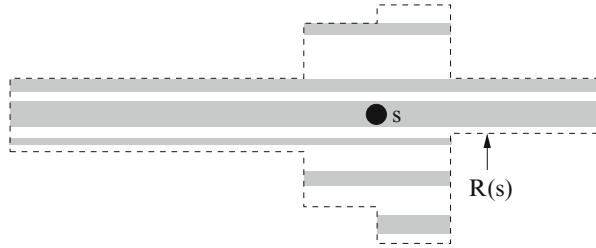
The area in the first quadrant of s enclosed together by the polygonal chain so defined, the x -axis and y -axis (with s as the origin), is part of $\mathcal{R}(s)$. $\mathcal{R}(s)$ is the union of the areas so defined in the four quadrants of s .

A region \mathcal{R} in the plane is said to be *rectilinearly convex* if \mathcal{R} is a connected region and, for any horizontal or vertical line L , $L \cap \mathcal{R}$ is either a single segment or empty. The following lemma characterizes some crucial structures of $\mathcal{R}(s)$.

Lemma 3 The gateway region $\mathcal{R}(s)$ of any point s has the following structures:

- $\mathcal{R}(s)$ is rectilinearly convex.
- For any point z in $\mathcal{R}(s)$, the points (z_x, s_y) and (s_x, z_y) are both in $\mathcal{R}(s)$.
- $\mathcal{R}(s)$ contains no vertices of $V_O \cup V_I$ in its interior.

Fig. 5 Horizontal obstacle strips across the gateway region $\mathcal{R}(s)$



[Lemma 3](#) implies that when obstacles are unweighted (i.e., their weights are all ∞), the interior of $\mathcal{R}(s)$ is either completely free of obstacle or is completely contained within an obstacle. However, when obstacles are weighted, there is an additional case in which the interior of $\mathcal{R}(s)$ contains parallel “strips” of obstacles which are all across $\mathcal{R}(s)$ (see [Fig. 5](#)).

From the above discussion, one can see that the set of gateways for the whole plane is the vertex set $V(G)$ of the reduced visibility graph G .

3.2.3 Algorithms for Rectilinear Path Queries

By making use of the structures of a gateway region $\mathcal{R}(s)$ ([Lemma 3](#)), W_s can be computed efficiently for any query point s , especially when $\mathcal{R}(s)$ is completely free of obstacle or is completely contained within an obstacle (e.g., in $O(\log^2 n)$ time by doing a binary search on the vertices of G along each projection cut line of s or in $O(\log n)$ time by using a *fractional cascading* data structure [[24, 25](#)]). But, there are still some difficulties. For example, amid weighted obstacles, each edge connecting s with a gateway point in W_s can penetrate as many as $O(n)$ obstacle strips across the gateway region $\mathcal{R}(s)$ ([Fig. 5](#)), making the computation of such a weighted edge length seemingly quite costly. However, by exploiting numerous geometric observations on these path problems, it is possible to compute W_s and the costs of all the edges adjacent to s in $O(\log n)$ time [[30, 32](#)].

The algorithms for various path query problems are summarized as follows:

- L_1 shortest paths amid general polygonal obstacles [[32](#)]. Here, the path edges can be in arbitrary directions, but the edge lengths are measured in the L_1 metric. It makes use of Mitchell’s single-source L_1 shortest path data structure [[71, 72](#)]. This L_1 path query data structure takes $O(n^2 \log^2 n)$ time and $O(n^2 \log n)$ space to build and supports a length query in $O(\log^2 n)$ time.
- Shortest A -distance paths amid general polygonal obstacles [[30](#)]. It constructs a generalized reduced visibility graph, which has $O(|A|n \log n)$ vertices and edges. It is shown that there are $O(|A| \log n)$ gateways for each query point. Since single-source shortest path data structure was not available before, both the single-source and general path query data structures need to be developed. The general path query data structure takes $O(|A|^2 n^2 \log^2 n)$ time and space to build and supports a length query in $O(|A|^2 \log^2 n)$ time.
- Rectilinear shortest paths amid weighted rectilinear polygonal obstacles [[32](#)]. It needs to develop both the single-source and general path query data structures.

The general path query data structure takes $O(n^2 \log^2 n)$ time and space to build and supports a length query in $O(\log^2 n)$ time.

- Rectilinear optimal bicriteria paths amid rectilinear polygonal obstacles based on combinations of the L_1 and link metrics [30]. It needs to develop both the single-source and general path query data structures. In addition, Yang, Lee, and Wong's special *segment-dragging* data structure [90] is generalized for the gateway-based procedure of answering path queries. This general path query data structure takes $O(n^2 \log^2 n)$ time and space to build and supports a length query in $O(\log^2 n)$ time.

3.3 Other Related Path Problems

In [15, 16], a simpler case of the problem of processing geometric shortest path queries was studied: Process rectilinear shortest path queries among disjoint *rectangular* rectilinear obstacles. The techniques developed involve a fast parallel computation of *staircase separators* and a parallel scheme for partitioning the boundaries of the obstacles in a way that ensures that the resulting path length matrices have a special *monotonicity property* that was apparently absent before applying the partitioning scheme. In particular, it was presented in [15, 18] optimal parallel algorithms for partitioning a set of m disjoint rectangles into two subsets of size $m/2$ each, by using an obstacle-avoiding staircase, called a *staircase separator* (see Fig. 6 for an example). It was shown that for rectilinear shortest path queries among rectangular obstacles, each query point needs to use at most two gateway points. The data structure was constructed sequentially in $O(n^2)$ time and space and in parallel in $O(\log^2 n)$ time using $O(n^2 / \log n)$ CREW PRAM processors, for supporting an $O(\log n)$ length query time [15, 16].

The staircase separator algorithms have been used, by the author and by others, in solving various geometric paths and Voronoi diagram problems sequentially and in parallel [18, 29, 50, 55, 79].

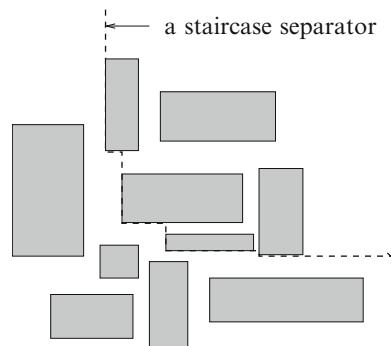


Fig. 6 A staircase separator partitioning a set of rectangular obstacles

Matrices with the special monotonicity property that are used in [15, 16] are called *monotone matrices* or *Monge matrices* [5, 6, 9]. Monotone matrices can be multiplied in the $(\min, +)$ semiring very efficiently, even in parallel. For example, two $n \times n$ monotone matrices can be multiplied in parallel in $O(\log n)$ time and $O(n^2 / \log n)$ CREW PRAM processors [5, 9]. Monotone matrices find numerous applications in many areas. In particular, it has been discovered that monotone matrices appear in many geometric and graph shortest path problems, yielding very efficient algorithms for such path problems [15–17, 19, 20, 64].

4 Approximate Shortest Path Queries

This section discusses a family of efficient algorithmic solutions for processing various length queries on *approximate* geometric shortest obstacle-avoiding paths among polygonal obstacles in the plane. The lengths of the paths considered are determined according to the L_p metric, where p is any positive integer. These algorithms achieve an interesting trade-off between the approximation factor, the query time, and the space and construction time of the data structures. In addition to the gateway paradigm, the algorithms are also based on other important paradigms for approximation of geometric and graph shortest paths.

Note that in a plane with multiple polygonal obstacles, the best-known data structures for even relatively simple cases of exact shortest path queries take super-quadratic space, as shown in the previous section. However, solutions of super-quadratic space can still be too expensive for many practical applications. Moreover, solutions suitable for many practical problems (e.g., in data compression and geographical information systems) often need not use exact shortest paths. In such situations, practitioners may be willing to sacrifice, to a certain degree, the optimality of their solutions to gain algorithmic simplicity and efficiency. Therefore, it makes sense to develop efficient solutions (e.g., of subquadratic space) for queries on approximate geometric shortest paths whose lengths are within a guaranteed small constant factor c of their corresponding exact shortest paths. Such approximate shortest paths will be referred to as *c-short paths* or simply *short paths*. For example, the *exact* shortest paths are 1-short paths.

One of the goals in this section, therefore, is to illustrate some trade-off between the approximation factor, the query time, and the space and construction time of the approximate shortest path query data structures developed in [10, 26].

Very few previous results were known for computing approximate geometric shortest paths in the plane. Notably, Clarkson [41] presented an algorithm for processing Euclidean $(1 + \epsilon)$ -short path queries amid polygonal obstacles, for any positive constant ϵ . Clarkson's algorithm uses $O(n \log n)$ time and $O(n)$ space for building the data structure and answers a short path query in $O(n \log n)$ time. As was mentioned in [41], it is possible to extend Clarkson's result as follows: In $O(n^2 \log n)$ time, an $O(n^2)$ -space data structure can be constructed for supporting a length query in $O(\log n)$ time; the details of this solution

is actually provided in [26]. Mitchell [71, 72] gave a different approach for computing $(1 + \epsilon)$ -short obstacle-avoiding paths in the plane.

4.1 Main Ideas

The main ideas of the approximation solutions are first discussed. Let V denote the set of n obstacle vertices. Let $G_V = (V, E)$ be the visibility graph of the obstacle scene with vertex set V and edge set E such that for any two distinct vertices u and $v \in V$, (u, v) is an edge in E if and only if the line segment \overline{uv} does not intersect the interior of any obstacle. The cost of edge $(u, v) \in E$ is the L_p distance $d(u, v)$ between u and v for a metric L_p . Clearly, $|E| = O(n^2)$ in the worst case. Let $Length(P)$ denote the length of an obstacle-avoiding path P .

It will use the vertex set V of the visibility graph G_V as the set of gateway points for the whole plane. To be able to compute efficiently a c -short path $P_S(p, q)$ between any two query points p and q in the plane, two difficulties need to be overcome:

1. “Discretize” the computation of $P_S(p, q)$, that is, compute $P_S(p, q)$ based on short paths $P_S(u, v)$, for some obstacle vertices u and $v \in V$.
2. Obtain short paths between obstacle vertices quickly.

To handle difficulty (1) in a fast manner (e.g., in polylogarithmic time) for each pair of query points requires the query procedure to examine $P_S(u, v)$ for only a small number of pairs of obstacle vertices u and v . To answer short path queries between arbitrary query points in the plane in polylogarithmic time and with a data structure in subquadratic space and construction time, a problem that one must get around is to avoid using general *ray shooting*. (In contrast, recall that many of the solutions for processing *exact* path queries in the previous section do make use of general ray shooting.) When two query points are visible to each other, the true shortest path between them is found naturally by ray shooting (in polylogarithmic time). However, a data structure that supports general ray-shooting operations in logarithmic time takes $O(n^2)$ space and $O(n^2 \log n)$ time to build [1], or one can use subquadratic space data structures and perform ray-shooting operations in superpolylogarithmic time (e.g., see [1–3]). Neither of those ray-shooting data structures gives a satisfactory solution.

Difficulty (2), in a sense, is the problem of finding a good “spanner.” Given a set S of n points in a d -dimensional space, a t -spanner is a graph G whose vertex set includes the points of S such that for any two points u and v of S , there is a u -to- v path in the graph G of length at most t times the distance between u and v in a chosen L_p metric. The problem of constructing various “good” spanners has attracted a considerable amount of attention recently (e.g., see [13] and the references given there). The “spanner” is not only for a finite set of points (which is the case for most spanner results) but for the entire obstacle-scattered plane. To resolve difficulty (2), it needs an efficient scheme for computing and maintaining short paths between any two obstacle vertices using only subquadratic space and time. Actually, the scheme involved makes use of certain spanners for discrete points.

The scheme for resolving difficulty (2), called *approximation of approximation*, is outlined as follows:

Approximation 1: Use a *planar spanner* for the gateway point set V , to approximate shortest paths between any two gateway points in the plane.

Approximation 2: Develop efficient algorithms and data structures for computing approximate shortest paths in such a spanner graph.

The rest of this section unfolds the solutions for resolving these difficulties.

4.2 Planar Spanners

As will be shown in the next subsection, planar spanners are particularly useful for computing geometric short paths. For the gateway point set V , a *planar spanner* is a graph $G_P = (V_P, E_P)$ such that:

1. The set V of obstacle vertices is a subset of V_P .
2. The edges of G_P represent straight-line segments that do not intersect the interior of any obstacle.
3. For any two obstacle vertices u and $v \in V$, there is a u -to- v path in G_P which corresponds to a c -short obstacle-avoiding path in the plane in an L_p metric;
4. G_P is a planar graph.

If $V_P = V$, then it calls G_P a *planar L_p c-spanner*. Otherwise, if G_P contains additional vertices (called *Steiner vertices*), then it calls G_P a *planar Steiner L_p c-spanner*. The real number $c \geq 1$, representing the approximation factor of short paths, is called the *stretch factor* of the spanner. [Figure 7](#) gives an example of a planar spanner of the visibility graph for a point set in the plane with no obstacle.

There were already several algorithms that construct planar L_2 c -spanners in $O(n \log n)$ time. The best-known stretch factor in L_2 is $c = 2$, which was discovered by Chew [[34, 36](#)].

Ideally, one would like to have planar spanners with a stretch factor of $1 + \epsilon$ for any given positive constant ϵ . So one interesting question is: What kind of planar spanners can achieve a $1 + \epsilon$ stretch factor? The following lemma, proved in [[10](#)], sheds some light on the answer to this question.

Lemma 4 *There exist point sets of size n in the plane such that any planar L_1 spanner that is a subgraph of the visibility graph on such a point set (modeling the L_1 distances among the given points) has a stretch factor ≥ 2 . Furthermore, the stretch factor of 2 is tight, since there is an $O(n \log n)$ -time algorithm for constructing such a planar L_1 2-spanner for the vertices of any polygonal obstacle scene in the plane.*

The $O(n \log n)$ -time algorithm for constructing a planar L_1 2-spanner (without using any Steiner vertices) hinges on computing a *constrained Delaunay triangulation* using a convex distance function [[35–37](#)] based on an isosceles triangle which

Fig. 7 The visibility graph (a) and a planar spanner (b) of five points in the plane

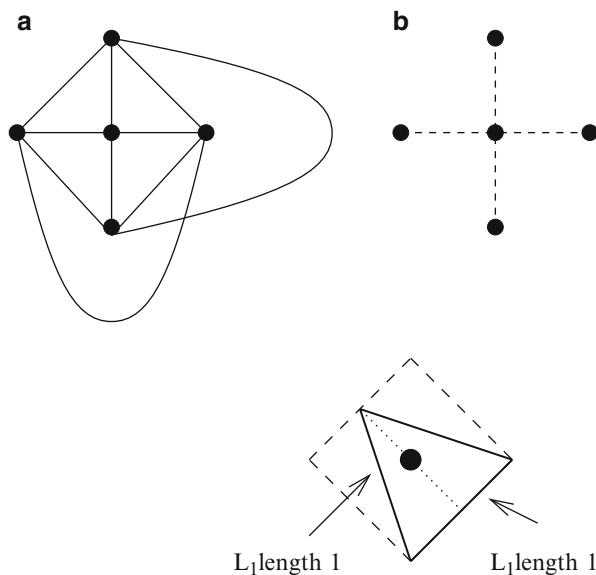
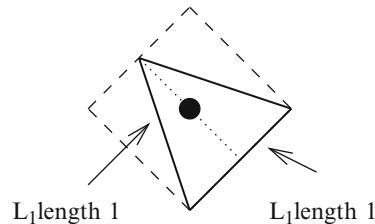


Fig. 8 The convex distance function is based on the isosceles triangle within an L_1 unit diamond



is within an L_1 unit diamond (shown in Fig. 8). This approach is similar to those used in [34–36].

Lemma 4 shows that if one wants a planar L_1 spanner with a stretch factor less than 2, then it is *necessary* to use Steiner vertices. Since any L_p metric is related to the L_1 metric by a fixed constant factor, this conclusion also holds for planar L_p spanners (i.e., Steiner vertices are needed for achieving planar L_p spanners with a $1 + \epsilon$ stretch factor).

The following lemma, proved in [10], shows that planar Steiner L_1 $(1 + \epsilon)$ -spanners exist and can be constructed efficiently.

Lemma 5 *Given a collection of disjoint polygonal obstacles in the plane with n vertices and given any constant $\epsilon > 0$, a planar L_1 $(1 + \epsilon)$ -spanner with $O(n/\epsilon^2)$ Steiner vertices can be constructed in $O(n \log n + n/\epsilon^2)$ time.*

The results of Lemma 5 have been extended to the L_2 metric (the constant factor associated with the number of Steiner vertices used by the resulting planar L_2 spanner depends on a higher-degree polynomial of $1/\epsilon$). It seems possible to even further extend this lemma to any L_p metric.

The $O(n \log n)$ -time algorithm for Lemma 5 is quite involved. It is based on a special *planar subdivision* introduced by Arya et al. [14], which was originally used for solving nearest neighbor searching problems. Interestingly, it is possible to apply this planar subdivision to constructing planar Steiner spanners for obstacle vertices. In [10], it first constructs the planar subdivision of Arya et al. [14] for a polygonal obstacle scene. Doing so in a straightforward manner, however, may result in a planar subdivision with $O(n^2)$ regions. It can be shown that it is possible

to group the $O(n^2)$ regions into only $O(n)$ regions, in $O(n \log n)$ time. Finally, it carefully creates grids on the resulting $O(n)$ regions of the subdivision. This gives a planar Steiner spanner with a $1 + \epsilon$ stretch factor.

In comparison with the planar L_1 and L_2 2-spanners (without any Steiner vertices) in [10, 34, 36], the planar Steiner $(1 + \epsilon)$ -spanners obtained in Lemma 5 have a better stretch factor but at a price of possibly using a large number of Steiner vertices.

Before concluding this subsection, it should be mentioned that all the presented $O(n \log n)$ -time algorithms for constructing planar spanners (with or without using Steiner vertices) are optimal in the algebraic computation tree model [22]. Specifically, the following lower-bound results were proved in [31]:

Lemma 6 *The problem of constructing geometric spanners, possibly containing Steiner points, for sets of points in the d -dimensional space, and the problem of computing approximate shortest paths amid a collection of polygonal obstacles in the plane, for any given approximation factor, all take $\Omega(n \log n)$ time to solve in the algebraic computation tree model.*

4.3 Short Paths in Planar Graphs

This subsection discusses several algorithmic techniques for computing and maintaining approximate shortest paths in weighted undirected planar graphs with nonnegative real edge costs. The algorithms all construct certain data structures for short path queries in planar graphs in subquadratic time and space and achieve an interesting trade-off between the approximation factor, the query time, and the space and construction time of the data structures. These solutions, of course, are immediately applicable to the planar spanners discussed in the previous subsection because such planar spanners are simply instances of weighted undirected planar graphs.

Let $G = (V, E)$ be a weighted undirected planar graph with nonnegative real costs on its edges, and let n denote the number of vertices of G . Note that one cannot simply store the all-pairs shortest paths in G , since that would require $O(n^2)$ space. Hence, different approaches for computing short paths in G must be used.

4.3.1 A General Approximation Paradigm

The first approach for computing short paths in undirected planar graphs hinges crucially on the following key observation which was proved in [26].

Lemma 7 *For two elements p and q in an environment \mathcal{R} , let \mathcal{R}' be a subset of \mathcal{R} such that a shortest path $P(p, q)$ between p and q in \mathcal{R} goes through an element in \mathcal{R}' . For p (resp., q), let p' (resp., q') be an element in \mathcal{R}' such that the length of the shortest path $P(p, p')$ (resp., $P(q, q')$) in \mathcal{R} is the shortest among all the paths between p (resp., q) and the elements*

in \mathcal{R}' ; that is, $\text{Length}(P(p, p')) = \min\{\text{Length}(P(p, w)) \mid w \in \mathcal{R}'\}$ (resp., $\text{Length}(P(q, q')) = \min\{\text{Length}(P(q, w)) \mid w \in \mathcal{R}'\}$). Then, $\text{Length}(P(p, p') \cup P(p', q') \cup P(q', q)) \leq 3 \times \text{Length}(P(p, q))$.

Lemma 7 states that the concatenation of the three shortest paths $P(p, p')$, $P(p', q')$, and $P(q', q)$ is a good approximation of the shortest path $P(p, q)$ (within at most *three times*). This observation is quite general since the environment \mathcal{R} can be geometric or graph-theoretic, so long as the paths involved in the observation are *undirected*. It immediately leads to the following general paradigm for processing short path queries in \mathcal{R} :

1. Find a subset \mathcal{R}' of \mathcal{R} (\mathcal{R}' is called a *separator* of \mathcal{R}), and use \mathcal{R}' to partition \mathcal{R} into connected components $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_k$. Ideally, \mathcal{R}' would give rise to a “balanced” partition of \mathcal{R} .
2. Compute and store shortest paths in \mathcal{R} from each element of \mathcal{R}' to other elements of \mathcal{R}' (i.e., an all-pairs shortest path computation among the elements of \mathcal{R}'). Also, compute and store, for each element p of \mathcal{R} , a shortest path in \mathcal{R} from p to its closest element p' in \mathcal{R}' (i.e., the Voronoi diagram on \mathcal{R} whose sites are the elements of \mathcal{R}'). Then, the shortest path information so prepared is sufficient for reporting an approximate shortest path (within a factor of 3 of the exact shortest path) between any $p \in \mathcal{R}_i$ and $q \in \mathcal{R}_j$, $1 \leq i < j \leq k$.
3. For every $i = 1, 2, \dots, k$, build the data structure recursively for \mathcal{R}_i , until \mathcal{R}_i becomes easily manageable.

In addition to the above general approximation paradigm, **Lemma 7** has several other important implications. For example, the size of a short path query data structure that is based on the above paradigm can be affected greatly by the size of the separator \mathcal{R}' because the all-pairs shortest paths among the elements of \mathcal{R}' may need to be represented explicitly. Therefore, “small-sized” separators such as Lipton and Tarjan’s planar separators [69] are particularly useful in this framework. However, large-sized separators, such as the *staircase separators* [15, 16, 18], can also be appropriate, especially when the all-pairs shortest paths among the elements of such separators can be made available without paying high computational costs. Hence, with this framework, results on various graph-theoretic and geometric separators can have a significant impact on algorithms for computing approximate shortest paths.

Now it is clear that this approximation paradigm is especially applicable to undirected planar graphs, due to Lipton and Tarjan’s *separator theorem* [69]. In fact, the approximation paradigm has led to a number of results on computing short geometric and graph paths in [10, 26].

4.3.2 3-Short Paths in Planar Graphs

When this paradigm is applied to a weighted undirected planar graph G , a two-phase recursive algorithm was developed for constructing a 3-short path query data structure [26]. At each recursion level of the algorithm, a planar separator [69], denoted by S , is computed and is used to partition a region of the graph G into

smaller regions. The Voronoi diagram on the region of G whose sites are the vertices of S is computed, based on Mehlhorn's notion of *Voronoi regions* in graphs [70] and making use of the optimal single-source shortest path algorithm in planar graphs by Klein et al. [66]. The (global) all-pairs shortest paths among the vertices of S are computed in the bottom-up and top-down phases of the algorithm, by repeatedly using Dobosiewicz's all-pairs shortest path algorithm [48]. The algorithm maintains a structure for the recursive graph-partitioning process, called the *partition tree* of G . Queries on 3-short paths between any two vertices of G are answered by first computing their *lowest common ancestor* [61, 85] in the partition tree.

As a result, the algorithm in [26] builds the 3-short path query data structure in $O(n \log n)$ space and $O(n^{3/2} / \sqrt{\log n})$ time, supporting a length query in $O(1)$ time. The data structure construction algorithm is further improved by using Frederickson's *hammock decomposition* technique [52], to $O(n \log n)$ space and $O(n \log n + q^{3/2} / \sqrt{\log q})$ time, where q , with $1 \leq q \leq n$, is the minimum number of faces needed to cover all the vertices of the planar graph.

4.3.3 c -Short Paths in Planar Graphs

The ideas and algorithm for computing 3-short paths in [26] inspired a further study of c -short paths in weighted undirected planar graphs, where $1 \leq c \leq 3$ [10]. In particular, the following results are presented in [10].

Lemma 8 *Given an n -vertex planar graph G and an arbitrary integer ρ with $1 \leq \rho \leq \sqrt{n}$, it is possible to construct a data structure in $O(n^2/\rho)$ time and space such that a length query on the exact shortest path between any two vertices in G can be answered in $O(\rho)$ time.*

Note that the algorithm for Lemma 8 is for computing *exact* shortest paths (i.e., 1-short paths) in G . It is based on the planar separators [69] and Frederickson's notion of r -division [51]. It partitions the planar graph G into a division consisting of a number of regions (depending on the value of ρ) and then computes shortest path trees in the entire graph G , with each of these shortest path trees rooting at a vertex on the boundary of such a region.

The following lemma, proved in [10], is for computing 2-short paths in planar graphs.

Lemma 9 *Let u and v be any two vertices of a planar graph G , and let S be a separator of G . If a shortest u -to- v path in G passes through a vertex in S , then $\min\{\text{Length}(P(u, b_u)) + \text{Length}(P(b_u, v)), \text{Length}(P(v, b_v)) + \text{Length}(P(b_v, u))\} \leq 2 \times \text{Length}(P(u, v))$, where $P(u, v)$ is a shortest u -to- v path in G and b_u , called a closest separator vertex of u on the separator S , is a vertex in S which satisfies $\text{Length}(P(u, b_u)) \leq \text{Length}(P(u, w))$ for any vertex w in S .*

Based on Lemma 9, an algorithm is developed for constructing a data structure for 2-short path queries in planar graphs. The 2-short path data structure construction algorithm, actually, is somewhat similar to the algorithm for the 3-short paths

in the previous subsubsection. One main difference between these two algorithms is that the computation of a Voronoi diagram in a region of G with the separator vertices as the sites is replaced by the computation of single-source shortest path trees in the region rooted at each separator vertex. The 2-short path algorithm achieves the following result.

Lemma 10 *Given an n -vertex planar graph G , it is possible to construct a data structure in $O(n^{3/2})$ time and space such that a length query on a 2-short path between any two vertices in G can be answered in $O(\log n)$ time.*

The query procedure supported by this 2-short path data structure takes $O(\log n)$ time because it needs to perform an $O(1)$ time computation (based on Lemma 9) at each of the $O(\log n)$ levels of the partition tree of G .

4.4 Short Path Queries Amid Obstacles in the Plane

This subsection shows how to put together the results on planar Steiner L_p c -spanners (Sect. 4.2) and on all-pairs short path queries in undirected planar graphs (Sect. 4.3), as well as other useful geometric structures, to solve the problem of processing L_p short path queries amid disjoint polygonal obstacles in the plane (with $p = 1, 2$). Several algorithms and data structures for the problem are given, which differ from each other in the approximation factors of the short paths they obtain and in their complexity bounds. The complexity bounds of these solutions also depend on a given value $\epsilon > 0$.

The L_p short path query data structures are also based on the gateway paradigm (Sect. 2). Recall that the gateway points in this case are simply all the n obstacle vertices. These path query data structures all consist of the following two major components:

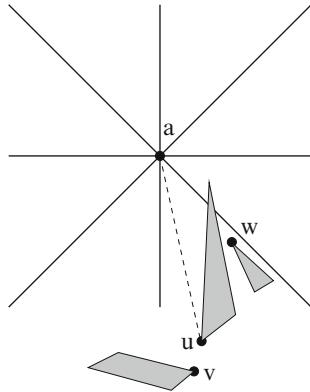
Part I. A data structure for answering queries on all-pairs short paths in a planar Steiner L_p c -spanner as discussed in Sect. 4.2

Part II. A data structure that, given any two query points s and t in the plane, quickly reduces the computation of a short s -to- t path in the plane to the computation of short paths between a constant number of pairs of gateway points (the number of pairs of gateway points involved depends on the value of ϵ)

Given the results in Sects. 4.2 and 4.3, the data structure of Part I can be easily constructed as follows. Let $\epsilon_1 > 0$ be a value depending on the given ϵ (ϵ_1 will be decided later). It first obtains a planar Steiner L_p $(1 + \epsilon_1)$ -spanner with $O(n)$ Steiner vertices, denoted by G_P , and then builds a data structure for answering all-pairs short path queries in the planar graph G_P .

Assume that, in $O(\log n)$ time, one can reduce the computation of a short path query between any two query points in the plane to the computation of short paths between $O(1)$ pairs of gateways (it will be shown later how to perform this $O(\log n)$ -time reduction with the $O(n)$ -space data structure of Part II). Once the

Fig. 9 Clarkson's cone system: to apex a , u is the closest visible obstacle vertex in the cone containing u



data structures of Parts I and II are available, it can, for example, process a short path query between any two query points in the plane in $O(n)$ time. This is done by first reducing such a query to computing $O(1)$ exact shortest paths in G_P and then applying to G_P the $O(n)$ -time single-source shortest path algorithm of [66]. Note that, in this case, the data structure of Part I needs to consist of only the graph G_P itself, and hence, it takes only $O(n)$ space and $O(n \log n)$ time to build. This $O(n)$ -time query-answering algorithm compares favorably with Clarkson's $O(n \log n)$ -time and $O(n)$ -space algorithm for L_2 $(1 + \epsilon)$ -short path queries among polygonal obstacles in the plane [41]. If the data structure of Part I supports queries on c -short paths in G_P (Sects. 4.3.2 and 4.3.3), then the length query time for a short path between any two query points in the plane is $O(\log n + f_c)$, where f_c is the length query time for a c -short path in G_P .

Let $\epsilon_2 > 0$ be a value depending on the given ϵ (ϵ_2 will be decided later). The data structure of Part II takes $O((n \log n)/\epsilon_2)$ time and $O(n/\epsilon_2)$ space to build. This data structure serves the purpose of, given two query points s and t in the plane, computing a c -short s -to- t path from the short paths between $O((1/\epsilon_2)^2)$ pairs of obstacle vertices. This data structure is given in [26] for processing L_2 short path queries in the plane. It is based on a modified version of Clarkson's cone system [41], as sketched below.

For a given value ϵ' , $0 < \epsilon' \leq 1$, Clarkson [41] constructed a set of cones in the plane with a common corner point a , called the *apex*, as follows: Let $\psi = \min\{\pi/12, \epsilon'\pi/2\}$ if $\epsilon' < 1/2$ and $\psi = \epsilon'\pi/6$ if $1/2 \leq \epsilon' \leq 1$; partition the plane into a set \mathcal{F}_ψ^a of cones with the point a as their apex such that the angle of each cone $C_a \in \mathcal{F}_\psi^a$ is smaller than ψ (see Fig. 9 for an example). If no particular apex point is referred to, the set of cones is denoted by \mathcal{F}_ψ . For two points a and b that are visible to each other, with b lying inside a cone $C_a \in \mathcal{F}_\psi^a$, the *approximate L_2 distance* $D_{a,b}^{C_a}$ from a to b is defined by $D_{a,b}^{C_a} = (b - a) \cdot U_{C_a}$, where U_{C_a} is a fixed unit vector contained in C_a . Based on this cone system, Clarkson built a sparse subgraph $G_{\epsilon'}$ of the visibility graph G_V in a plane with polygonal obstacles such that for any two obstacle vertices a and u , (a, u) is an edge of $G_{\epsilon'}$ if u is visible from a and there is

a cone $C_a \in \mathcal{F}_\psi^a$ containing u with $D_{a,u}^{C_a}$ being the minimum (approximate) distance among all the obstacle vertices in $C_a - \{a\}$ that are visible from a (see Fig. 9 for an example). $G_{\epsilon'}$ hence consists of all the vertices of G_V and of $O(n/\epsilon')$ edges and can be obtained in $O((n \log n)/\epsilon')$ time by using certain variants of Voronoi diagrams in the plane, called *conical Voronoi diagrams*, whose sites are the vertices of G_V [41]. By using $G_{\epsilon'}$ and an $O(n/\epsilon')$ -space data structure for a set of $O(1/\epsilon')$ conical Voronoi diagrams, a $(1 + \epsilon')$ -short path between any two points in the plane can be reported in $O(n \log n + n/\epsilon')$ time [41]. The geometric short path query data structures here include a modified version of Clarkson's cone system and several additional components, such as those for *oriented ray shooting* and for planar point location used in [26]. One of the needed modifications to Clarkson's cone system is to carefully choose a set of unit vectors for computing the approximate L_p distances in each cone (the choices of these unit vectors ensure the correctness of the query procedure [26]).

The data structure for Part II enables one to perform the following computation in $O((\log n)/\epsilon_2)$ time: Given any point s in the plane, compute a set P_s of $O(1/\epsilon_2)$ gateway points such that P_s is a subset of the obstacle vertices and such that each $v \in P_s$ ($v \neq s$) is the closest point among all the obstacle vertices in a cone $C_s \in \mathcal{F}_\psi^s$ (based on the approximate L_p distance defined by a unit vector for C_s) that are visible from s .

Given Parts I and II of the data structures, it can answer the length query for a geometric short path $P_S(s, t)$ between any two query points s and t as follows:

1. Compute the gateway point sets P_s and P_t using the set of $O(1/\epsilon_2)$ conical Voronoi diagrams.
2. For each of the $O((1/\epsilon_2)^2)$ pairs of gateway points $u \in P_s$ and $v \in P_t$, compute the length of a short u -to- v path in the planar spanner G_P (with the desired stretch factor).
3. Find the shortest length among all the short obstacle-avoiding paths from s to t via u and v so obtained. This gives the length of the short path $P_S(s, t)$.

The time complexity of this query procedure is $O((\log n)/\epsilon_2)$ (for step 1) plus $O((1/\epsilon_2)^2)$ times the time for a short path length query in the planar graph G_P (for step 2).

To complete the discussion of the geometric short path query data structures, it must be decided the values of ϵ_1 and ϵ_2 based on ϵ . Note that because G_P is a planar Steiner L_p $(1 + \epsilon_1)$ -spanner with $O(n)$ Steiner vertices and because the stretch factors of the short paths in G_P that the all-pairs short path query data structures report are $k \in \{1, 2, 3\}$, a short path in G_P so computed corresponds to a short obstacle-avoiding path in the plane with a stretch factor of $k(1 + \epsilon_1) = k + k\epsilon_1$. Also, note that another approximation error is induced by the above geometric short path query procedure and this error depends on ϵ_2 . The stretch factor of the short obstacle-avoiding paths thus computed, therefore, can be bounded by $k + k(\epsilon_1 + \epsilon_2)$, and one would like to have $k + k(\epsilon_1 + \epsilon_2) \leq k + \epsilon$. Choosing $\epsilon_1 = \epsilon_2 = \epsilon/(2k)$ will be sufficient, and this only increases the time and space complexity bounds of the geometric short path query data structures by a constant factor.

Table 1 The results for the geometric short path query data structures in the L_1 and L_2 metrics. Here, ϵ is an arbitrarily small positive constant, ρ is an arbitrary integer such that $1 \leq \rho \leq \sqrt{n}$, and q is the minimum number of faces needed to cover all the vertices of the planar spanner used by a data structure and $1 \leq q \leq n$

Stretch factor	Construction time	Space	Query time
$1 + \epsilon$	$O(n^2/\rho)$	$O(n^2/\rho)$	$O(\log n + \rho)$
$1 + \epsilon$	$O(n \log n)$	$O(n)$	$O(n)$
$2 + \epsilon$	$O(n^{3/2})$	$O(n^{3/2})$	$O(\log n)$
$3 + \epsilon$	$O(n \log n + q^{3/2}/\sqrt{\log q})$	$O(n \log n)$	$O(\log n)$

The results of the geometric short path query data structures presented above are summarized in [Table 1](#).

4.5 A Special Case

As mentioned in [Sect. 4.3.1](#), not only small-sized separators (e.g., [69]) are useful in the approximation paradigm but also large sized graph-theoretic and geometric separators can be effective. For example, Mitra and Bhattacharya [79] and Chen and Klenk [29] have used the *staircase separators*, which was introduced for parallel geometric shortest path algorithms and for other geometric path problems in [15, 16, 18], in designing data structures for approximate rectilinear shortest path queries among rectilinear rectangular obstacles. Mitra and Bhattacharya's data structure in [79] takes $O(n \log^2 n)$ space and $O(n \log^3 n)$ construction time and supports an $O(\log^2 n)$ time on length queries for 7-short paths. An improved data structure was given that takes $O(n \log n)$ space and $O(n \log^2 n)$ construction time and supports an $O(\log n)$ -time length query for 3-short paths [29]. Observe that an obstacle-avoiding staircase separator S (e.g., see [Fig. 6](#)), although can be of size $O(n)$, has the nice property that for any two points on S , there is a rectilinear shortest obstacle-avoiding path connecting the two points along S . Hence, the data structures for approximate shortest path queries in [29, 79] need not explicitly compute and store the all-pairs shortest paths among points on any such staircase separator, resulting in a significant saving in their space and construction time. It was also shown in [29] that in this special case, each query point needs to use at most two gateway points.

5 Some Other Geometric Shortest Path Query Results

This section sketches some other results for geometric shortest path query problems in the plane and in the 3-D space. Unless otherwise specified, we assume that the distance metric used in this section is Euclidean.

If the planar geometric environment contains only one simple polygon of n vertices, then a data structure can be constructed optimally in $O(n)$ time and space, which answers shortest path queries in $O(\log n)$ time each [56, 57].

In a planar geometric environment S that contains h (in the worst case, $h = O(n)$) polygonal obstacles of totally n vertices, the general shortest path query problem becomes much more difficult. While the single-source shortest path queries in S can be handled by a data structure which takes $O(n \log n)$ time to build and processes each query in $O(\log n)$ time [63], the general (two-point) query version has much higher time and space complexities. Known solutions for general two-point queries in S often reduce the problem to a higher-dimensional point location problem [88]. Chiang and Mitchell [38] presented several data structures for the general two-point query problem in this setting, including a data structure built in $O(n^{11})$ (resp., $O(n^{10} \log n)$) time and space for answering any query in $O(\log n)$ (resp., $O(\log^2 n)$) time and a data structure built in $O(n^{5+\epsilon})$ time and space with a sublinear query time, where $\epsilon > 0$ is any small constant. But, these time and space bounds are usually very high for practical usage. Even for some special cases on the planar geometric environment S with multiple polygonal obstacles, solutions for general two-point queries are still very expensive. For example, when both query points are constrained to lie on the boundaries of the obstacles in S , it takes $\tilde{O}(n^5)$ (resp., $O(n^{3+\epsilon})$) time and space to build a data structure with an $O(\log n)$ (resp., sublinear) query time, where the notation \tilde{O} hides a polylogarithmic factor, by Bae and Okamoto [21].

Guo et al. [58] developed a data structure built in $O(n^2 \log n)$ time and $O(n^2)$ space with an $O(h \log n)$ query time. While the time and space bounds in [58] are (nearly) quadratic, the $O(h \log n)$ query time is still $O(n \log n)$ in the worst case, no better than the time bound for computing single-source shortest paths in S [63].

The shortest path query problem on 3-D polyhedral surfaces has also been studied. For two-point queries on a *convex* polytope of n vertices, Agarwal et al. [4] presented a data structure built in $O(n^6 m^{1+\epsilon})$ time and space with an $O(n^{1/2}/m^{1/4})$ query time, for any fixed $1 \leq m \leq n^2$ and any constant $\epsilon > 0$; when the query points are all on the edges of the convex polytope, the time and space bounds of the data structure can be reduced by a factor of n [4]. Recently, Cook and Wenk [44] improved the solutions in [4] (reducing the bounds by a factor of n) based on a kinetic Voronoi diagram approach. For single-source queries on a (possibly non-convex) polyhedral surface, a data structure can be built in $O(n^2)$ time and space, by Chen and Han [33] (an improvement on the solution in [76]), with an $O(\log n)$ query time. If the polyhedral surface is *convex*, then a data structure can be constructed in $O(n \log n)$ time and space, by Schreiber and Sharir [86]. There are also some interesting efficient techniques for processing *approximate* shortest path queries on 3-D polyhedral surfaces (e.g., see [8, 59, 60]).

6 Conclusion

This chapter has discussed several newly developed algorithmic paradigms for processing geometric shortest path queries and related problems. It has shown how these general paradigms lead to efficient techniques for designing algorithms and

data structures for processing a variety of queries on exact and approximate shortest paths in a number of geometric and graph-theoretic settings.

There are many exciting and important problems on computing various geometric shortest paths that are yet to be solved. In [27], a number of research issues were raised on geometric optimal path planning. In addition to the research issues discussed in [27], the following open problems and research directions are likely to receive considerable attention in the future:

- In [10], it was shown that for the L_1 metric, it is possible to achieve a planar spanner (without any Steiner vertices) with a stretch factor of 2, and this stretch factor is tight. However, what about other L_p metrics for $p > 1$? For example, it is already known how to construct planar spanners (without any Steiner vertices) for the L_2 metric with a stretch factor of 2 [34, 36]. Can this stretch factor be improved for the L_2 metric? What is the tight lower bound for this stretch factor?
- The visibility-sensitive approach for processing queries on exact geometric shortest paths (Sect. 3.1) relates the query time with the visibility polygons of the query points, whose sizes can still be big. Can one reduce this query time for exact geometric shortest paths in the L_2 metric while still being able to use a data structure with a near-quadratic space and construction time?
- The approaches for processing queries on approximate geometric shortest paths (Sect. 4) crucially hinge on efficient schemes for computing approximate shortest paths in *planar spanners* of the obstacle vertices. There are many other geometric spanners that have a variety of useful properties (e.g., see [13, 41]). Can one develop efficient schemes for computing approximate shortest paths in these geometric spanners?
- The geometric shortest path query algorithms in this chapter all assume that the plane is perfectly “flat.” However, path-planning problems in practical situations are likely to occur in non-flat planes or even terrains. Distances in such non-flat geometric settings are usually more than the standard L_p distances. For example, in [7], it defines a distance (called *skew distance*) in a non-flat plane as the Euclidean distance plus a signed difference in height between two points. The skew distance is a natural generalization of the Euclidean distance. Efficient algorithms for processing shortest path queries in non-flat geometric settings with such generalized distance functions need to be developed.
- It is well known that computing geometric shortest paths in the 3-D space with polyhedral obstacles is NP-hard [23]. Several efficient approximation algorithms have been discovered for computing 3-D shortest paths [40, 41, 81]. However, there is so far no efficient approximation algorithm known for processing 3-D shortest path *queries* among polyhedral obstacles, even for the L_1 metric.
- All geometric shortest path query algorithms in this chapter assume that the geometric settings are *static*. However, in real applications, the geometric environments are likely to be *dynamic* (e.g., obstacles are moving, the weight factors of the regions are changing). Shortest path problems in dynamic geometric environments are much harder than the static versions. So far, no efficient algorithm is known for processing shortest path queries in dynamic geometric environments.

- The known algorithms for general two-point Euclidean shortest path queries in a planar geometric environment with multiple polygonal obstacles (e.g., [21, 38]) and on 3-D polyhedral surfaces (e.g., [4, 38, 44]) tend to have high time and space complexities. Reducing the time and space bounds of these solutions is important to practical applications.

Acknowledgements The work of the author was supported in part by the National Science Foundation under Grant CCF-0916606 and Grant CCF-1217906.

Recommended Reading

- P.K. Agarwal, Ray shooting and other applications of spanning trees with low stabbing number. SIAM J. Comput. **21**(3), 540–570 (1992)
- P.K. Agarwal, J. Matousek, Ray shooting and parametric search. SIAM J. Comput. **22**(4), 794–806 (1993)
- P.K. Agarwal, M. Sharir, Applications of a new space partitioning technique. Discret. Comput. Geom. **9**(1), 11–38 (1993)
- P.K. Agarwal, B. Aronov, J. O'Rourke, C.A. Schevon, Star unfolding of a polytope with applications. SIAM J. Comput. **26**(6), 1689–1713 (1997)
- A. Aggarwal, J. Park, Notes on searching in multidimensional monotone arrays, in *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, New York, 1988, pp. 497–512
- A. Aggarwal, M.M. Klawe, S. Moran, P. Shor, R. Wilber, Geometric applications of a matrix searching algorithm. Algorithmica **2**, 209–233 (1987)
- O. Aichholzer, F. Aurenhammer, D.Z. Chen, D.T. Lee, E. Papadopoulou, Skew Voronoi Diagrams. Int. J. Comput. Geom. Appl. **9**(3), 235–247 (1999)
- L. Aleksandrov, H. Djidjev, H. Guo, A. Maheshwari, D. Nussbaum, J.-R. Sack, Approximate shortest path queries on weighted polyhedral surfaces, in *Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science*, Stará Lesná, 2006, pp. 98–109
- A. Apostolico, M.J. Atallah, L. Larimore, H.S. McFaddin, Efficient parallel algorithms for string editing and related problems. SIAM J. Comput. **19**(5), 968–988 (1990)
- S. Arikati, D.Z. Chen, L.P. Chew, G. Das, M. Smid, C.D. Zaroliagis, Planar spanners and approximate shortest path queries among obstacles in the plane, in *Proceedings of the 4th Annual European Symposium on Algorithms*, Barcelona, Spain, 1996, pp. 514–528
- E.M. Arkin, R. Connelly, J.S.B. Mitchell, On monotone paths among obstacles, with applications to planning assemblies, in *Proceedings of the 5th Annual ACM Symposium on Computational Geometry*, Saarbrücken, 1989, pp. 334–343
- E.M. Arkin, J.S.B. Mitchell, S. Suri, Optimal link path queries in a simple polygon, in *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, Orlando, 1992, pp. 269–279
- S. Arya, G. Das, D.M. Mount, J.S. Salowe, M. Smid, Euclidean spanners: short, thin, and lanky, in *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, Las Vegas, 1995, pp. 489–498
- S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A. Wu, An optimal algorithm for approximate nearest neighbor searching. J. ACM **45**(6), 891–923 (1998)
- M.J. Atallah, D.Z. Chen, Parallel rectilinear shortest paths with rectangular obstacles. Comput. Geom. **1**(2), 79–113 (1991)
- M.J. Atallah, D.Z. Chen, On parallel rectilinear obstacle-avoiding paths. Comput. Geom. **3**(6), 307–313 (1993)
- M.J. Atallah, D.Z. Chen, Computing the all-pairs longest chains in the plane. Int. J. Comput. Geom. Appl. **5**(3), 257–271 (1995)

18. M.J. Atallah, D.Z. Chen, Applications of a numbering scheme for polygonal obstacles in the plane, in *Proceedings of the 7th Annual International Symposium on Algorithms and Computation*, Osaka, Japan, 1996, pp. 1–24
19. M.J. Atallah, D.Z. Chen, K.S. Klenk, Parallel algorithms for longest increasing chains in the plane and related problems. *Parallel Process. Lett.* **9**(4), 511–520 (1999)
20. M.J. Atallah, D.Z. Chen, O. Daescu, Efficient parallel algorithms for planar *st*-graphs. *Algorithmica* **35**(3), 194–215 (2003)
21. S.W. Bae, Y. Okamoto, Querying two boundary points for shortest paths in a polygonal domain, in *Proceedings of the 20th Annual International Symposium on Algorithms and Computation*, Honolulu, USA, 2009, pp. 1054–1063
22. M. Ben-Or, Lower bounds for algebraic computation trees, in *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, Boston, 1983, pp. 80–86
23. J. Canny, J.H. Reif, New lower bound techniques for robot motion planning problems, in *Proceedings of the 28th IEEE Annual Symposium on Foundations of Computer Science*, Los Angeles, 1987, pp. 49–60
24. B. Chazelle, L.J. Guibas, Fractional cascading: I. A data structuring technique. *Algorithmica* **1**(2), 133–162 (1986)
25. B. Chazelle, L.J. Guibas, Fractional cascading: II. Applications. *Algorithmica* **1**(2), 163–191 (1986)
26. D.Z. Chen, On the all-pairs Euclidean short path problem, in *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, 1995, pp. 292–301
27. D.Z. Chen, Developing algorithms and software for geometric path planning problems. *ACM Comput. Surv.* **28**(4es) (1996) Article 18, <http://www.acm.org/pubs/citations/journals/surveys/1996-28-4es/a18-chen/>
28. D.Z. Chen, Efficient algorithms for geometric shortest path query problems, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos, vol. 2 (Kluwer Academic, Boston, 1998), pp. 1–33
29. D.Z. Chen, K.S. Klenk, Rectilinear short path queries among rectangular obstacles. *Inf. Process. Lett.* **57**(6), 313–319 (1996)
30. D.Z. Chen, O. Daescu, K.S. Klenk, On geometric path query problems. *Int. J. Comput. Geom. Appl.* **11**(6), 617–645 (2001)
31. D.Z. Chen, G. Das, M. Smid, Lower bounds for computing geometric spanners and approximate shortest paths. *Discret. Appl. Math.* **110**(2–3), 151–167 (2001)
32. D.Z. Chen, K.S. Klenk, H.-Y.T. Tu, Shortest path queries among weighted obstacles in the rectilinear plane. *SIAM J. Comput.* **29**(4), 1223–1246 (2000)
33. J. Chen, Y. Han, Shortest paths on a polyhedron. *Int. J. Comput. Geom. Appl.* **6**(2) 127–144 (1996)
34. L.P. Chew, Planar graphs and sparse graphs for efficient motion planning in the plane. Computer Science Tech. Report, PCS-TR90-146, Dartmouth College, 1987
35. L.P. Chew, Constrained Delaunay triangulations. *Algorithmica* **4**(1), 97–108 (1989)
36. L.P. Chew, There are planar graphs almost as good as the complete graph. *J. Comput. Syst. Sci.* **39**(2), 205–219 (1989)
37. L.P. Chew, R.L. Drysdale, Voronoi diagrams based on convex distance functions, in *Proceedings of the 1st Annual ACM Symposium on Computational Geometry*, Baltimore, 1985, pp. 35–244
38. Y.-J. Chiang, J.S.B. Mitchell, Two-point Euclidean shortest path queries in the plane, in *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, 1999, pp. 215–224
39. Y.-J. Chiang, F.P. Preparata, R. Tamassia, A unified approach to dynamic point location, ray shooting, and shortest paths in planar maps. *SIAM J. Comput.* **25**(1), 207–233 (1996)
40. J. Choi, J. Sellin, C.-K. Yap, Approximate Euclidean shortest path in 3-space. *Int. J. Comput. Geom. Appl.* **7**(4), 271–295 (1997)
41. K.L. Clarkson, Approximation algorithms for shortest path motion planning, in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, New York, 1987, pp. 56–65

42. K.L. Clarkson, S. Kapoor, P.M. Vaidya, Rectilinear shortest paths through polygonal obstacles in $O(n(\log n)^2)$ time, in *Proceedings of the 3rd Annual ACM Symposium on Computational Geometry*, Waterloo, 1987, pp. 251–257.
43. K.L. Clarkson, S. Kapoor, P.M. Vaidya, Rectilinear shortest paths through polygonal obstacles in $O(n \log^{3/2} n)$ time, manuscript
44. A.F. Cook, C. Wenk, Shortest path problems on a polyhedral surface, in *Proceedings of the 11th International Symposium on Algorithms and Data Structures*, Banff, 2009, pp. 156–167
45. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd edn. (McGraw-Hill, New York, 2001)
46. M. de Berg, On rectilinear link distance. *Comput. Geom.* **1**, 13–34 (1991)
47. M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 3rd edn. (Springer, Berlin, 2008)
48. W. Dobosiewicz, A more efficient algorithm for the min-plus multiplication. *Int. J. Comput. Math.* **32**(1), 49–60 (1990)
49. H. Edelsbrunner, *Algorithms in Combinatorial Geometry* (Springer, Heidelberg, Germany, 1987)
50. H. ElGindy, P. Mitra, Orthogonal shortest route queries among axes parallel rectangular obstacles. *Int. J. Comput. Geom. Appl.* **4**(1), 3–24 (1994)
51. G.N. Frederickson, Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.* **16**(6), 1004–1022 (1987)
52. G.N. Frederickson, Planar graph decomposition and all pairs shortest paths. *J. ACM* **38**(1), 162–204 (1991)
53. M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM* **34**(3), 596–615 (1987)
54. M.T. Goodrich, R. Tamassia, Dynamic ray shooting and shortest paths via balanced geodesic triangulations. *J. Algorithms* **23**(1), 51–73 (1997)
55. S. Guha, I. Suzuki, Proximity problems for points on a rectilinear plane with rectangular obstacles. *Algorithmica* **17**(3), 281–307 (1997)
56. L.J. Guibas, J. Hershberger, Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.* **39**(2), 126–152 (1989)
57. L.J. Guibas, J. Hershberger, D. Leven, M. Sharir, R.E. Tarjan, Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica* **2**, 209–233 (1987)
58. H. Guo, A. Maheshwari, J.-R. Sack, Shortest path queries in polygonal domains, in *4th International Conference on Algorithmic Aspects in Information and Management*, Shanghai, 2008, pp. 200–211
59. S. Har-Peled, Approximate shortest paths and geodesic diameter on a convex polytope in three dimensions. *Discret. Comput. Geom.* **21**(2), 217–231 (1999)
60. S. Har-Peled, Constructing approximate shortest path maps in three dimensions. *SIAM J. Comput.* **28**(4), 1182–1197 (1999)
61. D. Harel, R.E. Tarjan, Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.* **13**(2), 338–355 (1984)
62. J. Hershberger, A new data structure for shortest path queries in a simple polygon. *Inf. Process. Lett.* **38**(5), 231–235 (1991)
63. J. Hershberger, S. Suri, An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.* **28**(6), 2215–2256 (1999)
64. X. Hu, D.Z. Chen, R. Sambandam, Efficient list-approximation techniques for floorplan area minimization. *ACM Trans. Design Autom. Electron. Syst.* **6**(3), 372–400 (2001)
65. M. Iwai, H. Suzuki, T. Nishizeki, Shortest path algorithm in the plane with rectilinear polygonal obstacles (in Japanese), in *Proceedings of the SIGAL Workshop*, Ryukoku University, Japan, 1994
66. P.N. Klein, S. Rao, M. Rauch, S. Subramanian, Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.* **55**(1), 3–23 (1997)

67. D.T. Lee, C.D. Yang, T.H. Chen, Shortest rectilinear paths among weighted obstacles. *Int. J. Comput. Geom. Appl.* **1**(2), 109–124 (1991)
68. D.T. Lee, C.D. Yang, C.K. Wong, Rectilinear paths among rectilinear obstacles. *Discret. Appl. Math.* **70**(3), 185–215 (1996)
69. R.J. Lipton, R.E. Tarjan, A separator theorem for planar graphs. *SIAM J. Appl. Math.* **36**(2), 177–189 (1979)
70. K. Mehlhorn, A faster approximation algorithm for the Steiner problem in graphs. *Inf. Process. Lett.* **27**(3), 125–128 (1988)
71. J.S.B. Mitchell, An optimal algorithm for shortest rectilinear paths among obstacles, in *Proceedings of the 1st Canadian Conference on Computational Geometry*, Montreal, Canada, 1989
72. J.S.B. Mitchell, L_1 shortest paths among polygonal obstacles in the plane. *Algorithmica* **8**(1), 55–88 (1992)
73. J.S.B. Mitchell, Shortest paths among obstacles in the plane. *Int. J. Comput. Geom. Appl.* **6**3, 309–332 (1996)
74. J.S.B. Mitchell, Geometric shortest paths and network optimization, in *Handbook of Computational Geometry*, ed. by J.-R. Sack, J. Urrutia (Elsevier Science, Amsterdam, 1998), pp. 635–701
75. J.S.B. Mitchell, Shortest paths and networks, in *Handbook of Discrete and Computational Geometry*, 2nd edn. ed. by J.E. Goodman, J. O'Rourke (CRC, Boca Raton, 2004), pp. 607–641
76. J.S.B. Mitchell, D.M. Mount, and C.H. Papadimitriou, The discrete geodesic problem. *SIAM J. Comput.* **16**(4), 647–668 (1987)
77. J.S.B. Mitchell, S. Suri, Geometric algorithms, in *Network Models*, ed. by M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser. *Handbook of Operations Research/Management Science* (Elsevier, Amsterdam, 1995), pp. 425–479
78. J.S.B. Mitchell, G. Rote, G. Woeginger, Minimum-link paths among obstacles in the plane. *Algorithmica* **8**(5–6), 431–459 (1992)
79. P. Mitra, B. Bhattacharya, Efficient approximation shortest-path queries among isothetic rectangular obstacles, in *Proceedings of the 3rd Workshop on Algorithms and Data Structures*, Montréal, 1993, pp. 518–529
80. K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms* (Prentice Hall, New York, 1993)
81. C.H. Papadimitriou, An algorithm for shortest-path motion in three dimensions. *Inf. Process. Lett.* **20**(5), 259–263 (1985)
82. M. Pocchiola, G. Vegter, Topologically sweeping visibility complexes via pseudotriangulations. *Discret. Comput. Geom.* **16**(4), 419–453 (1996)
83. M. Pocchiola, G. Vegter, The visibility complex. *Int. J. Comput. Geom. Appl.* **6**(3), 279–308 (1996)
84. F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction* (Springer, Berlin, 1985)
85. B. Schieber, U. Vishkin, On finding lowest common ancestors: simplification and parallelization. *SIAM J. Comput.* **17**(6), 1253–1262 (1988)
86. Y. Schreiber, M. Sharir, An optimal-time algorithm for shortest paths on a convex polytope in three dimensions. *Discret. Comput. Geom.* **39**(1–3), 500–579 (2008)
87. S. Schuierer, An optimal data structure for shortest rectilinear path queries in a simple rectilinear polygon. *Int. J. Comput. Geom. Appl.* **6**(2), 205–225 (1996)
88. M. Sharir, P.K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications* (Cambridge University Press, New York, 1995)
89. P. Widmayer, Y.F. Wu, C.K. Wong, On some distance problems in fixed orientations. *SIAM J. Comput.* **16**(4), 728–746 (1987)
90. C.D. Yang, D.T. Lee, C.K. Wong, Rectilinear path problems among rectilinear obstacles revisited. *SIAM J. Comput.* **24**(3), 457–472 (1995)

Energy Efficiency in Wireless Networks

Hongwei Du, Xiuzhen Cheng and Deying Li

Contents

1	Introduction	1156
2	Minimum Energy with Adjustable Power	1157
2.1	Symmetric Topological Control	1157
2.2	Asymmetric Topological Control	1162
2.3	Broadcast	1168
2.4	Asymmetric Power Requirement	1176
2.5	Unicast	1178
3	Minimum Energy with Unadjustable Power	1179
3.1	Operations in Multiradio Multichannel Wireless Networks	1179
3.2	Energy-Efficient Virtual Backbone	1182
3.3	Connected Sensor Cover	1183
4	Maximum Lifetime	1184
4.1	Sensor Coverage	1184
4.2	Garg-Könemann Algorithm	1188
4.3	Coverage Breach	1190
4.4	Maximum Lifetime Routing	1191
5	Conclusion	1191
	Cross-References	1191
	Recommended Reading	1191

H. Du (✉)

Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, People's Republic of China

e-mail: hwdu@hitsz.edu.cn

X. Cheng

Department of Computer Science, George Washington University, Washington, D.C., USA
e-mail: cheng@gwu.edu

D. Li

Department of Computer Science, Renmin University, Beijing, People's Republic of China
e-mail: deyingli@ruc.edu.cn

Abstract

Since batteries are power supply for devices in wireless networks, energy efficiency is an important issue. There are many combinatorial optimization problems in the study of wireless networks. They can be roughly divided into three classes, minimum total power with adjustable-power nodes, minimum total power with unadjustable-power nodes, and maximum lifetime with unadjustable-power nodes. In this survey, some classical problems are introduced together with their solutions in each class.

1 Introduction

Recently, *New York Times* [1] reported the following:

Mr. Obama is also asking Congress to make a one-time investment of \$5 billion to bring wireless coverage to rural areas, and is proposing to spend \$3 billion of the spectrum proceeds on research and development into new wireless technologies. And the president is calling for a \$10.7 billion commitment to support what the administration describes as a nationwide wireless broadband network for public safety.

This means that a high-speed next-generation wireless network is going to be constructed and the investigation and technology developments in wireless communication will get in another hot season.

There are many types of wireless networks. They all have advantage that it can be accessed at anywhere and anytime due to their mobility. They all involve many optimization problems in their developments. In particular, the wireless ad hoc network and the wireless sensor network usually consist of a collection of mobile hosts with limited resource supply and hence they have many optimization problems on resource distribution and management. *In this chapter, an overview on those problems is given. These problems will be classified based on their optimization objectives, and the relationship between them will be further introduced.*

There are two types of energy-saving techniques in the literature. In the first type, the transmission power level at each node is adjusted to minimize the total energy consumption [2–4]. In the second type, the transmission power level at each node is fixed, wireless nodes are scheduled into sleep or active state in order to minimize the total energy consumption [5–7] or to maximize the lifetime of network system [8–10]. Therefore, one can classify those optimization problems into three classes, total energy consumption minimization with adjustable power level ([Sect. 2](#)), total energy consumption minimization with fixed power level ([Sect. 3](#)), and lifetime maximization with fixed power level ([Sect. 4](#)). There is a very interesting relationship between [Sects. 3](#) and [4](#).

2 Minimum Energy with Adjustable Power

The energy of wireless devices is often supplied with batteries, which means the energy supply is usually limited. Due to this fact, the energy efficiency becomes an important issue in the study of wireless networks. The communication range of wireless station (node) is closely related to its energy consumption and antenna type. For omni-antenna at a station s , the signal power received at a location t is decreasing as the distance $d(s, t)$ is increasing. Suppose at s , the signal power is P_s . Then at location t , it is $\frac{P_s}{d(s,t)^\alpha}$ where α is a constant usually between 2 and 5 [11]. Suppose c is the quality threshold for the signal power, that is, in order to receive the signal correctly, the signal power has to be at least c . This means that for location t to receive the signal correctly, it must have $\frac{P_s}{d(s,t)^\alpha} \geq c$, that is,

$$P_s \geq c \cdot d(s, t)^\alpha.$$

Thus, the communication range at each station is a disk with radius r where r is determined by the power p through formula

$$p = cr^\alpha.$$

Often, the transmission quality threshold c is normalized to 1.

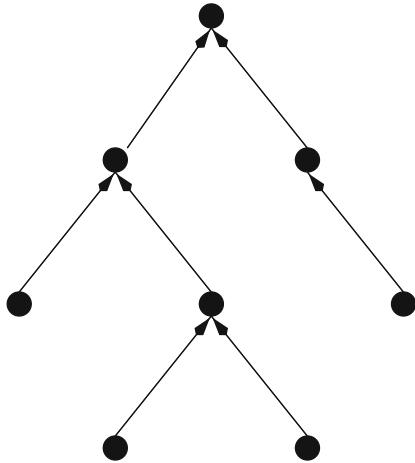
When the power p at each node is adjustable, one often studies the problem of minimizing the total energy consumption under certain network connection constraints in order to meet the requirement for certain duty. There exist many combinatorial optimization problems of this type in the literature. The following is a general mathematical formulation.

Consider a complete graph (or directed graph) $G = (V, E)$ with a distance table. For each $(u, v) \in E$, denote $w(u, v) = d(u, v)^\alpha$, the power requirement for arc (u, v) to be connected, that is, arc (u, v) is connected if and only if vertex u is assigned with a power $p(u) \geq w(u, v)$. The problem is to find a power assignment $p : V \rightarrow R^+$ such that the subgraph with vertex set V and edge set (or arc set) $\{(u, v) \mid p(u) \geq w(u, v), p(v) \geq w(v, u)\}$ (or $\{(u, v) \mid p(u) \geq w(u, v)\}$) satisfies certain property and total power $\sum_{u \in V} p(v)$ is minimized.

In this section, some examples will be presented. An interesting common aspect in those examples is that the minimum spanning tree plays an important role in design of good approximation algorithms for them.

2.1 Symmetric Topological Control

To keep nodes in a wireless network being able to communicate each other, the network is required to contain a spanning tree (in symmetric case) or a strongly connected spanning subgraph (in asymmetric case).

Fig. 1 An in-arborescence

In symmetric case, an (undirected) edge exists between two nodes u and v if and only if the communication range of u covers v and the communication range of v covers u . The topological control problem in this case can be stated as follows.

MIN-POWER SYMMETRIC CONNECTIVITY: Given a wireless network with power-adjustable nodes, find a power assignment to minimize the total power and to keep the network connected symmetrically.

The NP-hardness of this problem was first shown by Blough et al. [12]. To study its approximation solutions, consider a spanning tree T . The total required power for keeping the existence of T is

$$P(T) = \sum_{u \in V(T)} \max_{(u,v) \in E(T)} w(u, v)$$

where $V(T)$ is the node set of T and $E(T)$ is the edge set of T . Choose an arbitrary node as the root r . For every node u other than r , let e_u denote the edge incident to u on the path from u to r . Then e_u is unique and all e_u for $u \in V(T)$ form an in-arborescence A as shown in Fig. 1 and $P(T) \geq P(A)$.

Lemma 1 (In-Arborescence Lemma) *For any in-arborescence A , let T be the spanning tree obtained from removing all edge directions in A . Then,*

$$P(A) = w(T)$$

where $w(T) = \sum_{e \in E(T)} w(e)$ is the total weight of spanning tree T .

Proof

$$P(A) = \sum_{(v,u) \in E(A)} w(v, u) = w(T).$$

□

By In-Arborescence Lemma, it can be observed that $P(T) \geq \cdot w(T)$. On the other hand,

$$P(T) = \sum_{u \in V(T)} \max_{(u,v) \in E(T)} w(u, v) \leq \sum_{u \in V(T)} \sum_{v:(u,v) \in E(T)} w(u, v) = 2 \cdot w(T).$$

Suppose T^* is the minimum-weight spanning tree with edge weight $w(\cdot)$ and T^{opt} the optimal spanning tree for MIN-POWER SYMMETRIC CONNECTIVITY. Then,

$$P(T^*) \leq 2 \cdot w(T^*) \leq 2 \cdot w(T^{\text{opt}}) \leq 2 \cdot P(T^{\text{opt}}).$$

This means that T^* is a 2-approximation for MIN-POWER SYMMETRIC CONNECTIVITY.

The following lemma indicates that the minimum-weight spanning tree is actually the minimum-length spanning tree.

Lemma 2 *Let T and T^* be a spanning tree and the minimum-length spanning tree of a network, respectively. Then there is a one-to-one onto mapping $\sigma : E(T) \rightarrow E(T^*)$ such that for every $e \in E(T)$, $d(e) \geq d(\sigma(e))$.*

Proof For every edge $e \in E(T^*) \cap E(T)$, set $\sigma(e) = e$. Note that for each $e \in E(T) \setminus E(T^*)$, $e \cup T^*$ contains a cycle Q_e which contains edge e , and for every edge e' in Q_e , $d(e) \geq d(e')$. (Otherwise, it has $d(e) < d(e')$, which implies that $(T - e') \cup e$ would be a spanning tree with length smaller than T , contradicting the minimality of T).

One claims that for any subset $F \subseteq E(T) \setminus E(T^*)$, $\cup_{e \in F} Q_e$ contains at least $|F|$ edges in $E(T) \setminus E(T^*)$. For $|F| = 1$, it is trivial since, otherwise, T^* contains cycle Q_e for $e \in F$, contradicting that T^* is a tree. Next, consider $|F| \geq 2$. Fix an edge $e \in F$ and choose $e' \in Q_e \cap (E(T^*) \setminus E(T))$. Denote $T' = (T \cup e) - e'$. For any $\bar{e} \in F - \{e\}$, if $Q_{\bar{e}}$ does not contain e' , then set $Q'_{\bar{e}} = Q_{\bar{e}}$. If $Q_{\bar{e}}$ contains e' , then note that $Q_e \oplus Q_{\bar{e}}$ is a union of cycles in which there is one containing \bar{e} ; this one is defined to be $Q'_{\bar{e}}$. Clearly, $\cup_{\bar{e} \in F - \{e\}} Q'_{\bar{e}} \subset \cup_{e \in F} Q_e$. Each $Q'_{\bar{e}}$ is contained in $T' \cup \bar{e}$. By induction hypothesis, $\cup_{\bar{e} \in F - \{e\}} Q'_{\bar{e}}$ contains at least $|F| - 1$ edges in $E(T^*) \setminus E(T')$. Add back e' . It follows immediately that $\cup_{e \in F} Q_e$ contains at least $|F|$ edges in $E(T) \setminus E(T^*)$. This completes the proof of the claim.

From claim, one sees that there exists a one-to-one mapping σ from $E(T^*) \setminus E(T)$ to $E(T) \setminus E(T^*)$ such that for every $e \in E(T^*) \setminus E(T)$, $\sigma(e)$ is in Q_e and hence $d(e) \geq d(\sigma(e))$. \square

Now, it has the following.

Theorem 1 ([13]) *The minimum-length spanning tree is a polynomial-time 2-approximation for MIN-POWER SYMMETRIC CONNECTIVITY.*

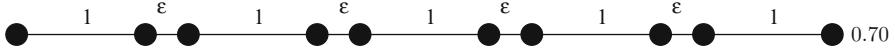


Fig. 2 $2n$ points on a line

Remark This fact is significant because the minimum-length spanning tree in Euclidean plane can be computed in $O(n \log n)$ time while computing directly the minimum-weight spanning tree with edge weight $w(\cdot)$ may need more time.

The following example [14] shows that 2 is tight for the performance ratio of minimum-length spanning tree as an approximation for MIN-POWER SYMMETRIC CONNECTIVITY.

Let v_1, v_2, \dots, v_{2n} be $2n$ points lying along a line in Fig. 2 with the distance as follows:

$$\begin{aligned} d(v_1, v_2) &= d(v_3, v_4) = \dots = d(v_{2n-1}, v_{2n}) = 1, \\ d(v_2, v_3) &= d(v_4, v_5) = \dots = d(v_{2n-2}, v_{2n-1}) = \varepsilon, \end{aligned}$$

where ε is a sufficiently small positive number. The minimum power-cost for keeping symmetric connectivity is

$$nc(1 + \varepsilon)^\alpha + (n - 1)c\varepsilon^\alpha + c \rightarrow (n + 1)c \text{ as } \varepsilon \rightarrow 0$$

which is achieved when $v_1, v_3, \dots, v_{2n-1}$ are assigned with power $c(1 + \varepsilon)^\alpha$, $v_2, v_4, \dots, v_{2n-2}$ are assigned with power $c\varepsilon^\alpha$, and v_{2n} is assigned with power $c1^\alpha$. The minimum spanning tree is the path $(v_1, v_2, \dots, v_{2n})$ with power-cost

$$2nc1^\alpha = 2nc.$$

Therefore, the performance ratio is

$$\frac{2nc}{nc(1 + \varepsilon)^\alpha + (n - 1)c\varepsilon^\alpha + c} \rightarrow 2$$

as $\varepsilon \rightarrow 0$ and $n \rightarrow \infty$.

Is there a polynomial-time approximation with performance ratio less than 2 for MIN-POWER SYMMETRIC CONNECTIVITY? The answer is yes.

To see it, one introduced a type of decomposition for tree.

A k -restricted decomposition of a tree T is a partition of T into a set of edge-disjoint subtrees each with at most k nodes. Consider a k -restricted decomposition of T , $Q = \{T_1, T_2, \dots, T_q\}$. The power-cost of Q is defined by

$$P(Q) = \sum_{i=1}^q P(T_i).$$

Calinescu et al. [15] indicated that the k -restricted decomposition plays a role in the study of MIN-POWER SYMMETRIC CONNECTIVITY the same as the k -restricted Steiner tree in the network Steiner minimum tree problem [16–19]. Algorithms in [16–19] can be modified into approximation algorithms for MIN-POWER SYMMETRIC CONNECTIVITY with performance ratios less than 2. For example, let *gain* of a subtree H is defined by

$$\text{gain}(H) = 2 \cdot \text{mst}(V) - 2 \cdot \text{mst}(V/H) - P(H)$$

where $\text{mst}(V)$ is the power-cost of minimum-length spanning tree T on V , that is,

$$\text{mst}(V) = \sum_{e \in E(T)} w(e),$$

and V/H is the set of nodes after contracting H . Let *loss* of a subtree H be defined by

$$\text{loss}(H) = c(H) - (\text{mst}(V) - \text{mst}(V/H)).$$

Then the algorithm in [18] can be written as follows.

Loss-Contracting Algorithm

Input: A complete graph $G = (V, E)$ with edge cost $w(u, v) = d(u, v)^\alpha$ and integer $3 \leq k < |V|$.

$T \leftarrow$ the minimum-length spanning tree of G ;

$H \leftarrow G$;

$ok \leftarrow 1$;

while $ok = 1$ **do**

find a k -restricted tree K with maximum $r = \text{gain}(K)/\text{loss}(K)$;

if $r < 0$

then $ok \leftarrow 0$

else $H \leftarrow H \cup K$

$V \leftarrow V/K$;

output H (a k -restricted decomposition connecting all nodes in G .)

By an argument similar to that in [18], it can be obtained by the following.

Theorem 2 *The above Loss-Contracting Algorithm produces an approximation solution for MIN-POWER SYMMETRIC CONNECTIVITY with performance ratio at most $\rho_k \left(1 + \frac{\ln(4\rho_k^{-1}-1)}{2}\right)$ where ρ_k is defined by*

$$\rho_k = \sup_T \min_Q \frac{P(Q)}{P(T)}$$

where T is over all trees and Q is over all k -restricted decompositions of T .

Althaus et al. [20] showed that $\rho_3 = 5/3$ equal to the inverse of the k -Steiner ratio determined by Du [21]. Li et al. [22] showed that this fact holds for general k , that is, ρ_k is equal to the inverse of the k -Steiner ratio determined in [23].

Theorem 3 ([22]) *For any $k \geq 2$,*

$$\rho_k = \frac{(r+1)2^r + s}{r2^r + s}$$

where $k = 2^r + s$, $0 \leq s < 2^r$.

As $k \rightarrow \infty$, it can be obtained that $\rho_k \rightarrow 1$. Therefore,

$$\rho_k \left(1 + \frac{\ln(4\rho_k^{-1} - 1)}{2} \right) \rightarrow 1 + (\ln 3)/2 < 1.55.$$

Thus, the following is obtained.

Corollary 1 *For sufficiently large k , **Loss-Contraction Algorithm** produces an approximation solution for MIN-POWER SYMMETRIC CONNECTIVITY with performance ratio at most 1.55.*

Although some new technique [24, 25] has been introduced to study MIN-POWER SYMMETRIC CONNECTIVITY, 1.55 is still currently the best known performance ratio of polynomial-time approximation for MIN-POWER SYMMETRIC CONNECTIVITY. Not every technique for design of approximation for Network Steiner Tree Problem can be easily applied to MIN-POWER SYMMETRIC CONNECTIVITY. Indeed, polynomial-time approximation for Network Steiner Tree Problem has been improved by Byrka et al. [26] with a new technique. However, it is an open problem whether this new technique can be applied to MIN-POWER SYMMETRIC CONNECTIVITY.

2.2 Asymmetric Topological Control

It is interesting to note that the minimum spanning tree is also a polynomial-time 2-approximation in asymmetric case as follows.

MIN-POWER STRONG CONNECTIVITY: Given a wireless network with power-adjustable nodes, find a power assignment to minimize the total power and to keep the network strongly connected.

In asymmetric case, an arc (directed edge) (u, v) exists if and only if the communication range of node u covers node v , that is, $p(u) \geq w(u, v)$. This problem was initially studied by Chen and Huang [27]. They showed that the minimum-length spanning tree gives 2-approximation. Kirousis et al. [13] showed the NP-hardness of the problem in three-dimensional Euclidean space with $\alpha = 2$ and presented

a polynomial-time 2-approximation. Clementi et al. [28, 29] showed that the NP-hardness remains in two-dimensional Euclidean plane. Cheng et al. [30] showed an alternative proof for the NP-hardness of MIN-POWER STRONG CONNECTIVITY in the two-dimensional Euclidean plane. Earlier efforts were also made by Ramanathan et al. [31], Wattenhofer et al. [32], and Hajiaghayi et al. [33].

Consider a directed network H . The total power for keeping H survived is

$$P(H) = \sum_{u \in V(H)} \max_{(u,v) \in E(H)} w(u, v).$$

When H is strongly connected, it can construct an in-arborescence A as follows:

- Pick up a root r arbitrarily.
- For every node u , construct a shortest path $p(u)$ from u to r .
- Let A be the in-arborescence obtained from the union $\cup_{u \in V(H)} p(u)$ by the following simplification: for each node u , if u has more than one out-edges, then keep one and delete all others.

Since $P(H) \geq P(A)$, by In-Arborescence Lemma, the following can be obtained.

Lemma 3 *For any strongly connected graph H and a node r of H , there is an in-arborescence A with root r , contained in H , such that*

$$P(H) \geq w(A)$$

where $w(A) = \sum_{(u,v) \in E(A)} w(u, v)$.

Now, suppose T^* is a minimum-length spanning tree. By Lemma 2, it can be obtained that

$$\begin{aligned} P(T^*) &\leq 2 \sum_{(u,v) \in E(T^*)} w(u, v) \\ &\leq 2w(A) \\ &\leq 2P(H) \end{aligned}$$

for every strongly connected graph H where A is an in-arborescence contained in H . This means that the minimum-length spanning tree is a polynomial-time 2-approximation for MIN-POWER STRONG CONNECTIVITY.

Theorem 4 ([27]) *The minimum-length spanning tree is a polynomial-time 2-approximation for MIN-POWER STRONG CONNECTIVITY.*

Calinescu [14] indicated that 2 is tight upper bound for the performance ratio of the minimum-length spanning tree. Actually, the example for MIN-POWER SYMMETRIC CONNECTIVITY also works here although Calinescu [14] constructed

a different example. Indeed, the example for MIN-POWER SYMMETRIC CONNECTIVITY means that there exists a spanning tree T such that for the minimum-length spanning tree T^* ,

$$\frac{P(T^*)}{P(T)} \rightarrow 2 \text{ as } \varepsilon \rightarrow 0, n \rightarrow \infty.$$

Let opt_{mpsc} be the objective function value of optimal solution for MIN-POWER STRONG CONNECTIVITY. Then, $P(T) \geq opt_{\text{mpsc}}$ and $P(T^*) \leq 2 \cdot opt_{\text{mpsc}}$. Therefore,

$$2 \geq \frac{T^*}{opt_{\text{mpsc}}} \geq \frac{P(T^*)}{P(T)} \rightarrow 2$$

as $\varepsilon \rightarrow 0$ and $n \rightarrow \infty$. Hence,

$$\frac{T^*}{opt_{\text{mpsc}}} \rightarrow 2 \text{ as } \varepsilon \rightarrow 0, n \rightarrow \infty.$$

Although many efforts have been made to improve the performance of approximation algorithms [13, 34–36] and the performance ratio less than 2 has been established in several special cases, it was a long-standing open problem whether there is a polynomial-time approximation algorithm with performance ratio less than 2 for MIN-POWER STRONG CONNECTIVITY. Recently, this open problem is solved by Calinescu [14]. He presented a greedy approximation with performance 1.992. The following is a sketch of his work.

The algorithm is a greedy approximation with submodular potential function. This submodular potential function is defined as follows:

Consider a minimum-length spanning tree T on given graph $G = (V, E)$. Let \hat{T} be obtained from T by replacing each edge with two arcs of opposite directions. \hat{T} is called the bidirected version of T .

For any vertex $u \in V$ and a real number $p \in \{w(u, v) \mid (u, v) \in E\}$, $S(u, p)$ denotes the directed star with center u and all arcs (u, v) with $w(u, v) \leq p$ where recall that $w(u, v) = d(u, v)^\alpha$. Let $Q(u, p)$ be the set of edges of T on a path between two vertices of $S(u, p)$.

For a collection \mathcal{A} of directed stars $S(u_i, p_i)$, define $f(\mathcal{A}) = \sum_{e \in Q(\mathcal{A})} w(e)$ where $Q(\mathcal{A}) = \cup_{S(u_i, p_i) \in \mathcal{A}} Q(u_i, p_i)$.

Lemma 4 *f is submodular and monotone increasing, that is, for any two collections \mathcal{A} and \mathcal{B} of directed stars $S(u_i, p_i)$,*

$$f(\mathcal{A}) + f(\mathcal{B}) \geq f(\mathcal{A} \cup \mathcal{B}) + f(\mathcal{A} \cap \mathcal{B}),$$

and

$$\mathcal{A} \subset \mathcal{B} \Rightarrow f(\mathcal{A}) \leq f(\mathcal{B}).$$

Proof By Lemma 2.25 in [37], it suffices to prove

$$\mathcal{A} \subset \mathcal{B} \Rightarrow \Delta_S f(\mathcal{A}) \geq \Delta_S f(\mathcal{B})$$

where S is one of directed stars $S(u_i, w_i)$ and $\Delta_S f(\mathcal{A}) = f(\mathcal{A} \cup \{S\}) - f(\mathcal{A})$. Clearly,

$$\begin{aligned}\Delta_S f(\mathcal{A}) &= \sum_{e \in Q(\mathcal{A} \cup \{S\})} w(e) - \sum_{e \in Q(\mathcal{A})} w(e) \\ &= \sum_{e \in Q(\{S\}) \setminus Q(\mathcal{A})} w(e) \\ &\geq \sum_{e \in Q(\{S\}) \setminus Q(\mathcal{B})} w(e) \\ &= \Delta_S f(\mathcal{B}).\end{aligned}$$

□

Let $\hat{Q}(u, p)$ be the set of arcs on the path from u to v in \hat{T} for $(u, v) \in S(u, p)$. Let $I_{\mathcal{A}}(S(u, p))$ denote the set of arcs in $\hat{Q}(u, p)$, but its undirected version not in $Q(\mathcal{A})$. Denote $w(T) = \sum_{e \in E(T)} w(e)$. The following is the greedy algorithm proposed by Calinescu [14]:

Calinescu's Greedy Algorithm

Let T be a minimum spanning tree.

```

 $\mathcal{A} \leftarrow \emptyset,$ 
 $M \leftarrow \hat{T};$ 
while  $f(\mathcal{A}) < w(T)$  do
     $(u, p) \leftarrow \operatorname{argmax}_{(u', p')} \frac{\Delta_{S(u', p')}(\mathcal{A})}{p'}$ 
     $M \leftarrow M \setminus I_{\mathcal{A}}(S(u, p))$ 
     $\mathcal{A} \leftarrow \mathcal{A} \cup \{S(u, p)\}$ 
output  $\cup_{S \in \mathcal{A}} E(S) \cup M.$ 
```

The following lemma shows the correctness of above algorithm.

Lemma 5 $H = (V, \cup_{S \in \mathcal{A}} E(S) \cup M)$ is a strongly connected spanning subgraph during the computation of Calinescu's Greedy Algorithm.

Proof Remove from T all edges each having two corresponding arcs in M . Then T is broken into several connected components T_1, T_2, \dots, T_k . It can be observed that every $V(T_i)$ for $1 \leq i \leq k$ induces a strongly connected subgraph of H . Then, the lemma follows immediately from this fact.

Initially, $M = \hat{T}$. Thus, every edge of T has two corresponding arcs in M . This means that each T_i is a vertex of T and hence strongly connected. Now, suppose at some stage of the algorithm, every $V(T_i)$ induces a strongly connected subgraph H_i of H . It is shown that after adding $S(u, p)$ to \mathcal{A} , this fact still holds.

For each $v \in V(S(u, p) - \{u\})$, suppose on the path from u to v in T , there are h edges $(x_1, y_1), (x_2, y_2), \dots, (x_h, y_h)$ each having its two corresponding arcs (x_i, y_i) and (y_i, x_i) in M . Further, it can be assumed that u and x_1 are in the same connected component T_0 , y_1 and x_2 are in the same connected component T_1, \dots, y_h and v are in the same connected component T_h . When $S(u, w)$ is added to \mathcal{A} , arcs $(x_1, y_1), (x_2, y_2), \dots, (x_h, y_h)$ would be deleted from M ; however, arc (u, v) is added to $\cup_{S \in \mathcal{A}} E(S)$. Therefore, in new H , $V(T_0) \cup V(T_1) \cup \dots \cup V(T_h)$ would induce a strongly connected subgraph because arcs $(u, v), (y_h, x_h), \dots, (y_1, x_1)$ form a directed cycle connecting $h + 1$ strongly connected subgraphs H_0, H_1, \dots, H_h into a bigger strongly connected subgraph. Note that after deleting arcs $(x_1, y_1), (x_2, y_2), \dots, (x_h, y_h)$ from M , $T_0 \cup (u, x_1) \cup T_1 \cup \dots \cup T_h$ becomes a new connected component. This would complete our proof. \square

To analyze the performance of Calinescu's Greedy Algorithm, the general theory introduced in [37] should be recalled.

Consider a monotone nondecreasing, submodular function $f : 2^X \rightarrow R$. Denote $\Omega(f) = \{A \subseteq X \mid f(A) = f(X)\}$. Let $c : X \rightarrow R$ be a nonnegative cost function. Consider the following SUBMODULAR-COVER problem:

$$\begin{aligned} \min \quad & c(A) = \sum_{x \in A} c(x) \\ \text{subject to} \quad & A \in \Omega(f). \end{aligned}$$

To compute an approximation of this problem, the following greedy algorithm may be employed.

Greedy Algorithm G

```

 $A \leftarrow \emptyset;$ 
while  $f(A) < f(X)$  do
    choose  $x \in X$  to maximize  $\frac{\Delta_x f(A)}{c(x)}$ 
    and  $A \leftarrow A \cup \{x\}$ ;
output  $A$ .

```

There is a general theorem on the performance of above algorithm.

Theorem 5 Suppose in Greedy Algorithm G, selected x always satisfies

$$\frac{\Delta_x f(A)}{c(x)} \geq 1.$$

Then, Greedy Algorithm G produces an approximation solution for SUBMODULAR-COVER with performance ratio at most

$$1 + \ln \frac{f(X) - f(\emptyset)}{opt}$$

where opt is the objective function value of optimal solution for SUBMODULAR-COVER.

Consider the MIN-STARS problem as follows:

$$\min \sum_{S(u,p) \in \mathcal{A}} p$$

subject to $f(\mathcal{A}) = w(T).$

Note that Calinescu's Greedy Algorithm is the same as the **Greedy Algorithm G** for the MIN-STARS problem. Based on this observation, it is easy to obtain the following corollary.

Corollary 2 *If $opt_{\min\text{-stars}} \leq \alpha opt_{\text{mpsc}}$ where opt_{mpsc} is the objective function value of optimal solution for MIN-POWER STRONG CONNECTIVITY, then Calinescu's Greedy Algorithm is a polynomial-time approximation for MIN-POWER STRONG CONNECTIVITY with performance ratio at most*

$$1 + \alpha + \alpha(1 - \ln \alpha).$$

Proof Note that when $f(\mathcal{A}) < w(T)$, M contains two antiparallel arcs (x, y) and (y, x) . In such a case, S can be selected to satisfy

$$\frac{\Delta_{S(u,p)} f(\mathcal{A})}{p} \geq 1,$$

since set $w' = w((x, y))$, it has the following that

$$\frac{\Delta_{S(x,p')} f(\mathcal{A})}{p'} \geq 1.$$

By [Theorem 5](#),

$$\sum_{S(u,p) \in \mathcal{A}} p \leq opt_{\min\text{-stars}} \left(1 + \ln \frac{w(T)}{opt_{\min\text{-stars}}} \right).$$

Therefore, output of Calinescu's Greedy Algorithm has total power at most

$$\begin{aligned} & w(T) + \sum_{S(u,p) \in \mathcal{A}} p \\ & \leq opt_{\text{mpsc}} + opt_{\min\text{-stars}} \left(1 + \ln \frac{w(T)}{opt_{\min\text{-stars}}} \right) \\ & \leq opt_{\text{mpsc}} + opt_{\min\text{-stars}} \left(1 + \ln \frac{opt_{\text{mpsc}}}{opt_{\min\text{-stars}}} \right) \\ & \leq opt_{\text{mpsc}} \left(1 + \alpha \left(1 + \ln \frac{1}{\alpha} \right) \right) \\ & \leq opt_{\text{mpsc}} (1 + \alpha(1 - \ln \alpha)), \end{aligned}$$

where note that function $x \left(1 + \ln \frac{opt_{\text{mpsc}}}{x} \right)$ is monotone nondecreasing for $0 < x < opt_{\text{mpsc}}$. \square

Calinescu [14] showed the following.

Lemma 6 $opt_{\text{min-stars}} \leq (4/5)opt_{\text{mpsc}}$.

The following follows immediately from [Corollary 2](#) and [Lemma 6](#).

Theorem 6 *There exists a polynomial-time approximation for MIN-POWER STRONG CONNECTIVITY with performance ratio at most $1 + (4/5)(1 + \ln(5/4))$.*

2.3 Broadcast

A broadcasting routing is for transmitting data from a source node s to all other nodes. Mathematically, it is an out-arborescence T rooted at s . Let (u, v_i) , $i = 1, \dots, k$ be all out-edges at node u in T . Then, the power-cost of u in the routing equals $p(u) = \max_{1 \leq i \leq k} w(u, v_i)$. The power-cost of broadcasting routing T is the sum of power-costs at all nodes in T . In general graph setting, it is unlikely to have polynomial-time $o(\log n)$ -approximation for the minimum power broadcasting problem [38, 39]. However, for geometric setting, the situation may be different. For example, in one-dimensional Euclidean space, the problem is polynomial-time solvable [38, 40, 41], and in d -dimensional Euclidean space, the minimum-length spanning tree is still a good approximation. For instance, consider this problem in Euclidean plane as follows.

MIN-POWER BROADCAST: Given a set of points S in the Euclidean plane and a source node $s \in S$, find a broadcasting routing from s to minimize the total power-cost of the routing.

This is an NP-hard problem [40]. It is an interesting fact that the minimum Euclidean spanning tree T is also a constant approximation for MIN-POWER BROADCAST. This is because of the existence of the following lemma, which plays an important role for MIN-POWER BROADCAST, similar to the role of [Lemma 1](#) for MIN-POWER SYMMETRIC CONNECTIVITY and the role of [Lemma 3](#) for MIN-POWER STRONG CONNECTIVITY.

Lemma 7 [*Out-Arborescence Lemma*] *For any out-arborescence T , there exists a minimum Euclidean spanning tree T^* such that*

$$P(T) \geq \frac{1}{6} \sum_{e \in E(T^*)} c \|e\|^\alpha.$$

The proof of [Lemma 7](#) relies on the following fact.

Lemma 8 *Let C be a disk with center x and radius R and P be a set of points lying in C . Assume $x \in P$. Then, for minimum spanning tree T^* on P ,*

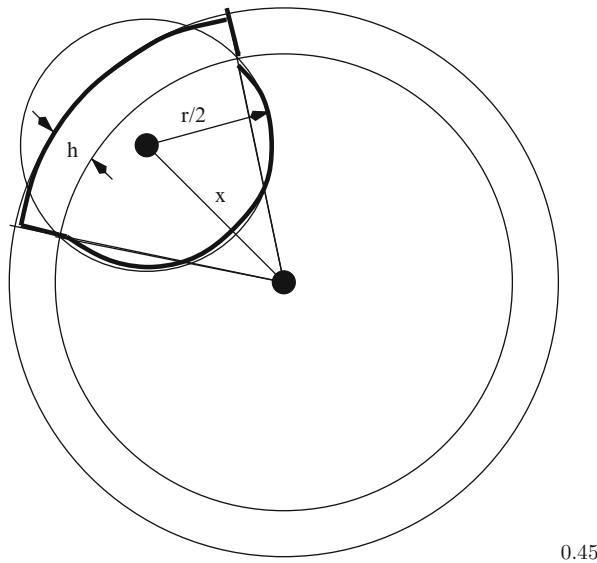


Fig. 3 Minimum spanning tree in a disk

$$\sum_{e \in T^*} \|e\|^\alpha \leq 6R^\alpha$$

where $\alpha \geq 2$.

To prove Lemma 7, for each node u of T , a smallest disk $D(u)$ to cover all out-edges at u is drawn. Let $V(u)$ be the set of all endpoints of out-edges at u . Let $T^*(u)$ be the minimum spanning tree on $V(u) \cup \{u\}$. By Lemma 8,

$$\sum_{e \in E(T^*(u))} c \|e\|^\alpha \leq 6cR(u)^\alpha$$

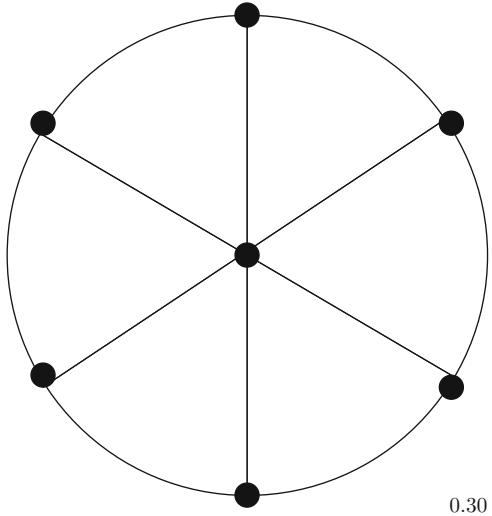
where $R(u)$ is the radius of $D(u)$. Note that $\cup_{u \in V} T^*(u)$ is a spanning tree on the given node set V . By Lemma 2,

$$\sum_{e \in E(T^*)} c \|e\|^\alpha \leq \sum_{u \in V} \sum_{e \in E(T^*(u))} c \|e\|^\alpha.$$

Therefore,

$$\sum_{e \in E(T^*)} c \|e\|^\alpha \leq \sum_{u \in V} 6cR(u)^\alpha = 6P(T).$$

This completes the proof of Lemma 7 (Fig. 3).

Fig. 4 6 is the best possible

The proof of [Lemma 8](#) given by Ambühl [\[42\]](#) is quite complicated. Actually, this result was obtained through a sequence of efforts [\[4, 43–46\]](#). Actually, Wan et al. [\[4\]](#) initiated this theoretical analysis, and they also indicated that the constant 6 is the best possible. An example which reaches this bound is shown in [Fig. 4](#).

The approach to establish the upper bound for this performance ratio is quite interesting. Here, a proof of a weaker bound is included to show the main idea.

Lemma 9 ([46]) Consider a circle C with center x and radius R in the Euclidean plane. Let P be a set of points lying in C . Assume $x \in P$. Then, for minimum Euclidean spanning tree T^* on P ,

$$\sum_{e \in E(T^*)} \|e\|^\alpha \leq 8R^\alpha$$

where $\alpha \geq 2$.

Proof Let $n(T^*, r)$ be the number of connected components of subgraph $T^*(r)$ of T^* where $T^*(r)$ has the node set P and the edge set consisting of all edges with length at most r . Note that $x \in P$ implies that every edge in T^* has length at most R . Therefore,

$$\sum_{e \in E(T^*)} \|e\|^\alpha = \alpha \int_0^R (n(T^*, r) - 1)r^{\alpha-1} dr.$$

For each node $u \in P$, draw a disk with center u and radius $r/2$. Then, for each connected component of $T^*(r)$, those disks form a connected region. Moreover, different connected components induce disjoint such regions. Note that each such

region contains at least one disk with radius $r/2$. Hence, its area is at least $\pi(r/2)^2$. It follows that the boundary of each connected region has length at least πr because circle gives the shortest boundary for surrounding a given amount of area. Let $a(P, r)$ denote the total area covered by those disks with radius $r/2$. Then, it has

$$\begin{aligned} a(P, R) &= \int_0^R d(a(P, r)) \\ &\geq \int_0^R n(T^*, r) \pi r d(r/2) \\ &= \frac{\pi}{2} \int_0^R (n(T^*, r) - 1) r dr + \frac{\pi R^2}{4} \\ &= \frac{\pi}{4} \sum_{e \in E(T^*)} \|e\|^2 + \frac{\pi R^2}{4}. \end{aligned}$$

Note that $a(P, R)$ is contained by a disk at x with radius $1.5R$. Thus,

$$\pi(1.5R)^2 \geq \frac{\pi}{4} \sum_{e \in E(T^*)} \|e\|^2 + \frac{\pi R^2}{4}.$$

Hence,

$$\sum_{e \in E(T^*)} \|e\|^2 \leq 8R^2.$$

Since for every $e \in E(T^*)$, $\|e\| \leq R$, one has

$$\sum_{e \in E(T^*)} (\|e\|/R)^\alpha \leq \sum_{e \in T} (\|e\|/R)^2 \leq 8.$$

□

By [Lemma 7](#), it is easy to see the following.

Theorem 7 *The minimum spanning tree induces a 6-approximation for MIN-POWER BROADCAST.*

All above results in this subsection are on networks with all nodes lying in the Euclidean plane. For networks with all nodes lying in the three-dimensional space, results are different.

For networks with all nodes lying in the d -dimensional Euclidean space, it has been shown that the minimum-length spanning tree would induce an 18.8-approximation for $d = 3$ [47] and $(3^d - 1)$ -approximation for $d > 3$ [44]. In [4, 40], it has been also shown that the lower bound for the performance ratio

of the minimum-length spanning tree is given by the d -dimensional kissing number n_d , which is the maximum number of nonoverlapping unit balls touching a unit ball. For $d = 3$, $n_d = 12$. In general, $n_d = 2^{cd(1+o(1))}$ where $0.2075 \leq c \leq 0.401$ for large d [48]; determination of exact value of n_d for large d is quite hard [49].

Is it possible to have an approximation with performance better than that of the minimum-length spanning tree for MIN-POWER BROADCAST? The answer is positive. Wan et al. [4] indicated that the BIP (Broadcasting Incremental Power) is a possible candidate because its performance ratio is between $13/3$ and 12 while the minimum-length spanning tree has performance ratio between 6 and 12 (note: in both cases, upper bound 12 has been improved to 6 by Ambühl [42]). To construct a BIP tree, starting from the source node, at each iteration, a new node is reached with smallest increasing power. Note that this increasing may have two ways. One is to assign power to an unassigned node, and the other one is to increase the power to a node with power assigned in previous iteration.

Caragiannis et al. [50] gave a greedy approximation with performance ratio 4.2 which is definitely better than both BIP and the minimum-length spanning tree. The result of Caragiannis et al. [50] is for any metric space that if the minimum-length spanning tree induces a ρ -approximation, then their greedy approximation has performance ratio $2 \ln \rho - 2 \ln 2 + 2$. It is 4.2 for $d = 2$, 6.49 for $d = 3$, and $2.20d + 0.61$ for $d \geq 4$. This greedy approximation is quite interesting, which is designed using the following properties of spanning tree.

Lemma 10 *Let T_1 and T_2 be two rooted spanning trees over the same set of nodes V . There exists a one-to-one mapping $\sigma : E(T_1) \rightarrow E(T_2)$ such that for any node u and its children v_1, \dots, v_k in T_1 , $(T_1 \cup \{(u, v_1), \dots, (u, v_k)\}) - \{\sigma((u, v_1)), \dots, \sigma((u, v_k))\}$ is a spanning tree.*

Lemma 11 *Let T be a spanning tree. Consider k edges $(u, v_1), \dots, (u, v_k)$. Suppose $(T \cup \{(u, v_1), \dots, (u, v_k)\}) - \{(w_1, y_1), \dots, (w_k, y_k)\}$ is a spanning tree. Then, for $\{(u, z_1), \dots, (u, z_h)\}$, there exists a subset F of h edges of T such that $(T \cup \{(u, v_1), \dots, (u, v_k), (u, z_1), \dots, (u, z_h)\}) - (\{(w_1, y_1), \dots, (w_k, y_k)\} \cup F)$ is a spanning tree.*

There is no proof for above two lemmas in [50]. They can be derived from the following.

Lemma 12 *Let T_1 and T_2 be two spanning trees. There exists a one-to-one onto mapping $\sigma : E(T_1) \rightarrow E(T_2)$ such that for any $A \subseteq E(T_1)$, $(E(T_2) \cup A) - \sigma(A)$ is the edge set of a spanning tree.*

However, it is an open problem whether Lemma 12 holds or not, which is an interesting open problem. Next, an application of above lemmas will be presented.

Consider a spanning tree T of a graph $G = (V, E)$ and an edge subset F of E . A *swap set* for T with F is an edge subset A of $E(T)$ such that $(E(T) \cup F) - A$ is the edge set of a spanning tree. Let $E(u, p) = \{(u, v) \in E(G) \mid w(u, v) \leq p\}$. $A(u, p)$ denotes a swap of T with $E(u, p)$. The greedy algorithm can be described as follows.

CFM Algorithm

```

 $T \leftarrow$  the minimum-length spanning tree of  $G$ .
 $ok \leftarrow 1$ ;
 $\mathcal{A} \leftarrow \emptyset$ ;
while  $ok = 1$  do begin
     $E(u, p) = \operatorname{argmax}_{E(u', p')} \frac{w(A(u', p'))}{p'}$ ;
    if  $\frac{w(A(u, p))}{p} > 2$ 
        then  $\mathcal{E} \leftarrow \mathcal{E} \cup \{E(u, p)\}$ 
             $(T \cup E(u, p)) \setminus A(u, p)$ 
            assign all edges in  $E(u, p)$  with weight 0
        else  $ok \leftarrow 0$ ;
output  $\mathcal{E}$  and  $T$ .

```

The following shows an upper bound for power-cost of broadcasting tree constructed from output of CFM Algorithm.

Lemma 13 *With output \mathcal{E} and T , a broadcasting tree can be constructed with powercost at most*

$$2 \sum_{E(u,p) \in \mathcal{E}} p + w(T).$$

Proof Suppose s is the source node. Turn tree T into a tree rooted at s and turn each edge into an arc by giving the direction from parent to child. This rooted tree (or out-arborescence) can be realized with the following power assignment. Consider an arc (x, y) in rooted tree T .

If edge (x, y) is not in $\cup_{E(u,p) \in \mathcal{E}} E(u, p)$, then assign $p(x) = w(x, y)$.

If $(x, y) \in E(u, p)$ for some $E(u, p) \in \mathcal{E}$, then assign $p(x) = p$. An important observation is that for each $E(u, p)$, there is at most one edge (u, v) in $E(u, p)$ such that arc (v, u) is in rooted tree T . In fact, if (v, u) is in the tree T with root s , then for every edge $(u, v') \in E(u, p)$, other than (u, v) , one can reach v' from u through arc (u, v') , that is, if (v', u) cannot be in T .

In above assignment, the total power is at most

$$2 \sum_{E(u,p) \in \mathcal{E}} p + w(T).$$

□

The performance of CFM is given in the following theorem.

Theorem 8 Suppose $\rho > 2$ is the performance ratio of the minimum-length spanning tree as an approximation for MIN-POWER BROADCAST. Then, the performance ratio of approximation solution obtained from CFM Algorithm is at most $2 \ln \rho - 2 \ln 2 + 2$.

Proof Let T_0 denote the minimum spanning tree at the initial step and T_i for $i = 1, \dots, k$ the tree obtained in the i th iteration. Let $E(u_i, p_i)$ be selected at the i th iteration. Clearly,

$$w(A(u_i, p_i)) = w(T_{i-1}) - w(T_i).$$

First, it can be observed that

$$\rho \leq \frac{w(T_{i-1}) - w(T_i)}{w(T_{i-1})} \cdot opt.$$

where opt is objective function value of optimal solution for MIN-POWER BROADCAST.

Suppose T^* is the optimal broadcast tree. Let $IN(T^*)$ denote the set of internal nodes of T^* . Denote by $B(u)$ the set of arcs going out from u . Then,

$$p(T^*) = \sum_{u \in IN(T^*)} \max_{(u,v) \in B(u)} w(u, v).$$

By Lemma 12, there exists a one-to-one onto mapping $\sigma : E(T^*) \rightarrow E(T_{i-1})$ such that $\sigma(B(u))$ is a swap set for T_{i-1} with $B(u)$. Denote $\rho_u = \max_{(u,v) \in B(u)} w(u, v)$. Then,

$$\frac{w(\sigma(B(u)))}{\rho_u} \geq \frac{\sum_{u \in IN(T^*)} w(\sigma(B(u)))}{\sum_{u \in IN(T^*)} \rho_u} = \frac{w(T_{i-1})}{opt}.$$

Therefore, there exists $u \in IN(T^*)$ such that

$$\frac{w(\sigma(B(u)))}{\rho_u} \geq \frac{w(T_{i-1})}{opt}.$$

Since $\sigma(B(u)) \subseteq A(u, \rho_u)$, it follows that

$$\frac{w(A(u, \rho_u))}{\rho_u} \geq \frac{w(T_{i-1})}{opt}.$$

It follows from greedy principle that

$$\frac{w(A(u_i, \rho_i))}{\rho_i} \geq \frac{w(A(u, \rho_u))}{\rho_u} \geq \frac{w(T_{i-1})}{opt},$$

that is,

$$\rho_i \leq \frac{w(T_{i-1}) - w(T_i)}{w(T_{i-1})} \cdot opt.$$

Now, by [Lemma 13](#), GFM Algorithm produces a broadcast tree with total power at most

$$\begin{aligned} & 2 \sum_{i=1}^k \frac{w(T_{i-1}) - w(T_i)}{w(T_{i-1})} \cdot opt + w(T_k) \\ & \leq 2 \sum_{i=1}^{k-1} \frac{w(T_{i-1}) - w(T_i)}{w(T_{i-1})} \cdot opt + 2\rho_k + w(T_k). \end{aligned}$$

Denote $\delta = 2opt - w(T_k)$. Note that

$$\frac{w(A(u_k, \rho_k))}{\rho_k} > 2.$$

Thus,

$$\begin{aligned} 2\rho_k + w(T_k) &= 2 \cdot \frac{w(A(u_k, \rho_k))}{w(A(u_k, \rho_k))/\rho_k} - \delta + 2opt \\ &\leq 2 \cdot \frac{w(A(u_k, \rho_k)) - \delta}{w(A(u_k, \rho_k))/\rho_k} + 2opt \\ &\leq 2 \cdot \frac{w(A(u_k, \rho_k)) - \delta}{w(T_{k-1})/opt} + 2opt \quad (*) \\ &= 2opt \cdot \left(\frac{w(A(u_k, \rho_k)) - \delta}{w(T_{k-1})} + 1 \right), \end{aligned}$$

Therefore, GFM Algorithm produces a broadcast tree with total power at most

$$\begin{aligned} & 2opt \left(\sum_{i=1}^{k-1} \frac{w(T_{i-1}) - w(T_i)}{w(T_{i-1})} + \frac{w(T_{k-1}) - 2opt}{w(T_{k-1})} + 1 \right) \\ &\leq 2opt \left(\sum_{i=1}^{k-1} \int_{w(T_0)-w(T_{i-1})}^{w(T_0)-w(T_i)} \frac{dt}{w(T_0)-t} + \int_{w(T_0)-w(T_{k-1})}^{w(T_0)-2opt} \frac{dt}{w(T_0)-t} + 1 \right) \\ &= 2opt \left(\int_0^{w(T_0)-2opt} \frac{dt}{w(T_0)-t} + 1 \right) \\ &= 2opt \left(\ln \frac{w(T_0)}{opt} - \ln 2 + 1 \right) \\ &\leq opt(2 \ln \rho - 2 \ln 2 + 2). \end{aligned}$$

□

The above proof contains a bug. The inequality sign at line (*) holds only if $w(A(u_k, \rho_k)) - \delta \geq 0$, that is, $w(T_{k-1}) \geq 2opt$. If $w(T_{k-1}) < 2opt$, is there some way to fix the proof? So far, nobody knows.

Dai and Wu [51], Li et al. [52, 53], and Guo and Yang [54] minimize the total energy cost of broadcast in wireless networks with directional antennas or adaptive antennas. Ghosh [55] designs distributed algorithm and Wu and Dai [56] design an broadcast algorithm with self-pruning technique. Agarwal et al. [57] study energy-efficient broadcast in wireless ad hoc network with hitch-hiking.

2.4 Asymmetric Power Requirement

In wireless networks studied in [58], each node x has its own rate $r(x)$ for power requirement to establish a link (x, y) , i.e., the power-cost for the existence of (x, y) is

$$p(x, y) = c\|(x, y)\|^\alpha / r(x)$$

instead of $c\|(x, y)\|^\alpha$. When $r(x) \neq r(y)$, it has the following that $p(x, y) \neq p(y, x)$.

Althaus et al. [20] also considered the case that arcs $p(u, v) \neq p(v, u)$ and studied the following problem:

MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENT:

Given a complete graph $G = (V, E)$ and power requirement $p : V \times V \rightarrow R^+$, find a minimum power-cost spanning tree where an (undirected) edge (u, v) exists iff the power at u is at least $p(u, v)$ and the power at v is at least $p(v, u)$.

For this problem, they showed that the minimum spanning tree with respect to edge weight $p(u, v) + p(v, u)$ is a 2-approximation. However, the proof contains a flaw because the power-cost of an out-arborescence has not been proved to be lower-bounded by the sum of its edge power requirements. In fact, Lemma 7 indicates that the power-cost of an out-arborescence can be lower bounded by one-sixth of the sum of edge power requirements of a minimum spanning tree which is not the out-arborescence itself. Therefore, it is still an open problem whether MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENT has a polynomial-time constant approximation or not.

Calinescu et al. [59] gave a positive answer in the case that

$$p(x, y) = cd(x, y)^\alpha / r(x),$$

where $d(x, y)$ is the distance between x and y in a metric space.

Theorem 9 *The minimum spanning tree with respect to edge weight $w(u, v) = p(u, v) + p(v, u)$ induces an approximation solution within a factor of*

$$\min_{r>1} 2 \max \left(2^\alpha + (r+1)^\alpha, \frac{r^\alpha}{r^\alpha - 1} \right)$$

from optimal for MIN-POWER SYMMETRIC CONNECTIVITY WITH ASYMMETRIC POWER REQUIREMENT.

This theorem follows immediately from the following lemma.

Lemma 14 [Rooted-Spanning Tree Lemma] For any rooted spanning tree T and a number $r > 1$, there exists a rooted spanning tree T_r such that

$$w(T_r) \leq 2 \max \left(2^\alpha + (r+1)^\alpha, \frac{r^\alpha}{r^\alpha - 1} \right) \cdot P(T)$$

where $w(T_r) = \sum_{(u,v) \in E(T_r)} (p(u,v) + p(v,u))$.

Proof Let $V^i(T)$ be the set of all internal nodes of T . Denote by $T(u)$ the subtree induced by an internal node u and all its children. Then,

$$\sum_{u \in V^i(T)} P(T(u)) \leq 2P(T).$$

For each $T(u)$, a tree $T_r(u)$ is constructed by modifying $T(u)$ in the following way:

- Sort all children x_1, x_2, \dots, x_k of u in ordering of nonincreasing in $d(u, x_i)$, that is, $d(u, x_1) \geq d(u, x_2) \geq \dots \geq d(u, x_k)$.
- For each $i = 1, \dots, k-1$, if $(d(u, x_i) \leq rd(u, x_{i+1}))$, then replace edge (u, x_i) by edge (x_i, x_{i+1}) .

Note that for each edge (x_i, x_{i+1}) in $T_r(u)$,

$$d(x_i, x_{i+1}) \leq d(u, x_i) + d(u, x_{i+1}) \leq (1+r)d(u, x_{i+1})$$

and

$$d(x_i, x_{i+1}) \leq d(u, x_i) + d(u, x_{i+1}) \leq 2d(u, x_i).$$

Thus,

$$p(x_i, x_{i+1}) \leq 2^\alpha p(x_i, u)$$

and

$$p(x_{i+1}, x_i) \leq (1+r)^\alpha p(x_{i+1}, u).$$

Therefore,

$$w(x_i, x_{i+1}) \leq (1+r)^\alpha p(x_i, u) + 2^\alpha p(x_{i+1}, u). \quad (1)$$

Let $(u, x_{i_1}), (u, x_{i_2}), \dots, (u, x_{i_h})$ be those edges of $T(u)$, which remain in $T_r(u)$. Then, it has the following that

$$d(u, x_{i_1}) \geq rd(u, x_{i_2}) \geq r^2 d(u, x_{i_3}) \geq \dots \geq r^{h-1} d(u, x_{i_h}).$$

Therefore,

$$\begin{aligned}
& p(u, x_{i_1}) + p(u, x_{i_2}) + \cdots + p(u, x_{i_h}) \\
& \leq p(u, x_{i_1})(1 + r^{-\alpha} + \cdots + r^{-(h-1)\alpha}) \\
& \leq p(u, x_{i_1}) \frac{1}{1 - r^{-\alpha}} \\
& \leq p(u, x_1) \frac{r^\alpha}{r^\alpha - 1}.
\end{aligned}$$

Combining with (1), it can be obtained

$$\begin{aligned}
w(T_r) &= \sum_{(u, x_i) \in E(T_r(u))} w(u, x_i) \\
&+ \sum_{(x_i, x_{i+1}) \in E(T_r(u))} w(x_i, x_{i+1}) \\
&\leq p(u, x_1) \frac{r^\alpha}{r^\alpha - 1} + \sum_{(u, x_i) \in E(T_r(u))} p(x_i, u) \\
&+ (1+r)^\alpha \sum_{(x_i, x_{i+1}) \in E(T_r(u))} (p(x_i, u) + p(x_{i+1}, u)) \\
&\leq P(T(u)) \max \left(\frac{r^\alpha}{r^\alpha - 1}, 2^\alpha + (1+r)^\alpha \right).
\end{aligned}$$

Now, let $T_r = \cup_{u \in V^i(T)} T_r(u)$. Then

$$\begin{aligned}
w(T_r) &\leq \max \left(\frac{r^\alpha}{r^\alpha - 1}, 2^\alpha + (1+r)^\alpha \right) \sum_{u \in V^i(T)} P(T(u)) \\
&\leq \max \left(\frac{r^\alpha}{r^\alpha - 1}, 2^\alpha + (1+r)^\alpha \right) \cdot 2P(T). \quad \square
\end{aligned}$$

2.5 Unicast

Calinescu et al. [15] studied a unicast problem as follows:

MIN-POWER UNICAST: Given a set V of points, a source $s \in V$, and a destination $t \in V$, find a minimum power-cost path between s and t .

First, they note that the minimum power-cost path is different from the minimum cost path. For example, consider three points s , x , and t which are three vertices of an isosceles triangle with a right angle at x . Suppose $\alpha = 2$ and $c = 1$. Then, $\|(s, t)\|^2 = \|(s, x)\|^2 + \|(x, t)\|^2$. However,

$$P((s, t)) = 2\|(s, t)\|^2,$$

and

$$P(s, x, t) = p(s) + p(x) + p(t) = \frac{3}{2}\|(s, t)\|.$$

Indeed, the calculation of power-cost of a path is not a simple sum of edge cost. This fact results in that the algorithm for computing the shortest path does not work on the geometric network on node set V .

Calinescu et al. [15] gave the following solution. First, construct an edge-weighted graph G with node set $V \times V$ and following edges:

- For every three nodes $u, v, w \in V$, construct an arc $((u, v), (u, w))$ with weight $\max(0, \|(u, w)\|^\alpha - \|(u, v)\|^\alpha)$.
- For every two nodes $u, v \in V$, construct two arcs $((u, v), (v, u))$ and $((v, u), (u, v))$ each with weight $\|(u, v)\|^\alpha$.

For example, the graph G on above three points s , x , and t contains nine nodes, and the path (s, x, t) corresponds to the path $((s, x), (x, s), (x, t), (t, x))$ with total weight equal to the power-cost $P(s, x, t)$ of path (s, x, t) .

Now, MIN-POWER UNICAST is reduced to the shortest path problem in constructed graph G . Therefore, *it has the following that*

Theorem 10 MIN-POWER UNICAST can be solved in polynomial-time.

3 Minimum Energy with Unadjustable Power

In real systems of wireless ad hoc networks, after network is configured, each node is assigned with a transmission power level which would not be changed for any request such as broadcast and unicast. In wireless sensor networks, the communication range of each sensor is often fixed and hence also in the same situation that the minimization of total energy consumption is under given transmission power level at each node. When power level at every node is the same, the total energy consumption can be measured through the total node-weight at transmission nodes. In this section, several optimization problems of this type is presented.

3.1 Operations in Multiradio Multichannel Wireless Networks

In works of Li et al. [7], Cagalj et al. [6], and Liang [5], the total energy consumption of broadcast in static wireless ad hoc networks is minimized under the same given power level at every node. The following is a general formulation.

Min Node-Weight Arborescence: Given a node-weighted digraph $G = (V, E)$ with a source node s , construct an out-arborescence rooted at s with the minimum total weight of internal nodes.

Liang [5] gave an efficient algorithm evaluated through simulations. Cagalj et al. [6] presented a polynomial-time $O(\log^3 n)$ -approximation. Li et al. designed a polynomial-time approximation algorithm with performance ratio $(1 + 2 \ln(n - 1))$. Those algorithms are presented for *Min Node-Weight Arborescence* in non-weighted case. However, it is easy to extend them to weighted case, corresponding to that power levels at different nodes may be different.

Li et al. [60] and Ma et al. [61] study the broadcast in multiradio multichannel multihop wireless networks. In a multiradio multichannel multihop wireless network (MR-MC network), there are totally C non-overlapping orthogonal frequency channels, $1, 2, \dots, c$. Each node v is equipped with some omni-directional radio interfaces. Suppose the following are done after configuration:

- (a) Each node v is assigned with a channel subset $B(v)$. This means that each node v can receive messages through any channel in subset $B(v)$ and transmit messages using any channel in subset $B(v)$.
- (b) The transmission power is fixed at a certain level $p(v)$ at each node v .

Assume that compared with transmission power, energy spending for receiving data at each node can be ignored. Then, energy consumption at each node v is $p(v)$ multiplying the number of forward channels, that is, channels used for sending messages.

The problem studied in [60, 61] is to find a forward scheme, that is, for each node v , find a channel subset $F(v) \subseteq B(v)$ such that data can be broadcasted from a given source node to all wireless nodes and the total energy consumption is minimized. For any broadcasting tree T , let $NL(T)$ be the set of its internal nodes. The following is the formulation.

BROADCAST IN MR-MC NETWORK: Consider a digraph $G = (V, E)$ with a source node s and a set of radio channels $C = \{1, 2, \dots, c\}$. Given each node v a set $B(v)$ of radio channels and a power $p(v)$, find an assignment $F : V \rightarrow 2^C$ to minimize

$$W(T) = \sum_{v \in NL(T)} |F(v)| \cdot p(v)$$

under the following conditions:

- (1) $F(v) \subseteq B(v)$ for every $v \in V$
- (2) There exists a broadcast tree T from source s satisfying condition that for each arc $(u, v) \in T$, $F(u) \cap B(v) \neq \emptyset$.

While Li et al. [60] assumed that transmission power level at all nodes is the same, Ma et al. [61] did not. Actually, *Min Node-Weight Arborescence* and **BROADCAST IN MR-MC NETWORK** are equivalent in terms of approximability.

Theorem 11 *There is a polynomial-time ρ -approximation for **BROADCAST IN MR-MC NETWORK** if and only if there is a polynomial-time ρ -approximation for **MIN NODE-WEIGHT ARBORESCENCE**.*

Proof **MIN NODE-WEIGHT ARBORESCENCE** can be seen as a special case of **BROADCAST IN MR-MC NETWORK**; in such a case, $B(v)$ contains only one channel which is the same for all $v \in V$. Thus, “only if” part is trivial.

To show “if” part, BROADCAST IN MR-MC NETWORK is transformed to a special case of MIN NODE-WEIGHT ARBORESCENCE.

In this transformation, first construct an auxiliary digraph $G_{\text{aux}} = (V_{\text{aux}}, E_{\text{aux}})$ based on the input digraph $G = (V, E)$ of BROADCAST IN MR-MC NETWORK as follows.

- (1) Create a source node s^* with weight $p(s^*) = 0$.
- (2) For each node v , create $|B(v)|$ nodes v_i with weight $p(v_i) = p(v)$ for $i \in B(v)$.
- (3) Connect s^* to every s_i for the input source node s of BROADCAST IN MR-MC NETWORK with arc (s^*, s_i) .
- (4) For each arc $(u, v) \in E$, add arcs (u_i, v_j) for $i \in B(u)$, $j \in B(v)$, and $i \in B(v)$.

Thus, one has

$$V_{\text{aux}} = \{s^*\} \cup \bigcup_{v \in V} \{v_j \mid j \in B(v)\}$$

with $p(s^*) = 0$, and $p(v_j) = p(v)$ for $j \in B(v)$, and

$$E_{\text{aux}} = \bigcup_{(u,v) \in E} \{(u_i, v_j) \mid (u, v) \in E, i \in B(u) \cap B(v), j \in B(v)\}.$$

This transformation has the following property: There exists a forward scheme F with cost at most W containing a broadcast tree from node s if and only if G_{aux} contains a broadcast tree with weight at most W from node s^* .

To show “only if” part of this property, suppose the existence of forward scheme F with cost at most W containing a broadcast tree T . Construct a broadcast tree T_{aux} for G_{aux} as follows:

Initially, set $T_{\text{aux}} = \emptyset$.

- (1) For each $i \in B(s)$, add arc (s^*, s_i) to T_{aux} .
- (2) For each arc (u, v) in T , choose a channel i from $F(u) \cap B(v)$ and add arcs (u_i, v_j) to T_{aux} for every $j \in B(v)$.

From the above construction, it is easy to see that for any path from s to a node v , T_{aux} contains a path from s^* to node v_j for any $j \in B(v)$. Therefore, T_{aux} being a broadcast tree follows from T being a broadcast tree. Moreover, only in case that arc (u, v) exists (i.e., u is an internal node of T) and $i \in F(u)$, u_i can be an internal node of T_{aux} . Therefore, the total weight of internal nodes other than s^* in T_{aux} is at most $\sum_{u \in NL(T)} |F(u)| \cdot p(u) \leq W$.

To show the “if” part of the property, suppose G_{aux} contains a broadcast tree T_{aux} with weight at most W . For each $u \in V$, define

$$F(u) = \{i \mid (u_i, v_j) \text{ in } T_{\text{aux}}\}.$$

Denote by T the subgraph induced by the arc set

$$\{(u, v) \mid (u_i, v_j) \text{ in } T_{\text{aux}}\}.$$

It follows from the definition of G_{aux} that for any arc (u, v) in the arc set, $i \in F(u) \cap B(v)$, and hence, $F(u) \cap B(v) \neq \emptyset$. Now, T is claimed to be a broadcast tree.

In fact, for each node $v \in V$, T_{aux} contains a path from s^* to v_j for any $j \in B(v)$ since T_{aux} is a broadcast tree. This path would induce a path from s to v . Finally, it is noticed that $i \in F(u)$ if and only if u_i is an internal node of T_{aux} . Therefore, $W(F) \leq \sum_{(u,i) \in NL(T_{\text{aux}})} p(u) \leq W$.

Now, the theorem can be proved easily from the property. First, it follows immediately from the property that the minimum weight for a forward scheme F to contain a broadcast tree is opt if and only if the minimum cost of an arborescence in G^{opt} is opt .

Next, suppose an out-arborescence T_{aux} with cost at most $\rho \cdot opt$ for G_{aux} can be computed in polynomial-time. Then, from the proof of the property, one may see that a forward scheme F can be constructed in polynomial time to contain a broadcast tree and to have cost at most $\rho \cdot opt$. This means that if there is a polynomial-time ρ -approximation for MIN NODE-WEIGHT ARBORESCENCE on input G_{aux} with weight $p(u_i) = p(u)$, then there is a polynomial-time ρ -approximation for BROADCAST IN MR-MC NETWORK. \square

The above theorem indicates that the study on broadcast in multiradio multi-channel wireless networks can be reduced to the study on broadcast in single-radio single-channel wireless networks. Du et al. [62] found that this is true for all network operations, such as gossiping, convergecast, multicast, and unicast.

For example, gossiping is another one of most fundamental communication primitives in wireless network. The energy-efficient gossiping in multiradio multichannel wireless networks can be formulated as follows.

GOSSIPING IN MR-MC NETWORK: Consider a graph $G = (V, E)$. Each node v is associated with a set $B(v)$ of radio channels and a weight $p(v)$. The problem is to assign each node v with a subset $F(v)$ of $B(v)$ such that the directed graph with node set V and arc set $\{(u, v) \in E \mid F(u) \cap B(v) \neq \emptyset\}$ is strongly connected.

Its corresponding problem in single-radio single-channel wireless networks can be described in the following way.

MIN-WEIGHT SCDS: Given a directed graph $G = (V, E)$ with nonnegative node weight $c : V \rightarrow R^+$, find the minimum-weight strongly connected dominating set where a subset C of nodes is called a *strongly connected dominating set* if every node $v \in V - C$ has an arc (v, u) ending at $u \in C$ and also an arc (w, v) coming from $w \in C$.

Du et al. [62] showed their equivalence as follows.

Theorem 12 *If min-weight SCDS has a polynomial-time ρ -approximation if and only if gossiping in MR-MC networks has a polynomial-time ρ -approximation.*

3.2 Energy-Efficient Virtual Backbone

There are several works on energy-efficient virtual backbone, that is, energy-efficient connected dominating sets in the literature [63–66]. Since they assume that the communication range at every station has the same radius, the minimization on

total power is the same as the minimization on the size of connected dominating set, which has been studied extensively. For a current research status, the reader may see [67].

3.3 Connected Sensor Cover

Motivated from study on coverage problem [68, 69] and connected dominating set problem, Gupta, Das, and Gu [70] proposed the connected sensor cover problem as follows.

CONNECTED SENSOR COVER (AREA): Consider a sensor network $G = (V, E)$ of n sensors in the Euclidean plane and a region A . Each sensor $s_i \in V$ is associated with a sensing region S_i . The problem is to select the minimum number of sensors such that A is covered by sensing regions of selected sensors and the subgraph induced by selected sensors is connected.

A subregion of A is called a *subelement* if the maximal region cannot be cut by any sensor's sensing region. Put a target in each subelement. Then, a set of sensors can cover A if and only if it can cover all targets. This means that the area coverage problem can be reduced to a target coverage problem.

CONNECTED SENSOR COVER (TARGET): Consider a sensor network $G = (V, E)$ of n sensors and a set of m targets. Each sensor $s_i \in V$ is associated with a sensing set S_i of targets. The problem is to select the minimum number of sensors such that every target appears in the sensing set of at least one selected sensor and the subgraph induced by selected sensors is connected.

If the sensor set and the target set is identical, then this problem is exactly the minimum connected dominating set problem. Thus, CONNECTED SENSOR COVER is NP-hard because the minimum connected dominating set problem is NP-hard. Gupta, Das, and Gu [70] designed a greedy approximation with the following performance.

Theorem 13 *There is a polynomial-time approximation for CONNECTED SENSOR COVER with performance ratio at most $(r - 1)(1 + \log m)$ where r is the link radius of the sensor network, that is, the maximum communication distance between two sensors with overlapping sensing sets.*

Wang et al. [71] and Zhang and Hou [72] study CONNECTED SENSOR COVER (AREA) in the case that each sensor s is associated with two disks with center s , sensing region and communication region. They found an interesting property specially for area coverage as follows.

Theorem 14 *Suppose the radius of communication region is at least twice the radius of sensing region. If a connected region is covered by a subset S of sensors, then S induced a connected subgraph.*

Zhou, Das, and Gupta [73] extended this property to k -coverage.

Theorem 15 Suppose the radius of communication region is at least twice the radius of sensing region. If a subset of sensors has k -coverage to a connected region, that is, every point in the region receives sensing from at least k sensors, then it induced a k -connected subgraph, that is, for any two sensors there are at least k node-disjoint paths between them.

Proof Suppose S is a subset of sensors with k -coverage. Then, for any $k-1$ sensors s_1, \dots, s_{k-1} , $S - \{s_1, \dots, s_{k-1}\}$ is still able to cover the target connected region. Therefore, the subgraph induced by $S - \{s_1, \dots, s_{k-1}\}$ is connected. \square

By above theorem, one need only to pay attention to coverage when design algorithm to construct connected sensor coverage. While many efforts [70, 72, 73] were made on design of greedy approximation with performance ratio $O(\log n)$, Funke et al. [74] showed that greedy approximation cannot have performance ratio better than $\Omega(\log n)$. They also proposed a fine grid algorithm with the following performance.

Theorem 16 Suppose in a uniform wireless sensor network, the communication radius R_c is at least twice of the sensing radius R_s . Also, suppose with reduced sensing radius $R'_s = (1 - \epsilon/\sqrt{2})R_s$, the target connected region can still be fully covered. Then, the fine grid algorithm can produce an approximation solution for CONNECTED SENSOR COVER with size within a factor of 12 from the size of optimal solution for sensing radius R'_s .

Zhou, Das, and Gupta [75] study CONNECTED SENSOR COVER in sensor networks with variable sensing radius and communication radius. Alam and Hass [76] and Bai et al. [77] study the problem in three-dimensional networks. Yu et al. [78] consider the problem in networks with directional antennas. Related to CONNECTED SENSOR COVER, there is also a sensor placement problem [79–82]. For a nice and complete survey about the problem, the reader may refer to [83].

4 Maximum Lifetime

With a certain resource, adjust usage of resource to maximize the lifetime of the system. This is a classic type of resource management problems. In wireless networks, there are also many optimization problems of this type on energy efficiency. Some examples will be discussed in this section.

4.1 Sensor Coverage

A result of randomly deploying a very large number of sensors into a certain region, possibly by an aircraft, is the existence of redundant sensors. Usually, such a result is utilized to increase the lifetime of network system by scheduling

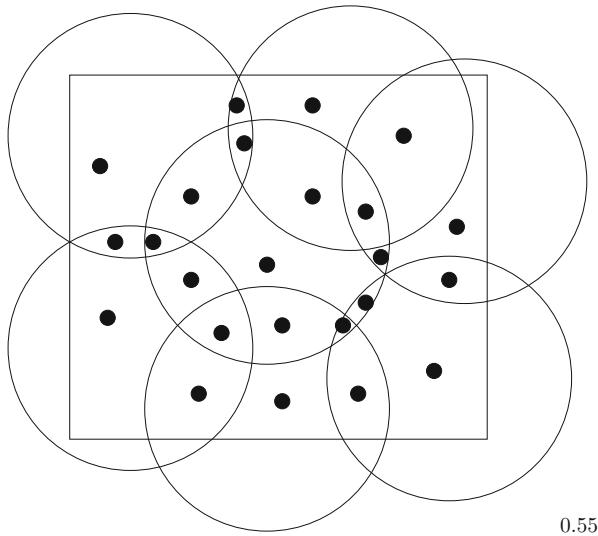


Fig. 5 Target subareas are represented by target points

active/sleep time of sensors such that sensor can be organized to work in different period so that the target area (or target points) can be covered in every period. This sensor activation scheduling with coverage of targets is called the *sensor coverage* problem. According to target type, there are two types problems: area coverage [71, 72, 84–86] and target coverage [8–10, 79, 87]. By target coverage, one usually means that targets are points.

AREA-COVERAGE. Given an area and a set of sensors each with a sensing area, find a sensor activation scheduling to maximize the lifetime of the sensor network where the sensor network is alive if and only if the given area is fully covered by active sensors.

TARGET-COVERAGE. Given a set of (point) targets and a set of sensors each with a sensing area, find a sensor activation scheduling to maximize the lifetime of the sensor network where the sensor network is alive if and only if the given targets are fully covered by active sensors.

AREA-COVERAGE can be reduced to TARGET-COVERAGE as follows: For every sensor, draw the boundary of its sensing area to divide the target area into small subareas; each subarea is a maximal area such that no boundary point of sensor's sensing area lies in its interior. Place a point-target in the interior of each subarea ([Fig. 5](#)). Then, a subarea is covered if and only if placed point-target is covered. Therefore, the given area is fully covered if and only if all point-targets are fully targets.

In the following, studying will be focused on TARGET-COVERAGE.

A simple scheduling is to divide sensors into disjoint subsets each of which fully covers all targets, called a *sensor cover*. For this type of scheduling, the sensor is activated only once, that is, once the sensor is activated, it keeps active until it dies. The specific formulation corresponding this scheduling is as follows.

DISJOINT SENSOR COVER: Given n targets r_1, \dots, r_n and m sensors s_1, \dots, s_m each with a sensing subset of targets, find the maximum number of disjoint sensor covers.

This problem is NP-hard. Various heuristics and approximation algorithms have been given in [85, 87, 88].

Cardei et al. [8] found that it is possible to increase the lifetime if each sensor is allowed to make more changes between active and sleeping. For example, suppose that sensor s_1 covers targets r_1 and r_2 , sensor s_2 covers targets r_2 and r_3 , and sensor s_3 covers targets r_1 and r_3 . Using disjoint-set strategy, the lifetime of this system is 1 unit, which is the lifetime of each sensor, because there do not exist two disjoint sensor covers. However, the following scheduling can obtain lifetime 1.5 unit time: In the first period of 0.5 unit time, sensors s_1 and s_2 are activated and sensor s_3 is put in sleeping; in the second period of 0.5 unit time, sensors s_2 keeps being active and s_3 is activated and sensor s_1 is put in sleeping; in the third period of 0.5 unit time, sensors s_1 is activated again and s_3 is still active.

The specific formulation on scheduling of nondisjoint sensor covers is as follows [8].

SENSOR-COVER-SCHEDULING: Given m sensors s_1, \dots, s_m each with a sensing subset of target set $\{r_1, \dots, r_n\}$, find a family of sensor cover S_1, \dots, S_p with time lengths t_1, \dots, t_p in $[0, 1]$, respectively, to maximize $t_1 + \dots + t_p$ subject to every sensor appears in cover sets with total time length at most 1.

This is still an NP-hard problem, which can be easily written as an 0-1 integer programming.

$$\begin{aligned} & \text{Maximize } t_1 + \dots + t_p \\ & \text{subject to } \sum_{j=1}^p x_{ij} t_j \leq 1 \text{ for } i = 1, \dots, m \\ & \quad \sum_{i \in C_k} x_{ij} \geq 1 \text{ for } k = 1, \dots, n \text{ and } j = 1, \dots, p, \\ & \quad x_{ij} = 0, 1, \text{ and } t_j \geq 0, \end{aligned}$$

where for each target r_k , C_k is the set of sensors whose sensing subset contains r_k . x_{ij} is the indicator that sensor s_i is in sensor cover S_j . The first inequality constraint means that for each sensor s_i , the total active time length does not exceed 1. The second inequality constraint means that at any time period, active sensors should form a sensor cover. Note that the first inequality constraint together with $t_j \geq 0$ implies $t_j \leq 1$. Therefore, $t_j \leq 1$ does not need to put in constraints explicitly.

Above integer programming is not linear since $x_{ij} t_j$ is not a linear term. However, it can be easily linearized by setting $y_{ij} = x_{ij} t_j$ as follows:

$$\text{Maximize } t_1 + \dots + t_p$$

$$\begin{aligned}
& \text{subject to } \sum_{j=1}^p y_{ij} \leq 1 \text{ for } i = 1, \dots, m \\
& \quad \sum_{i \in C_k} y_{ij} \geq t_j \text{ for } k = 1, \dots, n \text{ and } j = 1, \dots, p, \\
& \quad y_{ij} = 0, \text{ or } t_j, \text{ and } t_j \geq 0.
\end{aligned}$$

Cardei et al. [8] designed a heuristic using LP relaxation as follows.

By relaxing the condition $y_{ij} = 0, t_j$ to $0 \leq y_{ij} \leq t_j$, the following linear programming can be obtained from the above integer linear programming.

$$\begin{aligned}
& \text{Maximize } t_1 + \dots + t_p \\
& \text{subject to } \sum_{j=1}^p y_{ij} \leq 1 \text{ for } i = 1, \dots, m \\
& \quad \sum_{i \in C_k} y_{ij} \geq t_j \text{ for } k = 1, \dots, n \text{ and } j = 1, \dots, p, \\
& \quad 0 \leq y_{ij} \leq t_j.
\end{aligned}$$

The solution (y_{ij}^*, t^*) of this linear programming may not satisfy condition $y_{ij} = 0$ or t_j and even not necessarily satisfies $t_j \leq 1$. Therefore, the following rounding technique is employed: Set $t_j = \min_{k=1, \dots, m} \max_{i \in C_k} y_{ij}^*$. For each k , choose a i_k from C_k such that $y_{ij}^* \geq t_j$ and set $y_{ikj} = t_j$. For remaining i , set $y_{ij} = 0$.

This algorithm is employed iteratedly to utilize the remaining sensor lifetime. To do so, Cardei et al. [8] replace inequality $\sum_{j=1}^p x_{ij} t_j \leq 1$ by $\sum_{j=1}^p x_{ij} t_j \leq e_i$ where e_i is the remaining lifetime of sensor s_i .

No theoretical bound has been established for this heuristic. However, computational experiments show that the heuristic produces good solutions.

The nondisjoint model is better than the disjoint model also due to an interesting fact discovered in [89] that putting a sensor alternatively in active and sleeping states in a proper way may double its lifetime since the battery could be recovered in certain level during sleeping.

Berman et al. [90, 91] first designed an approximation algorithm for SENSOR-COVER-SCHEDULING with theoretical bound.

Theorem 17 *There exists a polynomial-time approximation for SENSOR-COVER-SCHEDULING with performance ratio $O(\log n)$ where n is the number of sensors.*

Ding et al. [92] found constant approximation for geometric version.

Theorem 18 Suppose all sensors and targets lie in the Euclidean plane and the sensing area of every sensor is a disk with the same radius. Then, there exists a polynomial-time $(4 + \epsilon)$ -approximation for SENSOR-COVER-SCHEDULING.

Both result of Berman et al. [90, 91] and Ding et al. [92] employ an algorithm of Garg and Könemann [93] and their theorem, which will be introduced in next subsection.

4.2 Garg–Könemann Algorithm

The first one who found the application of Garg–Könemann Algorithm in the study of lifetime maximization type of problems is Calnescu et al. [59].

Consider a graph $G = (V, E)$. A function $c : V \times V \rightarrow R^+$ is called a power requirement if edge (u, v) is established if and only if both the power at u and the power at v are not less than $c(u, v)$. With a power requirement c , $G = (V, E, c)$ is called a power requirement graph.

Calnescu et al. [59] consider the following general form of the minimum power problem and the maximum lifetime problem.

POWER ASSIGNMENT: Given a power requirement graph $G = (V, E, c)$, and a connectivity constraint Q , find a power assignment $p : V \rightarrow R^+$ to minimize the total power $\sum_{u \in V} p(u)$ under condition that supported subgraph H satisfies the connectivity constraint Q .

NETWORK LIFETIME: Given a power requirement graph $G = (V, E, c)$, a battery supply $b : V \rightarrow R^+$ and a connectivity constraint Q , find a feasible power schedule $PT = \{(p_1, t_1), (p_2, t_2), \dots, (p_m, t_m)\}$ to maximize $\sum_{i=1}^m t_i$, where schedule PT is feasible if $\sum_{i=1}^m p_i(u) \leq b(u)$ for every $u \in V$.

Suppose \mathcal{S} is the set of all subgraphs satisfying constraint Q . For each $H \in \mathcal{S}$, let $p_H : V \rightarrow R^+$ be the power assignment required by establishing subgraph H . Then, NETWORK LIFETIME can be formulated a linear integer programming as follows:

$$\begin{aligned} & \max \sum_{H \in \mathcal{S}} t_H \\ & \text{subject to } \sum_{H \in \mathcal{S}} p_H(u)t_H \leq b(u) \\ & \quad t_H \geq 0, \forall H \in \mathcal{S}. \end{aligned}$$

Now, the sum of coefficients for each column is exactly the total power for establishing H , which is called the *length* of the column. Computing the minimum column length is exactly the POWER ASSIGNMENT problem. Consider the following weighted version of POWER ASSIGNMENT.

WEIGHTED POWER ASSIGNMENT: Given a power requirement graph $G = (V, E, c)$, a connectivity constraint Q , and a node weight $y : E \rightarrow R^+$, find a power assignment $p : V \rightarrow R^+$ to minimize the total power $\sum_{u \in V} p(u)y(u)$ under condition that supported subgraph H satisfies the connectivity constraint Q .

Suppose WEIGHTED POWER ASSIGNMENT has an f -approximation. Then, a $(1 + \varepsilon)f$ -approximation can be computed for NETWORK LIFETIME. Indeed, the linear programming formulation of NETWORK LIFETIME falls in a general form of linear programming studied by Garg and Könemann [93] as follows:

$$\max\{c^T x \mid Ax \leq b, x \geq 0\}$$

where A is an $m \times n$ positive matrix and b and c are m -dimensional and n -dimensional positive vectors, respectively. n can be exponentially larger than m . However, The following Garg and Könemann Algorithm [93] assumes that the linear programming is actually implicitly given by vector b .

Garg–Könemann Algorithm

input: A vector $b \in R^m$, $\varepsilon > 0$, an algorithm F

for computing f -approximation of $\min_j \sum_{i=1}^m A(i, j) \frac{y(i)}{c(j)}$.
 Initially, $\delta = (1 + \varepsilon)((1 + \varepsilon)m)^{-1/\varepsilon}$.
 for $i = 1, \dots, m$, $y(i) \leftarrow \frac{\delta}{b(i)}$,
 $D \leftarrow m\delta$,
 $j = 0$;

while $D < 1$ **do**

find a column A_q using f -approximation F
 choose index p to minimize $\frac{b(p)}{A_q(p)}$
 $j \leftarrow j + 1$
 $x^j \leftarrow \frac{b(p)}{A_q(p)}$
 $A^j \leftarrow A_q$
 for $i = 1, \dots, m$, $y(i) \leftarrow y(i)(1 + \varepsilon \cdot \frac{b(p)}{A_q(p)} / \frac{b(i)}{A_q(i)})$
 $D \leftarrow b^T y$

output $\{(A^j, \frac{x^j}{\log_{1+\varepsilon} \frac{1+\varepsilon}{\delta}})\}_{j=1}^k$.

This algorithm would give a $(1 + \varepsilon)f$ -approximation for NETWORK LIFETIME. It is also proved in [93] that $k \leq m$.

Theorem 19 Consider a connectivity constraint Q and a power requirement graph $G = (V, E, c)$. For any f -approximation algorithm F for WEIGHTED POWER ASSIGNMENT, there exists a $(1 + \varepsilon)f$ -approximation algorithm for the corresponding NETWORK LIFETIME.

Garg–Könemann Algorithm successfully reduces the lifetime maximization problem to the minimum weighted power assignment problem. This gives a quite useful technique to study approximation of the lifetime maximization. With this technique, several approximation with good performance have been found in the literature [90–92, 94]. This approach may also be used in the problem studied in [95–97] to obtain some new results.

4.3 Coverage Breach

In above sensor coverage problems, the complete coverage is required during the system lifetime period. However, the number of randomly deployed sensors is often very large and the core network may not be able to provide a bandwidth large enough for receiving data from all sensors or from all sensors in a sensor cover. In this situation, a complete coverage is sometimes not available. Therefore, the partial coverage becomes an interesting subject in the study of sensor networks.

A target is in *breach* if it is not monitored by any sensor. There are three coverage breach optimization formulations studied in the literature [9, 10, 85, 90, 91]. For each formulation, there are two models, disjoint model and nondisjoint model. The following are formulations in disjoint model:

Max-lifetime coverage with breach (disjoint model): Given a set of sensors, $\{s_1, \dots, s_m\}$, a set of targets, $\{r_1, \dots, r_n\}$, and a nonnegative integer b , the problem is to partition the sensor set into maximum number of subsets such that for each sensor subset, there are at most b breaches.

Min coverage breach (disjoint model): Given a set of sensors, $\{s_1, \dots, s_m\}$, a set of targets, $\{r_1, \dots, r_n\}$, and a positive integer k , the problem is to partition the sensor set into k subsets to minimize the total number of breaches.

Min-max coverage breach (disjoint model): Given a set of sensors, $\{s_1, \dots, s_m\}$, a set of targets, $\{r_1, \dots, r_n\}$, and a positive integer k , the problem is to partition the sensor set into k subsets such that the maximum number of breaches for each sensor subset is minimized.

Corresponding formulations in disjoint model are as follows.

Max-lifetime coverage with breach (nondisjoint model): Given a set of sensors, $\{s_1, \dots, s_m\}$, a set of targets, $\{r_1, \dots, r_n\}$, and a nonnegative integer b , the problem is to find a family of sensor subsets S_1, \dots, S_p with time lengths t_1, \dots, t_p in $[0, 1]$, respectively, to maximize $t_1 + \dots + t_p$ subject to that every sensor appears in those sensor subsets with total time length at most 1 and for each of those sensor subsets, there are at most b breaches.

Min coverage breach (nondisjoint model): Given a set of sensors, $\{s_1, \dots, s_m\}$, a set of targets, $\{r_1, \dots, r_n\}$, and a positive integer k , the problem is to find a family of sensor subsets S_1, \dots, S_p with time lengths t_1, \dots, t_p in $[0, 1]$, respectively, to minimize $t_1 b_1 + \dots + t_p b_p$ where b_i is the number of breaches for sensor subset S_i , subject to that every sensor appears in those sensor subsets with total time length at most 1 and $t_1 + \dots + t_p = k$.

Min-max coverage breach (nondisjoint model): Given a set of sensors, $\{s_1, \dots, s_m\}$, a set of targets, $\{r_1, \dots, r_n\}$, and a positive integer k , the problem is to find a family of sensor subsets S_1, \dots, S_p with time lengths t_1, \dots, t_p in $[0, 1]$, respectively, to minimize $\max(b_1, \dots, b_p)$ where b_i is the number of breaches for sensor subset S_i , subject to that every sensor appears in those sensor subsets with total time length at most 1 and $t_1 + \dots + t_p = k$.

For min–max type of problem, there is a variation that the minimization is on the maximum breach interval, where a breach interval is a time period during which there exists a target not covered by any sensor at any time.

All above problems are NP-hard. However, not many efforts have been made on design of approximation algorithms [9, 10].

4.4 Maximum Lifetime Routing

The shortest path routing is very popular and a good routing strategy in many cases. However, it may not be good for energy efficiency in wireless networks. In fact, the wireless equipment has limited energy supply, often with batteries. Every message transmission would have energy consumption. If a routing strategy allows repeatedly using some stations, then network lifetime would be shorten. Thus, a maximum lifetime routing problem appeared [98–111].

Let (s, t) represent a routing request that a message is required to send from sensor s to sensor t . For simplicity of discussion, it is often assumed that all messages have the same length, which require the same amount of time to transmit.

Max-lifetime routing: Given a wireless sensor network and a sequence of routing requests $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$, find a routing strategy to maximize j such that routing requests $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ are successfully routed.

The problem has an online version [110] as follows.

Max-lifetime online routing: Given a wireless sensor network and a sequence of routing requests coming online, find an online routing strategy to maximize j such that first j routing requests can be successfully routed.

More information about the above problems can be found in [112, 119].

5 Conclusion

In this chapter, we have argued many combinatorial optimization problems in the study of power issues in wireless networks. The issues have been discussed into three categories: (1) minimum total power with adjustable-power nodes, (2) minimum total power with unadjustable-power nodes, and (3) maximum lifetime with unadjustable-power nodes. Latest results have been shown in the introduction of these problems.

Cross-References

- ▶ [Connected Dominating Set in Wireless Networks](#)
 - ▶ [Greedy Approximation Algorithms](#)
 - ▶ [Network Optimization](#)
 - ▶ [Optimization in Multi-Channel Wireless Networks](#)
 - ▶ [Optimizing Data Collection Capacity in Wireless Networks](#)
-

Recommended Reading

1. S.G. Stolberg, Obama Unveils Wireless Expansion Plan, *New York Times*, February 10, 2011
2. J.E. Wieselthier, G.D. Nguyen, A. Ephremides, On the construction of energy-efficient broadcast and multicast trees in wireless networks, in *IEEE INFOCOM*, 2000

3. J.E. Wieselthier, G.D. Nguyen, A. Ephremides, Energy-efficient broadcast and multicast trees in wireless networks. *Mobile Network Appl.* **7**, 481–492 (2002)
4. P.-J. Wan, G. Calinescu, X.-Y. Li, O. Frieder, Minimum energy broadcast routing in static ad-hoc wireless networks, in *Proc. IEEE INFOCOM*, 2001, pp. 1162–1171
5. W. Liang, Constructing minimum-energy broadcast trees in wireless ad hoc networks, in *Proc. MOBIHOC*, 2002
6. M. Cagalj, J.-P. Hubaux, C. Enz, Minimum-energy broadcast in all wireless networks: NP-completeness and distributed issues, in *Proc. MOBICOM*, 2002, pp. 172–182
7. D. Li, X. Jia, H. Liu, Energy efficient broadcast routing in static ad hoc wireless networks. *IEEE Trans. Mob. Comput.* **3**(2), 144–151 (2004)
8. M. Cardei, M. Thai, Y. Li, W. Wu, Energy-efficient target coverage in wireless sensor networks, in *IEEE INFOCOM*, 2005
9. M.X. Cheng, L. Ruan, W. Wu, Achieving minimum coverage breach under bandwidth constraints in wireless sensor networks, in *IEEE INFOCOM*, 2005
10. M.X. Cheng, L. Ruan, W. Wu, Coverage breach problems in bandwidth-constrained sensor networks. *TOSN* **3**(2), 12 (2007)
11. T. Rappaport, *Wireless Communications: Principles and Practice* (Prebtice-Hall, Englewood Cliff, NY, 1996)
12. D.M. Blough, M. Leoncini, G. Resta, P. Santi, On the symmetric range assignment problem in wireless ad hoc networks, in *Proc. of the IFIP 17th World Computer Congress TC1 Stream / 2nd IFIP International Conference on Theoretical Computer Science (TCS)*, 2002, pp. 71–82
13. L.M. Kirousis, E. Kranakis, D. Krizanc, A. Pelc, Power consumption in packer radio networks. *Theor. Comput. Sci.* **243**, 289–305 (2000)
14. G. Calinescu, Minimum-power strong connectivity. *Lect. Note Comput. Sci.* **6302**, 67–80 (2010)
15. G. Calinescu, I. Mandoiu, A. Zelikovsky, Symmetric Connectivity with minimum power consumption in radio networks, in *2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)* (Kluwer Academic, 2002), pp. 119–130
16. A. Zelikovsky, An 11/6-approximation algorithm for the network Steiner tree problem. *Algorithmica* **9**, 463–470 (1993)
17. D.-Z. Du, Y.-J. Zhang, On better heuristic for Steiner minimum trees *Math. Program. Ser. B* **57**, 193–202 (1992)
18. G. Robin, A. Zelikovsky, Improved Steiner tree approximation in graphs, in *SODA'2000*, 2000, pp. 770–779
19. H.J. Prummel, A. Steger, A new approximation algorithm for the Steiner tree problem with performance ration 5/3. *J. Algorithm* **36**, 89–101 (2000)
20. E. Althaus, G. Calinescu, I.I. Mandoiu, S. Prasad, N. Tchervenski, A. Zelikovsky, Power efficient range assignment in ad-hoc wireless networks, in *Proceedings of the IEEE Wireless Communications and Networking Conference(WCNC)*, IEEE, New Orleans, USA, March 2003, pp. 1889–1894
21. D.-Z. Du, On component-size bounded Steiner trees. *Discrete Appl. Math.* **60**, 131–140 (1995)
22. M. Li, S.L. Huang, X. Sun, X. Huang, Performance evaluation for energy efficient topologic control in ad hoc wireless networks. *Theor. Comput. Sci.* **326**(1–3), 399–408 (2004)
23. A. Borchers, D.-Z. Du, The k-Steiner ratio in graphs. *SIAM J. Comput.* **26** 857–869 (1997)
24. G. Kortsarz, V.S. Mirrokni, Z. Nutov, E. Tsankov, Approximating minimum-power degree and connectivity problems, in *LATIN*, 2008, pp. 423–435
25. M.T. Hajijadhai, G. Kortsarz, V.S. Mirrokni, Z. Nutov, Power optimization for connectivity problems. *Math. Program.* **110**, 195–208 (2007)
26. J. Byrka, F. Grandoni, T. Rothvoss, L. Sanitá, An improved LP-based approximation for Steiner tree, in *Proc. of 42th STOC*, 2010, pp. 583–592
27. W.T. Chen, N.F. Huang, The strongly connection problem on multihop packet radio networks. *IEEE Trans. Comm.* **37**(3), 293–295 (1989)

28. A.E.F. Clementi, P. Penna, R. Silvestri, Hardness results for the power range assignment problem in packet radio networks, in *RANDOM 1999 and APPROX 1999*, ed. by D.S. Hochbaum, K. Jansen, J.D.P. Rolim, A. Sinclair, LNCS, vol. 1671 (Springer, Heidelberg, 1999), pp. 197–208
29. A.E.F. Clementi, P. Penna, R. Silvestri, On the power assignment problem in radio networks. *J. Mob. Netw. Appl.* **9**(2), 125–140 (2004)
30. X. Cheng, B. Narahari, R. Simha, M.X. Cheng, D. Liu, Strong minimum energy topology in wireless sensor networks: NP-completeness and heuristic. *IEEE Trans. Mobile Comput.* **2**(3), 248–256 (2003)
31. R. Ramanathan, R. Rosales-Hain, Topology control of multihop wireless networks using transmit power adjustment, in *IEEE INFOCOM(2)*, 2000, pp. 404–413
32. R. Wattenhofer, L. Li, P. Bahl, Y.-M. Wang, Distributed topology control for wireless multihop ad-hoc networks, in *IEEE INFOCOM*, 2001
33. M.T. Hajiaghayi, N. Immorlica, V.S. Mirrokni, Power optimization in fault-tolerant topology control algorithms for wireless multi-hop networks, in *MOBICOM*, 2003, pp. 300–312
34. P. Carmi, M.J. Katz, Power assignment in radio networks with two power levels. *Algorithmica* **47**, 183–201 (2007)
35. C. Ambühl, A.E.F. Clementi, P. Penna, G. Rossi, R. Silvestri, On the approximability of the range assignment problem on radio networks in presence of selfish agents. *Theor. Comput. Sci.* **343**, 27–41 (2005)
36. M.X. Cheng, M. Cardei, J. Sun, X. Cheng, L. Wang, Y. Xu, D.-Z. Du, Topology control of ad hoc wireless networks for energy efficiency. *IEEE Trans. Comput.* **53**(12), 1629–1635 (2004)
37. D.-Z. Du, K.-I. Ko, X. Hu, *Design and Analysis of Approximation Algorithms*, Springer Optimization and Its Applications, Vol. 62 (Springer, New York, 2011)
38. I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, New results for energy-efficient broadcasting in wireless networks, in *Proceedings of the 13th Annual International Symposium on Algorithms and Computation (ISAAC'02)*, LNCS 2518 (Springer, 2002), pp. 332–343
39. I. Caragiannis, C. Kaklamanis, P. Kanellopoulos, Energy-efficient wireless network design. *Theor. Comput. Syst.* **39**(5), 593–617 (2006)
40. A.E.F. Clementi, P. Crescenzi, P. Penna, G. Rossi, P. Voccia, On the Complexity of Computing Minimum Energy Consumption Broadcast Subgraphs, in *Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, LNCS, vol. 2010, 2001, pp. 121–131
41. M. Cagalj, J. Hubaux, C. Enz, Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues, in *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking (MobiCom'02)* (ACM Press, 2002), pp. 172–182
42. C. Ambühl, An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks, in *Proceedings of 32nd International Colloquium on Automata, Languages and Programming*, 2005, pp. 1139–1150
43. H. Cai, Y. Zhao, On approximation ratios of minimum-energy multicast routing in wireless networks. *J. Combin. Optim.* **9**(3), 243–262 (2005)
44. M. Flammini, R. Klasing, A. Navarra, S. Perennes, Improved approximation results for the minimum energy broadcasting problem, in *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing*, 2004
45. R. Klasing, A. Navarra, A. Papadopoulos, S. Perennes, Adaptive broadcast consumption (abc), a new heuristic and new bounds for the minimum energy broadcast routing problem, in *Proceedings of 31st International Colloquium on Automata, Languages and Programming*, 2004, pp. 866–877
46. A. Navarra, Tighter bounds for the minimum energy broadcasting problem, in *Proceedings of 32nd International Colloquium on Automata, Languages and Programming*, 2005, pp. 313–322
47. A. Navarra, 3-d minimum energy broadcasting, in *SIROCCO 2006*, LNCS, vol. 4056, 2006, pp. 240–252

48. J.H. Conway, N.J.A. Sloane, “The kissing number probe” and “Bounds on kissing numbers”, Ch. 2.1 and Ch. 13, in *Sphere Packings, Lattices, and Groups*, 3rd edn. (Springer, New York, 1998)
49. C. Zhong, *Sphere Packing* (Springer, New York, 1999)
50. I. Caragiannis, M. Flammini, L. Moscardelli, An exponential improvement on the MST heuristic for minimum energy broadcasting in ad hoc wireless networks, in *ICALP*, 2007, pp. 447–458
51. F. Dai, J. Wu, Efficient broadcasting in ad hoc wireless networks using directional antennas. *IEEE Trans. Parallel Distr. Syst.* **17**(4), 335–347 (2006)
52. D. Li, Z. Li, L. Liu, Energy efficient broadcast routing in ad hoc sensor networks with directional antennas, in *WASA2008*, LNCS, vol. 5258, 2008, pp. 29–39
53. Z. Li, D. Li, Minimum energy broadcast routing in ad hoc and sensor networks with directional antennas, in *Proceedings of COCOA’2009* (Yellow Mountain, China, 2009), pp. 507–518
54. S. Guo, O. W. W. Yang, Minimum-energy multicast in wireless ad hoc networks with adaptive antennas: MILP formulations and heuristic algorithms. *IEEE Trans. Mobile Comput.* **5**(4), 333–346 (2006)
55. S.K. Ghosh, Energy efficient broadcast in distributed ad-hoc wireless networks, in *Proc. of IEEE 11th International Conference on Computational Science and Engineering (SCE)*, 2008, pp. 394–401
56. J. Wu, F. Dai, Broadcasting in ad hoc networks based on self-pruning, in *Proc. IEEE INFOCOM*, 2003, pp. 2240–2250
57. M. Agarwal, J.H. Cho, L. Gao, J. Wu, Energy efficient broadcast in wireless ad hoc networks with Hitch-hiking, in *Proc. IEEE INCOCOM*, 2004
58. Y.-C. Tseng, Y.-N. Chang, B.-H. Tzeng, Energy-efficient topology control for wireless ad hoc sensor networks. *J. Inform. Sci. Eng.* **20**, 27–37 (2004)
59. G. Calinescu, S. Kapoor, A. Olshevsky, A. Zelikovsky, Network lifetime and power assignment in ad-hoc wireless networks, in *Proc. of European Symp on Algorithms (ESA’03)*, LNCS 2832, September 2003, pp. 114–126
60. L. Li, B. Qin, C. Zhang, H. Li, Efficient Broadcasting in Multi-radio Multi-channel and Multi-hop Wireless Networks Based on Self-pruning, Lecture Notes in Computer Science, 2007, vol 4782, pp. 484–495
61. C. Ma, D. Li, H. Du, H. Ma, Y. Wang, W. Lee, Energy efficient broadcast in multiradio multichannel wireless networks, in *IEEE INFOCOM*, 2012
62. H. Du, L. Sheng, J. Kim, Energy efficient operations in multiradio multichannel wireless networks. *J. Combin. Optim.* (to appear in)
63. Y. Zeng, X. Jia, Y. He, Energy efficient distributed connected dominating sets construction in wireless sensor networks, in *Proceedings of the 2006 International Conference on Wireless Communication and Mobile Computing (IWCWC’06)*, July 3–6, Canada, 2006, pp. 797–802
64. J. Wu, F. Dai, M. Gao, I. Stojmenovic, On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks, in *Proc. IEEE Int’l ICOPP*, 2001, 346–356
65. J. Lee, B. Mans, Energy-efficient virtual backbone for reception-aware MANET, in *Proc. Vehicular Technology Conference*, 2006, pp. 1097–1101
66. J. Kim, J. Lim, K. Chae, Energy-efficient code dissemination using minimal virtual backbone in sensor networks, in *Proc. International Conference on Broadband, Wireless Computing, Communication and Applications*, 2010, pp. 149–154
67. H. Du, L. Ding, W. Wu, D. Kim, P.M. Pardalos, J. Willson, Connected dominating set in wireless networks, Chapter 42, in *Handbook of Combinatorial Optimization*, 2nd edn. (Springer, New York)
68. K. Lieska, E. Laitinen, J. Lahteenmaki, Radio coverage optimization with genetic algorithms, in *Proc. of IEEE Int. Symp. on Personal Indoor, and Mobile Radio Communication (PIMRC)*, 1998

69. S. Meguerdichian, F. Koushanfar, M. Potkonjak, M. Srivastava, Coverage problem in wireless ad-hoc sensor networks, in *Proc. of the Conf. on Computer Communications (INFOCOM)*, 2001
70. H. Gupta, S. Das, Q. Gu, Connected sensor cover: self-organization of sensor networks for efficient query execution, in *MobiHoc'03*, 2003, pp. 189–200
71. X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, C. Gill, Integrated coverage and connectivity configuration in wireless sensor networks, in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, Sensys'03*, Los Angeles, CA, November 2003, pp. 28–39
72. H. Zhang, J.C. Hou, Maintaining sensing coverage and connectivity in large sensor networks. *Ad Hoc Sensor Wireless Networks* **1**, 89–124 (2005)
73. Z. Zhou, S. Das, H. Gupta, Connected k-coverage problem in sensor networks, in *Proceedings of the 13th IEEE International Conference on Computer Communications and Networks (ICCCN'04)*, Chicago, IL, October 2004, pp. 373–378
74. S. Funke, A. Kesselman, F. Kuhn, Z. Lotker, M. Segal, Improved approximation algorithms for connected sensor cover. *Wireless Network* **13**(2), 153–164 (2007)
75. Z. Zhou, S. Das, H. Gupta, Variable radii connected sensor cover in sensor networks, in *Proc. of the IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, 2004
76. S. Alam, Z. Hass, Coverage and connectivity in three-dimensional networks, in *Proc. of ACM MobiCom*, 2006
77. X. Bai, C. Zhang, D. Xuan, J. Teng, W. Jia, Low-connectivity and full-coverage three-dimensional wireless sensor networks, in *Proc. of ACM MobiHoc*, 2009
78. Z. Yu, J. Teng, X. Bai, D. Xuan, W. Jia, Connected coverage in wireless networks with directional antennas, in *Proc. of IEEE INFOCOM*, 2011
79. K. Kar, S. Banerjee, Node placement for connected coverage in sensor networks, in *Proc. of WiOpt 2003: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003
80. X. Bai, S. Kumar, D. Xuan, Z. Yun, T.H. Lai, Deploying wireless sensor to achieve both coverage and connectivity, in *Proc. of ACM MobiHoc*, 2006
81. X. Bai, D. Xuan, Z. Yun, T.H. Lai, W. Jia, Complete optimal deployment patterns for full-coverage and k -connectivity ($k \leq 6$) wireless sensor networks, in *Proc. of ACM MobiHoc*, 2008
82. X. Han, X. Cao, E. Loyd, et al., Deploying directional sensor networks with guaranteed connectivity and coverage, in *Proc. of IEEE SECON*, 2008
83. A. Ghosh, S.K. Das, Coverage and connectivity issues in wireless sensor networks: a survey. *Pervasive Mobile Comput.* **4**(3), 303–334 (2008)
84. J. Carle, D. Simplot, Energy efficient area monitoring by sensor networks. *IEEE Comput.* **37**(2), 40–46 (2004)
85. S. Sljepcevic, M. Potkonjak, Power efficient organization of wireless sensor networks, in *IEEE International Conference on Communications*, 2001, pp. 472–476
86. D. Tian, N.D. Georganas, A coverage-preserving node scheduling scheme for large wireless sensor networks, in *Proc. of 1st ACM Workshop on Wireless Sensor Networks and Applications*, 2002
87. M. Cardei, D.-Z. Du, Improving wireless sensor network lifetime through power aware organization. *ACM Wireless Network* **11**(3), 333–340 (2005)
88. G. Calinescu, R. Ellis, On the lifetime of randomly deployed sensor networks, in *DialM-POMC The Fifth ACM SIGACTSIGOPS International Workshop on Foundation of Mobile Computing*, 2008
89. R. Hahn, H. Reichl, Batteries and power supplies for wearable and ubiquitous computing, in *Proc. 3rd Intl. Symposium on Wearable computers*, 1999
90. P. Berman, G. Calinescu, C. Shah, A. Zelikovsky, Efficient energy management in sensor networks, in *Ad Hoc and Sensor Networks, Wireless Networks and Mobile Computing*, ed. by Y. Xiao, Y. Pan, vol. 2 (Nova Science, 2005), pp. 1–16

91. P. Berman, G. Calinescu, C. Shah, A. Zelikovsky, Power Efficient Monitoring Management in Sensor Networks, in *IEEE Wireless Communication and Networking Conf (WCNC'04)*, Atlanta, March 2004, pp. 2329–2334
92. L. Ding, W. Wu, J. K. Willson, L. Wu, Z. Lu, W. Lee, Constant-approximation for target coverage problem in wireless sensor networks, in *Proc. of The 31st Annual Joint Conf. of IEEE Communication and Computer Society (INFOCOM)*, 2012
93. N. Garg, J. Kemann, Faster and simpler algorithms for multicommodity flows and other fractional packing problems, *Proc. 39th Annual Symposium on the Foundations of Computer Science*, 1998, pp. 300–309
94. D. Brinza, G. Calinescu, S. Tongngam, A. Zelikovsky, Energy-efficient continuous and event-driven monitoring, in *Proc. 2nd IEEE Intl Conf on Mobile Ad-Hoc and Sensor Systems (MASS 2005)*, 2005, pp. 167–169
95. M. Cardei, J. Wu, N. Lu, M.O. Pervaiz, maximum network lifetime with adjustable range, in *IEEE Intl. Conf. on Wireless and Mobile Computing, Networking and Communications (WiMob'05)*, Aug. 2005
96. M. Cardei, J. Wu, Energy-efficient coverage problems in wireless ad hoc sensor networks. *Comput. Comm.* **29**(4), 413–420 (2006)
97. M. Cardei, D. MacCallum, X. Cheng, M. Min, X. Jia, D. Li, D.-Z. Du, Wireless sensor networks with energy efficient organization. *J. Interconnection Network* **3**, 213–229 (2002)
98. J. Chang, L. Tassiulas, Routing for maximum system lifetime in wireless ad-hoc networks, in *37th Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sept. 1999
99. J. Chang, L. Tassiulas, Energy conserving routing in wireless ad-hoc networks, in *IEEE INFOCOM*, 2000
100. J. Chang, L. Tassiulas, Fast approximate algorithms for maximum lifetime routing in wireless ad-hoc networks. *IFIP-TC6 Networking 2000, LNCS*, vol 1815 (Springer, New York, 2000), pp. 702–713
101. R. Kannan, L. Ray, R. Kalidindi, S. Iyengar, Energy threshold constrained geographic routing protocol for wireless sensor networks. *Signal Process. Lett.* **1**(1), 79–84 (2003)
102. R. Kannan, S. Iyengar, Game theoretical models for reliable path-length and energy constrained routing with data aggregation in wireless sensor networks, in *IEEE JSAC*, 2004
103. M. Maleki, K. Dantu, M. Pedram, Power-aware source routing protocol for mobile ad hoc networks, in *SLPED'02*, 2002
104. S. Singh, M. Woo, C. Raghavendra, Power-aware routing in mobile ad hoc networks, in *ACM/IEEE MOBICOM*, 1998
105. A. Spyropoulos, C. Raghavendra, Energy efficient communications in ad hoc networks using directional antenna, in *IEEE INFOCOM*, 2002
106. I. Stojmenovic, X. Lin, Power-aware localized routing in wireless networks. *IEEE Trans. Parallel Distr. Syst.* **12**(11), 1122–1133 (2000)
107. C. Toh, H. Cobb, D. Scott, Performance evaluation of battery-life-aware routing optimization schemes for wireless ad hoc networks, in *IEEE ICC'01*, 2001
108. G. Zussman, A. Segall, Energy efficient routing in ad hoc disaster recovery networks, in *IEEE INFOCOM*, 2003
109. J. Park, S. Sahni, Maximum lifetime broadcasting in wireless networks. *IEEE Trans. Comput.* **54**(9), 1081–1090 (2005)
110. J. Park, S. Sahni, An online heuristic for maximum lifetime routing in wireless sensor networks. *IEEE Trans. Comput.* **55**(8), 1048–1056 (2006)
111. J. Park, S. Sahni, Maximum lifetime broadcasting in wireless networks, in *ACS/IEEE Intl. Conf. on Computer Systems and Applications (AICCSA)*, 2005
112. A. Michail, T. Ephremide, Energy-efficient routing for connection-oriented traffic in wireless ad-hoc networks. *Mobile Network Appl.* **8**, 517–533 (2003)

113. M. Kalantari, M. Shayman, Energy efficient routing in wireless sensor networks, in *Proc. Conference on Information Sciences and Systems*, March 2004
114. R. Madan, S. Lall, Distributed algorithms for maximum lifetime routing in wireless sensor networks. *IEEE Trans. Wireless Comm.* **5**(8), 2185–2193 (2006)
115. R. Madan, S. Cui, S. Lall, and A. Goldsmith, Cross-layer design for lifetime maximization in interference-limited wireless sensor networks, in *Proc. IEEE INFOCOM*, vol. 3, March 2005, pp. 1964–1975
116. Z. Wang, S. Basagni, E. Melachrinoudis, C. Petrioli, Exploiting sink mobility for maximizing sensor networks lifetime, in *Proc. 38th HICSS*, 2005
117. I. Papadimitriou, L. Georgiadis, Maximum lifetime routing to mobile sink in wireless sensor networks, in *Proc. SoftCOM*, Sept. 2005
118. M. Gatzianas, L. Georgiadis, A distributed algorithm for maximum lifetime routing in sensor networks with mobile sink. *IEEE Trans. Wireless Comm.* **7**(3), 984–993 (2008)
119. J. Li, S. Dey, Lifetime optimization for wireless sensor networks with outage probability constraints, in *Proc. 12th European Wireless*, 2006

Equitable Coloring of Graphs

Ko-Wei Lih

Contents

1	Introduction	1200
2	Bipartite Graphs	1202
3	Trees	1204
4	The Equitable Δ -Coloring Conjecture	1206
5	Split Graphs	1207
6	Outerplanar Graphs	1208
7	Planar Graphs	1209
8	Graphs with Low Degeneracy	1211
9	Graphs of Bounded Treewidth	1213
10	Kneser Graphs	1214
11	Interval Graphs and Others	1217
12	Random Graphs	1219
13	Graph Products	1219
13.1	Square Product	1219
13.2	Cross Product	1221
13.3	Strong Product	1222
14	List Equitable Coloring	1223
15	Graph Packing	1225
16	Equitable Δ -Colorability of Disconnected Graphs	1228
17	More on the Hajnal-Szemerédi Theorem	1233
18	Applications	1235
19	Related Notions of Coloring	1236
19.1	Equitable Edge-Coloring	1236
19.2	Equitable Total-Coloring	1238
19.3	Equitable Defective Coloring	1240
19.4	Equitable Coloring of Hypergraphs	1241
20	Conclusion	1242
	Cross-References	1242
	Recommended Reading	1242

K.-W. Lih

Institute of Mathematics, Academia Sinica, Taipei, Taiwan

e-mail: makwlih@sinica.edu.tw

Abstract

If the vertices of a graph G are colored with k colors such that no adjacent vertices receive the same color and the sizes of any two color classes differ by at most one, then G is said to be equitably k -colorable. The equitable chromatic number $\chi_=(G)$ is the smallest integer k such that G is equitably k -colorable. In the first introduction section, results obtained about the equitable chromatic number before 1990 are surveyed. The research on equitable coloring has attracted enough attention only since the early 1990s. In the subsequent sections, positive evidence for the important equitable Δ -coloring conjecture is supplied from graph classes such as forests, split graphs, outerplanar graphs, series-parallel graphs, planar graphs, graphs with low degeneracy, graphs with bounded treewidth, Kneser graphs, and interval graphs. Then three kinds of graph products are investigated. A list version of equitable coloring is introduced. The equitable coloring is further examined in the wider context of graph packing. Appropriate conjectures for equitable Δ -coloring of disconnected graphs are then studied. Variants of the well-known and significant Hajnal and Szemerédi Theorem are discussed. A brief summary of applications of equitable coloring is given. Related notions, such as equitable edge coloring, equitable total coloring, equitable defective coloring, and equitable coloring of uniform hypergraphs, are touched upon. This chapter ends with a short conclusion section. This survey is an updated version of Lih [102].

1 Introduction

A graph G consists of a vertex set $V(G)$ and an edge set $E(G)$. All graphs considered in this chapter are finite, loopless, and without multiple edges. Let $|G|$ and $\|G\|$ denote the number of vertices, also known as the *order*, and the number of edges of the graph G , respectively. If the vertices of a graph G can be partitioned into k sets V_1, V_2, \dots, V_k such that each V_i is an *independent set* (none of its vertices are adjacent), then G is said to be *k -colorable* and the k sets are called its *color classes*. Equivalently, a coloring can be viewed as a function $\pi : V(G) \rightarrow \{1, 2, \dots, k\}$ such that adjacent vertices are mapped to distinct numbers. The mapping π is said to be a (proper) *k -coloring*. All pre-images of a fixed i , $1 \leq i \leq k$, form a color class. The smallest number k , denoted by $\chi(G)$, such that G is k -colorable is called the *chromatic number* of G . The graph G is said to be equitably colored with k colors, or *equitably k -colorable*, if there is a k -coloring whose color classes satisfy the condition $\|V_i\| - \|V_j\| \leq 1$ for every pair V_i and V_j . The smallest integer k for which G is equitably k -colorable, denoted by $\chi_=(G)$, is called the *equitable chromatic number* of G . Suppose that the graph G of order n is equitably colored with k colors. If $n = qk + r$, where $0 \leq r < k$, then there are exactly r color classes of size $q + 1$ and exactly $k - r$ color classes of size q . The sizes of the color classes can be enumerated as $\lfloor n/k \rfloor, \lfloor (n+1)/k \rfloor, \dots, \lfloor (n+k-1)/k \rfloor$ in a nondecreasing order. Note that $\lfloor (n+t-1)/k \rfloor = \lceil (n+t-k)/k \rceil$ for $1 \leq t \leq k$.

The notion of an equitable coloring was first introduced in [115] by W. Meyer. His motivation came from Tucker's paper [138], in which vertices represent garbage collection routes and two such vertices are adjacent when the corresponding routes should not be run on the same day. Meyer thought that it would be desirable to have an approximately equal number of routes run on each of the 6 days in a week.

Let $\deg_G(v)$, or $\deg(v)$ for short, denote the degree of the vertex v in the graph G and $\Delta(G) = \max\{\deg(v) \mid v \in V(G)\}$. Let $\lceil x \rceil$ and $\lfloor x \rfloor$ denote, respectively, the smallest integer not less than x and the largest integer not greater than x . The main result obtained by Meyer was that a tree T can be equitably colored with $\lceil \Delta(T)/2 \rceil + 1$ colors. However, there were gaps in his proof. According to Guy's report [63], Egginton could extend Meyer's result to show that a tree T can be equitably colored with k colors, provided $k \geq \lceil \Delta(T)/2 \rceil + 1$. A finer result about trees is the following theorem by Bollobás and Guy [16].

Theorem 1 *A tree T is equitably 3-colorable if $|T| \geq 3\Delta(T) - 8$ or $|T| = 3\Delta(T) - 10$.*

The most interesting contribution made in Meyer's paper is to propose the following conjecture. It is also called the *Equitable Coloring Conjecture (ECC)*. Let K_n and C_n denote, respectively, a complete graph and a cycle on n vertices.

Conjecture 1 *Let G be a connected graph. If G is neither a complete graph K_n nor an odd cycle C_{2n+1} , then $\chi_=(G) \leq \Delta(G)$.*

Meyer was successful in verifying the *ECC* only for graphs with six or fewer vertices. Apparently the motivation of the *ECC* came from the following fundamental Brooks' Theorem [19].

Theorem 2 *Let G be a connected graph. If G is neither a complete graph K_n nor an odd cycle C_{2n+1} , then $\chi(G) \leq \Delta(G)$.*

The well-known Hajnal and Szemerédi Theorem [64], when rephrased in terms of equitable colorings, had already shown the following before Meyer's paper.

Theorem 3 *A graph G (not necessarily connected) is equitably k -colorable if $k \geq \Delta(G) + 1$.*

Let $\chi_=(G)$ denote the smallest integer n such that G is equitably k -colorable for all $k \geq n$. Then an equivalent formulation of **Theorem 3** is that $\chi_=(G) \leq \Delta(G) + 1$ holds for any graph G . There is a notable contrast between the equitable colorability and the ordinary colorability: $\chi_=(G)$ may in fact be greater than $\chi_-(G)$. This will be demonstrated later. Therefore, it makes sense to introduce the notion $\chi_*(G)$, and it shall be called the *equitable chromatic threshold* of G .

In an entirely different context, de Werra [39] treated color sequences and the majorization order among them. His results have consequences in equitable colorability. A sequence of nonnegative integers $h = (h_1, h_2, \dots, h_k)$ is called a *color sequence* for a given graph G if the following conditions hold:

1. $h_1 \geq h_2 \geq \dots \geq h_k \geq 0$.
2. There is a k -coloring of G such that the color classes V_1, V_2, \dots, V_k satisfy $|V_i| = h_i$ for $1 \leq i \leq k$.

The *majorization order*, also known as the *dominance order*, is a widely used notion in measuring the evenness of distributions. Marshall and Olkin [110] offer a comprehensive treatment of majorization. Let $\alpha : a_1 \geq a_2 \geq \dots \geq a_k \geq 0$ and $\beta : b_1 \geq b_2 \geq \dots \geq b_k \geq 0$ be two sequences of nonnegative integers. The sequence α is said to be *majorized* by the sequence β if the following two conditions hold:

1. $\sum_{i=1}^j a_i \leq \sum_{i=1}^j b_i$ for any j , $1 \leq j < k$.
2. $\sum_{i=1}^k a_i = \sum_{i=1}^k b_i$.

Let $K_{m,n}$ denote the complete bipartite graph whose parts are of size m and size n , respectively. A *claw-free* graph is a graph containing no $K_{1,3}$ as an induced subgraph. One of de Werra's results is the following:

Theorem 4 *Let G be a claw-free graph and $h = (h_1, h_2, \dots, h_k)$ be a color sequence of G . Then any sequence of nonnegative integers $h' = (h'_1, h'_2, \dots, h'_k)$ is also a color sequence if h' is majorized by h .*

Combining Theorems 2 and 4, it follows that, for a claw-free graph G , G is equitably k -colorable for all $k \geq \chi(G)$, or equivalently $\chi^*(G) = \chi_-(G) = \chi(G)$. Although this fact can be shown directly, it was first implicitly implied in de Werra's paper. It follows immediately that the *ECC* holds for claw-free graphs. Since every line graph is claw-free, the *ECC* holds for line graphs in particular. This was also obtained in Wang and Zhang [149].

This ends the history of pre-1990 activities on the equitable coloring of graphs.

2 Bipartite Graphs

A graph is called *r-partite* if its vertex set can be partitioned into r -independent sets V_1, V_2, \dots, V_r and *complete r-partite*, denoted by K_{n_1, n_2, \dots, n_r} , if every vertex in V_i is adjacent to every vertex in V_j whenever $i \neq j$ and $|V_i| = n_i \geq 1$ for every $1 \leq i \leq r$. By convention it is always assumed that $r \geq 2$ and $1 \leq n_1 \leq n_2 \leq \dots \leq n_r$. A graph is said to be *complete multipartite* if it is complete r -partite for some r . A bipartite graph is synonymous with a 2-partite graph. Let I_n denote the graph consisting of n isolated vertices. Then I_n is equitably k -colorable with color classes of size $\lfloor x \rfloor$ or $\lceil x \rceil$ if and only if $\lfloor x \rfloor \leq \lfloor n/k \rfloor \leq \lceil n/k \rceil \leq \lceil x \rceil$ or, equivalently, if and only if $\lceil n/\lceil x \rceil \rceil \leq k \leq \lfloor n/\lfloor x \rfloor \rfloor$.

Lih and Wu [102] first settled the *ECC* for bipartite graphs.

Theorem 5 *If a connected bipartite graph G is different from any complete bipartite graph $K_{n,n}$, then G can be equitably colored with $\Delta(G)$ colors.*

Theorem 6 *The complete bipartite graph $K_{n,n}$ can be equitably colored with k colors if and only if $\lceil n/\lfloor k/2 \rfloor \rceil - \lfloor n/\lceil k/2 \rceil \rfloor \leq 1$.*

The proof for the complete bipartite case is straightforward by considering the appropriate sizes of the color classes. One interesting point to note is that, for $k = n = \Delta(K_{n,n})$, the difference involved in [Theorem 6](#) is 0 when n is even and is 2 when n is odd. In view of [Theorem 5](#), one can conclude that, except the complete bipartite graphs $K_{2m+1,2m+1}$, every connected bipartite graph G can be equitably colored with $\Delta(G)$ colors. Clearly, $\chi(K_{2m+1,2m+1}) = \chi_-(K_{2m+1,2m+1}) = 2$, yet $\chi_-^*(K_{2m+1,2m+1}) = 2m+2$. There is a gap between the equitable chromatic number and the equitable chromatic threshold. Nevertheless, *ECC* holds for connected bipartite graphs.

In many cases, the equitable chromatic number is below the maximum degree. If additional constraints are imposed upon the graph, a better bound for the equitable chromatic number could be obtained. The following is a result of this type:

Theorem 7 *Let $G = G(X, Y)$ be a connected bipartite graph with two parts X and Y such that $\|G\| = e$. Suppose $|X| = m \geq n = |Y|$ and $e < \lfloor m/(n+1) \rfloor (m-n) + 2m$. Then $\chi_-(G) \leq \lceil m/(n+1) \rceil + 1$.*

The bound for the equitable chromatic number in the above theorem is indeed better than $\Delta(G)$ when there are at least two edges. The following conjecture was made by B.-L. Chen in a personal communication:

Conjecture 2 *Let G be a connected bipartite graph. Then $\chi_-(G) \leq \lceil \Delta(G)/2 \rceil + 1$.*

Chen proved its validity when the maximum degree is at least 53. It is also trivial to see that the conjecture holds for complete bipartite graphs. The Meyer-Egginton result about trees gives another positive evidence.

Wang and Zhang [[149](#)] established the validity of *ECC* for complete multipartite graphs. The exact value of the equitable chromatic number of a complete multipartite graph K_{n_1, n_2, \dots, n_r} was determined by Lam et al. [[99](#)].

Theorem 8 *Let M be the largest natural number such that $n_i \pmod M < \lceil n_i/M \rceil$ for $1 \leq i \leq r$. Then $\chi_-(K_{n_1, n_2, \dots, n_r}) = \sum_{i=1}^r \lceil n_i/(M+1) \rceil$.*

Blum et al. [[12](#)] also obtained a formula for $\chi_-(K_{n_1, n_2, \dots, n_r})$. For $r \geq 2$, let $K_{r(n)}$ denote the complete multipartite graph $K_{\underbrace{n, n, \dots, n}_r}$. [Theorem 6](#) has been generalized to $K_{r(n)}$ in [[104](#)].

Theorem 9 Let integers $n \geq 1$ and $k \geq r \geq 2$. Then $K_{r(n)}$ is equitably k -colorable if and only if $\left\lceil \frac{n}{\lfloor k/r \rfloor} \right\rceil - \left\lfloor \frac{n}{\lceil k/r \rceil} \right\rfloor \leq 1$.

Complete multipartite graphs also furnish us with examples to show that the inequalities $\chi(G) \leq \chi_=(G) \leq \chi^*(G)$ can be strict. For instance [103], let $G_1 = K_{\underbrace{1,1,\dots,1}_m, 2n+1}$, $G_2 = K_{\underbrace{2n+1,2n+1,\dots,2n+1}_m}$, and $G_3 = K_{\underbrace{2n+1,2n+1,4n+2,4n+2,\dots,4n+2}_m}$. Then

1. $\chi(G_1) = m + 1 < \chi_=(G_1) = \chi^*(G_1) = m + n + 1$.
2. $\chi(G_2) = \chi_=(G_2) = m < \chi^*(G_2) = m(n + 1)$.
3. $\chi(G_3) = m + 2 < \chi_=(G_3) = 2(m + 1) < \chi^*(G_3) = (m + 1)(2n + 1) + 1$.

As to the gap between the chromatic number and the equitable chromatic number, Wang and Zhang [149] proposed the following:

Conjecture 3 For any graph G , $\chi_-(G) - \chi(G) \leq \lfloor \Delta(G)/2 \rfloor$.

The upper bound is sharp in the sense that it can be attained by a star $K_{1,2n+1}$.

3 Trees

A graph is said to be *nontrivial* if it contains at least one edge. There is a natural way to regard a nontrivial tree T as a bipartite graph $T(X, Y)$. The technique used to prove the *ECC* for connected bipartite graphs can be applied to find the equitable chromatic number of a nontrivial tree when the sizes of the two parts differ by at most one. First try to cut the parts into classes of nearly equal size. If there are vertices remaining, then one can manage to find nonadjacent vertices in the opposite part to form a class of the right size. The following was established in Chen and Lih [26].

Theorem 10 Let $T = T(X, Y)$ be a nontrivial tree satisfying $|X| - |Y| \leq 1$. Then $\chi_-(T) = \chi^*(T) = 2$.

When the sizes of the two parts differ by more than one, the determination in [26] for the equitable chromatic number of a tree needs extra notation. For any vertex u of a graph G , an independent set containing u is called a *u-independent set*. Let $\alpha_u(G)$ denote the maximum size of a u -independent set in G .

Now suppose that G is partitioned into $\chi_-(G)$ parts of independent sets. Let v be an arbitrary vertex of G . Then the part containing v has size at most $\alpha_v(G)$, and other parts have size at most $\alpha_v(G) + 1$. It follows that $|G| \leq \alpha_v(G) + (\chi_-(G) - 1)$ ($\alpha_v(G) + 1$) = $\chi_-(G)(\alpha_v(G) + 1) - 1$.

Lemma 1 Let v be an arbitrary vertex of G , then $\chi_-(G) \geq \lceil (|G| + 1) / (\alpha_v(G) + 1) \rceil$.

An induction based on [Theorem 1](#) leads to the following:

Theorem 11 *Let $T = T(X, Y)$ be a tree satisfying $|X| - |Y| > 1$. Then $\chi_-(T) = \chi_-^*(T) = \max\{3, \lceil(|T| + 1)/(\alpha_u(T) + 1)\rceil\}$, where u is an arbitrary vertex of maximum degree in T .*

The following result was proved by Miyata et al. in an unpublished manuscript [116] without using [Theorem 1](#):

Theorem 12 *Let T be a tree and $k \geq 3$ be an integer. Then T is equitably k -colorable if and only if $k \geq \max_{v \in V(T)} \lceil(|T| + 1)/(\alpha_v(T) + 1)\rceil$.*

The inequality in the above theorem can be equivalently described as $\alpha_v(T) \geq \lfloor |T|/k \rfloor$ for any vertex v of T . This can be seen from the following equivalences which hold for any integer k and graph G :

$$\begin{aligned} k \geq \left\lceil \frac{|G| + 1}{\alpha_v(G) + 1} \right\rceil &\Leftrightarrow k \geq \frac{|G| + 1}{\alpha_v(G) + 1} \Leftrightarrow \alpha_v(G) \geq \frac{|G| - k + 1}{k} \Leftrightarrow \alpha_v(G) \\ &\geq \left\lfloor \frac{|G|}{k} \right\rfloor. \end{aligned}$$

Actually, the difference between characterizations in [26] and [116] is only apparent. In view of the following lemma, Chang [22] gave a simplified and unified proof for the more general case of a forest:

Lemma 2 *Let u be a vertex of a forest F . If $\lceil(|F| + 1)/(\alpha_u(F) + 1)\rceil > 3$, then u is the unique vertex of maximum degree in F .*

Theorem 13 *Let F be a forest and $k \geq 3$ be an integer. Then F is equitably k -colorable if and only if $\alpha_v(F) \geq \lfloor |F|/k \rfloor$ for any vertex v of F .*

To determine when a forest F is equitably 2-colorable needs a bit more work than that of a tree. Without loss of generality, suppose that F has r components such that each component tree T_i consists of two parts X_i and Y_i . The objective is to look for a partition of $\{1, 2, \dots, r\}$ into two parts A and B such that $\sum_{i \in A} |X_i| + \sum_{j \in B} |Y_j| = \lfloor |F|/2 \rfloor$.

Hansen et al. [66] introduced the notion of an *m-bounded coloring* of a graph G , i.e., a proper coloring of G such that each color class is of size at most m . The *m-bounded chromatic number* of G , denoted by $\chi_m(G)$, is the smallest number of colors required for an m -bounded coloring of G . This notion of colorability is closely related to equitable colorability via the following observation:

Observation. The graph G has an m -bounded coloring using k colors if and only if the graph G' obtained from G by adding $mk - |G|$ isolated vertices is equitably k -colorable.

The problem of determining the m -bounded chromatic number of a tree was left open in [66]. By modifying the techniques used in [26], Chen and Lih [25] were able to determine the m -bounded chromatic number of a tree.

Theorem 14 Let $T = T(X, Y)$ be a nontrivial tree and let u be a vertex of maximum degree. Then one of the following holds:

1. $\chi_m(T) = 2$ when $m \geq \max\{|X|, |Y|\}$.
2. $\chi_m(T) = \max\{3, \lceil |T|/m \rceil, \lceil (|T| - \alpha_u(T))/m \rceil + 1 \}$ when $m < \max\{|X|, |Y|\}$.

For a nontrivial tree $T = T(X, Y)$, suppose that $|X| = q_1m + r_1$ and $|Y| = q_2m + r_2$, where $0 \leq r_1, r_2 < m$. One can color the part X with $q_1 + 1$ colors and the part Y with $q_2 + 1$ colors. Hence, $\chi_m(T) \leq q_1 + q_2 + 2 \leq \lceil |T|/m \rceil + 1$ colors. On the other hand, it is obvious that $\lceil |T|/m \rceil \leq \chi_m(T)$. A tree T is said to be *class A* if $\chi_m(T) = \lceil |T|/m \rceil$ and *class B* if $\chi_m(T) = \lceil |T|/m \rceil + 1$. Jarvis and Zhou [73] gave an explicit characterization when a tree belongs to class B. Their proof can be used to determine $\chi_m(T)$ in $O(|T|^3)$ time and produce an optimal coloring even if m is part of the input. To make a comparison, it is known [14] that the problem of determining whether a bipartite graph can be m -bounded colored with three colors is NP-complete when m is part of the input. Bentz and Picouleau [10] studied a variation of m -bounded coloring of trees.

4 The Equitable Δ -Coloring Conjecture

Unlike the ordinary colorability of a graph, the equitable colorability does not satisfy monotonicity, namely, a graph can be equitably k -colorable without being equitably $(k + 1)$ -colorable. Therefore, the *ECC* does not fully reveal the true nature of the equitable colorability. It seems that the maximum degree plays a crucial role here. For instance, by Theorems 5 and 6 the following conjecture proposed by Chen et al. [28] holds for bipartite graphs. This conjecture is called the *equitable Δ -coloring conjecture (E Δ CC).*

Conjecture 4 Let G be a connected graph. If G is not a complete graph K_n , or an odd cycle C_{2n+1} , or a complete bipartite graph $K_{2n+1, 2n+1}$, then G is equitably $\Delta(G)$ -colorable.

The conclusion of the E Δ CC can be equivalently stated as $\chi_{=}^*(G) \leq \Delta(G)$. It is also immediate to see that the E Δ CC implies the *ECC*. In Chen, Lih, and Wu [28], E Δ CC was settled for graphs whose maximum degree is at least one-half of the order. The following two lemmas supplied the basic tools for the solution. Let G^c denote the complement graph of G . Let $\delta(G)$ and $\alpha'(G)$ denote the minimum degree and the edge-independence number of G , respectively.

Lemma 3 Let G be a disconnected graph. If G is different from K_n^c and $K_{2n+1,2n+1}^c$ for all $n \geq 1$, then $\alpha'(G) > \delta(G)$.

Lemma 4 Let G be a connected graph such that $|G| > 2\delta(G) + 1$. Suppose the vertex set of G cannot be partitioned into a set H of size $\delta(G)$ and an independent set I of size $|G| - \delta(G)$ such that each vertex of I is adjacent to all vertices of H . Then $\alpha'(G) > \delta(G)$.

Theorem 15 Let G be a connected graph with $\Delta(G) \geq |G|/2$. If G is different from K_n and $K_{2n+1,2n+1}$ for all $n \geq 1$, then G is equitably $\Delta(G)$ -colorable.

As pointed out by Yap, a close examination of the proof of [Theorem 15](#) in [28] reveals that a stronger result was obtained, namely, $\chi_=(G) \leq |G| - \alpha'(G^c) \leq \Delta(G)$. In a similar vein, Yap and Zhang [155] made an analysis of the complement graph and succeeded in verifying the *ECC* for connected graphs G such that $|G|/3 + 1 \leq \Delta(G) < |G|/2$. By combining [Theorem 15](#), their proof can be modified to establish the following stronger result:

Theorem 16 The *EΔCC* holds for all connected graphs G such that $\Delta(G) \geq (|G| + 1)/3$.

Along a different direction, one may try to tackle the *EΔCC* for special classes of graphs. By attaching appropriately chosen auxiliary graphs to a nonregular graph, attention may be restricted to regular graphs due to the following lemma [28]:

Lemma 5 The *EΔCC* holds if it does so for all regular graphs.

If the chromatic number of a connected cubic graph G is 2, then the *EΔCC* has already been established. It only needs to handle cubic graphs having chromatic number 3 to obtain the following [28]:

Theorem 17 The *EΔCC* holds for all connected graphs G such that $\Delta(G) \leq 3$.

In [85], Kierstead and Kostochka extended the above to $\Delta(G) \leq 4$.

5 Split Graphs

There are interesting results dealing with special families of graphs that provide positive evidence for the *EΔCC*. A connected graph G is called a *split* graph if its vertex set can be partitioned into two nonempty subsets $U = \{u_1, u_2, \dots, u_n\}$ and $V = \{v_1, v_2, \dots, v_r\}$ such that U induces a complete graph and V induces an independent set. Denote the split graph G as $G[U; V]$, and always assume that no vertex in V is adjacent to all vertices in U . A family of bipartite graphs $BG(k)$, $k \geq 1$, can be assigned to the given split graph $G[U; V]$ in the following way.

The vertex set of $BG(k)$ is $\{u_{ij} \mid 1 \leq i \leq n \text{ and } 1 \leq j \leq k\} \cup V$, and $\{u_{ij}, v_t\}$ is defined to be an edge of $BG(k)$ if and only if u_i and v_t are nonadjacent in G . Note that $BG(k)$ is a subgraph of $BG(k+1)$. The coloring of a split graph $G[U; V]$ is closely related to independent edges of the graphs $BG(k)$. For instance, any given set of independent edges in $BG(k)$ induces a partial coloring of G in the following standard way. The i -th color is used to color u_i and all those vertices in V that are matched by the edges to some u_{ij} , $1 \leq j \leq k$. Chen, Ko, and Lih [29] proved the following:

Theorem 18 *Let $G[U; V]$ be a split graph such that $|U| = n$ and $|V| = r$. Let $m = \max\{k \mid \alpha'(BG(k)) = kn\}$ if the set in question is nonempty; otherwise let m be zero. Then $\chi^*(G[U; V]) = n + \lceil(r - \alpha'(BG(m+1)))/(m+2)\rceil$.*

Once $\chi^*(G[U; V])$ is known, it is straightforward to verify that split graphs satisfy the $E\Delta CC$.

In [29], the m -bounded chromatic number of a split graph was obtained in addition to its equitable chromatic number.

Theorem 19 *Let $G = G[U; V]$ be a split graph such that $|U| = n$ and $|V| = r$. Let $m \geq 1$ be a given integer. Then $\chi_m(G[U; V]) = n + \lceil(r - \alpha'(BG(m-1)))/m\rceil$.*

6 Outerplanar Graphs

A graph is *planar* if it can be drawn on the Euclidean plane such that edges only meet each other at points representing the vertices of the graph. An *outerplanar* graph is a planar graph that has a drawing on the plane such that every vertex lies on the unbounded face. An *edge subdivision* is the operation of replacing an edge uv by a path uvw of length 2 in which w is a newly added vertex. A *subdivision* of a graph G is a graph obtained from G by a sequence of edge subdivisions. It is well known [18] that a graph is an outerplanar graph if and only if it has no subgraph that is a subdivision of K_4 or $K_{2,3}$. Yap and Zhang [156] settled the $E\Delta CC$ for outerplanar graphs.

Theorem 20 *If G is an outerplanar graph with $\Delta(G) \geq 3$, then G is equitably $\Delta(G)$ -colorable.*

Kostochka [91] proved the following result which answers a question posed at the end of [156]:

Theorem 21 *If G is an outerplanar graph with $\Delta(G) \geq 3$, then $\chi^*(G) \leq \Delta(G)/2 + 1$.*

Note that the bound cannot be weakened even for trees because the star $K_{1,2k-1}$ has no equitable k -coloring. An efficient algorithm for equitable k -coloring of

outerplanar graphs G with maximum degree at least 3 can be extracted from Kostochka's proof whenever $k \geq \Delta(G)/2 + 1$.

A fundamental phenomenon in equitable colorings can be noticed when one examines the equitable chromatic numbers of trees. Apart from $K_{1,n}$, "most" trees admit equitable colorings with few colors. This phenomenon happens for outerplanar graphs, too. Pemmaraju [125] showed the following:

Theorem 22 *An outerplanar graph G is equitably 6-colorable if $\Delta(G) \leq |G|/6$.*

The main stepping stone to Pemmaraju's result involves a special partition of a graph. A partition $V(G) = V_1 \cup V_2$ is called an *equitable 2-forest partition* if $||V_1| - |V_2|| \leq 1$ and each induced subgraph $G[V_i]$ is a forest. The following theorem and conjecture in [125] may have independent interest:

Theorem 23 *Any outerplanar graph has an equitable 2-forest partition.*

Conjecture 5 *The vertex set of any planar graph can be partitioned into two parts V_1 and V_2 such that $||V_1| - |V_2|| \leq 1$ and each part induces an outerplanar subgraph.*

Note that Chartrand et al. [23] have proved that the vertex set of a planar graph can be partitioned into two parts such that each part induces an outerplanar subgraph. They also conjectured that the edge set of a planar graph can be partitioned into two sets such that the subgraph induced by each of the sets is outerplanar. Recently, Gonçalves [60] has proved the validity of this conjecture.

A graph is called *series-parallel* if it contains no subgraph that is a subdivision of K_4 . This class of graphs can be characterized in a number of equivalent ways [18]. Clearly, outerplanar graphs are series-parallel graphs. Zhang and Wu [159] established the $E\Delta CC$ for series-parallel graphs.

Theorem 24 *If G is a series-parallel graph with $\Delta(G) \geq 3$, then G is equitably $\Delta(G)$ -colorable.*

Zhang and Wu also conjectured the following which generalizes Theorem 21:

Conjecture 6 *If G is a series-parallel graph with $\Delta(G) \geq 3$, then $\chi^*(G) \leq \Delta(G)/2 + 1$.*

7 Planar Graphs

To determine whether a planar graph with maximum degree 4 is 3-colorable is NP-complete [58]. For a given planar graph G with maximum degree 4, let G' be obtained from G by adding $2|G|$ isolated vertices. Then G is 3-colorable if and only if G' is equitably 3-colorable. Therefore, it is NP-complete to determine if a given

planar graph with maximum vertex degree 4 has an equitable coloring using at most 3 colors.

Zhang and Yap [160] proved the following:

Theorem 25 *A planar graph G is equitably $\Delta(G)$ -colorable if $\Delta(G) \geq 13$.*

Nakprasit [117] extended the above result to planar graphs with maximum degree at least 9. Thus, the $E\Delta CC$ holds for planar graphs with maximum degree at least 9. One can go further if extra conditions are imposed on the graph.

Theorem 26 ([118, 137]) *Let G be a C_4 -free planar graph. If $\Delta(G) \geq 7$, then the $E\Delta CC$ holds for G .*

Zhu and Bu [162] established the following results. Consequently, the $E\Delta CC$ holds for planar graphs G that are (i) C_3 -free and $\Delta(G) \geq 8$ or (ii) C_4 -free, C_5 -free, and $\Delta(G) \geq 7$.

Theorem 27 *Let G be a C_3 -free planar graph. Then $\chi^*(G) \leq \max\{\Delta(G), 8\}$.*

Theorem 28 *Let G be a C_4 -free and C_5 -free planar graph. Then $\chi^*(G) \leq \max\{\Delta(G), 7\}$.*

The *girth* of a graph G , denoted by $g(G)$, is defined to be the length of a shortest cycle in G . The girth of a forest is ∞ by convention. One may impose conditions on the girth of a planar graph to get tight bound for the equitable chromatic threshold. Wu and Wang [153] first established the following:

Theorem 29 *Let G be a planar graph with $\delta(G) \geq 2$.*

1. *If $g(G) \geq 14$, then $\chi^*(G) \leq 4$.*
2. *If $g(G) \geq 26$, then $\chi^*(G) \leq 3$.*

Luo et al. [108] improved these results further.

Theorem 30 *Let G be a planar graph with $\delta(G) \geq 2$.*

1. *If $g(G) \geq 10$, then $\chi^*(G) \leq 4$.*
2. *If $g(G) \geq 14$, then $\chi^*(G) \leq 3$.*

It remains an open problem to find the best possible girth conditions for 3- or 4-equitable colorability when the planar graph G satisfies $\delta(G) \geq 2$.

A well-known theorem of Grötzsch [61] states that the chromatic number of any planar graph of girth at least 4 is at most 3. Hence, the above theorem has an immediate consequence.

Corollary 1 *Let G be a non-bipartite planar graph with $\delta(G) \geq 2$. Then $\chi(G) = \chi^*(G)$ if $g(G) \geq 14$.*

8 Graphs with Low Degeneracy

A graph with small degeneracy can be regarded as “sparse.” A graph G is said to be d -degenerate if every subgraph H of G has a vertex of degree at most d in H . It is well known that graphs without edges are 0-degenerate, forests are exactly the 1-degenerate graphs, outerplanar graphs are 2-degenerate, and planar graphs are 5-degenerate. It follows from the definition that the vertices of every d -degenerate graph can be ordered as v_1, v_2, \dots, v_n so that for every $i < n$, vertex v_i has at most d neighbors v_j with $j > i$.

The following results were obtained by Zhu and Bu [163]:

Theorem 31 *Let G be a 2-degenerate graph. Then G is equitably 3-colorable if $\|G\| \leq \frac{2}{3}|G|$.*

Theorem 32 *Let G be a 2-degenerate graph. Then G is equitably 4-colorable if $\|G\| \leq \frac{3}{4}|G|$.*

Kostochka and Nakprasit [92] tried to find bounds on the equitable chromatic thresholds for d -degenerate graphs with a given maximum degree. However, the bound in [Theorem 21](#) on outerplanar graphs does not extend to all 2-degenerate graphs. To see this, consider the graph $G(d, \Delta) = K_d \vee I_{\Delta-d+1}$, where the join operation \vee connects each vertex in one graph to all vertices of the other graph. This graph is d -degenerate and of maximum degree Δ . In every proper coloring of $G(d, \Delta)$, each vertex in K_d forms a single color class. Hence, every equitable coloring of $G(d, \Delta)$ uses at least $d + \lceil(\Delta - d + 1)/2\rceil = \lceil(\Delta + d + 1)/2\rceil$ colors. In particular, $G(2, \Delta)$ uses at least $\lceil(\Delta + 3)/2\rceil$ colors for an equitable coloring, which is greater than $\lceil\Delta/2\rceil + 1$ for even Δ . Kostochka and Nakprasit showed that $\lceil(\Delta + d + 1)/2\rceil$ colors is enough to equitably color a d -degenerate graph G with maximum degree Δ provided Δ/d is large.

Theorem 33 *Let $2 \leq d \leq \Delta/27$ and G be a d -degenerate graph with maximum degree at most Δ . Then G is equitably k -colorable if $k \geq (\Delta + d + 1)/2$.*

The example $G(d, \Delta)$ shows that the bound on k cannot be decreased. The next corollary follows from this theorem since every planar graph is 5-degenerate.

Corollary 2 *Let $\Delta \geq 135$ and the maximum degree of the planar graph G be at most Δ . Then G is equitably k -colorable if $k \geq \Delta/2 + 3$.*

Let $k \geq 14d + 1$ and the d -degenerate graph G have maximum degree at most k . Then, for $\Delta = 2k - 1 - d$, G satisfies the condition of [Theorem 33](#).

Corollary 3 *Let $d \geq 2$. Then every d -degenerate graph with maximum degree at most k is equitably k -colorable if $k \geq 14d + 1$.*

In view of [Theorem 33](#), which is also true if $d = \Delta$ by the Hajnal and Szemerédi Theorem, the following was proposed in [92]:

Conjecture 7 *Let $2 \leq d \leq \Delta$ and G be a d -degenerate graph with maximum degree at most Δ . Then G is equitably k -colorable if $k \geq (\Delta + d + 1)/2$.*

A graph with “low degeneracy” is intuitively rather similar to a graph whose every subgraph has a “small average degree.” Kostochka and Nakprasit [93] proved the $E\Delta CC$ for graphs that have “small average degree” without restrictions on their subgraphs. The *average degree* of a graph G is defined to be $Ad(G) = 2\|G\|/|G|$.

Theorem 34 *Let $\Delta \geq 46$ and G be a graph of order at least 46 and maximum degree at most Δ . If $Ad(G) \leq \Delta/5$ and $K_{\Delta+1}$ is not a subgraph of G , then G is equitably Δ -colorable.*

An immediate consequence of this result is that the $E\Delta CC$ holds for d -degenerate graphs with maximum degree Δ if $d \leq \Delta/10$.

In Kostochka et al. [95], a result similar to [Theorem 22](#) was established for d -degenerate graphs.

Theorem 35 *Every d -degenerate graph G with maximum degree at most Δ is equitably k -colorable when $k \geq 16d$ and $\Delta \leq |G|/15$.*

If the restriction on Δ is removed, they proved the following:

Theorem 36 *Every d -degenerate graph G with maximum degree at most Δ is equitably k -colorable for any k , $k \geq \max\{62d, 31d|G|/(|G| - \Delta + 1)\}$.*

Corollary 4 *Every d -degenerate graph G with maximum degree at most $|G|/2 + 1$ is equitably k -colorable for any $k \geq 62d$.*

The proof of [Theorem 36](#) is constructive, and by extending their proof method, the following result of algorithmic nature was obtained:

Theorem 37 *There exists a polynomial time algorithm that produces an equitable k -coloring of G for every equitably m -colorable d -degenerate graph G if $k \geq 31dm$.*

A concept generalizing Pemmaraju’s “equitable 2-forest partition” was also introduced in [95]. An *equitable k -partition* of a graph G is a collection of subgraphs $\{G[V_1], G[V_2], \dots, G[V_k]\}$ of G induced by the vertex partition $\{V_1, V_2, \dots, V_k\}$ of $V(G)$ such that $||V_i| - |V_j|| \leq 1$ for every pair V_i and V_j . The following provides a tool for obtaining equitable colorings with few colors:

Theorem 38 Let $k \geq 3$ and $d \geq 2$. Then every d -degenerate graph has an equitable k -partition into $(d - 1)$ -degenerate graphs.

This is an extension of [Theorem 1](#) proved by Bollobás and Guy, which can be restated as follows: Any 1-degenerate graph G with $\Delta(G) \leq |G|/3$ can be equitably 3-partitioned into 0-degenerate graphs. Pemmaraju et al. [126] gave a direct generalization.

Theorem 39 For $d \geq 1$, every d -degenerate graph G with $\Delta(G) \leq |G|/3$ can be equitably 3-partitioned into $(d - 1)$ -degenerate graphs.

Repeated applications of this theorem can get the following:

Theorem 40 For $d \geq 1$, every d -degenerate graph G with $\Delta(G) \leq |G|/3^d$ can be equitably 3^d -colored.

In the same paper, the following conjecture was proposed:

Conjecture 8 There are functions $f(d) = O(d)$ and $g(d) = O(d)$ such that if G is a d -degenerate graph with $\Delta(G) \leq |G|/f(d)$, then G can be equitably $g(d)$ -colored.

9 Graphs of Bounded Treewidth

A *tree decomposition* of a graph G is a pair (T, \mathcal{F}) , with T a tree and $\mathcal{F} = \{X_i \subseteq V(G) \mid i \in V(T)\}$, that satisfies the following conditions:

1. $\bigcup_{i \in V(T)} X_i = V(G)$.
2. For every edge uv of G , there exists an $i \in V(T)$ such that X_i contains both u and v .
3. For all $i_1, i_2, i_3 \in V(T)$, $X_{i_1} \cap X_{i_3} \subseteq X_{i_2}$ if i_2 is on the path from i_1 to i_3 in T .

The *width* of the tree decomposition (T, \mathcal{F}) is defined to be $\max_{i \in V(T)} |X_i| - 1$; the *treewidth* of a graph G denoted by $\text{tw}(G)$ is the minimum width among all tree decompositions of G . It is a folklore result that every graph G of treewidth at most k has a vertex of degree at most k and has at most $k|G|$ edges. Hence, every graph of treewidth at most k is k -degenerate.

The class of graphs with treewidth at most k can be characterized in terms of partial k -trees. The class of k -trees is defined recursively as follows:

1. The complete graph K_k is a k -tree.
2. A k -tree G with $n + 1$ vertices ($n \geq k$) is constructed from a k -tree H with n vertices by adding a new vertex adjacent to and only to all vertices of a subgraph of H that is a K_k .

There are a number of alternative characterizations of k -trees [129]. A graph is called a *partial k -tree* if it is a subgraph of a k -tree. Forests are partial 1-trees and series-parallel graphs are partial 2-trees.

Theorem 41 ([139]) *A graph G is a partial k -tree if and only if G has treewidth at most k .*

A corollary of [Theorem 36](#) is the following:

Corollary 5 *Every graph G with treewidth w and maximum degree at most Δ is equitably k -colorable for any k , $k \geq \max\{62w, 31w|G|/(|G| - \Delta + 1)\}$.*

The equitable k -coloring problem can be stated as follows. A graph G and an integer k are given, and one asks whether G has an equitable k -coloring. For graphs of bounded treewidth, Bodlaender and Fomin [13] used the above result to establish the threshold for telling when the EQUITABLE k -COLORING problem is trivially solved and when it becomes to be solvable in polynomial time by their dynamic programming approach. It amounts to the following.

Theorem 42 *The equitable k -coloring problem can be solved in polynomial time on graphs of bounded treewidth.*

They also showed that such an approach cannot be extended to the equitable k -coloring with precoloring problem: A graph G , an integer k , and a precoloring π of G are given, and one asks whether there exists an equitable k -coloring of G extending π . For a graph G , a *precoloring* π of a subset U of vertices of G in k colors is a mapping $\pi : U \rightarrow \{1, 2, \dots, k\}$. A coloring of G with color classes V_1, V_2, \dots, V_k is said to extend the precoloring π if $u \in V_{\pi(u)}$ for every $u \in U$. The following was proved in [13]:

Theorem 43 *The equitable k -coloring with precoloring problem is NP-complete on trees.*

In the framework of parameterized complexity, e.g., [41, 51], and [119], a parameterized problem with the input size n and a parameter k is called *fixed parameter tractable* (FPT) if it can be solved in time $f(k) \cdot n^c$, where f is a function only depending on k and c is a constant. The basic complexity class for fixed parameter intractability is $\mathcal{W}[1]$. The equitable coloring problem was shown by Fellow et al. [47] to be $\mathcal{W}[1]$ -hard, parameterized by the treewidth plus the number of colors. However, the equitable coloring problem is FPT when parameterized by the vertex cover number as shown by Fiala et al. [48]. The vertex cover number of a graph G is the minimum size of a set $X \subseteq V(G)$ such that $V(G) \setminus X$ is an independent set.

10 Kneser Graphs

For integers $i \leq j$, let $[i, j] = \{i, i+1, \dots, j\}$ and $[n] = [1, n]$. If X is a set, then the collection of all k -subsets of X is denoted by $\binom{X}{k}$. The *Kneser graph* $\text{KG}(n, k)$ has $\binom{[n]}{k}$ as its vertex set. Two distinct vertices are adjacent in $\text{KG}(n, k)$ if they

have empty intersection as subsets. To exclude trivialities, it is always assumed that $n > 2k$ in $\text{KG}(n, k)$. The order of $\text{KG}(n, k)$ is clearly $\binom{n}{k}$. The *odd graph* O_k is the Kneser graph $\text{KG}(2k+1, k)$. Since it is easy to see that $\text{KG}(n, 1) = K_n$ and $\chi(\text{KG}(n, 1)) = \chi_-(\text{KG}(n, 1)) = \chi_-^*(\text{KG}(n, 1)) = n$, it is assumed that $k \geq 2$ throughout this section.

The following is a much celebrated result of Lovász [107] proved by topological method:

Theorem 44 *The chromatic number of $\text{KG}(n, k)$ is equal to $n - 2k + 2$.*

For $i \in [n]$, an i -flower \mathcal{F} of $\binom{[n]}{k}$ is a subcollection of $\binom{[n]}{k}$ such that all k -subsets in \mathcal{F} have i as a common element. It is clear that every i -flower is an independent set of $\text{KG}(n, k)$. Any independent set \mathcal{F} of $\text{KG}(n, k)$, also called an *intersecting family* of $\binom{[n]}{k}$, satisfies $A \cap B \neq \emptyset$ for all A and B in \mathcal{F} . The independence number $\alpha(\text{KG}(n, k))$ of $\text{KG}(n, k)$ was obtained by Erdős et al. [44].

Theorem 45 *Suppose \mathcal{F} is an intersecting family of $\binom{[n]}{k}$. Then*

$$|\mathcal{F}| \leq \binom{n-1}{k-1}.$$

Moreover, the equality holds if and only if \mathcal{F} is an i -flower for some $i \in [n]$. Hence, $\alpha(\text{KG}(n, k)) = \binom{n-1}{k-1}$.

There are independent sets of $\text{KG}(n, k)$ which are not flowers. Denote by $\alpha_2(\text{KG}(n, k))$, or simply $\alpha_2(n, k)$, the maximum size of independent sets \mathcal{H} of $\text{KG}(n, k)$ satisfying $\bigcap_{A \in \mathcal{H}} A = \emptyset$ and $\alpha_2(n, k)$ was obtained by Hilton and Milner [69].

Theorem 46 *Suppose \mathcal{H} is an intersecting family of $\binom{[n]}{k}$ with $\bigcap_{A \in \mathcal{H}} A = \emptyset$. Then*

$$|\mathcal{H}| \leq \binom{n-1}{k-1} - \binom{n-k-1}{k-1} + 1.$$

Moreover, the equality holds if \mathcal{H} is the family $\{A \in \binom{[n]}{3} \mid |A \cap [1, 3]| \geq 2\}$ when $k=3$ or \mathcal{H} is the family $\{A \in \binom{[n]}{k} \mid 1 \in A, |A \cap [2, k+1]| \geq 1\} \cup \{[2, k+1]\}$. Hence, $\alpha_2(n, k) = \binom{n-1}{k-1} - \binom{n-k-1}{k-1} + 1$.

Since every flower of $\binom{[n]}{k}$ is an independent set of $\text{KG}(n, k)$, it is natural to partition flowers to form an equitable coloring of $\text{KG}(n, k)$. If this is successful, then every k -subset of $[n]$ is in some flower. Hence, if f is an equitable m -coloring of $\text{KG}(n, k)$ such that every color class of f is contained in some flower, then $m \geq n - k + 1$. Otherwise, suppose $m \leq n - k$ and each color class $f^{-1}(i)$ is contained in some t_i -flower for $1 \leq i \leq m$. Since $|[n] \setminus \{t_1, t_2, \dots, t_m\}| \geq n - m \geq k$, one may

choose a k -subset $A \subseteq [n] \setminus \{t_1, t_2, \dots, t_m\}$. Since f is an equitable m -coloring, $A \in f^{-1}(i)$ for some i , i.e., $t_i \in A$. Thus, a contradiction is obtained.

In [24], Chen and Huang tried to show that $\text{KG}(n, k)$ is equitably m -colorable for all $m \geq n - k + 1$ by partitioning flowers of $\binom{[n]}{k}$ into m -independent sets whose sizes are as even as possible. They succeeded in establishing the following:

Theorem 47 Suppose that $m \geq n - k + 1$. Then $\text{KG}(n, k)$ is equitably m -colorable, i.e., $\chi_{=}(\text{KG}(n, k)) \leq \chi_{=}^*(\text{KG}(n, k)) \leq n - k + 1$.

Lemma 6 Suppose that $m \leq n - k$ and $\lfloor \binom{n}{k}/m \rfloor > \alpha_2(n, k)$. Then $\text{KG}(n, k)$ is not equitably r -colorable for all $r \leq m$, i.e., $\chi_{=}^*(\text{KG}(n, k)) \geq \chi_{=}(\text{KG}(n, k)) \geq m + 1$.

Theorem 48 If $\lfloor \binom{n}{k}/(n - k) \rfloor > \alpha_2(n, k)$, then $\chi_{=}(\text{KG}(n, k)) = \chi_{=}^*(\text{KG}(n, k)) = n - k + 1$.

Observe that $\binom{n}{k}/(n - k) = O(n^{k-1})$ and $\alpha_2(n, k) = O(n^{k-2})$. Hence, the following is true.

Corollary 6 Let k be fixed. Then $\chi_{=}(\text{KG}(n, k)) = \chi_{=}^*(\text{KG}(n, k)) = n - k + 1$ when n is sufficiently large.

Finally, $\chi_{=}(\text{KG}(n, 2))$, $\chi_{=}(\text{KG}(n, 3))$, and $\chi(O_k)$ were completely determined in [24].

Theorem 49 Assume $n \geq 5$. Then

$$\chi_{=}(\text{KG}(n, 2)) = \chi_{=}^*(\text{KG}(n, 2)) = \begin{cases} n - 2 & \text{if } n = 5 \text{ or } 6, \\ n - 1 & \text{if } n \geq 7. \end{cases}$$

Theorem 50 Assume $n \geq 7$. Then

$$\chi_{=}(\text{KG}(n, 3)) = \chi_{=}^*(\text{KG}(n, 3)) = \begin{cases} n - 4 & \text{if } 7 \leq n \leq 13, \\ n - 3 & \text{if } 14 \leq n \leq 15, \\ n - 2 & \text{if } n \geq 16. \end{cases}$$

Theorem 51 For $k \geq 1$, the odd graph O_k satisfies $\chi(O_k) = \chi_{=}(\text{KG}(n, k)) = \chi_{=}^*(\text{KG}(n, k)) = 3$.

Chen and Huang concluded their paper [24] by proposing the following:

Conjecture 9 If $n > 2k \geq 4$, then $\chi_{=}(\text{KG}(n, k)) = \chi_{=}^*(\text{KG}(n, k))$.

All results about Kneser graphs in this section were also independently obtained by Fidytek et al. [49].

11 Interval Graphs and Others

A graph G is called an *interval* graph if there exists a family $\mathcal{I} = \{I_v \mid v \in V(G)\}$ of intervals on the real line such that u and v are adjacent vertices if and only if $I_u \cap I_v \neq \emptyset$. Such a family \mathcal{I} is commonly referred to as an *interval representation* of G . Instead of intervals of real numbers, these intervals may be replaced by finite intervals on a linearly ordered set.

A *clique* of a graph G is a complete subgraph Q of G . A clique is called *maximal* if it is maximal in the set-inclusion order. For an interval graph G , Gillmore and Hoffman [59] showed that its maximal cliques can be linearly ordered as $Q_0 < Q_1 < \dots < Q_m$ so that for every vertex v of G , the maximal cliques containing v occur consecutively. The finite interval $I_v = [Q_i, Q_j]$ in this linear order is assigned to the vertex v if all the maximal cliques containing v are precisely Q_i, Q_{i+1}, \dots, Q_j . Again u and v are adjacent if and only if $I_u \cap I_v \neq \emptyset$. This representation of G is called a *clique path* representation of G . Conversely, the existence of a clique path representation implies that the graph is an interval graph.

Once a clique path representation is given, let $\text{left}(v)$ and $\text{right}(v)$ stand for the left and right endpoint, respectively, of the interval I_v . Then the following linear order on the vertices of G can be defined. Let $u < v$ if ($\text{left}(u) < \text{left}(v)$) or ($\text{left}(u) = \text{left}(v)$ and $\text{right}(u) < \text{right}(v)$). If u and v have the same left and right endpoints, choose $u < v$ arbitrarily. For any three vertices u , v , and w of G , this linear order satisfies the following condition. If $u < v < w$ and $uw \in E(G)$, then $uv \in E(G)$. The existence of a linear order satisfying this condition characterizes interval graphs [120]. Chen et al. [31] utilized this linear order to obtain the following:

Theorem 52 *Let G be a connected interval graph. If G is not a complete graph, then G is equitably $\Delta(G)$ -colorable.*

And they proceeded further to show the following:

Theorem 53 *Let G be an interval graph. Then $\chi_=(G) = \chi^*_=(G)$.*

A few other classes of special graphs have been investigated for their equitable colorability. For instance, central graphs and total graphs were studied in [2, 140]. Thorny graphs were studied in [56]. Additional examples can be found in [57, 76–78].

For a given graph G , the so-called *central graph* $C(G)$ of G is obtained from G by inserting a new vertex to each edge of G and then joining each pair of vertices of G which were nonadjacent in G . The total graph $T(G)$ of G has vertex set $V(G) \cup E(G)$ and edges joining all elements of this vertex set which are adjacent or incident in G . The notation P_n represents a path on n vertices.

Results obtained in [2, 140] are listed as follows:

1. $\chi_=(C(K_{1,n})) = n$.
2. $\chi_=(C(K_{n,n})) \geq n$ if $n \geq 3$.
3. $\chi_=(C(K_n)) = 3$.

$$4. \chi_=(C(P_n)) = \begin{cases} 1 & \text{if } n = 1, \\ 2 & \text{if } n = 2, \\ 3 & \text{if } n = 3, \\ 3 & \text{if } n = 4, \\ n/2 & \text{if } n \geq 5 \text{ is even,} \\ (n+1)/2 & \text{if } n \geq 5 \text{ is odd.} \end{cases}$$

$$5. \chi_=(C(C_n)) = \begin{cases} 2 & \text{if } n = 3, \\ 3 & \text{if } n = 4, \\ n/2 & \text{if } n \geq 5 \text{ is even,} \\ (n+1)/2 & \text{if } n \geq 5 \text{ is odd.} \end{cases}$$

$$6. \chi_=(T(K_{m,n})) = \begin{cases} n+1 & \text{if } m < n, \\ n+2 & \text{if } m = n, \end{cases}$$

$$7. \chi_=(T(P_n)) = 3.$$

$$8. \chi_=(T(C_n)) = 3 \text{ if } n \text{ is a multiple of 3.}$$

An edge in a graph is called an *pendant edge* if it is incident with a *leaf*, i.e., a vertex of degree 1. Trees are the smallest set of graphs that contains single vertex and is closed under the operation of attaching a pendant edge to a vertex. By analogy to this recursive definition of trees, graphs called edge-cacti, cacti, and thorny graphs can be defined as follows.

Edge-cacti constitute the smallest set of graphs that includes all cycles and is closed under the operation of attaching a cycle to a single edge, i.e., identifying this edge with some edge of the attached cycle. *Cacti* constitute the smallest set of graphs that contains single vertex and is closed under the operation of attaching a pendant edge or cycle to a vertex. *Thorny graphs* constitute the smallest set of graphs that includes single vertex and is closed under the following operations:

1. Attaching a pendant edge to a vertex
2. Attaching a cycle to a vertex
3. Attaching a cycle to an edge

Every thorny graph is connected, planar, and tripartite. All cacti, edge-cacti, and connected outerplanar graphs are thorny graphs. The following results were established in [56]:

Theorem 54 Any thorny graph without leaves and C_3 or C_5 is equitably 3-colorable.

Theorem 55 Any thorny graph without leaves and C_3 is equitably k -colorable for all $k \geq 4$.

Corollary 7 The following statements are true:

1. Any edge-cactus without C_3 is equitably k -colorable for all $k \geq 3$. Furthermore, if an edge-cactus G is bipartite, then $\chi_=(G) = 2$.
2. Any cactus without leaves and C_3 or C_5 is equitably k -colorable for all $k \geq 3$.
3. Any bipartite cactus without leaves is equitably k -colorable for all $k \geq 3$.
4. Any cactus without leaves and C_3 is equitably k -colorable for all $k \geq 4$.

12 Random Graphs

Let $G(n, p)$ denote the probability space of all labeled graphs of order n such that every edge appears randomly and independently with probability $p = p(n)$. The space $G(n, p)$ is said to possess a property \mathcal{P} *almost surely* if the probability that $G(n, p)$ satisfies \mathcal{P} tends to 1 as n tends to infinity. In [98], Krivelevich and Patkó conjectured the following:

Conjecture 10 *There exists a constant C such that if $C/n < p < 0.99$, then almost surely $\chi_-(G(n, p)) = (1 + o(1))\chi(G(n, p))$ holds.*

Partial results proved by them included the following:

1. If $n^{-1/5+\epsilon} < p < 0.99$ for some positive ϵ , then almost surely $\chi_-(G(n, p)) \leq (1 + o(1))\chi(G(n, p))$ holds.
2. There exists a constant C such that if $C/n < p < 0.99$, then almost surely $\chi_-(G(n, p)) \leq (2 + o(1))\chi(G(n, p))$ holds.
3. If $n^{-(1-\epsilon)} < p < 0.99$ for some positive ϵ , then almost surely $\chi_-(G(n, p)) \leq (2 + o(1))\chi(G(n, p))$ holds.
4. If $(\log^{1+\epsilon} n)/n < p < 0.99$ for some positive ϵ , then almost surely $\chi_-(G(n, p)) = O_\epsilon(\chi(G(n, p)))$ holds.

13 Graph Products

Given two graphs G_1 and G_2 , it is natural to use the Cartesian product $V(G_1) \times V(G_2)$ of the two vertex sets to be the vertex set of a new graph. There are several ways to define the edge set of such a product graph. Results on three different products will be surveyed. Their edge sets are defined as follows:

1. The *square* product $G_1 \square G_2$, also known as the Cartesian product:

$$E(G_1 \square G_2) = \{(u, x)(v, y) \mid (u = v \text{ and } xy \in E(G_2)) \text{ or } (x = y \text{ and } uv \in E(G_1))\}.$$
2. The *cross* product $G_1 \times G_2$, also known as the Kronecker, direct, tensor, weak tensor, or categorical product:

$$E(G_1 \times G_2) = \{(u, x)(v, y) \mid uv \in E(G_1) \text{ and } xy \in E(G_2)\}.$$
3. The *strong* product $G_1 \boxtimes G_2$, also known as the strong tensor product:

$$E(G_1 \boxtimes G_2) = \{(u, x)(v, y) \mid (u = v \text{ and } xy \in E(G_2)) \text{ or } (uv \in E(G_1) \text{ and } x = y) \text{ or } (uv \in E(G_1) \text{ and } xy \in E(G_2))\}.$$

Note that square and cross products are so named because the products of two single edges are a square and a cross, respectively, and $G_1 \boxtimes G_2 = (G_1 \square G_2) \cup (G_1 \times G_2)$.

13.1 Square Product

For the ordinary chromatic number, Sabidussi [130] proved a product theorem.

Theorem 56 For graphs G_1 and G_2 , $\chi(G_1 \square G_2) = \max\{\chi(G_1), \chi(G_2)\}$.

In Chen et al. [31], the following results were obtained:

Theorem 57 If G_1 and G_2 are equitably k -colorable, so is $G_1 \square G_2$.

Corollary 8 For graphs G_1 and G_2 , $\chi_{=}^*(G_1 \square G_2) \leq \max\{\chi_{=}^*(G_1), \chi_{=}^*(G_2)\}$.

Corollary 9 If $\chi(G_1) = \chi_{=}^*(G_1)$ and $\chi(G_2) = \chi_{=}^*(G_2)$, then $\chi(G_1 \square G_2) = \chi_{=}^*(G_1 \square G_2) = \max\{\chi(G_1), \chi(G_2)\}$.

Corollary 10 Let $G = G_1 \square G_2 \square \cdots \square G_n$, where each G_i is a path, a cycle, or a complete graph. Then $\chi(G) = \chi_{=}^*(G) = \max\{\chi(G_i) \mid 1 \leq i \leq n\}$.

Corollary 11 Suppose that G_1 and G_2 are nontrivial graphs. Then $G_1 \square G_2$ is equitably $\Delta(G_1 \square G_2)$ -colorable, i.e., the EΔCC holds for the square product of two graphs.

Even if $\chi(G_1) = \chi_{=}^*(G_1) = k$, the product $G_1 \square G_2$ may not be equitably k -colorable. An example is the product $K_{1,5} \square P_3$. If it is assumed that $\chi_{=}^*(G_1) = \chi_{=}^*(G_2) = k$, it may not lead to the conclusion $\chi_{=}^*(G_1 \square G_2) = k$. An example is $K_{1,2n} \square K_{1,2n}$. If $G_1 = K_{3,3}$ and $G_2 = K_{1,1,2}$, then $\chi_{=}^*(G_1 \square G_2) \leq \max\{\chi_{=}^*(G_1), \chi_{=}^*(G_2)\}$ is false.

Exact values for paths, cycles, and complete graphs are also determined in [31].

Theorem 58 The following hold for positive integers m and n :

$$\chi(P_m \square P_n) = \chi_{=}^*(P_m \square P_n) = \chi_{=}^*(P_m \square P_n) = 2.$$

$$\chi(C_m \square C_n) = \chi_{=}^*(C_m \square C_n) = \begin{cases} 2 & \text{if } m \text{ and } n \text{ are even,} \\ 3 & \text{otherwise.} \end{cases}$$

$$\chi(K_m \square K_n) = \chi_{=}^*(K_m \square K_n) = \chi_{=}^*(K_m \square K_n) = \max\{m, n\}.$$

Furmańczyk [54] also obtained a number of exact values of square products between cycles, paths, and hypercubes $Q_n = \underbrace{K_2 \square K_2 \cdots \square K_2}_n$.

Theorem 59 Let k, m, n , and r be positive integers. Then the following graphs have their equitable chromatic numbers all equal 2: $Q_r \square P_{2n}$, $Q_r \square C_{2n}$, and $Q_r \square Q_r$.

Besides $\chi_{=}^*(K_{m_1,n_1} \square K_{m_2,n_2}) \leq 4$, Lin in his Ph.D. dissertation [103] (also in [105]) determined more exact values. They are listed as follows, in which m, n , and r are assumed to be positive integers.

1. $\chi_{=}^*(P_{2r} \square K_{m,n}) = \chi_{=}^*(P_{2r+1} \square K_{m,n}) = 2$ except $\chi_{=}^*(P_2 \square K_{m,n}) = 4$ when $m + n + 2 < 3 \min\{m, n\}$.

2. $\chi_{=}^*(P_{2r+1} \square K_{m,n}) = \chi_{=}^*(P_{2r+1} \square K_{m,n}) = \begin{cases} 2 & \text{if } |m - n| \leq 1, \\ 3 & \text{otherwise.} \end{cases}$

3. $\chi_=(C_{2r} \square K_{m,n}) = \chi^*_=(C_{2r} \square K_{m,n}) = 2$ except $\chi^*_=(C_4 \square K_{m,n}) = 4$ when $m + n + 2 < 3 \min\{m, n\}$.
4. $\chi_=(C_{2r+1} \square K_{m,n}) = \chi^*_=(C_{2r+1} \square K_{m,n}) = 3$.
5. $\chi_=(K_{1,m} \square K_{1,n}) = \chi^*_=(K_{1,m} \square K_{1,n}) = \begin{cases} 4 & \text{if } (m-2)(n-2) > 5, \\ 3 & \text{otherwise.} \end{cases}$

Lin also proposed some interesting conjectures:

Conjecture 11 *If G_1 and G_2 are bipartite graphs, then $\chi^*_=(G_1 \square G_2) \leq 4$.*

Conjecture 12 *If G_1 and G_2 are connected graphs, then $\chi_=(G_1 \square G_2) \leq \chi(G_1)\chi(G_2)$.*

The connectedness is essential in the above conjecture because $\chi_=(K_{1,3} \square 3K_1) = \chi^*(K_{1,3} \square 3K_1) = 3 > 2 = \chi(K_{1,3})\chi(3K_1)$.

13.2 Cross Product

The most well-known conjecture for cross product is the one proposed by Hedetniemi [67].

Conjecture 13 *For graphs G_1 and G_2 , $\chi(G_1 \times G_2) = \min\{|G_1|, |G_2|\}$.*

This has been established for complete graphs in [42]. For two recent surveys on Hedetniemi's conjecture, the reader is referred to Sauer [131] and Zhu [161]. Furmańczyk [54] showed that $\chi_=(P_3 \times P_3) = 3 > 2 = \chi_=(P_3)$. Thus, $\chi_=(G_1 \times G_2) \leq \min\{\chi_=(G_1), \chi_=(G_2)\}$ does not hold in general. However, Chen et al. [31] gave the following upper bound.

Theorem 60 *For graphs G_1 and G_2 , $\chi_=(G_1 \times G_2) \leq \min\{|G_1|, |G_2|\}$.*

The upper bound is sharp in the case $\chi_=(K_m \times K_n) = \min\{m, n\}$. In general, $\min\{|G_1|, |G_2|\}$ is not an upper bound for $\chi^*(G_1 \times G_2)$. An example was given in [31] to show that $K_2 \times K_n$ is not equitably $(n+1)/2$ -colorable when $n > 1$ and $n \equiv 1 \pmod{4}$. Even $\chi^*(G_1 \times G_2) \leq \max\{\chi^*(G_1), \chi^*(G_2)\}$ fails in general. Examples are $\chi^*(K_{2,3} \times K_{2,3}) = 3 > 2 = \chi^*(K_{2,3})$ [31] and $\chi^*(P_3 \times P_3) = 3 > 2 = \chi^*(P_3)$ [54]. The following result was conjectured in [31] and later proved to be true in [103]:

Theorem 61 *For graphs G_1 and G_2 , $\chi^*(G_1 \times G_2) \leq \max\{|G_1|, |G_2|\}$.*

It suffices to prove the above theorem for the case $K_m \times K_n$. A slightly better upper bound was actually established in [103].

Theorem 62 For positive integers m and n ,

$$\chi_{=}^*(K_m \times K_n) \leq \left\lceil \frac{mn}{\min\{m, n\} + 1} \right\rceil.$$

As to exact values, the following were established in [31] and [54], respectively:

1. $\chi_{=}^*(C_m \times C_n) = \chi_{=}^*(C_m \times C_n) = \begin{cases} 2 & \text{if } mn \text{ is even,} \\ 3 & \text{otherwise.} \end{cases}$
2. $\chi_{=}^*(P_m \times K_{1,n}) = \begin{cases} 2 & \text{if } m \text{ is even or } n = 1, \\ 3 & \text{otherwise.} \end{cases}$

Lin in his Ph.D. dissertation [103] determined more exact values. They are listed as follows. Also see [104].

1. $\chi(P_m \times K_2) = \chi_{=}^*(P_m \times K_2) = \chi_{=}^*(P_m \times K_2) = 2$, where $m \geq 2$.
2. $\chi(P_{2m+1} \times K_n) = 2 < \chi_{=}^*(P_{2m+1} \times K_n) = \chi_{=}^*(P_{2m+1} \times K_n) = 3$, where $n \geq 3$, except $\chi_{=}^*(P_3 \times K_n) = \max\{\lceil \frac{3}{2} \lceil \frac{2n}{s} \rceil \rceil \mid s \nmid 2n \text{ and } \lceil \frac{3}{2} \lceil \frac{2n}{s} \rceil \rceil \leq \lceil \frac{3n}{4} \rceil\}$.
3. $\chi(P_{2m} \times K_n) = \chi_{=}^*(P_{2m} \times K_n) = \chi_{=}^*(P_{2m} \times K_n) = 2$, where $m > 1$ and $n \geq 3$.
4. $\chi_{=}^*(P_2 \times K_n) = \max\{2 \lceil \frac{n}{s} \rceil \mid s \nmid n \text{ and } 2 \lceil \frac{n}{2} \rceil \leq \lceil \frac{2n}{3} \rceil\}$, where $n \geq 3$.
5. $\chi(C_m \times K_2) = \chi_{=}^*(C_m \times K_2) = \chi_{=}^*(C_m \times K_2) = 2$, where $m \geq 3$.
6. $\chi(C_{2m+1} \times K_n) = \chi_{=}^*(C_{2m+1} \times K_n) = \chi_{=}^*(C_{2m+1} \times K_n) = 3$, where $m > 1$ and $n \geq 3$, except $\chi_{=}^*(C_5 \times K_n) = \chi_{=}^*(C_5 \times K_n) = 4$, when $n \geq 5$.
7. $\chi_{=}^*(C_3 \times K_n) = \begin{cases} \lceil \frac{3n}{4} \rceil & \text{if } n \equiv 2 \pmod{4}, \\ \max\{3 \lceil \frac{n}{s} \rceil \mid s \nmid n \text{ and } 3 \lceil \frac{n}{s} \rceil \leq \lceil \frac{3n}{4} \rceil\} & \text{otherwise,} \end{cases}$ where $n \geq 3$.
8. $\chi(C_{2m} \times K_n) = \chi_{=}^*(C_{2m} \times K_n) = \chi_{=}^*(C_{2m} \times K_n) = 2$, where $m > 1$ and $n \geq 3$.
9. $\chi_{=}^*(C_4 \times K_n) = \max\{2 \lceil \frac{2n}{s} \rceil \mid s \nmid 2n \text{ and } 2 \lceil \frac{2n}{s} \rceil \leq \lceil \frac{4n}{5} \rceil\}$, where $n \geq 3$.
10. $\chi_{=}^*(K_{m_1, n_1}, K_{m_2, n_2}) = \chi_{=}^*(K_{m_1, n_1}, K_{m_2, n_2}) = \min\left\{\left\lceil \frac{n_1}{m_1} \right\rceil, \left\lceil \frac{n_2}{m_2} \right\rceil\right\} + 1$, where $m_1 \leq n_1$ and $m_2 \leq n_2$.

13.3 Strong Product

The upper bound for the inequality $\chi(G_1 \boxtimes G_2) \leq \chi(G_1)\chi(G_2)$ is exact when both factors are complete graphs. As to lower bounds, there are those established by Vesztergombi [141] and Jha [74], respectively.

Theorem 63 For nontrivial graphs G_1 and G_2 , $\chi(G_1 \boxtimes G_2) \geq \max\{\chi(G_1), \chi(G_2)\} + 2$.

Theorem 64 For nontrivial graphs G_1 and G_2 , $\chi(G_1 \boxtimes G_2) \geq \chi(G_1) + \omega(G_2)$, where $\omega(G)$ denotes the clique number of the graph G , i.e., the largest order of a complete subgraph in G .

Furmańczyk [54] first investigated the equitable chromatic number of a strong product. Her results have been subsumed by those in [103]. Again, Lin's results are listed as follows:

1. $\chi(C_m \boxtimes K_n) = \chi_=(C_m \boxtimes K_n) = \chi^*_=(C_m \boxtimes K_n) = \left\lceil \frac{mn}{\lceil m/2 \rceil} \right\rceil$, where $m \geq 3$ and $n \geq 2$.
2. $\chi(P_m \boxtimes K_n) = \chi_=(P_m \boxtimes K_n) = \chi^*_=(P_m \boxtimes K_n) = 2n$, where $m \geq 2$ and $n \geq 2$.
3. $\chi(C_m \boxtimes C_n) = \chi_=(C_m \boxtimes C_n) = \chi^*_=(C_m \boxtimes C_n) = \begin{cases} 4 & \text{if } m \text{ and } n \text{ are even,} \\ 5 & \text{otherwise,} \end{cases}$ where $m, n \geq 4$.
4. $\chi(C_m \boxtimes P_n) = \chi_=(C_m \boxtimes P_n) = \chi^*_=(C_m \boxtimes P_n) = \begin{cases} 4 & \text{if } m \text{ is even,} \\ 5 & \text{otherwise,} \end{cases}$ where $m \geq 4$ and $n \geq 3$.
5. $\chi(P_m \boxtimes P_n) = \chi_=(P_m \boxtimes P_n) = \chi^*_=(P_m \boxtimes P_n) = \begin{cases} 4 & \text{if } mn \text{ is even,} \\ 5 & \text{otherwise,} \end{cases}$ where $m \geq 3$ and $n \geq 3$.

Conjecture 14 Suppose that G_1 has at least one edge. Then $\chi_=(G_1 \boxtimes G_2) \geq \chi_=(G_1) + 2\omega(G_2) - 2$ and $\chi^*_=(G_1 \boxtimes G_2) \geq \chi^*_=(G_1) + 2\omega(G_2) - 2$.

The conclusions of the above conjecture hold if the equitable chromatic number of threshold is replaced by the ordinary chromatic number [103].

14 List Equitable Coloring

A mapping L is said to be a *list assignment* for the graph G if it assigns a finite list $L(v)$ of possible colors, usually regarded as positive integers, to each vertex v of G . A list assignment L for G is *k-uniform* if $|L(v)| = k$ for all $v \in V(G)$. If G has a proper coloring π such that $\pi(v) \in L(v)$ for all vertices v , then G is said to be *L-colorable* or π is an *L-coloring* of G . The graph G is said to be *k-choosable* if it is *L-colorable* for every *k-uniform* list assignment L , *equitably L-colorable* if it has a $\lceil |G|/k \rceil$ -bounded *L-coloring* for a *k-uniform* list assignment L , and *equitably list k-colorable* or *equitably k-choosable* if it is equitably *L-colorable* for every *k-uniform* list assignment L .

The concept of list-coloring was introduced by Vizing [144] and independently by Erdős et al. [45]. However, it is not appropriate to generalize the ordinary equitable coloring to this list context. To see this, let every vertex except one of a graph G be assigned the list $[k]$ and the remaining vertex v be assigned the list $[k+1, 2k]$. Unless $|G| \leq k+1$, in every proper coloring, some colors are not used, the color on v appears once, and some other color appears at least $\lceil (|G|-1)/k \rceil$ times.

This explains why the list version of an equitable coloring is defined in terms of boundedness of color classes. This notion was first introduced by Kostochka et al. [94], and they proposed two conjectures that are analogue to the Hajnál-Szemerédi Theorem and the $E\Delta CC$.

Conjecture 15 *Every graph G is equitably k -choosable whenever $k > \Delta(G)$.*

Conjecture 16 *If G is a connected graph with maximum degree at least 3, then G is equitably $\Delta(G)$ -choosable, unless G is a complete graph or is $K_{r,r}$ for some odd r .*

When $\Delta(G) = 2$, it is easy to see the validity of Conjecture 16. Conjecture 15 has been proved for $\Delta(G) \leq 3$ independently by Pelsmajer [122] and Wang and Lih [146]. In [122], a graph G was shown to be equitably k -choosable when $k \geq 2 + \Delta(G)(\Delta(G) - 1)/2$. In [94], Kostochka et al. gave the following partial results to Conjecture 15 and 16:

Theorem 65 *If $k \geq \max\{\Delta(G), |G|/2\}$, then G is equitably k -choosable unless G contains K_{k+1} or $K_{k,k}$ (with k odd in the latter case).*

Theorem 66 *If G is a forest and $k \geq \Delta(G)/2 + 1$, then G is equitably k -choosable. Moreover, for all m there is a tree with maximum degree at most m that is not equitable $\lceil m/2 \rceil$ -choosable.*

Theorem 67 *If G is a connected interval graph and $k \geq \Delta(G)$, then G is equitably k -choosable unless $G = K_{k+1}$.*

Theorem 68 *If G is a 2-degenerate graph and $k \geq \max\{\Delta(G), 5\}$, then G is equitably k -choosable.*

Pelsmajer [123] provided more partial results.

Theorem 69 *Let G be a graph with treewidth w and $k \geq 3w - 1$. Then G is equitably k -choosable if:*

1. $w \leq 5$ and $k \geq \Delta(G) + 1$, or
2. $w \geq 5$ and $k \geq \Delta(G) + w - 4$.

A graph is said to be *chordal* if it has no induced cycle of length greater than three.

Corollary 12 *Let G be a chordal graph with maximum degree at most Δ . Then G is equitably k -choosable for $k \geq \max\{3\Delta - 4, \Delta + 1\}$.*

If vertices of degree 1 are removed recursively from a graph G , then the final graph has no vertices of degree 1 and is called the *core* of G . A graph is called a $\theta_{2,2,p}$ -graph if it consists of two vertices x and y and three internally disjoint paths

of lengths 2, 2, and p joining x and y . Erdős et al. [45] characterized 2-choosable graphs in terms of these concepts. Analogous to their result, Wang and Lih [146] gave the following characterization:

Theorem 70 *A connected graph G is equitably 2-choosable if and only if G is a bipartite graph satisfying the following two conditions:*

1. *The core of G is either a K_1 , an even cycle, or a $\theta_{2,2,2r}$ -graph, where $r \geq 1$.*
2. *G has two parts X and Y such that $|X| - |Y| \leq 1$.*

Bu and his collaborators have established a series of partial results for [Conjecture 15](#) and [16](#) in the class of planar graphs as follows [100, 162, 163]:

Theorem 71 *Let G be a C_4 -free and C_6 -free planar graph. Then G is equitably k -choosable when $k \geq \max\{\Delta(G), 6\}$.*

Theorem 72 *Let G be a C_3 -free planar graph. Then G is equitably k -choosable when $k \geq \max\{\Delta(G), 8\}$.*

Theorem 73 *Let G be a C_4 -free and C_5 -free planar graph. Then G is equitably k -choosable when $k \geq \max\{\Delta(G), 7\}$.*

Theorem 74 *Let G be an outerplanar graph. Then G is equitably k -choosable when $k \geq \max\{\Delta(G), 4\}$.*

Theorem 75 *Let G be an outerplanar graph of maximum degree 3. Then G is equitably 3-choosable.*

Recently, [Theorems 74](#) and [75](#) have been generalized to series-parallel graphs by Zhang and Wu [159]. The following result confirms [Conjecture 15](#) and [16](#) for series-parallel graphs:

Theorem 76 *If G is a series-parallel graph with $\Delta(G) \geq 3$, then G is equitably k -choosable if $k \geq \Delta(G)$.*

In the framework of parameterized complexity, the list equitable coloring problem was shown by Fellow et al. [47] to be $\mathcal{W}[1]$ -hard even for forests, parameterized by the number of colors on the lists.

15 Graph Packing

The equitable coloring problem can be stated in the language of graph packing; hence it can be studied in a wider context. Two graphs G_1 and G_2 of the same order are said to *pack* if G_1 is isomorphic to a subgraph of the complement G_2^c of G_2 , or, equivalently, G_2 is isomorphic to a subgraph of the complement G_1^c of G_1 .

This definition may be extended to n graphs G_1, G_2, \dots, G_n of the same order so that they pack if every pair of them pack:

Because the problem of whether G^c packs with the cycle $C_{|G|}$ is equivalent to the existence of a Hamiltonian cycle in G , two well-known sufficient conditions for the existence of a Hamiltonian cycle, due to Dirac [40] and Ore [121], can be cast in terms of graph packings.

Theorem 77 *If $\Delta(G) \leq |G|/2 - 1$, then G packs with $C_{|G|}$.*

Theorem 78 *If $\deg(u) + \deg(v) \leq |G| - 2$ for every edge uv in G , then G packs with $C_{|G|}$.*

A graph G is k -colorable if and only if G packs with a graph of the same order that is the union of k -cliques. Let $H(n, k)$ denote the graph of order n such that it has k components each of which is a clique of order $\lfloor n/k \rfloor$ or $\lceil n/k \rceil$. This graph is the complement of the well-known Turán graph in extremal graph theory. A graph G is equitably k -colorable if and only if G packs with $H(|G|, k)$. The Brooks' Theorem and the Hajnal and Szemerédi Theorem can now be stated as follows:

Theorem 79 *If $r \geq 3$, G is a connected graph with $\Delta(G) \leq r$ and G does not pack with the complement of any r -partite graph, then $G = K_{r+1}$.*

Theorem 80 *Let G satisfy $\Delta(G) \leq r$. Then G packs with $H(|G|, r + 1)$.*

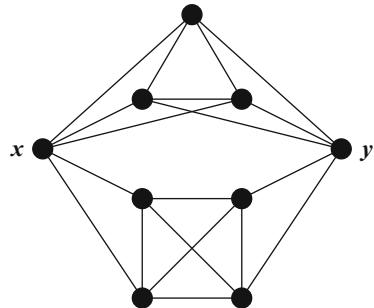
In view of Ore's theorem, the *Ore degree* of an edge uv , denoted by $\theta(uv)$, is defined to be $\deg(u) + \deg(v)$, and the *Ore degree of a graph* G is $\theta(G) = \max\{\theta(uv) \mid uv \in E(G)\}$. Following [81], those upper bounds in terms of Ore degree giving sufficient conditions for packing graphs are called *Ore-type* bounds. Those in terms of maximum degree are called *Dirac type*.

An obvious Dirac-type bound on the chromatic number of a graph G is $\chi(G) \leq \Delta(G) + 1$. The Brooks' Theorem characterizes the conditions for equality to hold: either G contains $K_{\Delta(G)+1}$ or $\Delta(G) = 2$ and G contains an odd cycle. An Ore-type bound on $\chi(G)$ can be obtained easily such that $\chi(G) \leq \lfloor \theta(G)/2 \rfloor + 1$. The bound is also attained at complete graphs. However, for small odd $\theta(G)$, there are more connected extremal graphs.

Theorem 81 *If $6 \leq k = \chi(G) = \lfloor \theta(G)/2 \rfloor + 1$, then G contains K_k .*

Kierstead and Kostochka [83] proposed the above as a conjecture and proved the statement when the lower bound 6 was replaced by 7. The theorem has been established recently by Rabern [127]. See [96] for further generalizations. The above theorem can be equivalently stated as follows: for $k \geq 6$, K_k is the only k -critical graph with maximum degree at most k whose vertices of degree k form an independent set. A k -critical graph is one that has chromatic number k , and any of its proper subgraphs has chromatic number less than k .

Fig. 1 The graph G with $\theta(G) = 9$ and $\chi(G) = 5$



The bound 6 is sharp as Fig. 1 gives a graph G with $\theta(G) = 9$ and $\chi(G) = 5$. This graph is adapted from [83]. Note that every 4-coloring of the subgraph induced by x and y and the upper three vertices assigns x and y the same color since the upper three vertices form a K_3 . On the other hand, every 4-coloring of the subgraph induced by x and y and the lower four vertices assigns x and y different colors since the lower four vertices form a K_4 . It follows that $\chi(G) > 4$.

Kierstead and Kostochka also constructed infinite families of connected graphs H with $\theta(H) \leq 7$ and $\chi(H) = 4$. Let G be a graph with $\theta(G) \leq 7$ and $\chi(G) = 4$. An example for such a graph G is illustrated in Fig. 2a. A graph G' with $\theta(G') \leq 7$ and $\chi(G') = 4$ can be constructed as follows. Choose a vertex v of G that has no neighbor of degree 4. Split v into two vertices v_1 and v_2 of degree at most two. Add two new adjacent vertices x_v and y_v , each of which is joined to both v_1 and v_2 . In this example, G' is depicted in Fig. 2b. By construction, $\theta(G') = 7$. Since v_1 and v_2 are adjacent to x_v and y_v , any 3-coloring of G' will assign the same color to v_1 and v_2 . But then a 3-coloring of G can be produced, contrary to the assumption $\chi(G) = 4$.

Kierstead and Kostochka [81] proved a generalization of the Hajnal and Szemerédi Theorem involving the Ore degree.

Theorem 82 *For every $r \geq 3$, each graph G with $\theta(G) \leq 2r + 1$ has an equitable $(r + 1)$ -coloring.*

This implies that the $E\Delta CC$ holds for graphs in which vertices of maximum degree form an independent set. In addition to K_{r+1} , the extremal graphs for the above theorem are $K_{m,2r-m}$ for every odd $0 < m \leq r$. The following Ore-type analogue of the $E\Delta CC$ was also proposed in [81] and, its truth for $r = 3$ was established.

Conjecture 17 *Let $r \geq 3$ and G be a connected graph with $\theta(G) \leq 2r$. If G differs from K_{r+1} and $K_{m,2r-m}$ for all odd m , then G is equitably r -colorable.*

A conjecture in the flavor of Conjecture 15 was proposed in [86], and positive evidence was provided for small θ .

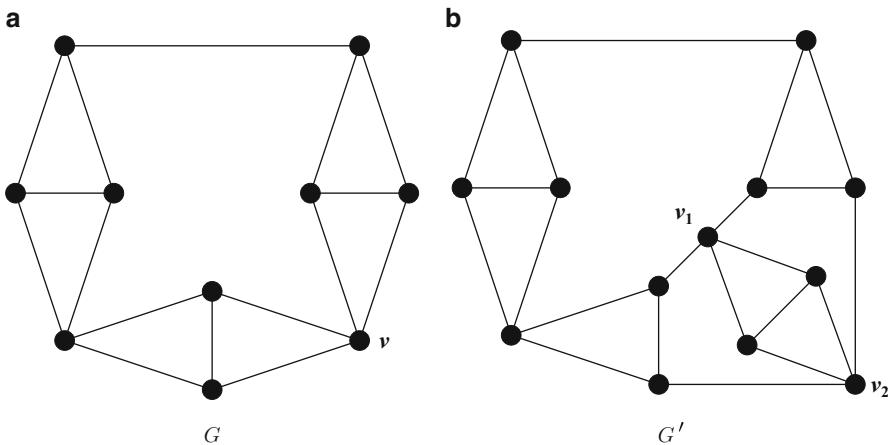


Fig. 2 A transformation from G to G'

Conjecture 18 *Every graph G is equitably $(\lceil \theta(G)/2 \rceil + 1)$ -choosable.*

Theorem 83 *If $\theta(G) \leq 6$, then G is equitably 4-choosable.*

In the graph packing area, a major outstanding conjecture was made independently by Bollobás and Eldridge [15] and Catlin [20, 21].

Conjecture 19 *Let G_1 and G_2 be two graphs of the same order n . If $(\Delta(G_1) + 1)(\Delta(G_2) + 1) \leq n + 1$, then G_1 and G_2 pack.*

The Hajnal-Szemerédi Theorem verifies the conjecture in the case when G_2 is the disjoint union of copies of a clique. Aigner and Brandt [1] and independently (for huge n) Alon and Fischer [4] settled the case $\Delta(G_1) \leq 2$. Csaba et al. [36] proved the case for $\Delta(G_1) = 3$ and huge n . Bollobás et al. [17] proved it in case that for $d \geq 2$, G_1 is d -degenerate, $\Delta(G_1) \geq 40d$, and $\Delta(G_2) \geq 215$. Kaul et al. [86] showed that for $\Delta(G_1), \Delta(G_2) \geq 300$, if $(\Delta(G_1) + 1)(\Delta(G_2) + 1) < 3n/5$, then G_1 and G_2 pack. Recently, Kun has announced a proof of the conjecture for graphs with at least 10^8 vertices.

The reader is referred to Wozniak [151] for a general survey on the graph packing area and to Kierstead et al. [79] for recent progress in Ore-type and Dirac-type bounds for graph packing problems.

16 Equitable Δ -Colorability of Disconnected Graphs

If a disconnected graph G is $\Delta(G)$ -colorable, then the conditions for G to be equitably $\Delta(G)$ -colorable are quite different. For an odd integer $r \geq 3$, G is equitably $\Delta(G)$ -colorable if $G = K_{r,r} \cup K_{r,r}$. However, G is not equitably

$\Delta(G)$ -colorable if $G = K_{r,r} \cup K_r$. The latter example can be generalized. A graph H is said to be r -equitable if r divides $|H|$, H is r -colorable, and every r -coloring of H is equitable. For an odd integer $r \geq 3$, if $H = K_{r,r}$ is a subgraph of G and $G - H$ is r -equitable, then G is not equitably r -colorable. Kierstead and Kostochka [84] gave a good description of the family of all r -equitable graphs so that all of them can be built up from simple examples in a straightforward way. Their approach to deal with disconnected graphs led them to propose the following conjecture.

Conjecture 20 Suppose that $\Delta(G) = r \geq 3$ and G is an r -colorable graph. Then G is not equitably r -colorable if and only if the following conditions hold:

1. r is odd.
2. G has a subgraph $H = K_{r,r}$.
3. $G - H$ is r -equitable.

Chen and Yen [27] have also found sufficient conditions for the nonexistence of equitable Δ -colorings for graphs that are not necessarily connected.

Theorem 84 Suppose that $\Delta(G) = r \geq 3$ and G is an r -colorable graph. Then G is not equitably r -colorable if the following conditions hold:

1. r is odd.
2. G has exactly one component $H = K_{r,r}$.
3. $\alpha(G - H) \leq |G - H|/r$.

Suppose that $\Delta(G) = r$ and G is an r -colorable graph such that G has exactly one $K_{r,r}$ component. If $G = K_{r,r}$, then $\alpha(G - K_{r,r}) = 0 = |G - K_{r,r}|/r$. Otherwise, $\alpha(G - K_{r,r}) \geq |G - K_{r,r}|/\chi(G - K_{r,r}) \geq |G - K_{r,r}|/\chi(G) \geq |G - K_{r,r}|/r$. Hence, Condition 3 can be replaced by an equality. Chen and Yen conjectured that those sufficient conditions are also necessary and established some positive evidence.

Conjecture 21 Suppose that $\Delta(G) = r \geq 3$ and G is an r -colorable graph. Then G is not equitably r -colorable if and only if the following conditions hold:

1. r is odd.
2. G has exactly one component $H = K_{r,r}$.
3. $\alpha(G - H) = |G - H|/r$.

Theorem 85 A bipartite graph G satisfying $\Delta(G) \geq 2$ is equitably $\Delta(G)$ -colorable if and only if G is different from $K_{r,r}$ for all odd $r \geq 3$.

Theorem 86 A graph G that is $\Delta(G)$ -colorable and satisfies $\Delta(G) \geq 1 + |G|/3$ is equitably $\Delta(G)$ -colorable if and only if G is different from $K_{r,r}$ for all odd $r \geq 3$.

Theorem 87 Conjecture 21 holds for $\Delta(G) = 3$.

It was shown in Chen et al. [32] that Conjecture 20 and 21 are in fact equivalent. The proof utilizes the following result that was established in Chen et al. [30].

Theorem 88 Let $r \geq \chi(G) \geq \Delta(G)$. Then there exists a proper coloring of G using r colors such that some color class has size $\alpha(G)$.

Further lemmas are needed in the equivalence proof.

Lemma 7 If G is r -colorable and $\alpha(G) = |G|/r$, then G is r -equitable.

Lemma 8 If G is r -equitable, then $r = \chi(G)$.

Lemma 9 Let $r \geq \Delta(G)$. If G is r -equitable, then $\alpha(G) = |G|/r$.

By Lemma 8, $\chi(G) = r \geq \Delta(G)$. By Theorem 88, there exists an r -coloring ϕ of G such that a color class of ϕ is of size $\alpha(G)$. Since G is r -equitable, ϕ is an equitable r -coloring and r divides $|G|$. Hence $\alpha(G) = |G|/r$.

Lemma 10 Let a graph G satisfy $\Delta(G) = r \geq 3$. If G is r -equitable, then G does not have $K_{r,r}$ as a subgraph.

Suppose on the contrary that G has a subgraph $H = K_{r,r}$. The subgraph H is a component of G since $r = \Delta(G)$. Hence, $|G| = |G - H| + |H| = |G - H| + 2r$ and $\alpha(G) = \alpha(G - H) + \alpha(H) = \alpha(G - H) + r$. By Lemma 9, $\alpha(G) = |G|/r = |G - H|/r + 2$. Therefore, $\alpha(G - H) = |G - H|/r + 2 - r \leq |G - H|/r - 1$. On the other hand, Lemma 8 implies $r = \chi(G) \geq \chi(G - H)$. Then $\alpha(G - H) \geq \lceil |G - H|/\chi(G - H) \rceil \geq \lceil |G - H|/r \rceil \geq |G - H|/r$. A contradiction is obtained.

Lemma 11 Let a graph G satisfy $\Delta(G) = r \geq \chi(G)$. If $r \geq 3$, G has a subgraph $H = K_{r,r}$, and $G - H$ is r -equitable, then G has exactly one $K_{r,r}$ component.

If $G = K_{r,r}$ or $\Delta(G - H) < r$, then G has exactly one $K_{r,r}$ component H . Now, suppose that $G \neq K_{r,r}$ and $\Delta(G - H) = r$. Since $r \geq 3$ and $G - H$ is r -equitable, $G - H$ does not have $K_{r,r}$ as a subgraph by Lemma 10. Therefore, G has exactly one $K_{r,r}$ component H .

Using Lemmas 7, 9, and 11, the equivalence of two conjectures follows.

Theorem 89 Conjecture 20 and 21 are equivalent.

In [84], Kierstead and Kostochka described their conjecture in terms of other equivalent conditions. An r -equitable graph G is said to be r -reducible if $V(G)$ has a partition $\{V_1, \dots, V_t\}$ into at least two parts such that all induced subgraphs $G[V_i]$ are r -equitable. If such a partition fails to exist, then G is called r -irreducible. Obviously, K_r is r -irreducible. It can be identified that there is one other 5-irreducible graph F_1 (Fig. 3), three other 4-irreducible graphs F_2, F_3, F_4 (Fig. 4), and six other 3-irreducible graphs F_5, \dots, F_{10} (Figs. 5 and 6). Together with K_r , the r -irreducible graphs in the list F_1, \dots, F_{10} are called r -basic graphs.

Fig. 3 The 5-irreducible graph F_1

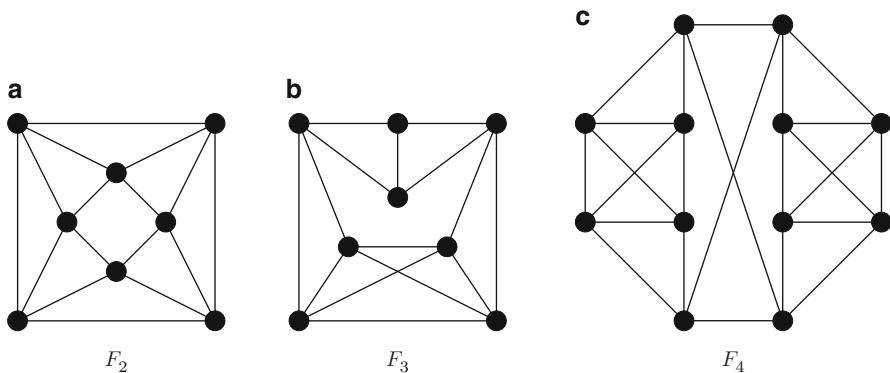
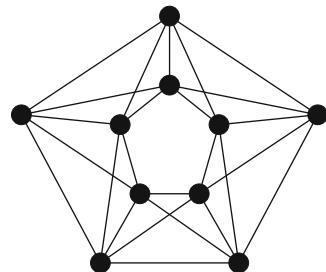


Fig. 4 The 4-irreducible graphs F_2 , F_3 , and F_4

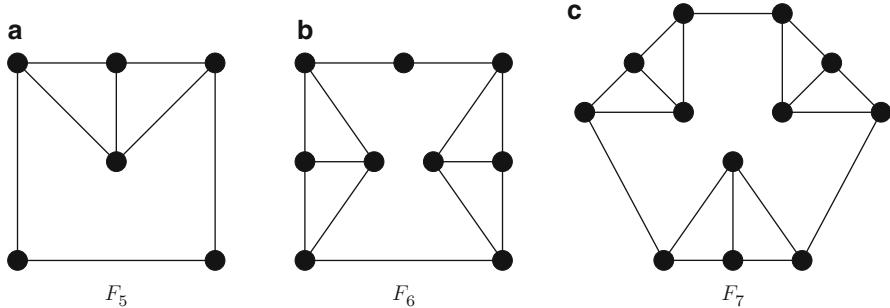


Fig. 5 The 3-irreducible graphs F_5 , F_6 , and F_7

An r -decomposition of G is a partition $\{V_1, \dots, V_t\}$ of $V(G)$ such that every induced subgraph $G[V_i]$ is r -basic. The graph G is called r -decomposable if it has an r -decomposition. A nearly equitable r -coloring of a graph G is an r -coloring of G such that exactly one color class has size $s - 1$, exactly one color class has size $s + 1$, and all other color classes have size s .

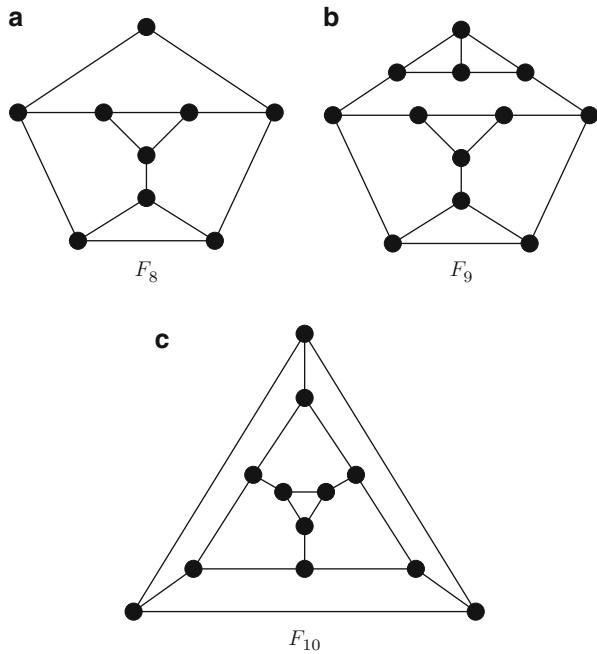


Fig. 6 The 3-irreducible graphs F_8 , F_9 , and F_{10}

Let $\mathcal{G}(r)$ be the class of all graphs whose maximum degree and chromatic number are less than or equal to r . Let $\mathcal{G}(r, n)$ denote the subclass of $\mathcal{G}(r)$ consisting of all graphs of order at most n .

Theorem 90 ([84]) *Let $G \in \mathcal{G}(r)$ and r divide $|G|$. Then the following are equivalent.*

1. *G is r -equitable.*
2. *G is r -decomposable.*
3. *G has an equitable r -coloring, but does not have a nearly equitable r -coloring.*

Conjecture 20 can now be re-stated as follows.

Conjecture 22 *Suppose that $\Delta(G) = r \geq 3$ and G is an r -colorable graph. Then G is not equitably r -colorable if and only if the following conditions hold.*

1. *r is odd.*
2. *G has a subgraph $H = K_{r,r}$.*
3. *$G - H$ is r -decomposable.*

For $r \geq 6$, this conjecture means that, if an r -colorable graph G with $\Delta(G) = r$ has no equitable r -coloring, then r is odd and there exists a partition $V(G) = V_0 \cup$

$V_1 \cup \dots \cup V_t$ such that $G[V_0] = K_{r,r}$ and $G[V_i] = K_r$ for $1 \leq i \leq t$. [Theorem 90](#) can be used to derive the corollaries below.

Corollary 13 *For all positive integers r and $n > r$, the EΔCC holds for all graphs in $\mathcal{G}(r, n)$ if and only if [Conjecture 22](#) holds for all graphs in $\mathcal{G}(r, n)$.*

Corollary 14 *Let $G \in \mathcal{G}(r)$ be r -equitable. Then G has a unique r -decomposition.*

Corollary 15 *There exists a polynomial time algorithm for deciding whether a graph $G \in \mathcal{G}(r)$ is r -equitable.*

The following conjecture proposed in [83] is a common extension of [Conjecture 17](#) and [22](#).

Conjecture 23 *Let $r \geq 3$. An r -colorable graph G with $\theta(G) \leq 2r$ has no equitable r -coloring if and only if r divides $|G|$, G has a subgraph $H = K_{m,2r-m}$ for some odd m , and $G - H$ is r -decomposable.*

This conjecture is proved to be equivalent to [Conjecture 17](#) for graphs with restricted order and Ore-degree.

Theorem 91 *Let $r \geq 3$. Assume that [Conjecture 17](#) holds for all graphs of order at most n and Ore-degree at most $2r$. Let G be an r -colorable graph of order n with $\theta(G) \leq 2r$. Then G has no equitable r -coloring if and only if r divides n , G has a subgraph $H = K_{m,2r-m}$ for some odd m , and $G - H$ is r -decomposable.*

It follows that [Conjecture 23](#) holds for $r = 3$.

17 More on the Hajnal-Szemerédi Theorem

The Hajnal-Szemerédi Theorem settled a conjecture raised by Erdős in 1964. The complete proof given in [64] was long and complicate, and did not produce a polynomial time algorithm. A simplification came when Kierstead and Kostochka [82] used a discharging argument in an approach similar to the original one. An analysis of their proof leads to a complexity results.

Theorem 92 *There is an algorithm of time complexity $O(n^5)$ that constructs an equitable $(r + 1)$ -coloring of any graph G with $|G| = n$ and $\Delta(G) \leq r$.*

An even shorter proof of the Hajnal-Szemerédi Theorem was included in the survey paper [86]. Independent of Kierstead and Kostochka, Mydlarz and Szemerédi also found polynomial time algorithm proof for the Hajnal-Szemerédi Theorem. These two groups finally worked together to refine their old methods and arrived at a faster algorithm [87].

Theorem 93 *There is an algorithm of time complexity $O(rn^2)$ that constructs an equitable $(r + 1)$ -coloring of any graph G with $|G| = n$ and $\Delta(G) \leq r$.*

However, the existence of an algorithmic version of [Theorem 82](#) is still open.

Conjecture 24 *There is a polynomial time algorithm that constructs an equitable $(r + 1)$ -coloring of any graph G with $\theta(G) \leq 2r + 1$.*

When one pays attention to the complement of the graph given in the Hajnal-Szemerédi Theorem, the theorem can be stated in the following form of which the first non-trivial case $r = 3$ had previously been solved by Corrádi and Hajnal [\[34\]](#).

Theorem 94 *Let G be a graph of order n with the minimum degree $\delta(G) \geq \frac{r-1}{r}n$. If r divides n , then G contains n/r vertex-disjoint cliques of size r .*

In order to extend the above to a multipartite version, the next conjecture was proposed in Csaba and Mydlarz [\[35\]](#). If all parts of an r -partite graph G have the same size, then G is called a *balanced r -partite graph*. Let V_1, V_2, \dots, V_r be the parts of such a graph G . The *proportional minimum degree* of G is defined as follows.

$$\tilde{\delta}(G) = \min_{1 \leq i \leq r} \min_{v \in V_i} \left\{ \frac{\deg(v, V_j)}{|V_j|} \mid j \neq i \right\}.$$

Conjecture 25 *Let G be a balanced r -partite graph of order rn . There exists a constant $M \geq 0$ such that, if $\tilde{\delta}(G)n \geq \frac{r-1}{r}n + M$, then G contains n vertex-disjoint cliques of size r .*

The extra additive constant M is necessary for the case of odd r . Partial results have been obtained by Fischer [\[50\]](#) and Johansson [\[75\]](#). The conjecture has been confirmed for $r = 3$ [\[109\]](#) and $r = 4$ [\[111\]](#). In [\[35\]](#), Csaba and Mydlarz proved a relaxed version of this conjecture. The proofs commonly used Szemerédi's Regularity Lemma [\[136\]](#) and the Blow-up Lemma [\[88\]](#) and are complicate. The cases for $r = 3$ and $r = 4$ have been reproved by Han and Zhao [\[65\]](#) using their absorbing method. The paper [\[80\]](#) by Keevash and Mycroft contains results about multipartite graphs with sufficiently large girth. Recently, [Conjecture 25](#) has been verified asymptotically by Lo and Markström [\[106\]](#). Balogh et al. [\[8\]](#) extended the result of Corrádi and Hajnal into the setting of sparse random graphs.

In order to find a understandable proof of the Hajnal-Szemerédi Theorem, Seymour [\[132\]](#) was motivated to pose the following conjecture. The *r -th power* of a graph is obtained by adding new edges joining vertices with distance at most r .

Conjecture 26 *Let G be a graph of order n with $\delta(G) \geq \frac{r}{r+1}n$. Then G contains the r -th power of C_n .*

[Theorem 77](#) is the case $r = 1$. The well-known Posa's conjecture (cf. [\[43\]](#)) is the case $r = 2$. Seymour's conjecture implies the Hajnal-Szemerédi Theorem since

any $r + 1$ consecutive vertices of the r -th power of a cycle induce K_{r+1} . Seymour's conjecture has been proved by Komolós et al. [89, 90] when the order of the graph is extremely large using Szemerédi's Regularity Lemma and the Blow-up Lemma.

Theorem 95 *For every positive integer r , there exists an integer N such that for every $n > N$ every graph G of order n with $\delta(G) \geq \frac{r}{r+1}n$ contains the r -th power of a hamiltonian cycle.*

18 Applications

Graph coloring can be regarded as a partition of resources problem. It is convenient in some situations to require color classes be of approximately the same size. Graph coloring is also a natural model for scheduling problems. Suppose that a number of jobs are given to be completed. A *conflict graph* can be constructed so that vertices represent jobs and two vertices are adjacent if there is a scheduling conflict between the jobs associated with these vertices. A proper coloring of the conflict graph corresponds to a conflict-free schedule. Some other examples are listed below.

1. The mutual exclusion scheduling problem [7, 134].
2. Scheduling in communication systems [70].
3. Round-the-clock scheduling [138].
4. Parallel memory systems [11, 37].
5. Load balancing in task scheduling [11, 97].
6. Constructing university timetables [55].

In applications, only algorithms with low time complexity could be utilized. Although checking $\chi_-(G) \leq 2$ can be achieved in polynomial time, the problem of deciding if $\chi_-(G) \leq 3$ is NP-complete even for line graphs of cubic graphs. The majority of known polynomial time results about equitable coloring are listed in [53] and [55]. Two competitive algorithms for approximate equitable coloring were also supplied. In [112, 113], and [114], Méndez-Díaz et al. gave a linear integer programming formulation for the equitable coloring problem. They studied its polyhedral structure and developed a cutting plane algorithm. Experiments on randomly generated graphs have been performed to make behavior comparisons with other algorithms of similar nature. Bahiense et al. [6] also gave two integer programming formulations based on representatives for the equitable coloring problem. They proposed a primal constructive heuristic, branching strategies, and branch-and-cut algorithms based on these formulations. Computational experiments were carried out on randomly generated graphs.

Applications of equitable colorings to other mathematical problems include the following.

Alon and Füredi [5] used equitable colorings to the determination of the threshold function for the edge probability that guarantees, almost surely, the existence of various sparse spanning subgraphs in random graphs.

Rödl and Ruciński [128] used equitable colorings to give a new proof of the Blow-Up Lemma.

Let $\mathcal{X} = \{X_1, X_2, \dots, X_n\}$ be a collection of random variables with $S = \sum_{i \in [n]} X_i$ and $\mu = E[S]$. The upper tail probability $\text{Prob}[S \geq (1 + \epsilon)\mu]$ and the lower tail probability $\text{Prob}[S \leq (1 - \epsilon)\mu]$ for $0 < \epsilon \leq 1$ are subjects of interest. A *dependency graph* for \mathcal{X} has vertex set $[n]$ and an edge set such that for each $i \in [n]$, X_i is mutually independent of all other X_j with j non-adjacent to i . In Pemmaraju [124], it is shown that a small equitable chromatic number for a dependency graph leads to sharp tail probability bounds that are roughly as good as those would have been obtained had the random variables been mutually independent. An example is an outerplanar dependency graph and [Theorem 23](#) plays an important role in the proof.

Pemmaraju also introduced an interesting notion of proportional coloring. For non-negative integer c and integer $\alpha \geq 1$, a proper coloring of G is said to be a (c, α) -coloring if all except at most c vertices are colored and $|V_i| \leq \alpha|V_j|$ for any pair of color classes V_i and V_j . [Theorem 1](#) can be extended to show that every tree has a $(1, 5)$ -coloring with two colors and every outerplanar graph has a $(2, 5)$ -coloring with four colors. Sharp tail probability bounds can be established if the dependency graph can be (c, α) -colored [124].

Pemmaraju believed that $\chi_-(G)$ should depend on the average degree rather than on the maximum degree and he proposed the conjecture below.

Conjecture 27 *There is a positive constant c such that, if a graph G has maximum degree at most $|G|/c$ and average degree d , then $\chi_-(G) = O(\chi(G) + d)$.*

The truth of this conjecture will immediately imply an $O(1)$ equitable chromatic number for most planar graphs and that will translate into extremely sharp tail probability bounds for the sum of random variables that have a planar dependency graph.

Janson et al. [71] and Janson and Ruciński [72] also used equitable colorings to get new bounds on tails of distributions of sums of random variables.

19 Related Notions of Coloring

In this section, some coloring notions related to equitable coloring will be discussed.

19.1 Equitable Edge-Coloring

An *edge-coloring* of a graph G is simply an assignment of colors to the edges of G . An edge- k -coloring of G assigns the k colors $\{1, 2, \dots, k\}$ to the edges of G . For a vertex v of G , let $c_i(v)$ denote the number of edges incident with v colored i . This edge- k -coloring is said to be *equitable* if, for each vertex v ,

$$|c_i(v) - c_j(v)| \leq 1 \quad (1 \leq i < j \leq k)$$

and *nearly equitable* if, for each vertex v ,

$$|c_i(v) - c_j(v)| \leq 2 \quad (1 \leq i < j \leq k).$$

The following was proved in [68].

Theorem 96 *If $k \geq 2$ does not divide the degree of any vertex, then the graph has an equitable edge- k -coloring.*

Hilton and de Werra made the observation at the end of their paper that the colorings can be so constructed that all color classes have equitable sizes.

An edge-coloring is said to be *proper* if any two edges are colored differently whenever they are incident to a common vertex. The smallest number of colors needed in a proper edge-coloring of G is called the *chromatic index* of G and denoted by $\chi'(G)$. Obviously, $\Delta(G) \leq \chi'(G)$. The above theorem reduces to the following well-known theorem of Vizing [142] when $k = \Delta(G) + 1$.

Theorem 97 *For any graph G , $\chi'(G) \leq \Delta(G) + 1$.*

An *edge cover coloring* of a graph G is a coloring of the edges of G so that, for every vertex v , each color appears at least once on some edge incident to v . The maximum number of colors needed for such a coloring is called the *edge cover chromatic index* of G and denoted by $\chi'_c(G)$. Let $k = \delta(G) + 1$. Applying Theorem 96 to the graph obtained from G by adjoining a pendant edge to each vertex v whose degree is a multiple of k , the following result of Gupta [62] is deduced.

Theorem 98 *For any graph G , $\delta(G) - 1 \leq \chi'_c(G) \leq \delta(G)$.*

Given $k \geq 2$, the k -core of a graph G is the subgraph of G induced by all vertices v whose degree is a multiple of k . A stronger form for Theorem 96 also appeared in [68].

Theorem 99 *Given $k \geq 2$, if the k -core of G contains no edges, then G has an equitable edge- k -coloring.*

The following recent result of Zhang and Liu [158] was first conjectured by Hilton and de Werra [68].

Theorem 100 *Given $k \geq 2$, if the k -core of G is a forest, then G has an equitable edge- k -coloring.*

A graph G is called *edge-equitable* if G has an equitable edge- k -coloring for any integer $k \geq 2$. If a graph is edge-equitable, then its chromatic index is equal to its maximum degree and its edge cover chromatic index is equal to its minimum degree. All bipartite graphs were shown to be edge-equitable in de Werra [38].

A *circuit* is a connected graph without any vertex of odd degree. A circuit is said to be odd, or even, according to its number of edges is odd, or even. It was stated in [38] that a connected graph has an equitable edge-2-coloring if and only if it is not an odd circuit. Wu [152] showed that a connected outerplanar graph is edge-equitable if and only if it is not an odd circuit. This can be generalized to series-parallel graphs. The following result was established by Song et al. [135].

Theorem 101 *Any connected series-parallel graph is edge-equitable if and only if it is not an odd circuit.*

Theorem 96 also has the following corollary.

Corollary 16 *Let $k \geq 2$. Then, for any graph G , there exists an edge- k -coloring of G such that, for each vertex v ,*

1. $|c_i(v) - c_j(v)| = 2$ for at most one pair $\{i, j\}$ of colors, and
2. $|c_s(v) - c_t(v)| \leq 1$ for all pairs $\{s, t\} \neq \{i, j\}$ of colors.

This can be proved by adjoining a pendant edge to each vertex of G whose degree is a multiple of k . Then apply Theorem 96 to this extended graph. The corresponding edge-coloring of G satisfies the above conditions. Thus, every graph has a nearly equitable edge- k -coloring for any given $k \geq 2$. In fact, Corollary 16 is equivalent to Theorem 96. To see this, suppose that k does not divide the degree of any vertex of G and the conditions of Corollary 16 are satisfied. For each vertex v of G , since $\deg(v)$ is not a multiple of k , it is not possible to have $|c_i(v) - c_j(v)| = 2$ for one pair $\{i, j\}$ of colors unless some second pair $\{s, t\} \neq \{i, j\}$ of colors satisfies $|c_s(v) - c_t(v)| = 2$ also. Thereby, an equitable edge-coloring with k colors can be produced.

For efficient algorithms for nearly equitable edge-coloring problem, the reader is referred to Shioura and Yagiura [133], Xie et al. [154], and references therein. These algorithms can produce color classes that have equitable sizes.

19.2 Equitable Total-Coloring

A *total- k -coloring* of a graph $G = (V, E)$ is a mapping $f : V(G) \cup E(G) \rightarrow \{1, 2, \dots, k\}$ such that any two adjacent or incident elements have distinct images. The *total chromatic number* $\chi''(G)$ is the smallest integer k such that G has a total- k -coloring. A total- k -coloring is said to be *equitable* if $||f^{-1}(i)| - |f^{-1}(j)|| \leq 1$ when $1 \leq i < j \leq k$. Fu [52] first studied equitable total-coloring under the name *equalized total coloring* and put forward the following tentative conjecture.

Conjecture 28 *For each $k \geq \chi''(G)$, there exists an equitable total- k -coloring of G .*

Fu proved this conjecture for complete graphs, complete bipartite graphs, trees, and complete split graph $K_m \vee I_n$. However, the tentative conjecture does not hold in general. Fu gave a family of counterexamples.

Theorem 102 *For any $n \geq 3$, let $G = nK_2 \vee I_{2n-1}$. Then $\chi''(G) = 2n + 1$, yet G has no equitable total- $(2n + 1)$ -coloring. However, G has an equitable total- k -coloring for any $k \geq 2n + 2 = \Delta(G) + 2$.*

This theorem prompted Fu to make the following refined conjecture.

Conjecture 29 *For each $k \geq \max\{\chi''(G), \Delta(G) + 2\}$, G has an equitable total- k -coloring.*

Fu also proved that G satisfies the above conjecture if $\Delta(G) = |G| - 2$, or G is a complete multipartite graph of odd order. When G is a multipartite graph of even order, the best Fu could prove was that G has an equitable total- k -coloring for any $k \geq \Delta(G) + 3$. Wang [145] proposed the following weaker conjecture in which the *equitable total chromatic index* $\chi''_=(G)$ of a graph G is defined to be the least integer k such that G has an equitable total- k -coloring. Wang proved the conjecture holds for graphs G with $\Delta(G) \leq 3$.

Conjecture 30 *For any graph G , $\chi''_=(G) \leq \Delta(G) + 2$.*

This weaker conjecture is powerful enough to imply the outstanding Total Coloring Conjecture, proposed independently by Behzad [9] and Vizing [143], which asserts $\chi''(G) \leq \Delta(G) + 2$ for any graph G .

For some classes of graphs, the upper bound in the above conjecture can be lowered to $\Delta(G) + 1$. Chungling et al. [33] proved the following for the square product of cycles.

Theorem 103 *Let $G = C_m \square C_n$. Then, for $m \geq 3$ and $n \geq 3$, $\chi''_=(G) = \Delta(G) + 1 = 5$.*

The *wheel* W_n is defined to be the join of a cycle C_n with a center vertex u . The *Mycielskian* $\mathbf{M}(W_n)$ of W_n is constructed as follows. For each vertex x of W_n , a new vertex x' is added. Join x_i and x'_j with an edge whenever x_i and x_j are adjacent in W_n . Finally, add one more new vertex w and make it adjacent to every new vertices x' . The *hypo-Mycielskian* $\mathbf{HM}(W_n)$ of W_n has the same vertex set as $\mathbf{M}(W_n)$ and all edges of $\mathbf{M}(W_n)$ except those of the form ux' , $x \neq u$. Note that $\Delta(\mathbf{HM}(W_n))$ is equal to 6 when $n = 3$ or 4 and is equal to $n + 1$ when $n \geq 5$.

Theorem 104 ([147]) *For $n \geq 3$, $\chi''_=(\mathbf{M}(W_n)) = \Delta(G) + 1$.*

Theorem 105 ([148]) *For $n \geq 3$, $\chi''_=(\mathbf{HM}(W_n)) = \Delta(G) + 1$.*

One question left open in [52] asks whether the existence of an equitable total- k -coloring implies that of an equitable total- $(k + 1)$ -coloring.

19.3 Equitable Defective Coloring

A reasonable relaxation of scheduling problem allows conflicts to happen to a certain level. The graph coloring model corresponds to this type of relaxation can be formulated as follows. A *d-defective coloring* is a coloring of the vertices of a graph in which monochromatic subgraphs have maximum degree at most d . An ordinary proper coloring is precisely a 0-defective coloring. A *defective coloring* simply means a 1-defective coloring, in which a vertex may share a color with at most one neighbor. The smallest number k such that the graph G has a 1-defective coloring is denoted by $\text{df}_1(G)$.

A graph G has an *equitable defective k -coloring*, or an *ED- k -coloring*, if G has a coloring using k colors that is both equitable and defective. This notion of coloring was introduced by Williams et al. [150]. Let $\chi_{\text{ED}}(G)$ denote the smallest integer k such that G has an ED- k -coloring, and $\chi_{\text{ED}}^*(G)$ the smallest integer k such that G has an ED- m -coloring for all $m \geq k$. It is clear that $\text{df}_1(G) \leq \chi_{\text{ED}}^*(G) \leq \chi_{\text{ED}}(G)$. These parameters may differ from each other by an arbitrarily large amount. The following example was provided in [150]. Consider $X = K_{\lceil n/2 \rceil}$ and $Y = I_{\lfloor n/2 \rfloor}$. Let G be the graph obtained from $X \vee Y$ by removing a matching between X and Y of size $\lfloor n/2 \rfloor$. Note that $\chi(G) = \chi_{\text{ED}}^*(G) = \lceil n/2 \rceil$. If X is colored with $|X|/2 = \lceil n/4 \rceil$ colors and Y is colored with one extra color, then $\text{df}_1(G) \leq \lceil n/4 \rceil + 1$. If a color class in an ED-coloring contains two vertices of X , then it cannot contain any other vertices. This forces every color class has at most three vertices. However, color classes of size three must contain at least two vertices of Y . Therefore, there are at most $|Y|/2$ color classes of size three. It follows that $\chi_{\text{ED}}(G) \geq \lceil 3n/8 \rceil$. It is easy to see that an ED-coloring with k colors exists for any $k \geq \lceil 3n/8 \rceil$, and hence $\chi_{\text{ED}}^*(G) = \lceil 3n/8 \rceil$.

Extending results in [108], Williams et al. proved the following.

Theorem 106 *If G is a planar graph with minimum degree $\delta(G) \geq 2$ and girth $g(G) \geq 10$, then $\chi_{\text{ED}}^*(G) \leq 3$.*

The condition $\delta(G) \geq 2$ is indispensable since $K_{1,n}$ has no ED- k -coloring when n is sufficiently large for any fixed k . The girth condition cannot be lower than 5 since $K_{2,n}$ has girth 4 but $\chi_{\text{ED}}^*(K_{2,n})$ is not bounded by any constant. Whether 5 is the smallest girth that a planar graph G can have so that $\chi_{\text{ED}}^*(G)$ can be bounded by a constant is an open question.

Recently, Fan et al. [46] have shown that a graph with maximum degree at most r admits an equitable ED- r -coloring and provided a polynomial-time algorithm for constructing such a coloring.

19.4 Equitable Coloring of Hypergraphs

A *hypergraph* \mathcal{H} , is an ordered pair (V, \mathcal{F}) , where \mathcal{F} is a family of subsets of the finite set V . Elements of V and \mathcal{F} are called *vertices* and *hyperedges* of \mathcal{H} , respectively. The number of hyperedges incident with a vertex of \mathcal{H} is called its *degree*. A hypergraph is said to be k -uniform if each hyperedge has precisely k -elements.

Let \mathcal{H} be a k -uniform hypergraph on n vertices. A *strong r -coloring* of \mathcal{H} is a partition of the vertices into r parts, called *color classes*, such that each hyperedge intersects each part. A strong r -coloring is called *equitable* if the size of each color class is either $\lceil n/r \rceil$ or $\lfloor n/r \rfloor$. Let $c(\mathcal{H})$ and $ec(\mathcal{H})$ denote the maximum possible number of color classes in a strong coloring and an equitable coloring of \mathcal{H} , respectively. Clearly, $1 \leq ec(\mathcal{H}) \leq c(\mathcal{H}) \leq k$. If no upper bounds are imposed on the maximum degree, then $ec(\mathcal{H}) = c(\mathcal{H}) = 1$ could happen even k is large. An example is the complete k -uniform hypergraph on $2k$ vertices that satisfies $c(\mathcal{H}) = 1$ and maximum degree less than 4^k .

Let $a \geq 1$ be any real number, and let $\epsilon > 0$ be small. Let k be sufficiently large such that there exists an integer in the interval $[k/(1 + \epsilon^2/4)a \ln k, k/(1 + \epsilon^2/8)a \ln k]$. For some γ in the interval $[\epsilon^2/8, \epsilon^2/4]$, the number $k/(1 + \gamma)a \ln k$ is an integer. Let \mathcal{H} be a k -uniform hypergraph with maximum degree at most k^a . Yuster [157] proved that there exists an equitable coloring of \mathcal{H} with $k/(1 + \gamma)a \ln k - \lceil k\sqrt{\gamma}/a \ln k \rceil > (1 - \epsilon)k/a \ln k$ colors and the following was established.

Theorem 107 *If $a \geq 1$ and \mathcal{H} is a k -uniform hypergraph with maximum degree at most k^a , then $ec(\mathcal{H}) \geq \frac{k}{a \ln k} (1 - o_k(1))$. The lower bound is asymptotically tight. For all $a \geq 1$, there exists k -uniform hypergraphs \mathcal{H} with maximum degree at most k^a and $c(\mathcal{H}) \leq \frac{k}{a \ln k} (1 + o_k(1))$.*

Note that results in Alon [3] have already implied that no equitable coloring of a k -uniform hypergraph could have more than $(k/\ln k)(1 + o_k(1))$ color classes. Yuster made the following remarks at the end of his paper [157].

Using more involved computations, Theorem 107 can be proved when a is not a constant but satisfies $a = a(k) = o(k/\ln k)$, i.e., the degree of \mathcal{H} is allowed to be any subexponential function of k .

It is possible to convert the proof of Theorem 107 into an algorithmic one. In terms of the number of vertices of the hypergraph, but not in its uniformity, a polynomial time algorithm can be found to produce an equitable partition with $(1 - o_k(1))ck/(a \ln k)$ color classes, where c is a fixed small constant depending only on a .

A special case of Theorem 107 gives the following interesting result about graphs. Let G be a k -regular graph. If k is sufficiently large, then G has an equitable

coloring with $(1 - o_k(1))(k / \ln k)$ colors such that each color class is a so-called *total dominating set*, that is a subset of the vertices such that each vertex of G has a neighbor in that subset.

20 Conclusion

The subject of graph coloring occupies a central position in graph theory. Historically speaking, much of the early development of graph theory was motivated by the attempts at solving the four-color conjecture. Later on, a large number of variants or generalizations of graph coloring problem have emerged. Graph coloring involves deep mathematical and computational issues. The possibilities for applications are also wide. When resources are allocated, a certain kind of graph coloring model may be lurking around. The requirement for even distribution is rather natural. The equitable coloring goes to the extreme to make color classes differ in size by at most one. This may be too stringent for applications in the real world. Yet it brings up hard questions to be addressed.

Although the concept of an equitable coloring of a graph was introduced in early 1970s, substantial research results have only been accumulated in the last 20 years. The importance of the equitable Δ -coloring conjecture has gradually been recognized. Positive evidence has been collected in this chapter.

Another fundamental phenomenon in equitable graph coloring is that in many graph classes “most” members admit equitable colorings with colors not extremely larger than their ordinary chromatic numbers. This phenomenon needs further investigated.

Equitable coloring of graphs can be formulated in terms of graph packings. This places equitable coloring in a wider context and gives it some previously unexpected connections and it may provide motivations to generate graph packing problems.

Cross-References

- ▶ [Advanced Techniques for Dynamic Programming](#)
- ▶ [Advances in Scheduling Problems](#)
- ▶ [Faster and Space Efficient Exact Exponential Algorithms: Combinatorial and Algebraic Approaches](#)
- ▶ [On Coloring Problems](#)
- ▶ [Online and Semi-online Scheduling](#)
- ▶ [Resource Allocation Problems](#)

Recommended Reading

1. M. Aigner, S. Brant, Embedding arbitrary graphs of maximum degree two. *J. Lond. Math. Soc. II Ser.* **48**, 39–51 (1993)
2. M.M. Akbar Ali, K. Kaliraj, J. Vernold Vivin, On equitable coloring of central graphs and total graphs. *Electron. Notes Discret. Math.* **33**, 1–6 (2009)

3. N. Alon, Transversal numbers of uniform hypergraphs. *Graphs Combin.* **6**, 1–4 (1990)
4. N. Alon, E. Fischer, 2-factors in dense graphs. *Discret. Math.* **152**, 13–23 (1996)
5. N. Alon, Z. Füredi, Spanning subgraphs of random graphs. *Graphs Combin.* **8**, 91–94 (1992)
6. L. Bahiense, Y. Frota, T.F. Noronha, C.C. Ribeiro, A branch-and-cut algorithm for the equitable coloring problem using a formulation by representatives. *Discret. Appl. Math.* doi:10.1016/j.dam.2011.10.008
7. B. Baker, E. Coffman, Mutual exclusion scheduling. *Theor. Comput. Sci.* **162**, 225–243 (1996)
8. J. Balogh, C. Lee, W. Samotij, Corrádi and Hajnal’s theorem for sparse random graphs. arXiv:1011.5443
9. M. Behzad, Graphs and their chromatic numbers. Ph.D. dissertation, Michigan State University, U.S.A., 1965
10. C. Bentz, C. Picouleau, Locally bounded k -colorings of trees. *RAIRO-Oper. Res.* **43**, 27–33 (2009)
11. J. Blazewics, K. Ecker, E. Pesch, G. Schmidt, J. Weglarz, *Scheduling Computer and Manufacturing Processes* (Springer, Berlin, 2001)
12. D. Blum, D. Torrey, R. Hammack, Equitable chromatic numbers of complete multipartite graphs. *MO. J. Math. Sci.* **15**, 75–81 (2003)
13. H.L. Bodlaender, F.V. Fomin, Equitable colorings of bounded treewidth graphs. *Theor. Comput. Sci.* **349**, 22–30 (2005)
14. H.L. Bodlaender, K. Jansen, On the complexity of scheduling incompatible jobs with unit-times, in *Foundations of Computer Science 1993. Lecture Notes in Computer Science*, vol. 711 (Springer, Berlin/New York, 1993), pp. 291–300
15. B. Bollobás, S.E. Eldridge, Maximal matchings in graphs with given maximal and minimal degrees. *Congr. Numer.* **15**, 165–168 (1976)
16. B. Bollobás, R.K. Guy, Equitable and proportional coloring of trees. *J. Comb. Theory Ser. B* **34**, 177–186 (1983)
17. B. Bollobás, A.V. Kostochka, K. Nakprasit, Packing d -degenerate graphs. *J. Comb. Theory Ser. B* **98**, 85–94 (2008)
18. A. Brandstadt, V.B. Le, J.P. Spinrad, *Graph Classes: A Survey* (SIAM, Philadelphia, 1999)
19. R.L. Brooks, On colouring the nodes of a network. *Proc. Camb. Philos. Soc.* **37**, 194–197 (1941)
20. P.A. Catlin, Subgraphs of graphs I. *Discret. Math.* **10**, 225–233 (1974)
21. P.A. Catlin, On the Hajnal-Szemerédi theorem on disjoint cliques. *Util. Math.* **17**, 163–177 (1980)
22. G.J. Chang, A note on equitable colorings of forests. *Eur. J. Comb.* **30**, 809–812 (2009)
23. G. Chartrand, D. Geller, S. Hedetniemi, Graphs with forbidden subgraphs. *J. Comb. Theory Ser. B* **10**, 12–41 (1971)
24. B.-L. Chen, K.-C. Huang, The equitable colorings of Kneser graphs. *Taiwan. J. Math.* **12**, 887–900 (2008)
25. B.-L. Chen, K.-W. Lih, A note on the m -bounded chromatic number of a tree. *Eur. J. Comb.* **14**, 311–312 (1993)
26. B.-L. Chen, K.-W. Lih, Equitable coloring of trees. *J. Comb. Theory Ser. B* **61**, 83–87 (1994)
27. B.-L. Chen, C.-H. Yen, Equitable Δ -coloring of graphs. *Discret. Math.* **312**, 1512–1517 (2012)
28. B.-L. Chen, K.-W. Lih, P.-L. Wu, Equitable coloring and the maximum degree. *Eur. J. Comb.* **15**, 443–447 (1994)
29. B.-L. Chen, M.-T. Ko, K.-W. Lih, Equitable and m -bounded coloring of split graphs, in *Combinatorics and Computer Science. Lecture Notes in Computer Science*, vol. 1120 (Springer, Berlin/New York, 1996), pp. 1–5
30. B.-L. Chen, K.-C. Huang, C.-H. Yen, Chromatic coloring with a maximum color class. *Discret. Math.* **308**, 5533–5537 (2008)
31. B.-L. Chen, K.-W. Lih, J.-H. Yan, Equitable colorings of interval graphs and products of graphs. arXiv:0903.1396

32. B.-L. Chen, K.-W. Lih, C.-H. Yen, Equivalence of two conjectures on equitable coloring of graphs. *J. Comb. Optim.* doi:10.1007/s10878-011-9429-8
33. T. Chunling, L. Xiaohui, Y. Yuansheng, L. Zhihe, Equitable total coloring of $C_m \square C_n$. *Discret. Appl. Math.* **157**, 596–601 (2009)
34. K. Corrádi, A. Hajnal, On the maximal number of independent circuits in a graph. *Acta Math. Acad. Sci. Hung.* **14**, 423–439 (1963)
35. B. Csaba, M. Mydlarz, Approximate multipartite version of the Hajnal-Szemerédi theorem. *J. Comb. Theory Ser. B* **102**, 395–410 (2012)
36. B. Csaba, A. Shokoufandeh, E. Szemerédi, Proof of a conjecture of Bollobás and Eldridge for graphs of maximum degree three. *Combinatorica* **23**, 35–72 (2003)
37. S.K. Das, I. Finocchi, R. Petreschi, Conflict-free star-access in parallel memory systems. *J. Parallel Distrib. Comput.* **66**, 1431–1441 (2006)
38. D. de Werra, Equitable colorations of graphs. *Rev. Française Informat. Recherche Opérationnelle* **R-3**, 3–8 (1971)
39. D. de Werra, Some uses of hypergraph in timetabling. *Asis-Pac. J. Oper. Res.* **2**, 2–12 (1985)
40. G. Dirac, Some theorems on abstract graphs. *Proc. Lond. Math. Soc.* **2**, 69–81 (1952)
41. R.G. Downey, M.R. Fellows, *Parametrized Complexity* (Springer, New York, 1999)
42. D. Duffus, B. Sands, R.E. Woodrow, On the chromatic number of the product of graphs. *J. Graph Theory* **9**, 487–495 (1985)
43. P. Erdős, Problem 9, in *Theory of Graphs and Its Applications*, ed. by M. Fieldler (Czech Academy of Science Publishing, Prague, 1964), p. 159
44. P. Erdős, C. Ko, R. Rado, Intersection theorems for systems of the finite sets. *Q. J. Math. Oxf.* (2) **12**, 31–320 (1961)
45. P. Erdős, A.L. Rubin, H. Taylor, Choosability in graphs. *Congr. Numer.* **26**, 125–157 (1980)
46. H. Fan, H.A. Kierstead, G. Liu, T. Molla, J.-L. Wu, X. Zhang, A note on relaxed equitable coloring of graphs. *Inf. Process. Lett.* **111**, 1062–1066 (2011)
47. M.R. Fellows, F.V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, S. Szeider, C. Thomassen, On the complexity of some colorful problems parameterized by treewidth. *Inf. Comput.* **209**, 143–153 (2011)
48. J. Fiala, P.A. Golovach, J. Kratochvíl, Parameterized complexity of coloring problems: treewidth versus vertex cover. *Theor. Comput. Sci.* **412**, 2513–2523 (2011)
49. R. Fidytek, H. Furmańczyk, P. Żyliński, Equitable coloring of Kneser graphs. *Discuss. Math. Graph Theory* **29**, 119–142 (2009)
50. E. Fischer, Variants of the Hajnal-Szemerédi theorem. *J. Graph Theory* **31**, 275–282 (1999)
51. J. Flum, M. Grohe, *Parametrized Complexity Theory* (Springer, New York, 2006)
52. H.-L. Fu, Some results on equalized total coloring. *Congr. Numer.* **102**, 111–119 (1994)
53. H. Furmańczyk, Equitable coloring of graphs, in *Graph Colorings*, ed. by M. Kubale. Contemporary Mathematics, vol. 352 (American Mathematical Society, Providence, 2004), pp. 35–53
54. H. Furmańczyk, Equitable coloring of graph products. *Opusc. Math.* **26**, 31–44 (2006)
55. H. Furmańczyk, M. Kubale, The complexity of equitable vertex coloring of graphs. *J. Appl. Comput. Sci.* **13**, 97–107 (2005)
56. H. Furmańczyk, K. Giaro, M. Kubale, Equitable 4-coloring of cacti and edge-cacti in polynomial time. *Int. J. Pure Appl. Math.* **27**, 377–389 (2006)
57. H. Furmańczyk, K. Kaliraj, J. Vernold Vivin, Equitable coloring on corona graphs of graphs with complete graphs, cycles and paths. *Manuscript* (2010)
58. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman and Company, San Francisco, 1979)
59. P.C. Gilmore, A.J. Hoffman, Characterization of comparability graphs and interval graphs. *Can. J. Math.* **16**, 539–548 (1964)
60. D. Gonçalves, Edge partition of planar graphs into two outerplanar graphs, in *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing* (ACM, New York, 2005), pp. 504–512

61. H. Grötzsch, Zur Theorie der diskreten Gebilde. VII. Ein Dreifarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg. Math.-Nat. Reihe* **8**, 109–120 (1958/1959)
62. R.P. Gupta, On decompositions of a multigraph into spanning subgraphs. *Bull. Am. Math. Soc.* **80**, 500–502 (1974)
63. R.K. Guy, Monthly research problems, 1969–1975. *Am. Math. Mon.* **82**, 995–1004 (1975)
64. A. Hajnal, E. Szemerédi, Proof of a conjecture of Erdős, in *Combinatorial Theory and Its Applications*, ed. by P. Erdős, A. Rényi, V. T. Sós, vol. II. *Colloquia Mathematica Societatis Janos Bolyai*, vol. 4 (North-Holland, Amsterdam, 1970), pp. 601–623
65. J. Han, Y. Zhao, On multipartite Hajnal-Szemerédi theorems. arXiv:1203.2667
66. P. Hansen, A. Hertz, J. Kuplinsky, Bounded vertex colorings of graphs. *Discret. Math.* **111**, 305–312 (1993)
67. S. Hedetniemi, Homomorphism of graphs and automata. *Univ. Michigan Technical Report 03105-44-7*, 1966
68. A.J. W. Hilton, D. de Werra, A sufficient condition for equitable edge-colourings of simple graphs. *Discret. Math.* **128**, 179–201 (1994)
69. A.J.W. Hilton, E.C. Milner, Some intersection theorems for systems of finite sets. *Q. J. Math. Oxford* (2) **18**, 369–384 (1967)
70. S. Irani, V. Leung, Scheduling with conflicts, and applications to traffic signal control, in *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, Philadelphia, 1996), pp. 85–94
71. S. Janson, T. Łuczak, A. Ruciński, *Random Graphs* (Wiley-Interscience, New York, 2000)
72. S. Janson, A. Ruciński, The infamous upper tail. *Random Struct. Algorithms* **20**, 317–342 (2002)
73. M. Jarvis, B. Zhou, Bounded vertex coloring of trees. *Discret. Math.* **232**, 145–151 (2001)
74. P.K. Jha, Hypercubes, median graphs and products of graphs: some algorithmic and combinatorial results. Ph.D. Dissertation, Iowa State Univ., 1990
75. R. Johansson, Triangle factors in a balanced blown-up triangle. *Discret. Math.* **211**, 249–254 (2000)
76. K. Kaliraj, J. Vernold Vivin, On equitable coloring of helm graph and gear graph. *Int. J. Math. Comb.* **4**, 32–37 (2010)
77. K. Kaliraj, J. Veninstine Vivik, Equitable coloring of corona graphs. *J. Comb. Math. Comb. Comput.* to appear
78. K. Kaliraj, J. Veninstine Vivik, J. Vernold Vivin, Equitable coloring on corona graph of graphs. *J. Comb. Math. Comb. Comput.* **81**, 191–197 (2012)
79. H. Kaul, A.V. Kostochka, G. Yu, On a graph packing conjecture of Bollobás, Eldridge, and Catlin. *Combinatorica* **28**, 469–485 (2008)
80. P. Keevash, R. Mycroft, A multipartite Hajnal-Szemerédi theorem. arXiv:1201.1882
81. H.A. Kierstead, A.V. Kostochka, An Ore-type theorem on equitable coloring. *J. Comb. Theory Ser. B* **98**, 226–234 (2008)
82. H.A. Kierstead, A.V. Kostochka, A short proof of the Hajnal-Szemerédi theorem on equitable colouring. *Comb. Probab. Comput.* **17**, 265–270 (2008)
83. H.A. Kierstead, A.V. Kostochka, Ore-type versions of Brooks' theorem. *J. Comb. Theory Ser. B* **99**, 298–305 (2009)
84. H.A. Kierstead, A.V. Kostochka, Equitable versus nearly equitable coloring and the Chen-Lih-Wu conjecture. *Combinatorica* **30**, 201–216 (2010)
85. H.A. Kierstead, A.V. Kostochka, Every 4-colorable graph with maximum degree 4 has an equitable 4-coloring. *J. Graph Theory* **71**, 31–48 (2012)
86. H.A. Kierstead, A.V. Kostochka, G. Yu, Extremal graph packing problems: Ore-type versus Dirac-type, in *Surveys in Combinatorics 2009*. London Mathematical Society Lecture Note Series, vol. 365 (Cambridge University Press, Cambridge, 2009), pp. 113–135
87. H.A. Kierstead, A.V. Kostochka, M. Mydlarz, E. Szemerédi, A fast algorithm for equitable coloring. *Combinatorica* **30**, 217–224 (2010)

88. J. Komolós, G. Sárkőzy, E. Szemerédi, Blow-up lemma. *Combinatorica* **17**, 109–123 (1997)
89. J. Komolós, G. Sárkőzy, E. Szemerédi, On the Pósa-Seymour conjecture. *J. Graph Theory* **29**, 167–176 (1998)
90. J. Komolós, G. Sárkőzy, E. Szemerédi, Proof of Seymour's conjecture for large graphs. *Ann. Comb.* **1**, 43–60 (1998)
91. A.V. Kostochka, Equitable colorings of outerplanar graphs. *Discret. Math.* **258**, 373–377 (2002)
92. A.V. Kostochka, K. Nakprasit, Equitable colourings of d -degenerate graphs. *Comb. Probab. Comput.* **12**, 53–60 (2003)
93. A.V. Kostochka, K. Nakprasit, On equitable Δ -coloring of graphs with low average degree. *Theor. Comput. Sci.* **349**, 82–91 (2005)
94. A.V. Kostochka, M.J. Pelsmajer, D.B. West, A list analogue of equitable coloring. *J. Graph Theory* **44**, 166–177 (2003)
95. A.V. Kostochka, K. Nakprasit, S.V. Pemmaraju, On equitable coloring of d -degenerate graphs. *SIAM J. Discret. Math.* **19**, 83–95 (2005)
96. A.V. Kostochka, L. Rabern, M. Stiebitz, Graphs with chromatic number close to maximum degree. *Discret. Math.* **312**, 1273–1281 (2012)
97. J. Krarup, D. de Werra, Chromatic optimisation: limitations, objectives, uses, references. *Eur. J. Oper. Res.* **11**, 1–19 (1982)
98. M. Krivelevich, B. Patkós, Equitable coloring of random graphs. *Random Struct. Algorithms* **35**, 83–99 (2009)
99. P.C.B. Lam, W.C. Shiu, C.S. Tong, Z.F. Zhang, On the equitable chromatic number of complete n -partite graphs. *Discret. Appl. Math.* **113**, 307–310 (2001)
100. Q. Li, Y. Bu, Equitable list coloring of planar graphs without 4- and 6-cycles. *Discret. Math.* **309**, 280–287 (2009)
101. K.-W. Lih, The equitable coloring of graphs, in *Handbook of Combinatorial Optimization*, vol. 3, ed. by D.-Z. Du, P.M. Pardos (Kluwer, Boston, 1998), pp. 543–566
102. K.-W. Lih, P.-L. Wu, On equitable coloring of bipartite graphs. *Discret. Math.* **151**, 155–160 (1996)
103. W.-H. Lin, Equitable colorings of graph products. Ph.D. Dissertation, National Taiwan University, Taiwan, 2009
104. W.-H. Lin, G.J. Chang, Equitable colorings of Kronecker products of graphs. *Discret. Appl. Math.* **158**, 1816–1826 (2010)
105. W.-H. Lin, G.J. Chang, Equitable colorings of Cartesian products of graphs. *Discret. Appl. Math.* **160**, 239–247 (2012)
106. A. Lo, K. Markström, A multipartite version of the Hajnal-Szemerédi theorem for graphs and hypergraphs. *Combin. Prob. Comput.* **22**, 97–111 (2013)
107. L. Lovász, Kneser's conjecture, chromatic number, and homotopy. *J. Comb. Theory Ser. A* **25**, 319–324 (1978)
108. R. Luo, J. S. Sereni, D.C. Stephen, G. Yu, Equitable coloring of sparse planar graphs. *SIAM J. Discret. Math.* **24**, 1572–1583 (2010)
109. Cs. Magyar, R. Martin, Tripartite version of the Corrádi-Hajnal theorem. *Discret. Math.* **254**, 289–308 (2002)
110. A.W. Marshall, I. Olkin, *Inequalities: Theory of Majorization and Its Applications* (Academic, New York, 1979)
111. R. Martin, E. Szemerédi, Quadripartite version of the Hajnal-Szemerédi theorem. *Discret. Math.* **308**, 4337–4360 (2008)
112. I. Méndez-Díaz, G. Nasini, D. Severín, A polyhedral approach for the graph equitable coloring problem. *Discret. Appl. Math.* doi:10.1016/j.dam.2012.11.018
113. I. Méndez-Díaz, G. Nasini, D. Severín, A linear integer programming approach for the equitable coloring problem. in: II Congreso de Matematica Aplicada, Computacional e Industrial, Argentina (2009)

114. I. Méndez-Díaz, G. Nasini, D. Severín, Polyhedral results for the equitable coloring problem. *Electron. Notes Discret. Math.* **37**, 159–164 (2011)
115. W. Meyer, Equitable coloring. *Am. Math. Mon.* **80**, 920–922 (1973)
116. D. Miyata, S. Tokunaga, A. Kaneko, Equitable coloring of trees. Manuscript
117. K. Nakprasit, Equitable colorings of planar graphs with maximum degree at least nine. *Discret. Math.* **312**, 1019–1024 (2012)
118. K. Nakprasit, K. Nakprasit, Equitable colorings of planar graphs without short cycles. *Theoret. Comput. Sci.* **465**, 21–27 (2012)
119. R. Niedermeier, *Invitation to Fixed Parameter Algorithms* (Oxford University Press, Oxford/New York, 2006)
120. S. Olariu, An optimal greedy heuristic to color interval graphs. *Inf. Process. Lett.* **37**, 21–25 (1991)
121. O. Ore, Note on Hamilton circuits. *Am. Math. Mon.* **67**, 55 (1960)
122. M.J. Pelsmajer, Equitable list-coloring for graphs of maximum degree 3. *J. Graph Theory* **47**, 1–8 (2004)
123. M.J. Pelsmajer, Equitable list coloring and treewidth. *J. Graph Theory* **61**, 127–139 (2009)
124. S.V. Pemmaraju, Equitable coloring extends Chernoff-Hoeffding bounds, in *Approximation, Randomization, and Combinatorial Optimization*, Berkeley, CA, 2001. Lecture Notes in Computer Science, vol. 2129 (Springer, 2001), pp. 285–296
125. S.V. Pemmaraju, Coloring outerplanar graphs equitably. Manuscript (2001)
126. S.V. Pemmaraju, K. Nakprasit, A.V. Kostochka, Equitable colorings with constant number of colors, in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, MD, 2003 (ACM, 2003), pp. 458–459
127. L. Rabern, Δ -critical graphs with small high vertex cliques. *J. Comb. Theory Ser. B* **102**, 126–130 (2012)
128. V. Rödl, A. Ruciński, Perfect matchings in ϵ -regular graphs and the blow-up lemma. *Combinatorica* **19**, 437–452 (1999)
129. D.J. Rose, On simple characterizations of k -trees. *Discret. Math.* **7**, 317–322 (1974)
130. G. Sabidussi, Graphs with given group and given graph-theoretical properties. *Can. J. Math.* **9**, 515–525 (1957)
131. N. Sauer, Hedetniemi's conjecture – a survey. *Discret. Math.* **229**, 261–292 (2001)
132. P. Seymour, Problem section, in *Combinatorics: Proceedings of the British Combinatorial Conference, 1973*, ed. by T.P. McDonough, V.C. Mavron (Cambridge University Press, Cambridge, 1974), pp. 201–202
133. A. Shioura, M. Yagiura, A fast algorithm for computing a nearly equitable edge coloring with balanced conditions. *J. Graph Algorithms Appl.* **14**, 391–407 (2010)
134. B.F. Smith, P.E. Bjorstad, W.D. Gropp, *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations* (Cambridge University Press, Cambridge/New York, 1996)
135. H. Song, J. Wu, G. Liu, The equitable edge-coloring of series-parallel graphs, in *Computational Science – ICCS 2007*, Beijing, China, 2007. Lecture Notes in Computer Science, vol. 4489 (Springer, 2007), pp. 457–460
136. E. Szemerédi, Regular partitions of graphs, in *Colloques Internationaux C.N.R.S. No. 260 Problèmes Combinatoires et Théorie des Graphes*, Orsay, 1976, pp. 399–401
137. X. Tan, Equitable Δ -coloring of planar graphs without 4-cycles, in *Proceedings of ISORA'10*, Chengdu-Jiuzhaigou, China (ORSC & APORC, 2010), pp. 400–405
138. A.C. Tucker, Perfect graphs and an application to optimizing municipal services. *SIAM Rev.* **15**, 585–590 (1973)
139. J. van Leeuwen, Graph algorithms, in *Handbook of Theoretical Computer Science, A: Algorithms and Complexity Theory* (The MIT Press, Cambridge, 1990)
140. J. Vernold Vivin, K. Kaliraj, M.M. Akbar Ali, Equitable coloring on total graph of bigraphs and central graph of cycles and paths. *Int. J. Math. Math. Sci.* doi:10.1155/2011/279246 (2011)

141. K. Vesztergombi, Some remarks on the chromatic number of the strong product of graphs. *Colloq. Math. Soc. János Bolyai* **25**, 143–149 (1997)
142. V.G. Vizing, On an estimate of the chromatic class of a p -graph. *Diskret. Analiz.* **3**, 25–30 (1964). (in Russian)
143. V.G. Vizing, Some unsolved problems in graph theory. *Uspehi Mat. Nauk* **23**, 117–134 (1968). (in Russian)
144. V.G. Vizing, Coloring the vertices of a graph in prescribed colors. *Diskret. Analiz.* **29**, 3–10 (1976). (in Russian)
145. W. Wang, Equitable total coloring of graphs with maximum degree 3. *Graphs Comb.* **18**, 677–685 (2002)
146. W. Wang, K.-W. Lih, Equitable list coloring of graphs. *Taiwan. J. Math.* **8**, 747–759 (2004)
147. H. Wang, L. Sun, Equitable total chromatic number of Mycielski graphs of two classes of graphs. *Sci. Technol. Rev.* **206**, 29–30 (2005)
148. H. Wang, J. Wei, The equitable total chromatic number of the graph $HM(W_n)$. *J. Appl. Math. Comput.* **24**, 313–323 (2007)
149. W. Wang, K. Zhang, Equitable colorings of line graphs and complete r -partite graphs. *Syst. Sci. Math. Sci.* **13**, 190–194 (2000)
150. L. Williams, J. Vandenbussche, G. Yu, Equitable defective coloring of sparse planar graphs. *Discret. Math.* **312**, 957–962 (2012)
151. M. Woźniak, Packing of graphs. *Discret. Math.* **362**, 1–78 (1997)
152. J. Wu, The equitable edge-coloring of outerplanar graphs. *J. Comb. Math. Comb. Comput.* **36**, 247–253 (2001)
153. J. Wu, P. Wang, Equitable coloring planar graphs with large girth. *Discret. Math.* **308**, 985–990 (2008)
154. X. Xie, M. Yagiura, T. Ono, T. Hirata, U. Zwick, An efficient algorithm for the nearly equitable edge coloring problem. *J. Graph Algorithms Appl.* **12**, 383–399 (2008)
155. H.P. Yap, Y. Zhang, On equitable colorings of graphs. Manuscript (1996)
156. H.P. Yap, Y. Zhang, The equitable-colouring conjecture holds for outerplanar graphs. *Bull. Inst. Math. Acad. Sin.* **25**, 143–149 (1997)
157. R. Yuster, Equitable coloring of k -uniform hypergraphs. *SIAM J. Discret. Math.* **16**, 524–532 (2003)
158. Z. Zhang, G. Liu, Equitable edge-colorings of simple graphs. *J. Graph Theory* **66**, 175–197 (2011)
159. X. Zhang, J.-L. Wu, On equitable and equitable list colorings of series-parallel graphs. *Discret. Math.* **311**, 800–803 (2011)
160. Y. Zhang, H.P. Yap, Equitable colorings of planar graphs. *J. Comb. Math. Comb. Comput.* **27**, 97–105 (1998)
161. X. Zhu, A survey on Hedetniemi's conjecture. *Taiwan. J. Math.* **2**, 1–24 (1998)
162. J. Zhu, Y. Bu, Equitable list colorings of planar graphs without short cycles. *Theor. Comput. Sci.* **407**, 21–28 (2008)
163. J. Zhu, Y. Bu, Equitable and equitable list colorings of graphs. *Theor. Comput. Sci.* **411**, 3873–3876 (2010)

Faster and Space Efficient Exact Exponential Algorithms: Combinatorial and Algebraic Approaches

Dongxiao Yu, Yuexuan Wang, Qiang-Sheng Hua and Francis C.M. Lau

Contents

1	Introduction	1250
2	Combinatorial Methods	1250
2.1	Zeta Transform and Möbius Transform	1251
2.2	Subset Convolution	1252
2.3	Variants of Subset Convolution	1257
2.4	Inclusion-Exclusion	1259
2.5	Space Efficient Exact Algorithms	1262
2.6	Faster Exact Algorithms in Bounded-Degree Graphs	1268
3	Algebraic Methods	1271
3.1	Algebraic Sieving	1271
3.2	Polynomial Circuit-Based Algorithms	1282
4	Conclusion	1286
5	Further Reading	1287
	Cross-References	1289
	Recommended Reading	1289

Abstract

Exponential algorithms, whose time complexity is $O(c^n)$ for some constant $c > 1$, are inevitable when exactly solving NP-complete problems unless $\mathbf{P} = \mathbf{NP}$. This chapter presents recently emerged combinatorial and algebraic techniques for designing exact exponential time algorithms. The discussed

D. Yu • F.C.M. Lau (✉)

Department of Computer Science, The University of Hong Kong, Hong Kong, People's Republic of China

e-mail: dxyu@cs.hku.hk; fcmlau@cs.hku.hk

Y. Wang • Q.-S. Hua

Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, People's Republic of China

e-mail: wangyuexuan@tsinghua.edu.cn; qshua@mail.tsinghua.edu.cn; qshua@cs.hku.hk

techniques can be used either to derive faster exact exponential algorithms or to significantly reduce the space requirements while without increasing the running time. For illustration, exact algorithms arising from the use of these techniques for some optimization and counting problems are given.

1 Introduction

It is a folklore that any NP-complete problem can be exactly solved in exponential time via exhaustive search. Whether there is a faster way than this kind of brute-force approach to solve any such problem is still unclear. Nonetheless, many researchers have found exact exponential time algorithms that are faster than trivial exhaustive search for quite many NP-hard optimization problems such as the traveling salesman problem (TSP) [39]. The study of exact exponential algorithms for NP-hard problems has received increasing attention since the seminal survey by Woeginger [55] in 2001. As a result, some classical techniques such as inclusion-exclusion (e.g., [19, 20, 32]) have been resurrected, and some new techniques, such as fast subset convolution [12] and the algebraic methods (e.g., [9]), have been developed. Unfortunately, for some optimization problems, although it is possible to obtain faster exponential algorithms, exponential space is required, which renders these algorithms not practical for real-life use. This chapter presents some important recently resurrected or emerged combinatorial and algebraic approaches for designing faster exact exponential time algorithms and discusses how these techniques may be used to reduce the space complexity while not sacrificing the time complexity.

Notation. Throughout this chapter, a modified big-O notation is used to hide a polylogarithmic factor: for a positive real constant d and a function τ , $O^*(\tau)$ denotes a time complexity of the form $O(\tau \log^d \tau)$.

Structure of the chapter. The main theme of this chapter is to introduce two classes of methods for designing either faster or space-efficient exact exponential time algorithms. Section 2 discusses the combinatorial methods, which covers fast subset convolution and inclusion-exclusion based algorithms. Section 3 presents the recently emerged algebraic methods, which covers the algebraic sieving method and the polynomial circuit-based algebraic method. Each of these two sections will first explain how to use these techniques to design faster exact exponential time algorithms and then discuss how they may be used to reduce the space complexity while not increasing the running time. Various examples to illustrate the techniques will also be given. Section 4 concludes the chapter and Sect. 5 highlights the related work on these two classes of methods.

2 Combinatorial Methods

There are two well-developed and commonly used combinatorial techniques – fast subset convolution and inclusion-exclusion. The fast subset convolution can compute the convolution of any two given functions over some subset lattice in

$O^*(2^n)$ time, whereas a direct evaluation needs $\Omega(3^n)$ time. This fast algorithm is based on fast algorithms for the zeta transform and other related transforms which are actually accelerated dynamic programming algorithms. Thus, it needs exponential space since dynamic programming approaches have to store all the useful auxiliary information. Surprisingly, for certain special type of input functions called singletons, this exponential space could be circumvented. Furthermore, several variants of subset convolution, such as the covering product and the packing product [12], also play an important role in the design of faster exponential time exact algorithms.

The inclusion-exclusion principle is a classical sieving method: to determine the size of a set, the set is first overcounted, then subtract from this count, add again, subtract again, until finally arriving at the exact number of elements. This allows to count combinatorial objects indirectly, which is better than direct counting which could be inefficient or even impossible. Furthermore, although it needs to go through all subsets, the inclusion-exclusion technique is powerful in providing polynomial space exact algorithms. Combining it with some dynamic programming-based algorithms, such as the fast zeta transform, gives more surprising results. In fact, the inclusion-exclusion technique solves optimization problems via solving the corresponding counting problems, and hence, it is an important alternative for designing exact algorithms for counting problems.

Before the detailed descriptions of these two techniques, some essential tools needs to be introduced – the fast zeta transform and the fast Möbius transform.

2.1 Zeta Transform and Möbius Transform

Denote by R an arbitrary (possibly noncommutative) ring and by N a set of n elements, $n \geq 0$. Let f be a function that associates with every subset $S \subseteq N$ an element $f(S)$ of the ring R .

The zeta transform of f is the function $f\zeta$ that associates every $S \subseteq N$ with an element in R :

$$f\zeta(S) = \sum_{X \subseteq S} f(X). \quad (1)$$

The Möbius transform $f\mu$ of f is defined as

$$f\mu(S) = \sum_{X \subseteq S} (-1)^{|S \setminus X|} f(X). \quad (2)$$

Given the zeta transform $f\zeta$, the original function f can be obtained by the following formula which is often called Möbius inversion:

$$f(X) = \sum_{S \subseteq X} (-1)^{|X \setminus S|} f\zeta(S). \quad (3)$$

For more details of these two transforms, please refer to [27].

2.1.1 Fast Zeta Transform and Fast Möbius Transform

The straightforward way to compute the zeta transform evaluates $f\zeta(S)$ at every $S \subseteq N$, using $O(3^n)$ ring additions in total. This can be improved to use only $O(n2^n)$ ring operations by applying the Yates's method [37, 58], and the resulting algorithm is commonly called the fast zeta transform. Without loss of generality, assume that $N = \{1, 2, \dots, n\}$. To compute $f\zeta$, let initially $f\zeta_0(S) = f(S)$ for every $S \subseteq N$. Then iterate for $j = 1, 2, \dots, n$ and $S \subseteq N$ with the following recurrence:

$$f\zeta_j(S) = \begin{cases} f\zeta_{j-1}(S), & \text{if } j \notin X \\ f\zeta_{j-1}(S \setminus \{j\}) + f\zeta_{j-1}(S), & \text{otherwise.} \end{cases} \quad (4)$$

It can be verified by induction on j that the above recurrence gives $f\zeta_n(S) = f\zeta(S)$ for any $S \subseteq N$ in $O(n2^n)$ ring operations. The Möbius transform can be computed in a similar manner. Initially, let $f\mu_0(S) = f(S)$ for every $S \subseteq N$. Then iterate for $j = 1, 2, \dots, n$ and $S \subseteq N$ as follows:

$$f\mu_j(S) = \begin{cases} f\mu_{j-1}(S), & \text{if } j \notin X \\ -f\mu_{j-1}(S \setminus \{j\}) + f\mu_{j-1}(S), & \text{otherwise.} \end{cases} \quad (5)$$

Similarly, it can be proved that $f\mu_n(S) = f\mu(S)$ for any $S \subseteq N$.

Theorem 1 *Let N be a set of n elements. Then the zeta transform and the Möbius transform on the subset lattice of N can be computed in $O(n2^n)$ ring operations.*

2.2 Subset Convolution

Definition 1 (Subset convolution) Given a universe set N of n elements, f and g are functions defined on subsets of N , which associate every $S \subseteq N$ with an element of a ring R , respectively. The convolution $f * g$ for all $S \subseteq N$ is defined as follows:

$$f * g(S) = \sum_{X \subseteq S} f(X)g(S \setminus X). \quad (6)$$

The subset convolution can yield solutions to many hard computational problems. In particular, by embedding the integer max-sum or min-sum semiring into the sum-product ring, the technique can be used to implement many dynamic programming-based algorithms.

2.2.1 Fast Subset Convolution

In principle, the fast subset convolution consists of several efficient dynamic programming instances. In the fast subset convolution, the evaluation of (6) can

be achieved via the product of “ranked” extensions of the classical zeta transforms of f and g on the subset lattice, followed by a “ranked” Möbius transform. The fast subset convolution is summarized in [Algorithm 1](#).

For every $k = 0, 1, \dots, n$ and $S \subseteq N$, the ranked zeta transform of f is defined as

$$f\zeta(k, S) = \sum_{X \subseteq S, |X|=k} f(X). \quad (7)$$

Based on the above defined ranked zeta transform, the inversion can be achieved by

$$f(X) = \sum_{S \subseteq X} (-1)^{|S \setminus X|} f\zeta(|X|, S). \quad (8)$$

Although the above formula seems redundant, it will provide a key for fast evaluation of the subset convolution. Note that the ranked zeta transform can be computed in $O(n^2 2^n)$ ring operations by carrying out the fast zeta transform (4) independently for every $k = 0, 1, \dots, n$. Similarly, the ranked inversion (8) can be computed in $O(n^2 2^n)$ ring operations by carrying out the fast Möbius transform (5) independently for each $k = 0, 1, \dots, n$.

For the two ranked zeta transforms, $f\zeta$ and $g\zeta$, define a new convolution $f\zeta \otimes g\zeta$ for all $k = 0, 1, \dots, n$ and $S \subseteq N$ by

$$f\zeta \otimes g\zeta(k, S) = \sum_{j=0}^k f\zeta(j, S)g\zeta(k-j, S). \quad (9)$$

Lemma 1 $f * g(S) = \sum_{X \subseteq S} (-1)^{|S \setminus X|} f\zeta \otimes g\zeta(|S|, S)$

Proof

$$\begin{aligned} \sum_{X \subseteq S} (-1)^{|S \setminus X|} f\zeta \otimes g\zeta(|S|, S) &= \sum_{X \subseteq S} (-1)^{|S \setminus X|} \sum_{j=0}^{|S|} f\zeta(j, S)g\zeta(|S| - j, S) \\ &= \sum_{X \subseteq S} (-1)^{|S \setminus X|} \sum_{j=0}^{|S|} \sum_{\substack{U, V \subseteq X \\ |U|=j, |V|=|S|-j}} f(U)g(V). \end{aligned} \quad (10)$$

Because X ranges over all subsets of S , for any ordered pair (U, V) of subsets of S satisfying the condition that $|U| + |V| = |S|$, the term $f(U)g(V)$ occurs exactly once in the sum with sign $(-1)^{|S \setminus X|}$ for subsets X of S iff $U \cup V \subseteq X$. Then putting the terms associated with each pair (U, V) together, by the Binomial Theorem, the coefficient of $f(U)g(V)$ is

$$\sum_{x=|U \cup V|}^{|S|} \binom{|S| - |U \cup V|}{x - |U \cup V|} (-1)^{|S|-x} = \begin{cases} 1, & \text{if } |U \cup V| = |S| \\ 0, & \text{otherwise.} \end{cases} \quad (11)$$

The conditions that $|U| + |V| = |S|$ and $|U \cup V| = |S|$ imply that $U \cup V = S$ and $U \cap V = \emptyset$; and it follows that

$$\begin{aligned} \sum_{X \subseteq S} (-1)^{|S \setminus X|} f\zeta \circledast g\zeta(|S|, S) &= \sum_{U \cup V = S, U \cap V = \emptyset} f(U)g(V) \\ &= f * g(S). \end{aligned} \quad (12)$$

□

Theorem 2 *The subset convolution over an arbitrary ring can be evaluated in $O(n^2 2^n)$ ring operations using the fast subset convolution as shown in [Algorithm 1](#).*

Proof The correctness is established by [Lemma 1](#). In [Algorithm 1](#), the time is mainly consumed in lines 2, 6, and 9. By [Theorem 1](#), the time consumed in lines 2 and 9 is $O(n^2 2^n)$, and the time needed in line 6 is $O(n^2 2^n)$ according to Formula (9). Thus, the time complexity of [Algorithm 1](#) is $O(n^2 2^n)$. □

The following theorem is implied by [Theorem 2](#).

Theorem 3 *The subset convolution over the integer sum-product ring can be computed in $O^*(2^n \log M)$ time, provided that the range of the input functions is $\{-M, -M + 1, \dots, M\}$.*

Proof By [Theorem 2](#), the subset convolution can be computed in $O(n^2 2^n)$ ring operations. Thus, it suffices to note that any intermediate results, for which ring operations are performed, are $O(n \log M)$ -bit integers. This is the case because the ranked zeta transform of an input function can be computed with integers between $-M 2^n$ and $M 2^n$, and it follows that the convolution of ranked transforms

Algorithm 1 Fast subset convolution

Input: A universe set N of n elements and functions f, g defined on 2^U

Output: The subset convolution $f * g$ for every set $S \subseteq N$

- 1: **For** $i = 0$ to n **do**
 - 2: Compute the ranked zeta transforms $f\zeta(k, X), g\zeta(k, X)$ for every $X \subseteq N$ using the fast zeta transform.
 - 3: **End For**
 - 4: **For** $k = 0$ to n **do**
 - 5: **For** every $X \subseteq N$ **do**
 - 6: Compute the convolution $f\zeta \circledast g\zeta(k, X)$
 - 7: **End For**
 - 8: **End For**
 - 9: Compute the subset convolution $f * g$ for every set $S \subseteq N$ based on Formula (12) using the fast Möbius transform.
-

can be computed with $O(n \log M)$ -bit integers. Furthermore, the ranked Möbius transform is computed by adding and subtracting $O(n \log M)$ -bit integers for $O(2^n)$ times. \square

An obvious disadvantage of the fast subset convolution algorithm is its inapplicability to semirings where additive inverses need not exist. For some special semirings usually appearing in combinatorial optimization problems, for example, the integer max-sum and integer min-sum semirings, one can embed these semirings into the integer sum-product ring.

Theorem 4 *The subset convolution over the integer max-sum (min-sum) semiring can be computed in $O^*(2^n M)$ time, provided that the range of the input functions is $\{-M, -M + 1, \dots, M\}$.*

Proof Here, we only provide the proof for the max-sum semiring, as that for the min-sum semiring is similar. Without loss of generality, assume that the range of the input functions is $\{0, 1, \dots, M\}$; otherwise, the correct output can be obtained by first adding M to each value of both input functions, then computing the convolution, and finally subtracting $2M$ from the computed values.

Let f, g be the two input functions. Let $\beta = 2^n + 1$ and $M' = \beta^M$. Define new functions $f' = \beta^f$ and $g' = \beta^g$ which map the subsets of N to $\{0, 1, \dots, M'\}$. By Theorem 3, the subset convolution $f' * g'$ over the integer sum-product ring can be computed in $O^*(2^n \log M') = O^*(2^n M)$ time. It remains to show that for all $S \subseteq N$, the value for $\max_{T \subseteq S} \{f(T) + g(S \setminus T)\}$ can be efficiently deduced given the value of $f' * g'(S)$. Observe that $f' * g'(S)$ can be expressed as

$$\begin{aligned} f' * g'(S) &= \sum_{T \subseteq S} \beta^{f(T)+g(S \setminus T)} \\ &= \alpha_0(S) + \alpha_1(S)\beta + \cdots + \alpha_{2M}(S)\beta^{2M}. \end{aligned} \tag{13}$$

Due to the choice of β , each coefficient $\alpha_r(S)$ is uniquely determined and equals the number of subsets T of S for which $f(T) + g(S \setminus T) = r$. Thus, for each $S \subseteq N$, the largest r for which $\alpha_r(S) > 0$ corresponds to the value of $f * g(S)$. Clearly, this can be found in $O(M)$ time. \square

Example 1 (Partitioning Problem) The generic problem of partitioning is to divide an n -element set N into k disjoint subsets such that each of which satisfies some desired property specified by an indicator function f on the subsets of N . Given N , k , and f as input, the task is to decide whether there exists a partition $\{S_1, S_2, \dots, S_k\}$ of N such that $f(S_c) = 1$ for each $c = 1, 2, \dots, k$. Many classical graph partitioning problems are of this form. For example, in graph coloring, $f(S) = 1$ iff S is an independent set in the input graph with the vertices N . Likewise, in domatic partitioning, f is the indicator of dominating sets. Note that the number of valid partitions of N is given by $f^{*k}(N)$, where f^{*k} is defined as

$$f^{*k} = \underbrace{f * \dots * f}_{k \text{ times}}.$$

Thus, the valid partitions can be counted by $O(\log k)$ subset convolutions using the doubling trick – computing only convolutions f^{*2^i} . \square

Example 2 (Steiner Tree Problem) Given an undirected graph $G = (V, E)$, a weight $w(e) > 0$ for each edge $e \in E$, and a set of vertices $K \subseteq V$, the Steiner tree problem is to find a subgraph H of G that connects the vertices in K and has the minimum total weight $\sum_{e \in E(H)} w(e)$ among all such subgraphs of G . Here, we only consider the case where the weight of each edge is bounded by a constant. Obviously, H is necessarily a tree. A Steiner tree always refers to such an optimal subgraph. The fast subset convolution can be used to accelerate the famous Dreyfus-Wagner algorithm [25].

For a vertex subset $Y \subseteq V$, denote the total weight of a Steiner tree connecting Y in G by $W(Y)$. The Dreyfus-Wagner algorithm is based on the following dynamic programming called Dreyfus-Wagner recurrence: for $|Y| \geq 3$, and all $q \in Y$, $X = Y \setminus \{q\}$.

$$W(\{q\} \cup X) = \min\{W(\{p\}) + g_p(X) : p \in V\}, \quad (14)$$

$$g_p(X) = \min\{W(\{p\} \cup D) + W(\{p\} \cup (X \setminus D)) : \emptyset \subset D \subset X\}. \quad (15)$$

The Steiner tree problem can be solved by computing the weight $W(K)$ via the above recursion. For the base case, observe that for $|Y| = 1$ the weight $W(Y) = 0$ and for $|Y| = 2$ the weight $W(Y)$ can be determined by a shortest-path computation based on the edge weight $w(e)$, for example, Johnson's algorithm [35]. A bottom-up evaluation of $W(K)$ takes $O^*(3^k n + 2^k n^2 + nm)$ time, where $n = |V|$, $m = |E|$ and $k = |K|$. Once the values $W(\{p\} \cup Y)$ and $g_p(Y)$ for all $Y \subset K$ and $p \in V$ are computed and stored, an actual Steiner tree is easy to construct by tracing backward a path of optimal choices in (14) and (15).

The fast subset convolution over the min-sum semiring can expedite the evaluation of the Dreyfus-Wagner recursion in (15). Here the fast subset convolution needs to be implemented in a level-wise manner. For each level $l = 2, 3, \dots, k-1$, assume the value $W(\{q\} \cup X)$ has been computed and stored for all $X \subset K$ and $q \in V \setminus X$. Define the function f_p for all $X \subseteq K$ and $p \in V$ as

$$f_p(X) = \begin{cases} W(\{p\} \cup X), & \text{if } 1 \leq |X| \leq l-1 \\ \infty, & \text{otherwise.} \end{cases} \quad (16)$$

Obviously, $g_p(X) = f_p * f_p(X)$ for all $X \subseteq K$ and $|X| = l$. Then applying the fast subset convolution over the min-sum semiring, by Theorem 4, the Steiner tree problem can be solved in $O^*(2^k n^2 + nm)$ time. \square

2.3 Variants of Subset Convolution

2.3.1 Covering Product and Intersecting Covering Product

Definition 2 Given two functions f, g defined on 2^N which associate each subset of N to an element of a ring R , the covering product $f *_c g(S)$ for each $S \subseteq N$ is defined as

$$f *_c g(S) = \sum_{U, V \subseteq S, U \cup V = S} f(U)g(V). \quad (17)$$

Theorem 5 *The covering product over an arbitrary ring can be evaluated in $O(n^2 2^n)$ ring operations.*

Proof The proof is mainly based on the following Lemma.

Lemma 2 *For any $S \subseteq N$, the following holds:*

$$(f *_c g)\zeta(S) = f\zeta(S) \cdot g\zeta(S). \quad (18)$$

Proof

$$\begin{aligned} (f *_c g)\zeta(S) &= \sum_{X \subseteq S} (f *_c g)(S) \\ &= \sum_{X \subseteq S} \sum_{U \cup V = X} f(U)g(V) \\ &= \sum_{U \subseteq S} f(U) \sum_{V \subseteq U} g(V) \\ &= f\zeta(S) \cdot g\zeta(S). \end{aligned} \quad (19)$$

The third equality comes from the fact that for each $U, V \subseteq X$, there exists exactly one $X \subseteq S$ such that $U \cup V = X$. \square

By Lemma 2, $f *_c g$ can be obtained by computing the Möbius transform of $f\zeta(S) \cdot g\zeta(S)$. Then an $O^*(2^n)$ time algorithm for the covering product is as follows: first, compute $f\zeta$ and $g\zeta$ using the fast zeta transform (4), then multiply $f\zeta$ by $g\zeta$ for all $S \subseteq N$, and finally, compute $f *_c g$ using the fast Möbius transform (5). \square

Definition 3 Given two functions f, g defined on 2^N which associate each subset of N to an element of a ring R , the intersecting covering product $f *_c g(S)$ for each $S \subseteq N$ is defined as

$$f *_c g(S) = \sum_{\substack{U, V \subseteq S \\ U \cup V = S, U \cap V \neq \emptyset}} f(U)g(V). \quad (20)$$

An $O^*(2^n)$ time algorithm can be immediately derived from the fact that $f *_{ic} g = f *_c g - f * g$.

Theorem 6 *The intersecting covering product over an arbitrary ring can be evaluated in $O(n^2 2^n)$ ring operations.*

Furthermore, more precise control over the allowed intersection cardinalities $|U \cap V| = l$ besides the $l = 0$ ($f * g$) and $l > 0$ ($f *_{ic} g$) cases can be obtained by modifying (9).

Example 3 (Minimum Connected Spanning SubHypergraph (MCSH)) A hypergraph is a pair $H = (V, \mathcal{E})$, where V is a finite set and \mathcal{E} is a set consisting of subsets of V . A hypergraph $J = (W, F)$ is a subhypergraph of H if $W \subseteq V$ and $F \subseteq \mathcal{E}$. A subhypergraph is spanning if $V = W$. A path in a hypergraph H is a sequence $(x_1, E_1, x_2, E_2, \dots, E_l, x_{l+1})$ such that (a) $x_1, x_2, \dots, x_{l+1} \in V$ are all distinct, (b) $E_1, E_2, \dots, E_l \in \mathcal{E}$ are all distinct, and (c) $x_i, x_{i+1} \in E_i$ for all $i = 1, 2, \dots, l$. Such a path joins x_1 to x_{l+1} . A hypergraph is connected if for all distinct $x, y \in V$ there exists a path joining x to y .

Given a hypergraph $H = (V, \mathcal{E})$ and a weight $w(E) > 0$ for each hyperedge $E \in \mathcal{E}$, the minimum connected spanning subhypergraph problem is to find a connected spanning subhypergraph of H that has the minimum total weight or to assert that none exists. Here, the weight of every edge is assumed to be bounded by a constant. The MCSH problem can be solved in $O^*(2^n)$ time using the intersecting covering product over the min-sum semiring, where $n = |V|$.

First, define the k -th power of the intersecting covering product for all $k = 2, 3, \dots$ by

$$f *_{ic}^k = f *_{ic} (f *_{ic}^{k-1}). \quad (21)$$

Here, the order in which the products are evaluated matters because the intersecting covering product is not associative. Define the function f for all $E \subseteq V$ by

$$f(E) = \begin{cases} w(E), & \text{if } E \in \mathcal{E} \\ \infty, & \text{otherwise.} \end{cases} \quad (22)$$

Now, observe that (a) an MCSH can be constructed by augmenting a connected subhypergraph of H one hyperedge at a time, and (b) at most $n-1$ hyperedges occur in an MCSH of H . Thus, $f *_{ic}^k(V) < \infty$ is the minimum weight of a connected spanning subhypergraph of H consisting of k hyperedges. By storing the function values $f *_{ic}^k$ for each $k = 1, 2, \dots, n-1$, the actual MCSH can be determined by tracing back the computation one edge at a time.

2.3.2 Other Variants

Another important variant of the subset convolution is the packing product which is defined for all $S \subseteq N$ as

$$f *_p g(S) = \sum_{U,V \subseteq S, U \cap V = \emptyset} f(U)g(V). \quad (23)$$

Given f and g , the packing product can be evaluated in $O(n^2 2^n)$ ring operations by first computing the subset convolution $f * g$ and then convolving the result with vector $\vec{1}$ with all entries being equal to 1 based on the following fact:

$$f *_p g = f * g * \vec{1}. \quad (24)$$

Some further variations are possible by restricting the domain, for example, to any hereditary family of subsets of N . For more details, please refer to [12].

2.4 Inclusion-Exclusion

The inclusion-exclusion technique is based on a fundamental counting principle – the inclusion-exclusion principle. Clearly, it is a natural approach for solving counting problems. Here, we mainly focus on demonstrating its power in solving decision versions of optimization problems when direct computation is inefficient or even impossible. Another important feature of the inclusion-exclusion technique is that it does not need exponential space in principle. Hence, this technique is also a good choice for deriving polynomial space algorithms.

Lemma 3 (Inclusion-exclusion principle) *Let B be a finite set with subsets $A_1, \dots, A_n \subseteq B$. With the convention that $\cap_{i \in \emptyset} A_i = B$, the following holds:*

1. *The number of elements of B which lie in none of the A_i is*

$$|\bigcap_{i=1}^n \overline{A_i}| = \sum_{X \subseteq \{1, \dots, n\}} (-1)^{|X|} |\bigcap_{i \in X} A_i|. \quad (25)$$

2. *Let $w : B \rightarrow R$ be a real-valued weight function extended to the domain 2^B by setting $w(A) = \sum_{e \in A} w(e)$. Then*

$$w(|\bigcap_{i=1}^n \overline{A_i}|) = \sum_{X \subseteq \{1, \dots, n\}} (-1)^{|X|} w(|\bigcap_{i \in X} A_i|). \quad (26)$$

Proof The Lemma is proved by analyzing the contribution of every element $e \in B$ to both sides of the expression. If e is not contained by any A_i , it contributes $w(e)$ to the left hand side. To the right hand side it contributes $w(e)$ exactly once, namely, in the term corresponding to $X = \emptyset$. Conversely, assume that e lies in A_i for all $i \in I \neq \emptyset$. Its contribution to the left hand side is 0. On the right hand side, since e lies in the intersection $\cap_{i \in X} A_i$ for every $X \subseteq I$, by the Binomial Theorem, its total contribution is

$$\sum_{X \subseteq I} (-1)^{|X|} w(e) = w(e) \sum_{i=0}^{|I|} (-1)^i \binom{|I|}{i} = 0. \quad (27)$$

□

Example 4 (Set Partition) Given a set system (N, \mathcal{F}) and an integer k , where $\mathcal{F} \subseteq 2^N$, a k -partition of the given set system is a tuple (S_1, \dots, S_k) over \mathcal{F} such that $\cup_{i=1}^k S_i = N$ and $S_i \cap S_j = \emptyset$ for any $1 \leq i \neq j \leq k$. Denote $P_k(\mathcal{F})$ as the number of such k -partitions. The set partition problem is to determine the minimum k such that there is a k -partition of (N, \mathcal{F}) , that is, the minimum k such that $P_k(\mathcal{F}) > 0$. Clearly, this problem can be solved by computing $P_k(\mathcal{F})$ for at most n different k values. The following lemma shows how to compute $P_k(\mathcal{F})$ by making use of the inclusion-exclusion principle.

Lemma 4 Let $a_k(X)$ denote the number of k -tuples (S_1, \dots, S_k) over \mathcal{F} for which $S_i \cap X = \emptyset$ ($1 \leq i \leq k$) and $\sum_{i=1}^k |S_i| = n$. Then

$$p_k(\mathcal{F}) = \sum_{X \subseteq N} (-1)^{|X|} a_k(X). \quad (28)$$

Proof This is a direct application of Lemma 3, where B is the set of k -tuples (S_1, \dots, S_k) over \mathcal{F} satisfying $\sum_{i=1}^k |S_i| = n$, and $A_i \subseteq B$ are those k -tuples that avoid $\{i\}$. Then $p_k(\mathcal{F})$ is exactly the number of k -tuples that lie in none of the A_i , and $|\cap_{i \in X} A_i| = a_k(X)$. □

Write $f\zeta^{(l)}(Y)$ for the number of sets $S \in \mathcal{F}$ with $|S| = l$ and $S \subseteq Y$, and recall that it is the zeta transform of the indicator function

$$f^{(l)}(S) = \begin{cases} 1, & \text{if } S \in \mathcal{F}, |S| = l \\ 0, & \text{otherwise.} \end{cases} \quad (29)$$

Thus, the fast zeta transform (4) computes a table containing $f\zeta^{(l)}(Y)$ for all l and Y in $O^*(2^n)$ time.

Once these values have been computed, $a_k(X)$ can be obtained for any fixed $X \subseteq N$ by dynamic programming in time polynomial in k and n as follows. Define $g(j, m)$ to be the number of j -tuples (S_1, \dots, S_j) for which $S_c \cap X = \emptyset$ ($1 \leq c \leq j$) and $\sum_{c=1}^j |S_j| = m$, formally

$$g(j, m) = \sum_{l_1 + \dots + l_j = m} \prod_{c=1}^j f\zeta^{(l_c)}(N \setminus X). \quad (30)$$

Then $a_k(X) = g(k, n)$, and it can be computed by the following recursion:

$$g(j, m) = \sum_{l=0}^m g(j-1, m-l) f\zeta^{(l)}(N \setminus X), \quad (31)$$

observing that $g(1, m) = f\zeta^{(m)}(N \setminus X)$. Finally, summing up $a_k(X)$ according to (28) gives an $O^*(2^n)$ time algorithm for computing $P_k(\mathcal{F})$. \square

Example 5 (Cover Polynomial) Denote $x^i = \frac{x!}{(x-i)!}$. A Hamiltonian path of a graph is a walk that contains all the nodes exactly once. If the walk is cyclic, it is called a Hamiltonian cycle. The cover polynomial of a directed graph $D = (V, A)$ is defined as (cf. [13, 22])

$$\sum_{i,j} C_V(i, j) x^i y^j, \quad (32)$$

where $C_V(i, j)$ can be interpreted as the number of ways to partition V into i -directed paths and j -directed cycles of D . The so-called computing cover polynomial is to compute the coefficient $C_V(i, j)$.

For $X \subseteq V$, denote by $h(X)$ the number of Hamiltonian paths in $D[X]$, and denote by $c(X)$ the number of Hamiltonian cycles in $D[X]$. Define $h(\emptyset) = c(\emptyset) = 0$. Note that for all $x \in V$, $h(\{x\}) = 1$ and $c(\{x\})$ is the number of loops incident on x . Then $C_V(i, j)$ can be expressed as:

$$C_V(i, j) = \frac{1}{i!j!} \sum_{X_1, \dots, X_i, Y_1, \dots, Y_j} h(X_1) \cdots h(X_i) c(Y_1) \cdots c(Y_j), \quad (33)$$

where the sum is over all $(i + j)$ -tuples $(X_1, \dots, X_i, Y_1, \dots, Y_j)$ such that it is a partition of V .

The expression for $C_V(i, j)$ can be reformulated using the principle of inclusion-exclusion. Let z be a polynomial indeterminate. Define for every $U \subseteq V$ the polynomials

$$H_U(z) = \sum_{X \subseteq U} h(X) z^{|X|}, \quad C_U(z) = \sum_{Y \subseteq U} c(Y) z^{|Y|}. \quad (34)$$

Viewed as set functions, $H_U(z)$ and $C_U(z)$ are zeta transforms of the set functions $h(X)z^{|X|}$ and $c(Y)z^{|Y|}$, respectively. By inclusion-exclusion,

$$C_V(i, j) = \frac{1}{i!j!} \sum_{U \subseteq V} (-1)^{|V \setminus U|} \{z^n\} H_U(z)^i C_U(z)^j. \quad (35)$$

Based on the above formula, an $O^*(2^n)$ time algorithm can be obtained. For $S \subseteq V$, let $w_S(s, t, l)$ denote the number of directed walks of length l from vertex s to vertex t in $D[S]$. Define $w_S(s, t, l) = 0$ if $s \notin S$ or $t \notin S$. By inclusion-exclusion again,

$$\begin{aligned} h(X) &= \sum_{s,t \in V} \sum_{S \subseteq X} (-1)^{|X \setminus S|} w_S(s, t, |X| - 1), \\ c(Y) &= \frac{1}{|Y|} \sum_{s \in V} \sum_{S \subseteq Y} (-1)^{|Y \setminus S|} w_S(s, s, |X|). \end{aligned} \quad (36)$$

Observing $w_S(s, t, l)$ can be computed in polynomial time, the fast Möbius transform (5) can compute $h(X)$ and $c(Y)$ for every $X, Y \subseteq V$ in $O^*(2^n)$ time. Then H and C can be evaluated using the fast zeta transform according to (34). Finally, given H and C , (35) can be evaluated in $O^*(2^n)$ time. \square

2.5 Space Efficient Exact Algorithms

2.5.1 Polynomial Space Algorithm Based on Subset Convolution

For some special input functions whose zeta transforms can be determined easily, polynomial space algorithms for some combinatorial optimization problems can be derived via the subset convolution and the covering product. A particular type of input functions called singletons is used here to demonstrate how to efficiently compute the subset convolution and the covering product values for the element set N using only polynomial space. A function $f : U \rightarrow R$ is called a singleton if there exists an element $e \in U$ such that $f(x) = 0$ unless $x = e$.

Theorem 7 *Given two functions f, g defined on 2^N which associate each subset of N to an element of a ring R , if f, g , then (1) $f *_c g(N)$ can be computed in $O^*(2^n)$ time and polynomial space, and (2) $f * g(N)$ can be computed in $O^*(2^n)$ time and polynomial space.*

Proof (1) Assume that $f(X_f) = e_f$ and $g(X_g) = e_g$, where $X_f, X_g \subseteq N$ and $e_f, e_g \in R$. Since, f, g are singletons, $f(X) = 0$ for any $X \subseteq N$ other than X_f . The same result is also true for g and any subsets other than X_g . The following Lemma is proved in Sect. 2.3.1.

Lemma 5 *For any $S \subseteq N$, the following holds:*

$$(f *_c g)\zeta(S) = f\zeta(S) \cdot g\zeta(S). \quad (37)$$

Note that for singletons f, g , the zeta transforms for every $Y \subseteq N$ can be easily given as

$$f\zeta(Y) = \begin{cases} e_f, & \text{if } X_f \subseteq Y \\ 0, & \text{if otherwise.} \end{cases} \quad (38)$$

$$g\zeta(Y) = \begin{cases} e_g, & \text{if } X_g \subseteq Y \\ 0, & \text{if otherwise.} \end{cases} \quad (39)$$

Thus, the zeta transform of $f *_c g$ for every subset $Y \subseteq N$ can be computed in polynomial time after getting the zeta transforms $f\zeta$ and $g\zeta$. Then $f *_c g(N)$ can be obtained in $O^*(2^n)$ time by computing the Möbius inversion (3). The space used is polynomial since it is not necessary to store $(f *_c g)\zeta$ values for every $Y \subseteq N$.

(2) Denote $h = f * g$. Define a relaxation of a function $f : 2^N \rightarrow R$ as a sequence of functions $f_i : 2^N \rightarrow R$, for $0 \leq i \leq |N|$, such that for every $0 \leq i \leq |N|$, $Y \subseteq N$:

$$f_i(Y) = \begin{cases} f(Y), & \text{if } i = |Y| \\ 0, & \text{if } i < |Y|. \end{cases} \quad (40)$$

Observe that a function f can have many different relaxations, since there are no restrictions on what $f_i(X)$ should be when $i > |X|$. The basic idea of the proof is that for the input functions f and g , there exists a relaxation $\{h_i\}$ of h such that the zeta transform of $h_{|N|}$ can be computed efficiently. Since $h(N) = h_{|N|}(N)$, $h(N)$ can be obtained by computing the Möbius inversion (3) of $h\zeta_{|N|}$.

For input functions f and g , define $f_i = f$ and $g_i = g$ for all i . It is easy to verify that $\{f_i\}$ and $\{g_i\}$ are relaxations of f and g , respectively. Then define for all i , $h_i = \sum_{j=0}^i f_j *_c g_{i-j}$.

Lemma 6 *As defined above, $\{h_i\}$ is a relaxation of h .*

Proof By the definition of the covering product, for every $X \subseteq N$

$$h_i(X) = \sum_{j=0}^i \sum_{Y \cup Z=X} f_j(Y)g_{i-j}(Z). \quad (41)$$

Consider an arbitrary summand $f_j(Y)g_{i-j}(Z)$. There are two cases:

Case 1 ($|X| > i$) Since $Y \cup Z = X$, $|Y| + |Z| \geq |X| > i$. It concludes that either $|Y| > j$ and $f_j(Y) = 0$ or $|Z| > i - j$ and $g_{i-j}(Z) = 0$, because $\{f_i\}$ and $\{g_i\}$ both are relaxations of f and g respectively. Thus, $f_j(Y)g_{i-j}(Z) = 0$. Since $f_j(Y)g_{i-j}(Z)$ is an arbitrary term in the summation for $h_i(X)$, it follows that $h_i(X) = 0$.

Case 2 ($|X| = i$) If $|Y| + |Z| > i$, either $|Y| > j$ or $|Z| > i - j$ and $f_j(Y)g_{i-j}(Z) = 0$ which is analogous to the first case. If $|Y| + |Z| = i$, then Y and Z are disjoint. Since $\{f_i\}$ and $\{g_{i-j}\}$ are relaxations of f and g respectively, only $f_{|Y|}(Y)g_{|Z|}(Z)$ will contribute to the sum. Thus,

$$\begin{aligned} h_i(X) &= \sum_{j=0}^i \sum_{Y \cup Z=X} f_j(Y)g_{i-j}(Z) \\ &= \sum_{Y \cup Z=X, Y \cap Z=\emptyset} f_{|Y|}(Y)g_{|Z|}(Z) \end{aligned} \quad (42)$$

$$\begin{aligned}
&= \sum_{Y \cup Z = X, Y \cap Z = \emptyset} f(Y)g(Z) \\
&= h(X).
\end{aligned}$$

□

By [Lemma 5](#) and the definition of h_i in (41), it follows that $h\xi_i = \sum_{j=0}^i f\xi_j g\xi_{i-j}$. Thus, the zeta transform of $h_{|N|}(X)$ for every $X \subseteq N$ can be computed in polynomial time given the zeta transforms for $\{f_i\}$ and $\{g_i\}$. From (1), the zeta transforms of $\{f_i\}$ and $\{g_i\}$ can be obtained efficiently. Similar to what is done in (1), $h(N) = h_{|N|}(N)$ can be obtained in $O^*(2^n)$ time and polynomial space by computing the Möbius inversion

$$h_{|N|}(N) = \sum_{X \subseteq N} (-1)^{|N \setminus X|} h\xi_{|N|}(X). \quad (43)$$

□

Example 6 (Cover Polynomial Revisited) Define

$$C_Y(i, j) = \frac{1}{i!j!} \sum_{l_1 + \dots + l_{i+j} = n-i} h_{l_1} *_c \dots *_{l_i} h_{l_i} *_c c_{l_{i+1}} \dots c_{l_{i+j}}(Y), \quad (44)$$

where $h_l(Y)$ and $c_l(Y)$ are the number of Hamiltonian paths and Hamiltonian cycles of length l in $D[Y]$, respectively (here, the parameter l is introduced in order to obtain efficient computable zeta transforms). Note that paths and cycles with l edges contain $l+1$ and l nodes, respectively, which follows that the sum of the lengths of the paths and cycles in such a partition will be $n-i$. Moreover, if V is covered, the paths and cycles are disjoint because of the size restriction. The above definition for $C_V(i, j)$ is equivalent to that in [Example 5](#). Note that $h_l(Y)$ can be replaced by $h'_l(Y)$ which denotes the number of walks of length l in $D[Y]$. Similarly, $c_l(Y)$ can be replaced by $c'_l(Y)$ which is the number of cyclic walks of length l . Using the technique introduced in [Theorem 7](#), $C\xi_Y(i, j)$ can be computed using the following formula given $h\xi'_l(Y)$ and $c\xi'_l(Y)$:

$$C\xi_Y(i, j) = \frac{1}{i!j!} \sum_{l_1 + \dots + l_{i+j} = n-i} \prod_{t=1}^i h\xi'_{l_t}(Y) \prod_{p=1}^j c\xi'_{l_{i+p}}(Y). \quad (45)$$

Finally, $h\xi'_l(Y)$ and $c\xi'_l(Y)$ can be computed in polynomial time using standard dynamic programming (cf. [44]). Then $C_V(i, j)$ can be obtained by computing the Möbius inversion as described in [Theorem 7](#) (by a similar argument as used in proving [Lemma 1](#), it can be proved that all extra items introduced in the process of relaxing h_l and c_l are canceled through computing the Möbius inversion). Putting everything together will provide an $O^*(2^n)$ time and polynomial space algorithm for counting cover polynomial. □

Combining with the algebraic method to be introduced in Sect. 3.2, the technique introduced in Theorem 7 can be used to give polynomial space algorithms for the traveling salesman problem, the Steiner tree problem, the weighted set cover problem, etc., without increasing the running times as have been derived in the fastest solutions to these problems [12, 20, 28, 36].

2.5.2 Polynomial Space Exact Algorithms Based on Inclusion-Exclusion

By the inclusion-exclusion principle (25), if $|\bigcap_{i \in X} A_i|$ can be computed in polynomial time for each $X \subseteq \{1, \dots, n\}$, there exists a polynomial space algorithm evaluating Eq. (25) in $O^*(2^n)$ time.

Example 7 (Counting Hamiltonian Paths) Given a graph $G = (V, E)$, a Hamiltonian path is a walk that contains each node exactly once. The Counting Hamiltonian Path problem is to count the number of Hamiltonian paths. The following inclusion-exclusion based algorithm is due to Karp [36]: define the universe B as all walks of length $n - 1$ in G , and define A_v as all walks of length $n - 1$ containing v for each $v \in V$. By the inclusion-exclusion principle, the number of Hamiltonian paths is

$$h = \sum_{X \subseteq V} (-1)^{|X|} |\bigcap_{v \in X} \overline{A_v}|. \quad (46)$$

For $X \subseteq V$ and $s \in X$, let $w_k(s, X)$ be the number of walks from s of length k in $G[X]$. Then

$$|\bigcap_{v \in X} \overline{A_v}| = \sum_{s \in V \setminus X} w_{n-1}(s, V \setminus X). \quad (47)$$

For fixed $X \subseteq V$, $w_k(s, X)$ can be computed in polynomial time using the following recurrence:

$$w_k(s, X) = \begin{cases} 1, & \text{if } k = 0 \\ \sum_{t \in N(s) \cap X} w_{k-1}(t, X), & \text{otherwise.} \end{cases} \quad (48)$$

Notice that $w_k(s, X)$ is at most $O(n^n)$ which needs polynomial bits to represent. Thus, $|\bigcap_{v \in X} \overline{A_v}|$ can be computed in polynomial space and time, which provides an $O^*(2^n)$ time and polynomial space algorithm to evaluate Eq. (46).

Example 8 (Counting Perfect Matchings) Given two undirected graphs F and G , let $\text{sub}(F, G)$ denote the number of distinct copies of a graph F contained in a graph G . Clearly, if F is a matching of $n/2$ edges, where n is the vertex number of G , then $\text{sub}(F, G)$ is the number of perfect matchings in G .

A homomorphism from F to G is a mapping from the vertex set of F to that of G such that the image of every edge of F is an edge of G . Let $\text{hom}(F, G)$ and $\text{inj}(F, G)$ be the numbers of homomorphisms and injective homomorphisms from F to G , respectively. Furthermore, let $\text{aut}(F, F)$ denote the number of automorphisms, that

is, bijective homomorphisms, from F to itself. The following equation switches the focus to computing $\text{inj}(F, G)$, since $\text{aut}(F, F)$ for a graph F on n_F vertices can be computed in subexponential time [2]

$$\text{sub}(F, G) = \frac{\text{inj}(F, G)}{\text{aut}(F, F)}. \quad (49)$$

Let S be a given subset of $V(G)$, then a homomorphism f from F to G is called S -saturating if (a) $S \subseteq f(V(F))$ and (b) for all $v \in S$, $|f^{-1}(v)| = 1$. Let $S - \text{hom}(F, G)$ denote the number of S -saturating homomorphisms. By the inclusion-exclusion principle,

$$\text{inj}(F, G) = \sum_{W \in V(G) \setminus S} (-1)^{|W|} S - \text{hom}(F, G[V(G) \setminus W]). \quad (50)$$

Note that if F is a matching of $n/2$ edges, $S - \text{hom}(F, G[V(G) \setminus W])$ can be computed in polynomial time and polynomial space using the following equation, which gives rise to an $O^*(2^n)$ time polynomial space algorithm for counting the number of perfect matchings in G . Let $S = \{v_1, \dots, v_a\}$, then

$$\begin{aligned} S - \text{hom}(F, G[V(G) \setminus W]) &= \binom{\frac{n}{2}}{a} a! \left(\prod_{i=1}^a (2 \deg_{V(G) \setminus W}(v_i)) \right. \\ &\quad \left. |2E(G[V(G) \setminus (W \cup S)])|^{\frac{n}{2}-a}, \right) \end{aligned} \quad (51)$$

where $\deg_{V(G) \setminus W}(v_i)$ is the number of vertices adjacent to v_i in $V(G) \setminus W$.

2.5.3 Linear Space Exact Algorithms Based on Linear Space Fast Zeta Transform

As shown before, inclusion-exclusion together with the fast zeta transform can provide efficient exact algorithms for many hard problems, for example, the covering problem, the partitioning problem, and the packing problem (also cf. [19, 20, 40]). However, all these algorithms need $O^*(2^n)$ space to carry out the fast zeta transform. The following linear space fast zeta transform ([Algorithm 2](#)) can help these algorithms achieve a linear space bound without increasing the running time.

Theorem 8 (Linear Space Zeta Transform) *Suppose f vanishes outside a family \mathcal{F} of subsets of N and the member of \mathcal{F} can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$. Then the values $f \zeta(X)$, for every $X \subseteq U$, can be listed in time $O^*(2^n)$ and space $O^*(|\mathcal{F}|)$ using [Algorithm 2](#).*

Proof Partition U into U_1 and U_2 such that $|U_1| = n_1$, $|U_2| = n_2$, where $n_2 = \lceil \log |\mathcal{F}| \rceil$ and $n_1 = n - n_2$. The correctness of [Algorithm 2](#) is established by the following Lemma.

Algorithm 2 Linear space algorithm for zeta transform

Input: A universe set N of n elements and a functions f defined on a family \mathcal{F} of subsets of N

Output: The zeta transform $f\zeta$ for all sets in 2^N

```

1: For each  $X_1 \subseteq U_1$  do
2:   For each  $Y_2 \subseteq U_2$  do
3:     set  $g(Y_2) \leftarrow 0$ 
4:   End For
5:   For each  $Y \in \mathcal{F}$  do
6:     If  $Y \cap U_1 \subseteq X_1$ , then
7:       set  $g(Y \cap U_2) \leftarrow g(Y \cap U_2) + f(Y)$ 
8:     End For
9:   Compute  $h \leftarrow g\zeta$  using the fast zeta transform on  $2^{U_2}$ 
10:  For each  $X_2 \subseteq U_2$  do
11:    output the value  $h(X_2)$  as the value  $f\zeta(X_1 \cup X_2)$ 
12:  End For
13: End For

```

Lemma 7

$$f\zeta(X_1 \cup X_2) = h(X_2) \quad (52)$$

Proof

$$\begin{aligned}
h(X_2) &= \sum_{Y_2 \subseteq X_2} g(Y_2) \\
&= \sum_{Y_2 \subseteq X_2} \sum_{Y \in \mathcal{F}} [Y \cap U_1 \subseteq X_1][Y \cap U_2 = Y_2] f(Y) \\
&= \sum_{Y \in \mathcal{F}} [Y \cap U_1 \subseteq X_1][Y \cap U_2 \subseteq X_2] f(Y) \\
&= \sum_{Y \in \mathcal{F}, Y \subseteq X_1 \cup X_2} f(Y) \\
&= \sum_{Y \subseteq X_1 \cup X_2} f(Y) \\
&= f\zeta(X_1 \cup X_2).
\end{aligned} \tag{53}$$

□

Observe that the algorithm runs in $O^*(2^{n_1}(2^{n_2} + |\mathcal{F}|))$ time and $O^*(2^{n_2})$ space. By the assigned values of n_1 and n_2 , the algorithm thus runs in $O^*(2^n)$ time and $O^*(|\mathcal{F}|)$ space. □

In [Algorithm 2](#), because the computations are independent for each $X_1 \subseteq U_1$, they can be executed in parallel on $O^*(2^n/|\mathcal{F}|)$ processors.

Example 9 (Counting k -Partitions) The problem and the notations are defined in [Example 4](#). It is convenient to express the k -partitions in terms of generating polynomials. Based on the principle of inclusion-exclusion, (28) can be reformulated as follows in which $a_k(X)$ is replaced by a polynomial over an intermediate z :

$$d_k(\mathcal{F}) = \sum_{X \subseteq N} (-1)^{|N \setminus X|} \left(\sum_{i=0}^n a_i(X) z^i \right)^k, \quad (54)$$

where the coefficient $a_j(X)$ is the number of subsets $Y \subseteq X$ that belong to \mathcal{F} and are of size j . Now, $d_k(\mathcal{F})$ is a polynomial whose coefficient of the monomial z^n is the number of k -partitions. To evaluate this expression, note that the polynomial $a(X) = \sum_{i=0}^n a_i(X) z^i$ is equal to the zeta transform $g\zeta(X)$, where $g(Y) = [Y \in \mathcal{F}] z^{|Y|}$. Now the linear space fast zeta transform operating in a ring of polynomials lists the polynomials $a(X)$ for all $X \subseteq N$ in $O^*(2^n)$ time and $O^*(|\mathcal{F}|)$ space. \square

Similar ideas to that used in the linear space fast zeta transform can give space efficient algorithms for other elementary transforms and problems, for example, the intersection transform, the disjoint sum, and the chromatic polynomial. An important feature of this type of algorithms is that they admit efficient parallelization as discussed before. In particular, some algorithms [\[17\]](#) of this type can achieve a trade-off between time and space complexities by adjusting the number of processors executing the algorithm in parallel.

2.6 Faster Exact Algorithms in Bounded-Degree Graphs

For bounded-degree graphs, faster exact algorithms can be derived for certain hard problems. The basic idea is to exploit some special properties so that only a special set of subsets of the vertex set, but not all subsets, need to be considered; then, based on a projection theorem presented by Chung et al. [\[23\]](#), which can be used to bound the size of the special set, the running time can be reduced for bounded-degree graphs. In this section, an inclusion-exclusion based algorithm will exemplify how to derive faster exact algorithms for bounded-degree graphs. This method is not only applicable to inclusion-exclusion based algorithms but also to some other types of algorithms. The following lemma is the key to employ this method.

Lemma 8 ([23]) *Let V be a finite set with subsets A_1, A_2, \dots, A_r such that every $v \in V$ is contained in at least δ subsets. Let \mathcal{F} be a family of subsets of V . For each $1 \leq i \leq r$ define the projections $\mathcal{F}_i = \{F \cap A_i : F \in \mathcal{F}\}$. Then,*

$$|\mathcal{F}|^\delta \leq \prod_{i=1}^r |\mathcal{F}_i|. \quad (55)$$

Example 10 (TSP in bounded-degree graphs) Given a graph G with nodes V and distant function $d : V \times V \rightarrow N$, the traveling salesman problem (TSP) is to find a minimum-weighted Hamiltonian cycle based on the distance function that

minimizes the total distance. Assume that the maximum degree of G is $\Delta = O(1)$. A set W of vertices is a connected dominating set if every vertex $v \in V$ is either in W or adjacent to a vertex in W and the induced subgraph $G[W]$ is connected. Denote by \mathcal{CD} the family of connected dominating sets of G .

The starting point is the algorithm by Karp [36], and, independently, by Kohn et al. [39]. Assume that the distance function is bounded by a constant, that is, $d(u, v) \in \{0, 1, \dots, B\} \cup \{\infty\}$ for any pair of vertices u, v , where $B = O(1)$.

The algorithm can be conveniently described in terms of generating polynomials. Select an arbitrary reference vertex $s \in V$, and let $U = V \setminus \{s\}$. For each $X \subseteq U$, denote by $q(X)$ the polynomial over the indeterminate z for which the coefficient of each monomial z^w counts the directed closed walks (in the complete directed graph with vertex set V and edge weights given by d) through s that (i) avoid the vertices in X , (ii) have length n , and (iii) have finite weight w . Then $q(X)$ can be computed in time polynomial in n by the following recurrence and setting $q(X) = p(n, s)$ for a fixed $X \subseteq U$. Initialize the recurrence for each vertex $u \in V \setminus X$ with

$$p(0, u) = \begin{cases} 1, & \text{if } u = s \\ 0, & \text{otherwise.} \end{cases} \quad (56)$$

For convenience, define $z^\infty = 0$. For each length $l = 1, 2, \dots, n$ and each vertex $u \in V \setminus X$, let

$$p(l, u) = \sum_{v \in V \setminus X} p(l-1, v) z^{d(v,u)}. \quad (57)$$

Here, note that due to the assumption on bounded weights, each $p(l, u)$ has at most a polynomial number of monomials with nonzero coefficients.

By the principle of inclusion-exclusion, the coefficients of the monomials in the polynomial

$$Q = \sum_{X \subseteq U} (-1)^{|X|} q(X) \quad (58)$$

count, by weight, the number of directed Hamiltonian cycles. It follows immediately that the traveling salesman problem can be solved in time $O^*(2^n)$.

For bounded-degree graphs, faster algorithms can be derived by analyzing (58) in more details. It will be convenient to work with a complemented form of (58). For each $S \subseteq U$, let $r(S) = q(U \setminus S)$. Then (58) can be rewritten as

$$Q = (-1)^n \sum_{S \subseteq U} (-1)^{|S|} r(S). \quad (59)$$

Observe that the induced subgraph $G[\{s\} \cup S]$ need not be connected. Associate with each $S \subseteq U$ the unique $f(S) \subseteq U$ such that $G[\{s\} \cup f(S)]$ is the connected component of $G[\{s\} \cup S]$ that contains s . It follows that $r(S) = r(f(S))$ for all $S \subseteq U$. This observation enables the following partition of the subsets of U into 9 f -preimages of constant r -value. For each $T \subseteq U$, let $f^{-1}(T) = \{S \subseteq U : f(S) = T\}$. Then rewrite (59) in the partition form

$$Q = (-1)^n \sum_{T \subseteq U} r(T) \sum_{S \in f^{-1}(T)} (-1)^{|S|}. \quad (60)$$

Lemma 9

$$\sum_{S \in f^{-1}(T)} (-1)^{|S|} = \begin{cases} (-1)^{|T|}, & \text{if } \{s\} \cup T \in \mathcal{CD} \\ 0, & \text{otherwise.} \end{cases} \quad (61)$$

Proof Consider an arbitrary $T \subseteq U$. The preimage $f^{-1}(T)$ is clearly empty if $G[\{s\} \cup T]$ is not connected. Thus, in the following assume that $G[\{s\} \cup T]$ is connected. For a set $W \subseteq V$, denote $\bar{N}(W)$ the set of vertices in W or adjacent to at least one vertex in W . Observe that $f(S) = T$ holds for an $S \subseteq U$ iff $T \subseteq S$ and $S \cap \bar{N}(\{s\} \cup T) = T$. In particular, if $V \setminus \bar{N}(\{s\} \cup T)$ is nonempty, then $f^{-1}(T)$ contains equally many even- and odd-sized subsets. Conversely, if $V \setminus \bar{N}(\{s\} \cup T)$ is empty (which means that $\{s\} \cup T$ is a dominating set of G), then $f^{-1}(T) = \{T\}$. \square

Using the above lemma to simplify (60), we have

$$Q = (-1)^n \sum_{T \subseteq U, \{s\} \cup T \in \mathcal{CD}} (-1)^{|T|} r(T). \quad (62)$$

To arrive at an algorithm with running time $|\mathcal{CD}|n^{O(1)}$, it suffices to list the elements of \mathcal{CD} with delay bounded by $n^{O(1)}$. Observe that \mathcal{CD} is an upclosed family of subsets of V , that is, if a set is in the family, then so are all of its supersets. Furthermore, whether a given $W \subseteq V$ is in \mathcal{CD} can be decided in polynomial time. These observations enable listing the sets in \mathcal{CD} in a top-down, depth-first manner. Thus, the only remaining task is to bound the size of \mathcal{CD} .

Lemma 10 *Let V be a finite set with r elements and with subsets A_1, A_2, \dots, A_r such that every $v \in V$ is contained in exactly δ subsets. Let \mathcal{F} be a family of subsets of V and assume that there is a log-concave function $f \geq 1$ and $0 \leq s \leq r$ such that the projections $\mathcal{F}_i = \{F \cap A_i : F \in \mathcal{F}\}$ satisfy $|\mathcal{F}_i| \leq f(|A_i|)$ for each $s+1 \leq i \leq r$. Then,*

$$|\mathcal{F}| \leq f(\delta)^{r/\delta} \prod_{i=1}^s 2^{|A_i|/\delta}. \quad (63)$$

Proof Let $a_i = |A_i|$ and note that $\sum_{i=1}^r a_i = \delta r$. By Lemma 8,

$$|\mathcal{F}|^\delta \leq \prod_{i=1}^s 2^{a_i} \prod_{i=s+1}^r f(a_i) \leq \prod_{i=1}^s 2^{a_i} \prod_{i=1}^r f(a_i). \quad (64)$$

Since f is log-concave, Jensen's inequality gives

$$\frac{1}{r} \sum_{i=1}^r \log f(a_i) \leq \log f \left(\left(\sum_{i=1}^r a_i \right) / r \right) = \log f(\delta). \quad (65)$$

Taking exponential and combining with (64) gives $|\mathcal{F}|^\delta \leq f(\delta)^r \prod_{i=1}^s 2^{|A_i|}$, which yields the claimed bound. \square

In order to bound the size of \mathcal{CD} , we need only to consider the special case where A_i are defined in terms of neighborhoods of the vertices of G . For each $v \in V$, define the closed neighborhood $N(v)$ by $N(v) = \{v\} \cup \{u \in V : u \text{ and } v \text{ are adjacent in } G\}$. Begin by defining the subsets A_v for $v \in V$ as $A_v = N(v)$. Then, for each $u \in V$ with degree $d(u) < \Delta$, add u to $\Delta - d(u)$ of the sets A_v not already containing it (it does not matter which ones). This ensures that every $u \in V$ is contained in exactly $\Delta + 1$ sets A_v . Combining with the following given bound on the size of \mathcal{CD} , an $O^*(\beta_\Delta^n)$ time algorithm can be obtained as described above, where $\beta_\Delta = (2^{\Delta+1} - 2)^{1/(\Delta+1)}$.

Lemma 11 *An n -vertex graphs with maximum vertex degree Δ has at most $\beta_\Delta^n + n$ connected dominating sets, where $\beta_\Delta = (2^{\Delta+1} - 2)^{1/(\Delta+1)}$.*

Proof Let $\mathcal{CD}' = \mathcal{CD} \setminus \{\{v\} : v \in V\}$. Then for every $C' \in \mathcal{CD}'$ and every A_v , $C' \cap A_v \neq \{v\}$. Thus, the number of sets in the projection $\mathcal{CD}'_v = \{F \cap A_v : F \in \mathcal{CD}'\}$ is at most $2^{|A_v|} - 2$, since $F \cap A_v \neq \emptyset$ for any $F \in \mathcal{CD}'$. To obtain the bound, apply Lemma 10 with the log-concave function $f(x) = 2^x - 1$ and $s = 0$. \square

3 Algebraic Methods

The application of algebraic methods in designing exact algorithms can be dated back to the famous paper by Kohn et al. [39] on solving the traveling salesman problem using the generating polynomials (for details, please refer to Example 10), in which by involving an intermediate x , a polynomial $G(x) = \sum a_i x^{e_i}$ is produced, where e_i denotes the cost of a possible tour and a_i denotes the number of tours having this cost. Then the minimal tour cost, that is, the smallest exponent, can be extracted by evaluating $G(x)$ at a specific value close to 0 and using a logarithmic technique. In this section, two recently developed algebraic methods are introduced – algebraic sieving method and polynomial circuit-based algebraic method.

3.1 Algebraic Sieving

Sieving methods are commonly used in designing exact and parameterized algorithms and have a long history. For example, the inclusion-exclusion principle is a typical sieving method by which all possibilities are first counted, and then, false contributions are canceled out. Here, a new algebraic sieving method which makes use of determinants over a field of characteristic two to achieve sieving is introduced.

The basic idea is to construct a polynomial in the underlying variables over a finite field in which at least one unique monomial exists for each required structure (e.g., Hamiltonian cycle) and no monomials results from unwanted structures (e.g., non-Hamiltonian cycle covers). The existence of structures being looked for ultimately emerges as a polynomial testing problem. Then the old fingerprint idea, often attributed to Freivalds (cf. [43]), is applied in which the polynomial is evaluated in a randomly chosen point in a finite field. The fingerprint result is used to judge whether the constructed polynomial is zero polynomial or not. The Schwartz-Zippel Lemma (cf. [43]) ensures that such judgment will be correct with high probability. The key point of this method is how to sieve the unwanted monomials to obtain the final required polynomial. So far only determinants over a finite field are taken advantage of to achieve sieving. Whether there exists some other more powerful tools that can be used for sieving under the algebraic framework needs further studying. Before going into the details, some notations need to be defined.

The determinant of an $n \times n$ matrix \mathbf{A} over an arbitrary ring R can be defined by the Leibniz formula:

$$\det(\mathbf{A}) = \sum_{\sigma:[n] \rightarrow [n]} \operatorname{sgn}(\sigma) \prod_{i=1}^n A_{i,\sigma(i)}, \quad (66)$$

where the summation is over all permutations of n elements, and sgn is a function called the sign of the permutation which assigns either 1 or -1 to a permutation. The permanent of \mathbf{A} is defined by

$$\operatorname{per}(\mathbf{A}) = \sum_{\sigma:[n] \rightarrow [n]} \prod_{i=1}^n A_{i,\sigma(i)}. \quad (67)$$

Denote $GF(2^k)$ as a finite field of characteristic two. It is well known that the determinant of \mathbf{A} over $GF(2^k)$ coincides with the permanent, since in such a field every element serves as its own additive inverse, in particular the element 1; the sgn function identically maps 1 to every permutation. Moreover, although the determinant is a sum of an exponential number of terms, it can be computed in polynomial time using for instance LU factorization of the matrix which can be found in any textbook on linear algebra. In fact, to compute the determinant is not harder than square matrix multiplication (cf. [21]). Hence, the determinant can be computed in $O(n^\omega)$ field operations where ω is the Coppersmith-Winograd exponent [24]. The following properties and lemma will be used in deriving algebraic sieving algorithms.

Property 1 \mathbf{A} is a matrix over $GF(2^k)$, then $\operatorname{per}(\mathbf{A}) = \det(\mathbf{A})$.

Property 2 For a given $n \times n$ matrix \mathbf{A} , $\det(\mathbf{A})$ can be computed in $O(n^\omega)$ time.

Lemma 12 (Schwartz-Zippel) Let $P(x_1, \dots, x_n)$ be a non zero n -variate polynomial of total degree d over a field F . Pick $r_1, \dots, r_n \in F$ uniformly at random, then

$$\Pr_r(P(r_1, \dots, r_n) = 0) \leq \frac{d}{|F|}. \quad (68)$$

3.1.1 k -Dimensional Matching

A hypergraph $H = (V, E)$ consists of a set V of n vertices and a multiset E of hyperedges which are subsets of V . In particular, hyperedges may include only one (or even no) vertex and may appear more than once. In a k -uniform hypergraph each edge $e \in E$ has size $|e| = k$. Given a vertex subset U of V , the projected hypergraph of H on U , denoted as $H[U] = (U, E[U])$, is a hypergraph on U where there is one edge $e_U \in E[U]$ for every $e \in E$, defined by $e_U = e \cap U$.

Definition 4 (k -Dimensional Matching) Given a k -uniform hypergraph $H = (V_1 \cup V_2 \cup \dots \cup V_k, E)$, with $E \subseteq V_1 \times V_2 \times \dots \times V_k$, where $|V_i| = n$ for $1 \leq i \leq k$, the k -dimensional matching problem asks whether there exists a k -dimensional matching $S \subseteq E$ such that $\bigcup_{e \in S} e = V_1 \cup V_2 \cup \dots \cup V_k$ and $\forall e_1 \neq e_2 \in S: e_1 \cap e_2 = \emptyset$.

In the following, each hyperedge e is associated with a variable x_e over $GF(2^m)$ for some $m \geq \log n + 1$. The next lemma shows how to construct a polynomial only consisting of monomials representing k -dimensional matchings by sieving unwanted monomials.

Lemma 13 Denote \mathcal{M} as the family of all k -dimensional matchings. Let $V = \bigcup_{i=1}^k V_i$ and $U = V_1 \cup V_2$, then

$$\sum_{X \subseteq V \setminus U} P(H, U, X) = \sum_{M \in \mathcal{M}} \prod_{e \in M} x_e, \quad (69)$$

where the computation is over a multivariate polynomial ring over $GF(2^m)$, and

$$P(H, U, X) = \sum_{M' \in \mathcal{M}'} \prod_{e \in M'} x_e, \quad (70)$$

where \mathcal{M}' is the family of all perfect matchings in $H[U]$ avoiding X , that is, for every $e \in M'$, $e \cap X = \emptyset$.

Proof For every $M \in \mathcal{M}$, M only avoids \emptyset . Thus, the monomial $\prod_{e \in M} x_e$ for every k -dimensional matching M emerges in the constructed polynomial precisely once. For a non- k -dimensional matching M' , it avoids all subsets of $V \setminus (\bigcup_{e \in M'} e)$. Then the monomial $\prod_{e \in M'} x_e$ for M' will be considered in $P(H, U, X')$ for every $X' \subseteq V \setminus (\bigcup_{e \in M'} e)$. It is well known that a nonempty set has even number of subsets. Hence, all of these $\prod_{e \in M'} x_e$ cancel each other since the operations are in a field of characteristic two. \square

From [Lemma 13](#), $\sum_{X \subseteq V \setminus U} P(H, U, X)$ consists of monomials representing k -dimensional matchings of H . Thus, by evaluating $\sum_{X \subseteq V \setminus U} P(H, U, X)$ on a randomly chosen point $r_1, \dots, r_{|E|} \in GF(2^m)$, the result is nonzero with probability at least $\frac{1}{2}$ if there exists at least one k -dimensional matching in H by [Lemma 12](#). If repeating the evaluation for polynomial times, the error probability can be exponentially small. The remaining work is how to efficiently compute $P(H, U, X)$ on a randomly chosen point $r_1, \dots, r_{|E|} \in GF(2^m)$ for every $X \subseteq V \setminus U$.

Clearly, $H[U]$ is a bipartite multigraph. Define its Edmonds matrix $\mathbf{A}(H, V_1, V_2)$ as

$$\mathbf{A}(H, V_1, V_2)_{ij} = \sum_{e=(i,j) \in E[U], i \in V_1, j \in V_2} x_e. \quad (71)$$

Denote \mathcal{M} as the family of all perfect matchings in $H[U]$. Then the following lemma generalizes Edmonds' work [26] for the case of bipartite multigraphs.

Lemma 14 $\det(\mathbf{A}(H, V_1, V_2)) = \sum_{M \in \hat{\mathcal{M}}} \prod_{e \in M} x_e$, where the computation is over a multivariate polynomial ring over $GF(2^m)$ and the summation is over all perfect matchings $\hat{\mathcal{M}}$ in $H[V_1 \cup V_2]$.

Proof By the definition of determinant, the summation is over all products of n of the matrix elements in which every row or column is used exactly once. In terms of the bipartite graph, this corresponds to a perfect matching in the graph since rows and columns represent the two vertex sets, respectively. Furthermore, the converse is also true. For every perfect matching there is a permutation describing it. Hence, the mapping is one-to-one. The inner product counts all choices of edges producing a matching described by a permutation σ since

$$\prod_{i=1}^n A_{i,\sigma(i)} = \prod_{i=1}^n \sum_{e=(i,\sigma(i))} x_e = \sum_{M \in \mathcal{M}(\sigma)} \prod_{e \in M} x_e, \quad (72)$$

where $\mathcal{M}(\sigma)$ is the set of all perfect matchings $\{e_1, \dots, e_n\}$ such that $e_i = (i, \sigma(i))$.

Let $H_X[U]$ denote the projected hypergraph of H on U by only projecting edges disjoint to X in H on U . Based on the above Lemma, $P(H, U, X)$ on a randomly chosen point $r_1, \dots, r_{|E|} \in GF(2^m)$ for every $X \subseteq V \setminus U$ can be computed by first constructing the Edmonds matrix of $H_X[U]$, with the variables replaced by the random sample points $r_1, \dots, r_{|E|}$, and then computing its determinant. Putting everything together generates a Monte Carlo algorithm with exponentially small error probability for the k -dimensional matching problem. The runtime bound is easily seen to be $O^*(2^{(k-2)n})$ since $|U| = |V_1| + |V_2| = 2n$ and $P(H, U, X)$ for every $X \subseteq V \setminus U$ can be evaluated in $O(n^\omega)$ time by [Lemma 14](#) and [Property 2](#).

Theorem 9 The k -dimensional matching problem on kn vertices can be solved by a Monte Carlo algorithm with an exponentially small failure probability in n and $O^*(2^{(k-2)n})$ runtime.

3.1.2 Hamiltonicity

An undirected graph $G = (V, E)$ on n vertices is said to be Hamiltonian if it has a Hamiltonian cycle, a vertex order $(v_0, v_1, \dots, v_{n-1})$ such that $v_i v_{i+1} \in E$ for all i . The indices are enumerated modulo n , that is, $v_{n-1} v_0$ is also an edge. The problem of detecting if a graph is Hamiltonian is called the Hamiltonicity problem. Clearly, this problem is a special case of the traveling salesman problem.

Before the algebraic sieving algorithm for the Hamiltonicity problem was derived, the dynamic programming recurrence that solves the general TSP in $O(n^2 2^n)$ time introduced by Bellman [6, 7] and independently by Held and Karp [30] in the early 1960s was the strongest result for this special problem. The algebraic sieving for the Hamiltonicity problem is obtained by reducing the problem to a variant of the cycle cover counting problem called labeled cycle cover sum, and then making use of the sieving algorithm to solve this variant. The algorithm will be described after introducing some useful notations.

In a directed graph $D = (V, A)$, a cycle cover is a subset $C \subseteq A$ such that for every vertex $v \in V$, there is exactly one arc $a_{v1} \in C$ starting from v and exactly one arc $a_{v2} \in C$ ending in v . The graphs considered have no loops. Denote $cc(D)$ as the family of all cycle covers of D , and $hc(D) \subseteq cc(D)$ as the set of Hamiltonian cycle covers. A Hamiltonian cycle cover consists of one big cycle passing through all vertices. The remaining cycle covers (which have more than one cycle in $cc(D) \setminus hc(D)$) are called non-Hamiltonian cycle covers. For undirected graphs, $hc(G)$ includes the Hamiltonian cycles with orientation, that is, traversed in both directions. For a Hamiltonian cycle $H \in hc(G)$ for an undirected graph G , an arc $uv \in G$ infers that the cycle is oriented from u to v along the edge uv . Denote $g : A \rightarrow B$ as a surjective function from the domain A to the codomain B and denote the preimage of every $b \in B$ as $g^{-1}(b)$, that is, $g^{-1}(b) = \{a \in A : g(a) = b\}$. For a matrix A , denote by $A_{i,j}$ the element at row i and column j . The labeled cycle cover sum problem is defined as follows:

Definition 5 Given a directed graph $D = (V, A)$, a finite set L of labels, and a function $f : A \times 2^L \rightarrow R$ on some codomain ring R , the labeled cycle cover sum problem is to compute the following defined term over R .

$$\Lambda(D, L, f) = \sum_{C \in cc(D)} \sum_{g: L \rightarrow C} \prod_{a \in C} f(a, g^{-1}(a)), \quad (73)$$

where the inner sum is over all surjective functions g .

In the above definition, f is introduced to make all non-Hamiltonian cycle covers cancel out in the sum. To do this, as before, the computations will be done over $GF(2^k)$ denoting a finite field of characteristic two. In what follows, a directed graph is bidirected if for every arc uv it has an arc in the opposite direction, that is, vu . In order to make sure that the Hamiltonian cycle cover will not be canceled as well, an asymmetry around an arbitrarily chosen special vertex s is defined. A function $f : A \times 2^L \rightarrow GF(2^k)$ is an s -oriented mirror function if

$f(uv, Z) = f(vu, Z)$ for all Z and all $u \neq s, v \neq s$. The following lemma captures how the non-Hamiltonian cycle covers vanish, which also implies the nonexistence of false positives in the resulting algorithms.

Lemma 15 *Given a bidirected graph $D = (V, A)$, a finite set L , and a special vertex $s \in V$, let f be an s -oriented mirror function with codomain $GF(2^k)$. Then*

$$\Lambda(D, L, f) = \sum_{H \in hc(D)} \sum_{g: L \rightarrow H} \prod_{a \in H} f(a, g^{-1}(a)). \quad (74)$$

Proof By the definition of the labeled cycle cover sum, a labeled cycle cover is a tuple (C, g) with $C \in cc(D)$ and $g : L \rightarrow C$. The labeled non-Hamiltonian cycle covers can be partitioned into dual pairs as follows such that both cycle covers in every pair contribute the same term to the sum. For a labeled non-Hamiltonian cycle cover (C, g) , let \mathbf{C} be the first cycle of C not passing through s (note that such cycle must exist since the cycle cover consists of at least two cycles and all cycles are vertex disjoint). Here, “first” is w.r.t. any fixed order of the cycles. Then the dual cycle cover of (C, g) is defined as $R(C, g) = (C', g')$, where $C' = C$ except for the cycle \mathbf{C} which is reversed in C' , that is, every arc $uv \in \mathbf{C}$ is replaced by the arc in the opposite direction vu in C' , and g'^{-1} is identical to g^{-1} on $C \setminus \mathbf{C}$, and is defined by $g'^{-1}(uv) = g^{-1}(vu)$ for all arcs $uv \in \mathbf{C}$. In other words, the reversed arcs preserve their original labeling. Note that such a dual cycle cover always exists since D is bidirected and $(C, g) \neq R(C, g), (C, g) = R(R(C, g))$. Hence, the mapping uniquely pairs up the labeled non-Hamiltonian cycle covers.

Since f is an s -oriented mirror function and has $f(uv, Z) = f(vu, Z)$ for all arcs uv not incident on s , (C, g) and $R(C, g)$ contribute the same product term to the sum for computing $\Lambda(D, L, f)$. Finally, since the computations are done in a field of characteristic two, all these terms will be canceled. \square

When limiting the computations over $GF(2^k)$, the labeled cycle cover sum can be computed efficiently via determinant. Permanent has a natural interpretation as the sum of weighted cycle covers in a directed graph. Formally, let $D = (V, A)$ be a directed graph with weights $w : A \rightarrow GF(2^k)$, and define a $|V| \times |V|$ matrix with rows and columns representing the vertices V

$$A_{i,j} = \begin{cases} w(ij), & \text{if } ij \in A \\ 0, & \text{otherwise.} \end{cases} \quad (75)$$

Then,

$$per(A) = \sum_{C \in cc(D)} \prod_{a \in C} w(a). \quad (76)$$

By [Property 1](#) and [2](#), $per(A)$ is identical with $det(A)$ and can be computed in $O(n^\omega)$ time. Define a polynomial in an indeterminate r as follows, with r aiming at controlling the total rank of the subsets used as labels in the labeled cycle covers:

$$p(f, r) = \sum_{Y \subseteq L} det \left(\sum_{Z \subseteq Y} r^{|Z|} M_f(Z) \right), \quad (77)$$

where

$$M_f(Z)_{i,j} = \begin{cases} f(ij, Z), & \text{if } ij \in A \\ 0, & \text{otherwise.} \end{cases} \quad (78)$$

Lemma 16 *For a directed graph D , a set L of labels, and any function $f : A \times L \rightarrow GF(2^k)$,*

$$[r^{|L|}]p(f, r) = \Lambda(D, L, f), \quad (79)$$

where $[r^{|L|}]p(f, r)$ denotes the coefficient of $r^{|L|}$ in $p(f, r)$.

Proof $p(f, r)$ can be rewritten as

$$\begin{aligned} p(f, r) &= \sum_{Y \subseteq L} \sum_{C \in cc(D)} \sum_{q:C \rightarrow 2^Y \setminus \{\emptyset\}} \prod_{a \in C} r^{|q(a)|} f(a, q(a)) \\ &= \sum_{C \in cc(D)} \sum_{q:C \rightarrow 2^L \setminus \{\emptyset\}} \sum_{\substack{\cup_{a \in C} q(a) \subseteq Y \subseteq L \\ \cup_{a \in C} q(a) = L}} \prod_{a \in C} r^{|q(a)|} f(a, q(a)). \end{aligned} \quad (80)$$

For functions $q : C \rightarrow 2^L \setminus \{\emptyset\}$ such that $\cup_{a \in C} q(a) \subset L$, that is, whose union over the elements do not cover all of L , the innermost summation is executed an even number of times with the same term (there are $2^{|L - \cup_{a \in C} q(a)|}$ equal terms). Since the computations are over $GF(2^k)$, these cancel. Then,

$$p(f, r) = \sum_{C \in cc(D)} \sum_{\substack{q:C \rightarrow 2^L \setminus \{\emptyset\} \\ \cup_{a \in C} q(a) = L}} r^{\sum_{a \in C} |q(a)|} \prod_{a \in C} f(a, q(a)), \quad (81)$$

in which the coefficient of $r^{|L|}$

$$[r^{|L|}]p(f, r) = \sum_{C \in cc(D)} \sum_{\substack{q:C \rightarrow 2^L \setminus \{\emptyset\} \\ \cup_{a \in C} q(a) = L \\ \forall a \neq b : q(a) \cap q(b) = \emptyset}} \prod_{a \in C} f(a, q(a)), \quad (82)$$

since $\cup_{a \in C} q(a) = L$, $\cup_{a \in C} |q(a)| = |L|$ implies $\forall a \neq b : q(a) \cap q(b) = \emptyset$. Inverting the function q will give the labeled cycle cover sum as defined in [Definition 5](#). \square

The above lemma is the base identity that enables a relatively efficient algorithm for computing the labeled cycle cover sum.

Lemma 17 *The labeled cycle cover sum $\Lambda(D, L, f)$ over a field $GF(2^k)$ on a directed graph D on n vertices, and with $2^k > |L|n$, can be computed in $O((|L|^2n + |L|n^{1+\omega})2^{|L|})$ arithmetic operations over $GF(2^k)$, where ω is the Coppersmith-Winograd matrix multiplication coefficient.*

Proof The labeled cycle cover sum is evaluated via the identity in Lemma 16. Observing that $p(f, r)$ as a polynomial in r has maximum degree $|L|n$, to recover one of its coefficients (the one for $r^{|L|}$), one needs to evaluate the polynomial for $|L|n$ choices of r and to use interpolation to solve for the coefficient being sought. This can be done for instance by using a generator g of the multiplicative group in $GF(2^k)$ and evaluating the polynomial in the points $r = g^0, g^1, \dots, g^{|L|n-1}$. The requirement $2^k > |L|n$ assures the distinctness of these points, and hence, the interpolation is possible. For every fixed r , the algorithm begins by tabulating $T(Y) = \sum_{Z \subseteq Y} r^{|Z|} M_f(Z)$ for all $Y \subseteq L$ through the fast zeta transform (4) in $O(|L|2^{|L|})$ field operations. Then $p(f, r) = \sum_{Y \subseteq L} \det(T(Y))$ can be evaluated in $O(n^\omega 2^{|L|})$ operations by Property 2. Once all values are computed, the polynomial time Lagrange interpolation can be used to compute the coefficient. Summing up the number of field operations required over all $|L|n$ values of r , the lemma follows. \square

Next, f will be defined to associate the argument arc, and label set with a nonconstant multivariate polynomial such that $f(su, X)$ and $f(us, X)$ will not share variables for any $su, us \in A$ to ensure that the Hamiltonian cycles oriented in opposite directions will contribute different terms to the sum. Based on the definition of f , the labeled cycle cover sum is represented by a polynomial in the underlying variables with one unique monomial per oriented Hamiltonian cycle and without monomials resulting from non-Hamiltonian cycle covers, which is a consequence of Lemma 15. Then, the fingerprint idea is employed to evaluate the polynomial in a randomly chosen point, and the Schwarz-Zippel Lemma ensures that with high probability, it can be detected whether the polynomial resulting from the labeled cycle cover sum is identically zero (i.e., no Hamiltonian cycles) or not (there is at least one Hamiltonian cycle) by identifying the evaluating result with the polynomial. For simplicity of analysis, assume that n is even.

For a uniformly randomly chosen partition $V = V_1 \cup V_2$ with $|V_1| = |V_2| = n/2$ and a fixed Hamiltonian cycle $H = (v_0, \dots, v_{n-1})$ in G . An arc $v_i v_{i+1}$ is called unlabeled by V_2 if both v_i and v_{i+1} belong to V_1 , and the set of all unlabeled arcs is denoted as $U(H)$. The remaining arcs are referred to as labeled by V_2 and denote the set of all labeled arcs as $L(H)$. Define $hc(G, V_2, m)$ as the subset of $hc(G)$ of Hamiltonian cycles which have precisely m unlabeled arcs by V_2 . Assign every edge $uv \in E$ two variables x_{uv} and x_{vu} , and x_{uv} and x_{vu} are identified except when $u = s$ or $v = s$.

Consider a complete bidirected graph $D = (V_1, F)$ and use V_2 as some of the labels. In addition to V_2 , a set L_m of size m of extra labels aiming at handling arcs unlabeled by V_2 . For each edge uv in $G[V_1]$ and every element $d \in L_m$, new variables $x_{uv,d}$ and $x_{vu,d}$ are introduced. And $x_{uv,d}$ coincides with $x_{vu,d}$ except when $u = s$ or $v = s$. For two vertices $u, v \in V$, and a subset $X \subseteq V$, define $\mathcal{P}_{u,v}(X)$ as the family of all simple paths in G from u to v passing through exactly the vertices in X . For $uv \in F$ and $X \subseteq V_2$, define

$$f(uv, X) = \sum_{P \in \mathcal{P}_{u,v}(X)} \prod_{wz \in P} x_{wz}. \quad (83)$$

For every arc $uv \in F$ such that uv is an edge in $G[V_1]$, and $d \in L_m$, define $f(uv, \{d\}) = x_{uv,d}$. In other points f is set to zero.

Lemma 18 *With $G, D, V_2, U(\cdot), L(\cdot), m, L_m$ and f defined as above,*

- (i) $\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc(G, V_2, m)} \left(\prod_{uv \in L(H)} x_{uv} \right) \left(\sum_{\sigma: U(H) \rightarrow L_m} \prod_{uv \in U(H)} x_{uv, \sigma(uv)} \right);$
- (ii) $\Lambda(D, V_2 \cup L_m, f)$ is a zero polynomial iff $hc(G, V_2, m) = \emptyset$.

Proof (i) Since D is bidirected and f is s -oriented mirror function, by Lemma 15,

$$\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc(D)} \sum_{g: V_2 \cup L_m \rightarrow H} \prod_{a \in H} f(a, g^{-1}(a)), \quad (84)$$

where the inner sum is over all surjective functions g . In order to prove the claim, it needs to expand the Hamiltonian cycles in D into Hamiltonian cycles of G . Note that the arcs of a Hamiltonian cycle in D in the sum above are labeled either by an element of L_m or by a nonempty subset of V_2 , since only in such cases f is nonzero. Next, the definition of labeled and unlabeled arcs are extended (which were defined previously for Hamiltonian cycles in G only). For a Hamiltonian cycle $H \in hc(D)$ labeled by the surjective function $g : V_2 \cup L_m \rightarrow H$, an arc $uv \in H$ is called labeled by V_2 if $g^{-1}(uv) \subseteq V_2$ and unlabeled by V_2 if $g^{-1}(uv) \in L_m$.

Since every arc uv unlabeled by V_2 , $g^{-1}(uv)$ is an element of L_m , only the Hamiltonian cycles in D with exactly m arcs unlabeled by V_2 leave a nonzero contribution. Then,

$$\begin{aligned} \Lambda(D, V_2 \cup L_m, f) &= \sum_{H \in hc(D)} \sum_{\substack{H_L \cup H_U = H \\ H_L \cap H_U = \emptyset \\ |H_U| = m}} \left(\sum_{g: V_2 \rightarrow H_L} \prod_{a \in H_L} f(a, g^{-1}(a)) \right) \cdot \\ &\quad \left(\sum_{\sigma: H_U \rightarrow L_m} \prod_{a \in H_U} f(a, \sigma(a)) \right), \end{aligned} \quad (85)$$

where the summation is over all surjective functions g and one-to-one functions σ . Replace f in the above expression, then

$$\begin{aligned} \Lambda(D, V_2 \cup L_m, f) = & \sum_{H \in hc(D)} \sum_{\substack{H_L \cup H_U = H \\ H_L \cap H_U = \emptyset \\ |H_U| = m \\ \forall a \in H_U : a \in E}} \left(\sum_{g: V_2 \rightarrow H_L} \prod_{a \in H_L} \sum_{P \in \mathcal{P}_{u,v}(g^{-1}(uv))} \prod_{w \in P} x_{wz} \right) \cdot \\ & \left(\sum_{\sigma: H_U \rightarrow L_m} \prod_{uv \in H_U} x_{uv, \sigma(uv)} \right), \end{aligned} \quad (86)$$

Every vertex in V_2 is mapped to precisely one arc in F on a Hamiltonian cycle H in D by g . Moreover, such a cycle H leaves a nonzero result iff there are precisely m arcs in the cycle not being mapped to by g (i.e., unlabeled by V_2) and they are also edges in G . Rewriting the expression as a sum of Hamiltonian cycles in G ,

$$\Lambda(D, V_2 \cup L_m, f) = \sum_{H \in hc(G, V_2, m)} \left(\prod_{uv \in L(H)} x_{uv} \right) \left(\sum_{\sigma: U(H) \rightarrow L_m} \prod_{uv \in U(H)} x_{uv, \sigma(uv)} \right). \quad (87)$$

(ii) If the graph G has no Hamiltonian cycles, the sum is clearly zero. For the other direction, each Hamiltonian cycle contributes a set of $m!$ different monomials per orientation of the cycle (one for each permutation σ), in which there is one variable per edge along the cycle. Monomials resulting from different Hamiltonian cycles are different since for each of two different Hamiltonian cycles, one has an edge that the other does not have. Every pair of opposite directed Hamiltonian cycles along the same undirected Hamiltonian cycle also traverse s through opposite directed arcs. Since the variables tied to the opposite directed arcs incident on s are different, these are also unique monomials in the sum. \square

The above Lemma describes how to transform the input graph G given m , V_1 , V_2 into a symbolic labeled cycle cover sum $\Lambda(D, V_2 \cup L_m, f)$ on a bidirected graph D on $n/2$ vertices and $n/2 + m$ labels $V_2 \cup L_m$. The next lemma shows that only the cases $m \leq n/4$ need to be considered.

Lemma 19 *Let $G = (V, E)$ be a Hamiltonian undirected graph. For a uniformly randomly chosen equal partition $V_1 \cup V_2 = V$, $|V_1| = |V_2| = n/2$,*

$$Pr \left(\sum_{i=0}^{n/4} |hc(G, V_2, i)| > 0 \right) \geq \frac{1}{n+1}. \quad (88)$$

Proof Consider one Hamiltonian cycle (v_0, \dots, v_{n-1}) in G . Let X denote the random variable representing the number of unlabeled arcs by V_2 . Clearly, $\Pr(v_i) = \frac{1}{2}$ for any i , and $\Pr(v_i, v_{i+1} \in V_1) < \frac{1}{4}$ since the events $v_i \in V_1$ and $v_{i+1} \in V_1$ are almost independent. Hence, the expected number of arcs unlabeled by V_2 , $E(X) < n/4$. By Markov's inequality, $\Pr(X > (1 + 1/n)E(X)) < \frac{1}{1+1/n}$, and the lemma follows. \square

Before discussing the algorithm, the only remaining problem is how to calculate f for all subsets of V_2 . This can be achieved by running a variant of the Bellman-Held-Karp recursion. Formally, let $\hat{f} : (V \times V) \times 2^V \rightarrow GF(2^k)$ be defined by

$$\hat{f}(uv, X) = \sum_{P \in \mathcal{P}_{u,v}(X)} \prod_{w \in P} x_{wz}. \quad (89)$$

Then $f(uv, X) = \hat{f}(uv, X)$ for $u, v \in V_1, X \subseteq V_2$, and the recursion

$$\hat{f}(uv, X) = \begin{cases} x_{uv}, & \text{if } |X| = 0 \\ \sum_{w \in X, uw \in E} x_{uw} \hat{f}(wv, X \setminus \{w\}), & \text{otherwise.} \end{cases} \quad (90)$$

can be used to tabulate $f(\cdot, X)$ on $X \subseteq V_2$. The running time is $O^*(2^{|V_2|})$.

Next, the algorithm is described in detail. First, pick a partition $V_1 \cup V_2 = V$ uniformly at random with $|V_1| = |V_2| = n/2$. Then, the algorithm loops over m , the number of edges unlabeled by V_2 along a Hamiltonian cycle from 0 to $n/4$. For each value of m , by Lemma 18, the problem is transformed to determine whether the symbolic labeled cycle cover sum $\Lambda(D, V_2 \cup L_m, f)$ is a nonzero polynomial. As described in Lemma 17, $\Lambda(D, V_2 \cup L_m, f)$ will be evaluated in a randomly chosen point over the field $GF(2^k)$ with $2^k > cn$ for some $c > 1$. By Lemma 12, with probability at least $1 - 1/c$ it will result in a nonzero answer iff $\Lambda(D, V_2 \cup L_m, f)$ was a nonzero polynomial (i.e., G has a Hamiltonian cycle with m edges unlabeled by V_2). Then repeat the algorithm for every $m \leq n/4$, and by Lemma 19, with probability at least $(1 - 1/c)/(n + 1)$, the presence of a Hamiltonian cycle can be detected if it exists. Running the algorithm a polynomial number of times in n , the probability of failure will be reduced exponentially small. The runtime is bounded by a polynomial factor in m and n of the runtime in Lemma 17. The worst case occurs for $m = 4/n$ in which case the time bound is $O^*(2^{3n/4})$.

Theorem 10 *There is a Monte Carlo algorithm detecting whether an undirected graph with n vertices is Hamiltonian or not in $O^*(2^{3n/4})$ time, with no false positives and no false negatives with probability exponentially small in n .*

3.2 Polynomial Circuit-Based Algorithms

This method only needs polynomial space. The problems to be solved will be translated into polynomials represented by polynomial circuits, where the translation is done in such a way that the instance considered is a “yes”-instance iff the coefficient of a special monomial in the polynomial is nonzero; this allows us only to compute the coefficient of the special monomial. A discrete Fourier transform-based technique provides an efficient way to achieve the goal.

3.2.1 Coefficient Interpolation

Given a set R and a set of variables X , an arithmetic circuit C over $(R, +, *)$ is a directed acyclic graph in which each source gate is labeled by a variable $x \in X$ or an element of R and other gates is either an addition gate or a product gate. The size $|C|$ of a circuit C is the number of gates in C . The depth $\Delta(C)$ of a circuit C is the length of a longest directed path in C . Denote by \mathbb{N} and \mathbb{C} the naturals and the complex numbers, respectively.

Computing the coefficient of some special monomial can be abstracted as the following coefficient interpolation problem which can be solved efficiently based on the discrete Fourier transform.

Definition 6 (Coefficient Interpolation) Given an arithmetic circuit C over $(\mathbb{N}, +, *)$ and variables x_1, x_2, \dots, x_β over \mathbb{N} that computes a polynomial $P(x_1, \dots, x_\beta)$, together with β integers t_1, t_2, \dots, t_β , the problem is to compute the coefficient of the term $x_1^{t_1} x_2^{t_2} \cdots x_\beta^{t_\beta}$.

Denote the coefficients of P as a β -dimensional array $\{c_{n_1, n_2, \dots, n_\beta}\}$. Furthermore, assume that the coefficients of any polynomial outputted by a gate of C are bounded by m and any exponent of x_i in the terms of P is bounded by $N_i - 1$ for $i \leq \beta$. Note that P can also be interpreted as a polynomial over the complex numbers. The discrete Fourier transform of the β -dimensional array $\{c_{n_1, n_2, \dots, n_\beta}\}$ is the array $\{C_{n_1, n_2, \dots, n_\beta}\}$ given by the following formula:

$$\begin{aligned} C_{k_1, k_2, \dots, k_\beta} &= \frac{1}{N_1 N_2 \cdots N_\beta} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cdots \sum_{n_\beta=0}^{N_\beta-1} c_{n_1, n_2, \dots, n_\beta} \prod_{i=1}^{\beta} \omega_{N_i}^{k_i n_i} \\ &= P(\omega_{N_1}^{k_1}, \omega_{N_2}^{k_2}, \dots, \omega_{N_\beta}^{k_\beta}), \end{aligned} \quad (91)$$

where ω_{N_i} is the i -th complex root of unity, given by $e^{-2\pi i / N_i}$. Then the inverse discrete Fourier transform can give the coefficient $c_{n_1, n_2, \dots, n_\beta}$, as follows:

$$\begin{aligned}
c_{t_1, t_2, \dots, t_\beta} &= \frac{1}{N_1 N_2 \cdots N_\beta} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cdots \sum_{n_\beta=0}^{N_\beta-1} C_{n_1, n_2, \dots, n_\beta} \prod_{i=1}^{\beta} \omega_{N_i}^{-t_i n_i} \\
&= \frac{1}{N_1 N_2 \cdots N_\beta} \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} \cdots \sum_{n_\beta=0}^{N_\beta-1} P(\omega_{N_1}^{n_1}, \omega_{N_2}^{n_2}, \dots, \omega_{N_\beta}^{n_\beta}) \prod_{i=1}^{\beta} \omega_{N_i}^{(N_i - t_i) n_i}.
\end{aligned} \tag{92}$$

Thus, by evaluating the polynomial P for $N_1 N_2 \cdots N_\beta$ different arrays of values, which can be done using the arithmetic circuit C , $c_{t_1, t_2, \dots, t_\beta}$ can be computed based on the above formula. Hence, $c_{t_1, t_2, \dots, t_\beta}$ can be computed with $O(N_1 N_2 \cdots N_\beta |C|)$ real multiplications and additions, and only $O(\beta + |C|)$ real numbers need to be stored at any point of the calculation.

In order to overcome the problem that the real number cannot be exactly stored and worked with, the binary representations of the numbers with the bits after the l most significant bits truncated will be used instead during the execution of the algorithm. Here, the number l is chosen such that if rounding the resulting estimate c' of $c_{t_1, t_2, \dots, t_\beta}$ to the nearest integer, $c_{t_1, t_2, \dots, t_\beta}$ can be correctly obtained. Observe that addition of these estimates can be done in time $O(l)$, and that multiplication can be carried out in $O^*(l)$ time using a fast algorithm for integer multiplication, such as the Fürer's algorithm [29]. Hence, the total time spent by the algorithm is $O^*((N_1 N_2 \cdots N_\beta)l)$ and the total space is $O((\beta + |C|)l)$. The following lemma makes sure the existence of such a choice for l .

Lemma 20 ([42]) *Let C be an arithmetic circuit over $(\mathbb{C}, +, *)$ computing a polynomial $P(x_1, x_2, \dots, x_\beta)$ such that (a) all constants of C are positive integers, (b) all coefficients of P are at most m , and (c) P has degree d (d) C has depth α . Let $x_1, x_2, \dots, x_\beta \in \mathbb{C}$, $x'_1, x'_2, \dots, x'_{\beta'} \in \mathbb{C}$ be such that for every i , $\|x_i\| = 1$ and $\|x_i - x'_i\| \leq \epsilon$. Let p' be the result of applying the circuit C to $x'_1, x'_2, \dots, x'_{\beta'}$ using the floating point arithmetic, truncating intermediate results after l bits. Suppose $(\epsilon + 2 \cdot 2^{-l})(4md)^\alpha \leq 1$. Then $\|p' - P(x_1, \dots, x_\beta)\| \leq (\epsilon + 2 \cdot 2^{-l})(4md)^\alpha$.*

For a fixed value of l , one can compute for every $i \leq \beta$ an estimate of ω_{N_i} whose distance ϵ to ω_{N_i} is at most $2 \cdot 2^{-l}$. Now, given $n_1, \dots, n_\beta, t_1, \dots, t_\beta, N_1, \dots, N_\beta$, based on the circuit C for computing P , a circuit C' can be constructed for computing

$$P' = P(x_1^{n_1}, x_2^{n_2}, \dots, x_\beta^{n_\beta}) \prod_{i=1}^{\beta} x_i^{(N_i - t_i) n_i}. \tag{93}$$

Clearly, the depth of C' is $\alpha + \log N + \log \beta$, where $N = \max_i N_i$. Furthermore, the degree of C' is at most $d + \beta N$, and the largest coefficient is at most md . Choose l such that $10 \cdot 2^{-l} (4(d + \beta N)md)^{\alpha + \log N + \log \beta} \leq 1$. Applying Lemma 20 on C' yields an estimation error for

$$P'(\omega_{N_1}^{n_1}, \omega_{N_2}^{n_2}, \dots, \omega_{N_\beta}^{n_\beta}) = P(\omega_{N_1}^{n_1}, \omega_{N_2}^{n_2}, \dots, \omega_{N_\beta}^{n_\beta}) \prod_{i=1}^{\beta} \omega_{N_i}^{(N_i - t_i)n_i}, \quad (94)$$

which is less than $\frac{1}{2}$. The total estimation error for $c_{t_1, t_2, \dots, t_\beta}$ is less than $\frac{N_1 N_2 \cdots N_\beta}{2 N_1 N_2 \cdots N_\beta} = \frac{1}{2}$. Thus, c' rounded to the nearest integer is $c_{t_1, t_2, \dots, t_\beta}$. Finally, observe that $l = ((\alpha + \log d)(\log m + \log d))$, which yields the following concluding theorem.

Theorem 11 *The coefficient interpolation problem can be solved in $O^*(N_1 \cdots N_\beta (\Delta(C) + \log d)(\log m + \log d))$ time and $O(\beta + |C|(\Delta(C) + \log d)(\log m + \log d))$ space.*

Remark Here, it is necessary to point out that the indegree of the constructed circuit C is implicitly required to be bounded by a constant.

By transforming the problems to polynomials, the above technique can provide pseudo-polynomial time polynomial space algorithms for several hard problems, for example, the Knapsack problem.

Example 11 (Knapsack Problem) Given a set $S = \{s_1, \dots, s_n\}$ of n items each of which has positive integer weight w_i and value v_i , and two positive integers w and v , the Knapsack problem asks whether there exists a subset S' of items whose total weight is at most w and total value is at most v . It may further assume that all items have weight at most w and value less than v , since otherwise this problem is trivial. The Knapsack problem can be reduced to coefficient interpolation as follows.

Define the polynomial $P_S(x, y) = \sum_{i=0}^{nw} \sum_{j=0}^{nv} c_{ij} x^i y^j$, where c_{ij} is the number of item sets $S' \subseteq S$ whose total weight is exactly i and the total value of the items not in S' is exactly j . The Knapsack problem is equivalent to checking whether there exists a weight $w' \leq w$ and value $v' \leq (\sum_{s_i \in S} v_i) - v$ such that $c_{w'v'}$ is nonzero. Clearly, this can be done by solving coefficient interpolation for each pair of possible values for w' and v' , but this would incur an unnecessary overhead of vw in the running time. Instead, define the polynomial $P'_S(x, y) = p_S(x, y)(\sum_{i=0}^{nw} x^i)(\sum_{j=0}^{nv} y^j)$ and set c'_{ij} to be the coefficient of the term $x^i y^j$ in P'_S . Then it suffices to check whether c'_{wv^*} is nonzero, where $v^* = \sum_{s_i \in S} v_i - v$. The remaining task is to construct a polynomial circuit C' for P'_S such that faster polynomial space algorithms can be obtained by applying Theorem 11.

Note that $P_S(x, y)$ can be factorized into $P_S(x, y) = \prod_{i=1}^n (x^{w_i} + y^{v_i})$ which yields a circuit of size $O(n \log w + n \log v)$ and depth $O(\log w + \log v + \log n)$ for computing P_S . Based on the following given circuit of size and depth $O(\log n + \log w)$ for computing $\sum_{s_i \in S} x^{w_i}$ and circuit of size and depth $O(\log n + \log v)$ for computing $\sum_{s_i \in S} y^{v_i}$, a circuit C' of size $O(n \log w + n \log v)$ and depth $O(\log w + \log v + \log n)$ for computing P'_S can be constructed.

Lemma 21 *There exists a circuit of size and depth $O(\log m)$ that computes $\sum_{i=0}^m x^i$.*

Proof The following recurrence can be turned into a circuit for computing $\sum_{i=0}^m x^i$.

$$\sum_{j=0}^p x^j = \begin{cases} (x^{p/2} + 1) \sum_{j=0}^{p/2} x^j, & \text{if } p \text{ is even} \\ x(x^{\lfloor p/2 \rfloor} + 1) \sum_{j=0}^{\lfloor p/2 \rfloor} x^j + 1, & \text{otherwise.} \end{cases} \quad (95)$$

To make the circuit it needs to construct gates evaluating $x^{\lfloor p/2 \rfloor}$, $x^{\lfloor p/4 \rfloor}$, $x^{\lfloor p/8 \rfloor}$, etc. This can be done using an identical trick to the one above by observing that $x^p = (x^{p/2})^2$ if p is even and $x^p = x(x^{\lfloor p/2 \rfloor})^2$ if p is odd. \square

Finally, note that the coefficients of P_S are at most 2^n , and hence, the coefficients of P'_S are at most $2^n w v n^2$. Also note that the degree of P'_S is at most $2nv + 2nw$. Then applying [Theorem 11](#), one can get an algorithm for the Knapsack problem with time complexity $O^*(n^4 vw(\log v + \log w)^2)$ and space complexity $O^*(n^2(\log v + \log w)^2)$. \square

3.2.2 Combination with Subset Convolution

By introducing some special gates (e.g., the subset convolution gate and the covering product gate introduced in [Sect. 2.2](#)) in the constructed polynomial circuits, some polynomial space algorithms for many other problems can be derived, for example, the TSP problem, the weighted Steiner tree problem, and the weighted set cover problem. Based on the polynomial space result for the subset convolution and the covering product ([Theorem 7](#) in [Sect. 2.5.1](#)) and using the embedding technique (introduced in [Sect. 2.2](#)), the following result can be obtained by [Theorem 11](#). Whether some other operations and a wider range of input functions can be involved in the polynomial circuit framework to give polynomial space algorithms or faster exponential space algorithms for hard problems, as well as whether this approach can be applied to a wider range of problems, need further studies.

Theorem 12 ([42]) *Let C be a circuit over $\{f : 2^V \rightarrow \mathcal{M}\}$ with operation gates including subset convolution, covering product, addition, and product gates. Let \hat{C} be the circuit over $\{\hat{g} : 2^V \rightarrow \mathbb{N}\}$ obtained by interpreting C over $(\mathbb{N}, +, *)$, and let u be such that $u \geq h(Y)$ where h is the output of any gate of \hat{C} and $Y \subseteq V$. If the input function is a singleton and is bounded by W , then it can be decided whether $f_{out}(V) \leq k$ in $O^*(2^{|V|} W \log u)$ time and $O^*(\log u \log W)$ space.*

Example 12 (Weighted Set Cover) Given a set U , a family $\mathcal{F} = \{S_1, \dots, S_l\} \subseteq 2^U$, a weight function $w : \mathcal{F} \rightarrow \mathbb{N}$, and an integer t . A set cover of U is a family $\mathcal{C} \subseteq \mathcal{F}$ such that $U \subseteq \sum_{S \in \mathcal{C}} S$. The weight $w(\mathcal{C}) = \sum_{S \in \mathcal{C}} w(S)$. The weighted set cover problem is to decide whether there exists a set cover of U with weight at most t .

Define $m_i(Y)$ as the minimum weight of a set cover \mathcal{C} of Y such that $|\mathcal{C}| = i$. Then the problem is to determine whether $m_n(U) \leq t$. $m_i(Y)$ can be computed using the following recurrence:

$$m_i(Y) = \begin{cases} 0, & \text{if } i = 0, Y = \emptyset \\ \infty, & i = 0, Y \neq \emptyset \\ \min_{1 \leq j \leq l} m_{\lceil i/2 \rceil}(Y * m_{\lfloor i/2 \rfloor}(Y * [Y \subseteq S_j]), & \text{otherwise,} \end{cases} \quad (96)$$

where $*$ denotes the subset convolution operation. By replacing the min operation by max and making use of the embedding technique (in [Algorithm 1](#)) to turn the operations over the max-sum semiring to be done over the product-sum ring, it can be proved that the input function is a singleton and the above recurrence can be turned into a circuit of depth $O(\log |U|)$. Applying [Theorem 12](#), a polynomial space algorithm with running time $O^*(2^{|U|} W)$ can be obtained, where W is the maximum weight assigned to sets in \mathcal{F} .

4 Conclusion

This chapter summarizes some recently resurrected or newly developed combinatorial and algebraic techniques for designing either faster or space efficient exact exponential time algorithms for NP-hard problems. Detailed examples to illustrate these techniques have been given. Despite significant progresses in the past few years, the area is still largely unexplored with many open problems, and new techniques for solving these problems are very much in need. For example, can the graph coloring problem be exactly solved in time faster than $O^*(2^n)$? Can a deterministic $O^*(c^n)$ time exact algorithm be derived for TSP where $c < 2$? It appears that using the inclusion-exclusion technique alone cannot break the $O^*(2^n)$ barrier of the graph coloring problem since it needs to go through all the subsets. As [Example 12](#) shows, one promising avenue along which to design either faster or space efficient exact algorithms is to combine multiple techniques. For graph coloring, by combining the combinatorial methods and the algebraic methods, an $O^*(c^n)$ ($c < 2$) time exact exponential time algorithm might eventually be obtained. More specifically, it is worthwhile to try applying some efficient combinatorial computing tools such as fast zeta transform in an algebraic framework. Furthermore, as Alan Perlis once said [41], “*for every polynomial-time algorithm you have, there is an exponential algorithm that I would rather run,*” exponential time exact algorithms with a small constant base would be preferable to polynomial time algorithms for some problem instances of moderate or reasonably large sizes. For example, an $O^*(1.01^n)$ exact algorithm may run much faster than a polynomial time algorithm with running time $O(n^3)$ for relatively large values of n , such as 2000. Besides designing faster exact algorithms, designing space efficient (in particular polynomial or even linear space) exact algorithms is also very important since compared to exponential time consumption, exponential space

consumption could be more costly from the standpoint of computing resource usages. All in all, developing practical polynomial space exact algorithms for real-life use will need more devoted effort in the future. Although some of the open problems listed in Woeginger’s survey [55] have been solved, many other ones remain open, some of which can be found in [56, 57].

5 Further Reading

The fast zeta transform comes from Yates’s work [58]. The general theory of Möbius inversions on posets is developed by Rota [48]. Björklund et al. introduced the linear space fast zeta transform in [17]. A variant of fast zeta transform – trimmed zeta transform—is introduced in [16], which is used to give faster exact algorithms for the important algorithmic tool disjoint sum in the design of parameterized algorithms [15]. For details on Yates’s algorithm and Möbius inversion, please refer to Knuth’s book [38] and Fedor and Kratsch’s book [27].

The fast algorithm for subset convolution comes from Björklund et al. [12]. There is also an FFT-based implementation for the fast subset convolution [27]. The fast algorithms for covering product and intersecting covering product are introduced by Björklund et al. [12], whereas the proof of Lemma 2 comes from Nederlof’s paper [44]. The exact algorithms for Examples 1–3 are due to Björklund et al. [12]. The fast subset convolution has also been used to derive faster exact algorithms for counting spanning forests [44], computing Tutte polynomial [13], computing f -width and rank-width [46]. For more information on applications of the fast subset convolution, please refer to [27]. The polynomial space algorithms for the covering product and the subset convolution are based on Nederlof’s work [44] and Lokshtanov and Nederlof’s work [42]. The exact algorithm for computing cover polynomial in Example 6 is due to [44]. Subset convolution can also be used to derive faster parameterized algorithms. For example, van Rooij, Bodlaender, and Rossmanith [52] gave an $O(2^t n)$ time parameterized algorithm for counting perfect matchings in graphs of treewidth at most t based on a generalized fast subset convolution algorithm. Vassilevska and Williams [54] introduced a new approach for computing permanent of rectangular matrix based on the fast subset convolution.

The application of inclusion-exclusion to combinatorial optimization goes back to Ryser’s formula for the permanent [49], which remains the most effective way to count the number of matchings in a bipartite graph. The first explicit reference to combinatorial optimization is for TSP by Kohn, Gottlieb, and Kohn [39] and a concise paper by Karp [36] applying the idea to a number of hard problems. Karp’s paper is very compact and has probably gone slightly unnoticed, as it focuses on cases where the technique offers reduction in space as compared to a dynamic programming algorithm but at the expense of a possible slight increase in running time. Later, Bax and Franklin [3–5] used similar methods to count paths and cycles in general graphs. These techniques provide an $O^*(2^n)$ time algorithm for counting Hamiltonian paths [4, 36].

The proof of the inclusion-exclusion principle comes from Björklund et al.'s paper [20]. Exact algorithms for the set partition problem in [Example 4](#) and for computing cover polynomial in [Example 6](#) are due to Björklund et al. [13, 20]. In the conference versions [19, 40] of [20] and some other papers [11, 31–33], exact algorithms using inclusion-exclusion for other partitioning and covering problems are given. The inclusion-exclusion technique is also used to give a faster exact algorithm for computing the permanent of a matrix over rings and finite commutative semirings [18]. The polynomial space exact algorithm for counting Hamiltonian paths in [Example 7](#) is due to Karp [36]. The inclusion-exclusion approach for counting subgraph isomorphisms using homomorphism as demonstrated in [Example 8](#) is developed by Amini, Fomin, and Saurabh [1]. Nederlof [44] further developed inclusion-exclusion techniques to derive a number of polynomial space algorithms. Additionally, the approach based on a combination of branching and inclusion-exclusion is developed in [45, 47, 51, 53]. For the recent progress on the algorithmic uses of inclusion-exclusion, please refer to the survey by Husfeldt [34].

The technique used for deriving faster exact algorithms in bounded graphs is introduced by Björklund et al. in [14, 16]. The exact algorithm for TSP in bounded graphs in [Example 10](#) comes from [14].

The application of algebraic methods in designing exact algorithms can be traced back to the famous paper by Kohn et al. [39] on solving the traveling salesman problem using generating polynomials. Björklund introduced the algebraic sieving method in some recent papers [8, 9]. The exact algorithms for k -dimensional matching and Hamiltonicity come from [8] and [9], respectively. The polynomial circuit-based approach for deriving polynomial space exact algorithms is developed by Lokshtanov and Nederlof in [42]. Faster polynomial space exact algorithms for subset sum were also proposed in the same paper, and by combining with subset convolution, they gave polynomial space exact algorithms for the Steiner tree problem as well as the TSP problem. Very recently, by using a new algebraic method based on computing the hafnian over an arbitrary ring, Björklund [10] gave a faster polynomial space algorithm for counting the number of perfect matchings.

This chapter focuses only on the recently resurrected or newly developed combinatorial and algebraic approaches for designing either faster or space efficient exact algorithms. Many recent successes in using these approaches suggest that these approaches could play a significant role in the design of exponential exact algorithms for many other NP-hard problems. Other than these approaches, there are also other classical methods for tackling NP-hard problems, such as branching and local search methods. For detailed illustrations of these classical methods, please refer to Woeginger's survey [55]. Schöning in his concise survey [50] investigated how randomization can be used to design randomized exponential time algorithms. Most recently, perhaps in response to the rapid development of exact algorithms research, Fomin and Kratsch have written an excellent text [27] on exact exponential algorithms. Both classical methods and newly developed techniques such as measure and conquer are covered by this book. The book discusses also combinatorial methods for which our chapter tries to fill in some of missing details.

On top of that, this chapter demonstrates how to employ these techniques to design space efficient exact algorithms without increasing the running time, as well as how the newly developed algebraic approaches can be utilized to design either faster or space efficient exact algorithms.

Acknowledgements This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00302, the National Natural Science Foundation of China Grant 61103186, 61073174, 61033001, 61061130540, the Hi-Tech research and Development Program of China Grant 2006AA10Z216, and Hong Kong RGC-GRF grants 714009E and 714311.

Cross-References

- ▶ [Advanced Techniques for Dynamic Programming](#)
 - ▶ [Algorithmic Aspects of Domination in Graphs](#)
 - ▶ [On Coloring Problems](#)
 - ▶ [Variations of Dominating Set Problem](#)
-

Recommended Reading

1. O. Amini, F.V. Fomin, S. Saurabh, Counting subgraphs via homomorphisms, in *ICALP*, Rhodes (1), 2009, pp. 71–82
2. L. Babai, W.M. Kantor, E.M. Luks, Computational complexity and the classification of finite simple groups, in *FOCS*, 1983, Tucson, pp. 162–171
3. E.T. Bax, Inclusion and exclusion algorithm for the Hamiltonian path problem. *Inf. Process. Lett.* **47**(4), 203–207 (1993)
4. E.T. Bax, Algorithms to count paths and cycles. *Inf. Process. Lett.* **52**(5), 249–252 (1994)
5. E.T. Bax, J. Franklin, A finite-difference sieve to count paths and cycles by length. *Inf. Process. Lett.* **60**(4), 171–176 (1996)
6. R. Bellman, Bottleneck problems and dynamic programming. *Proc. Natl. Acad. Sci.* **39**, 947–951 (1953)
7. R. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, 1957)
8. A. Björklund, Exact covers via determinants, in *STACS*, Nancy, 2010, pp. 95–106
9. A. Björklund, Determinant sums for undirected Hamiltonicity, in *FOCS*, Las Vegas, 2010, pp. 23–26
10. A. Björklund, Counting perfect matchings as fast as Ryser, in *SODA*, 2012, Kyoto, pp. 914–921
11. A. Björklund, T. Husfeldt, Exact algorithms for exact satisfiability and number of perfect matchings. *Algorithmica* **52**(2), 226–249 (2008)
12. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Fourier meets möbius: fast subset convolution, in *STOC*, San Diego, 2007, pp. 67–74
13. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Computing the tutte polynomial in vertex-exponential time, in *FOCS*, Philadelphia, 2008, pp. 677–686
14. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, The traveling salesman problem in bounded degree graphs, in *ICALP*, Reykjavik, 2008, pp. 198–209
15. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Counting paths and packings in halves, in *ESA*, Copenhagen, 2009, pp. 578–586
16. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Trimmed moebius inversion and graphs of bounded degree. *Theory Comput. Syst.* **47**, 637–654 (2010)

17. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Covering and packing in linear space, in *ICALP*, Bordeaux, (1), 2010, pp. 727–737
18. A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Evaluation of permanents in rings and semirings. *Inf. Process. Lett.* **110**(20), 867–870 (2010)
19. A. Björklund, T. Husfeldt, Inclusion-exclusion algorithms for counting set partitions, in *FOCS*, Berkeley, 2006, pp. 575–582
20. A. Björklund, T. Husfeldt, M. Koivisto, Set partitioning via inclusion-exclusion. *SIAM J. Comput.* **39**(2), 546–563 (2009)
21. J.R. Bunch, J.E. Hopcroft, Triangular factorization and inversion by fast matrix multiplication. *Math. Comput.* **28**, 231–236 (1974)
22. F.R.K. Chung, R.L. Graham, On the cover polynomial of a digraph. *J. Comb. Theory B* **65**(2), 273–290 (1995)
23. F.R.K. Chung, P. Frankl, R.L. Graham, J.B. Shearer, Some intersection theorems for ordered sets and graphs. *J. Comb. Theory A* **43**, 23–37 (1986)
24. D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions. *J. Symb. Comput.* **9**, 251–280 (1990)
25. S.E. Dreyfus, R.A. Wagner, The Steiner problem in graphs. *Networks* **1**, 195–207 (1971/1972)
26. J. Edmonds, Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Stand.* **71B**(4), 241–245 (1967)
27. F.V. Fomin, D. Kratsch, *Exact Exponential Algorithms* (Springer, Berlin/Heidelberg, 2010)
28. B. Fuchs, W. Kern, D. Mölle, S. Richter, P. Rossmanith, X. Wang, Dynamic programming for minimum Steiner trees. *Theory Comput. Syst.* **41**(3), 493–500 (2007)
29. M. Fürer, Faster integer multiplication, in *STOC*, San Diego, 2007, pp. 57–66
30. M. Held, R.M. Karp, A dynamic programming approach to sequencing problems. *J. Soc. Ind. Appl. Math.* **10**(1), 196–210 (1962)
31. Q.-S. Hua, D. Yu, Y. Wang, F.C.M. Lau, Exact algorithms for set multicover and multiset multicover problems, in *ISAAC*, Honolulu, 2009, pp. 34–44
32. Q.-S. Hua, Y. Wang, D. Yu, F.C.M. Lau, Set multi-covering via inclusion-exclusion. *Theor. Comput. Sci.* **410**(38–40), 3882–3892 (2009)
33. Q.-S. Hua, Y. Wang, D. Yu, F.C.M. Lau, Dynamic programming based algorithms for set multicover and multiset multicover problems. *Theor. Comput. Sci.* **411**(26–28), 2467–2474 (2010)
34. T. Husfeldt, Invitation to algorithmic uses of inclusion-exclusion, in *ICALP*, Zurich, (2), 2011, pp. 42–59
35. D.B. Johnson, Efficient algorithms for shortest paths in sparse networks. *J. ACM* **24**, 1–13 (1977)
36. R.M. Karp, Dynamic programming meets the principle of inclusion and exclusion. *Oper. Res. Lett.* **1**, 49–51 (1982)
37. R. Kennes, Computational aspects of the Moebius transform of a graph. *IEEE Trans. Syst. Man Cybern.* **22**, 201–223 (1991)
38. D.E. Knuth, *The art of computer programming, Vol. 3: Seminumerical algorithms*, 3rd edn. (Addison-Wesley, Upper Saddle River, 1998)
39. S. Kohn, A. Gottlieb, M. Kohn, A generating function approach to the traveling salesman problem, in *In ACM '77: Proceedings of the 1977 Annual Conference* (ACM, New York, 1977), pp. 294–300
40. M. Koivisto, An $O^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion, in *FOCS*, Berkeley, 2006, pp. 583–590
41. R. Lipton, Fast exponential algorithms, <http://rjlipton.wordpress.com/2009/02/13/polynomial-vs-exponential-time/>
42. D. Lokshtanov, J. Nederlof, Saving space by algebraization, in *STOC*, Cambridge, 2010, pp. 321–330
43. R. Motwani, P. Raghavan, *Randomized Algorithms* (Cambridge University Press, Cambridge/New York, 1995)

44. J. Nederlof, Fast polynomial-space algorithms using Möbius inversion: improving on Steiner tree and related problems, in *ICALP*, Rhodes, (1), 2009, pp. 713–725
45. J. Nederlof, J.M.M. van Rooij, Inclusion/exclusion branching for partial dominating set and set splitting, in *IPCO*, Chennai, 2010, pp. 204–215
46. S.i. Oum, Computing rank-width exactly. *Inf. Process. Lett.* **109**(13), 745–748 (2009)
47. D. Paulusma, J.M.M. van Rooij, On partitioning a graph into two connected subgraphs, in *ISAAC*, Honolulu, 2009, pp. 1215–1224
48. G.C. Rota, On the foundations of combinatorial theory. I. Theory of Möbius functions. *Z. Wahrscheinlichkeitstheorie und Verw. Gebiete* **2**, 340–368 (1964)
49. H.J. Ryser, *Combinatorial Mathematics*. Number 14 in Carus mathematical monographs (Mathematical Association of America, Buffalo, 1963)
50. U. Schöning, Algorithmics in exponential time, in *STACS*, Stuttgart, 2005, pp. 36–43
51. J.M.M. van Rooij, Polynomial space algorithms for counting dominating sets and the domatic number, in *CIAC*, Rome, 2010, pp. 73–84
52. J.M.M. van Rooij, H.L. Bodlaender, P. Rossmanith, Dynamic programming on tree decompositions using generalised fast subset convolution, in *ESA*, Copenhagen, 2009, pp. 566–577
53. J.M.M. van Rooij, J. Nederlof, T.C. van Dijk, Inclusion/exclusion meets measure and conquer: exact algorithms for counting dominating sets, in *ESA*, Copenhagen, 2009, pp. 554–565
54. V. Vassilevska, R. Williams, Finding, minimizing, and counting weighted subgraphs, in *STOC*, Bethesda, 2009, pp. 455–464
55. G.J. Woeginger, Exact algorithms for NP-hard problems: a survey, in *Combinatorial Optimization*, 2001, Springer-Verlag New York, pp. 185–208
56. G.J. Woeginger, Space and time complexity of exact algorithms: some open problems, in *IWPEC*, Bergen, 2004, pp. 281–290
57. G.J. Woeginger, Open problems around exact algorithms. *Discret. Appl. Math.* **156**(3), 397–405 (2008)
58. F. Yates, The design and analysis of factorial experiments. Technical Communication No. 35, Commonwealth Bureau of Soil Science, Harpenden, UK, 1937

Fault-Tolerant Facility Allocation

Hong Shen and Shihong Xu

Contents

1	Introduction	1294
2	Related Work	1295
3	Problem Formulation	1297
4	The Algorithm	1299
5	Analysis	1302
6	Conclusion	1307
	Recommended Reading	1307

Abstract

The problem of *Fault-Tolerant Facility Allocation (FTFA)* is a relaxation of the classical *Fault-Tolerant Facility Location (FTFL)* problem (Jain K, Vazirani VV (2000) An approximation algorithm for the fault tolerant metric facility location problem. In: APPROX '00: proceedings of the third international workshop on approximation algorithms for combinatorial optimization, London, UK. Springer, pp 177–183). Given a set of sites, each containing a set of identical facilities and associated with an operating cost for each facility, a set of clients, where each client-site pair has a (distance-based) connection cost and each client has a connection requirement *FTFA*, requires to compute a connection scheme between clients and sites such that each client is allocated a desired number of facilities and the total combined facility (operating) cost and connection cost

H. Shen (✉)

School of Computer Science, The University of Adelaide, Adelaide, SA, Australia

School of Information Science and Technology, Sun Yat-sen University, Guangzhou, People's Republic of China

e-mail: hong.shen@adelaide.edu.au

S. Xu

School of Computer Science, The University of Adelaide, Adelaide, SA, Australia

e-mail: shihong.xu@adelaide.edu.au

for all clients to access their required facilities is minimum. Compared with the *FTFL* problem which restricts that at most one facility can be opened at each site, the *FTFA* problem is less constrained and hence incurs a smaller cost. This chapter introduces our recent work on this problem (Xu S, Shen H (2009) The fault-tolerant facility allocation problem. In ISAAC, pp 689–698) and shows that the metric version of *FTFA*, i.e., the connection costs satisfy triangle inequality, is solvable in polynomial time within approximation ratio 1.861, which is better than the best approximation ratio 2.076 for metric *FTFL* (Swamy C, Shmoys DB (2008) Fault-tolerant facility location. ACM Trans Algorithms 4(4):1–27) known that time.

1 Introduction

Facility location [9] is one of the classical problems that has been widely studied in the field of operation research. In this problem, we are given a set \mathcal{F} of n_f sites, each holding one facility, and a set \mathcal{C} of n_c clients: every facility $i \in \mathcal{F}$ is associated with a nonnegative number f_i as the facility operating cost, and every facility-client pair $(i, j), i \in \mathcal{F}, j \in \mathcal{C}$ is associated with a connection cost c_{ij} to access facility i from client j . The objective is to open a subset of the facilities in \mathcal{F} and connect each client to an open facility so that the total cost is minimized.

Recently a generalization of the facility location problem, namely, *Fault-Tolerant Facility Allocation (FTFA)*, was studied in [27]. In this problem, each site allows an unlimited number of facilities to be opened, and each client requires a desired number of connections for fault-tolerance purpose, i.e., r_j connections to open facilities for client $j \in \mathcal{C}$. *FTFA* requires to allocate each site a proper number of facilities and further each client the required number of facilities so that the total combined facility cost and connection cost is minimized. The *FTFA* problem can be formulated by the following integer linear program:

$$\begin{aligned} & \text{minimize} && \sum_{i \in \mathcal{F}} f_i y_i + \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}} c_{ij} x_{ij} \\ & \text{subjected to} && \forall j \in \mathcal{C} : \sum_{i \in \mathcal{F}} x_{ij} \geq r_j \\ & && \forall i \in \mathcal{F}, j \in \mathcal{C} : y_i \geq x_{ij} \\ & && \forall i \in \mathcal{F}, j \in \mathcal{C} : x_{ij}, y_i \in \mathbb{Z}^+ \end{aligned} \tag{1}$$

In this formulation, nonnegative integer y_i indicates the number of facilities opened at site i , and x_{ij} indicates the number of connections established between site i and client j . The first constraint ensures QoS wrt client's connection requirement, i.e., established connections must satisfy client's required connection degree, achieving the desired fault-tolerance. The second constraint ensures there are enough opened facilities at site i to be connected from client j . In this chapter, the metric version of the problem, i.e., the connection costs satisfy triangle inequality, is considered.

The formulation of *FTFA* is similar to the well-studied *Fault-Tolerant Facility Location (FTFL)* problem [11, 12, 16, 25] which has the same objective function and

constraints as *FTFA* except the range of variants: x_{ij} and y_i are nonnegative integers (i.e., \mathbb{Z}^+) in *FTFA* but binary integers (i.e., 0 or 1) in *FTFL* for any $i \in \mathcal{F}, j \in \mathcal{C}$. Without the restriction on the maximum number of facilities that can be opened at each site, *FTFA* is less constrained and hence incurs a smaller total cost. The *FTFA* problem is applicable in surrogate server deployment in a Content Distribution Network. In such an application, multiple facilities (surrogate servers or ATMs in an ATM network) can be deployed at one site if necessary, which share the duty to serve clients together. It is noticed that the *FTFA* problem becomes the classical *Uncapacitated Facility Location (UFL)* problem when connectivity requirement $r_j = 1$ for all $j \in \mathcal{C}$. It is not hard to see that the hardness of *UFL*, *FTFA*, and *FTFL* complies with the following relation:

$$UFL \subseteq FTFA \subseteq FTFL.$$

Here, the second inclusion is implied by a special *FTFL* problem which has a set of facilities distributed by groups. Let $\mathcal{F}' = \mathcal{F} \times \{1, 2, \dots, R\}$, $R = \max_{j \in \mathcal{C}} r_j$ be the number of identical facilities in each group. Using this setting, the *FTFA* problem can be solved by *FTFL* algorithms because the number of facilities at any site is no more than R in any optimal solution of the *FTFA* problem. However, we notice that the existing algorithms for *FTFL* are not as efficient as the algorithms for *UFL*, in both approximation ratio and time complexity: most *FTFL* algorithms employ an LP routine which is rather time-consuming, and the best known approximation ratio for *FTFL* is 2.09 which is considerably worse than the best known 1.5 approximation ratio for *UFL*. Therefore, in order to achieve a better result than that from applying *FTFL* algorithms directly, we shall take the full advantage of existing methods for *UFL* in solving the *FTFA* problem. In fact, the *FTFA* problem can be regarded as a *UFL* problem with an additional constraint: for a given *FTFA*, consider a *UFL* problem with the aforementioned \mathcal{F}' and city set $\mathcal{C}' = \{(j, p), j \in \mathcal{C}, 1 \leq p \leq r_j\}$, where (j, p) is the p -th port of city j . In this setting, we need to ensure no parallel connections are made between any facility-city pair, i.e., different ports of a city must be connected with different facilities. This constraint is nontrivial and as a result, *FTFA* becomes harder to solve than *UFL*, yielding the first inclusion. In the subsequent sections, we will show how to deal with this constraint and obtain better single-factor and bi-factor approximation solutions to *FTFA* than that for *FTFL*.

2 Related Work

The facility location problem and its variants occupy a central place in operations research [9]. For the simplest problem – the maximization variant of *Uncapacitated Facility Location* – Cornuejols et al. [8] obtained a $(1 - e^{-1})$ -approximation algorithm. The first approximation algorithm for the minimization variant is a greedy algorithm due to [13], which is $O(\log n)$ -approximation in the general (nonmetric) case. The *UFL* problem has found extensive applications since this

works, and one of the most widely studied problems is *metric UFL*. In this problem, the function of connection cost forms a metric, i.e., the connection costs between facilities and cities satisfy triangle inequality. Existing algorithms for the *metric UFL* problem mainly use two types of techniques, i.e. LP rounding and primal-dual algorithms.

The first constant-factor approximation algorithm for the *metric UFL* problem was due to Shmoys et al. [23]. They gave a 3.16-approximation algorithm using the filtering technique of Lin and Vitter [19] to round the optimal solution of a linear programming relaxation. Chudak improves the approximation ratio to 1.736 [7] and 1.582 [24] also by rounding an optimal fractional solution to a linear program. These results are achieved through linear programming and rounding.

Charikar and Guha obtained 1.853- and 1.728-approximation algorithms [4] by using primal-dual theory and greedy augmentation; Jain et al. presented greedy algorithms based on dual-fitting and factor-revealing LP technique, achieving approximation guarantees 1.861 [14, 22] and 1.61 [14, 18]. Different from traditional primal-dual schema [15, 26], dual fitting relaxes the feasibility of the dual solution. Suppose the dual solution becomes feasible after being shrunk by a factor, then this factor is the approximation ratio of the algorithm. They also studied the trade-off [14] between facility and connection cost and got a series of bi-factor approximation guarantees. Mahdian et al. further improved the approximation ratio to 1.52 [20] by adding a scaling and greedy augmentation procedure to the JMS algorithm. Recently, Jaroslow [2] presented a 1.5-approximation algorithm by modifying Chudak and Shmoys's algorithm [6] and then combining it with a bi-factor result of JMS algorithm [18]. This is currently the best known result for the problem touching the approximability limit.

Fault-Tolerant Facility Location [16] is a generalization of *UFL*, where connectivities at different cities (e.g., the number of distinct facilities that serves a city) are specified to meet fault-tolerant requirements. Guha et al. obtained a 3.16-approximation algorithm by rounding the optimal fractional solution to the problem and further improve the result to 2.41 by employing a greedy local improvement step [12]. Swamy and Shmoys presented a 2.076-approximation by using LP rounding [25]. Recently, Byrka et al. [2] applied the dependent-rounding technique to FTFL and achieved the current best approximation ratio 1.725. All these results hold for both uniform connectivity case and nonuniform connectivity case (general case). Guha and Khuller proved that the best approximation ratio (lower bound) to *UFL* is 1.463 [10], assuming $\text{NP} \not\subseteq \text{DTIME}[n^{O(\log \log n)}]$; this result also holds for fault-tolerant version of the problem.

The *k-Median* problem [19] has also been studied extensively [1, 3, 5], and the best known approximation ratio for this problem is $3 + \varepsilon$ [1]. Jain et al. study a new problem called *k-Facility* which is a combination of the *k-Median* and *UFL* problems and achieve a 6-approximation algorithm [17] and further a 4-approximation [14] based on the JMS algorithm [14, 18]. The *Fault-Tolerant k-Facility* problem was also studied by Swamy and Shmoys [25], and they achieved

a 4-approximation algorithm for a special case of the problem where all cities have an identical connectivity requirement. Performance of their algorithm is unknown for the general case.

3 Problem Formulation

In order to obtain a suitable dual for formulation (1) from which a satisfactory approximation ratio can be derived, we need to first transform (1) into an equivalent formulation by applying a greedy approach of multiphase connection establishment as follows.

Without loss of generality, assume $\mathcal{R} = \{1, 2, 3, \dots, R\}$; otherwise, we may add dummy clients with the missing connectivity requirements. Further assume r_j ports at each client and R facilities at each site. All ports of a client must be connected in the order from 1 to r_j , and all facilities at a site can be opened, if necessary, in the order from 1 to R . We use multiphase algorithms to establish connections for certain cities in each phase. More specifically, we establish one connection for all cities in $\mathcal{C}^p = \{j \in \mathcal{C} : r_j \geq p\}$ in phase p . We use vector $y_i^p \in \{0, 1\}$ to denote whether the p -th facility at site i is opened in phase p and $x_{ij}^p \in \{0, 1\}$ whether the p -th port of a city is connected with a facility at site i . It is clear that $y_i = \sum_{p \in \mathcal{R}} y_i^p$, $x_{ij} = \sum_{p \in \mathcal{R}} x_{ij}^p$, and $x_{ij}^p = 0$ if $p > r_j$. With this said, formulation (1) can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in \mathcal{F}} \sum_{p \in \mathcal{R}} (f_i y_i^p + \sum_{j \in \mathcal{C}} c_{ij} x_{ij}^p) \\ \text{subjected to} \quad & \forall p \in \mathcal{R}, j \in \mathcal{C}^p : \sum_{i \in \mathcal{F}} x_{ij}^p \geq 1 \\ & \forall p \in \mathcal{R}, j \in \mathcal{C}^p, i \in \mathcal{F} : \sum_{p \in \mathcal{R}} y_i^p \geq \sum_{p \in \mathcal{R}} x_{ij}^p \\ & \forall p \in \mathcal{R}, j \in \mathcal{C}^p, i \in \mathcal{F} : x_{ij}^p, y_i^p \in \{0, 1\}. \end{aligned} \quad (2)$$

Note that the second constraint does not necessarily imply $y_i^p \geq x_{ij}^p$ for any $p \in \mathcal{R}$ which otherwise will make the problem a simple aggregation of the UFL problem. Now consider the situation that $y_i^p < x_{ij}^p$, i.e., $y_i^p = 0$ and $x_{ij}^p = 1$ for some p . Since $\sum_{p \in \mathcal{R}} y_i^p \geq \sum_{p \in \mathcal{R}} x_{ij}^p$, for any (i, j, p) with $y_i^p < x_{ij}^p$, there must exist a $q < p$ that satisfies $y_i^q - x_{ij}^q > 0$. This observation implies that a connection can be established with a facility opened at an earlier phase. Let f_i^p be the cost paid to open a facility at site i in phase p . That is,

$$f_i^p = \begin{cases} f_i & \text{if a new facility of site } i \text{ must be opened in phase } p; \\ 0 & \text{otherwise.} \end{cases}$$

As an open facility can be accessed for free, the cost paid to the site is equal to zero if no new facility has to be opened. We use $z_i^p \in \{0, 1\}$ to denote whether site i is involved in the new established connections in phase p , i.e., $z_i^p = 1$ if $\sum_{j \in \mathcal{C}^p} x_{ij}^p > 0$ and 0 otherwise. Apparently, $y_i^p = 1 \Rightarrow z_i^p = 1$ because for any i ,

$\max_{j \in \mathcal{C}} \sum_{q=1}^p x_{ij}^q > \sum_{q=1}^{p-1} y_i^q \geq \max_{j \in \mathcal{C}} \sum_{q=1}^{p-1} x_{ij}^q$ implies $\sum_{j \in \mathcal{C}} x_{ij}^p > 0$, i.e., $z_i^p = 1$. Therefore, we have $z_i^p \geq y_i^p$ for any i and p . This yields $f_i^p = f_i$ if $y_i^p < z_i^p$ and 0 otherwise. Since the object of formulation (2) carries a minimization function and $f_i^p z_i^p \geq f_i^p y_i^p \geq 0$, it is easy to see that with proper case reduction, we can equivalently transform the object to

$$\text{minimize} \sum_{p \in \mathcal{R}} \sum_{i \in \mathcal{F}} (f_i^p z_i^p + \sum_{j \in \mathcal{C}^p} c_{ij} x_{ij}^p),$$

and at the same time the second constraint to

$$z_i^p \geq x_{ij}^p, \forall p \in \mathcal{R}, j \in \mathcal{C}^p, i \in \mathcal{F}.$$

As a result, formulation (2) can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \sum_{p \in \mathcal{R}} \sum_{i \in \mathcal{F}} (f_i^p z_i^p + \sum_{j \in \mathcal{C}^p} c_{ij} x_{ij}^p) \\ \text{subjected to} \quad & \forall p \in \mathcal{R}, j \in \mathcal{C}^p : \sum_{i \in \mathcal{F}} x_{ij}^p \geq 1 \\ & \forall p \in \mathcal{R}, j \in \mathcal{C}^p, i \in \mathcal{F} : z_i^p - x_{ij}^p \geq 0 \\ & \forall p \in \mathcal{R}, j \in \mathcal{C}^p, i \in \mathcal{F} : x_{ij}^p, z_i^p \in \{0, 1\}. \end{aligned} \tag{3}$$

Noting $\sum_{p \in \mathcal{R}} z_i^p$ is not necessarily equal to y_i , we set $y_i = \max_{j \in \mathcal{C}} \sum_{p \in \mathcal{R}} x_{ij}^q$.

The LP relaxation of this formulation can be obtained by allowing x_{ij} and y_i to be nonnegative real numbers. The dual problem of this LP relaxation can be easily derived as follows:

$$\begin{aligned} \text{maximize} \quad & \sum_{p \in \mathcal{R}} \sum_{j \in \mathcal{C}^p} \alpha_j^p \\ \text{subjected to} \quad & \forall p \in \mathcal{R}, i \in \mathcal{F} : \sum_{j \in \mathcal{C}^p} \beta_{ij}^p \leq f_i \\ & \forall p \in \mathcal{R}, i \in \mathcal{F}, j \in \mathcal{C}^p : \alpha_j^p - \beta_{ij}^p \leq c_{ij} \\ & \forall p \in \mathcal{R}, i \in \mathcal{F}, j \in \mathcal{C}^p : \alpha_j^p, \beta_{ij}^p \geq 0. \end{aligned} \tag{4}$$

We use the same interpretation for dual variables, α_j^p and β_{ij}^p , as in [14, 22], which will be explained in detail later.

Lemma 1 For any feasible solution (α, β) to dual problem (4) and feasible solution (x, y) to primal problem (2), we have $\sum_{p \in \mathcal{R}} \sum_{j \in \mathcal{C}^p} \alpha_j^p \leq \sum_{i \in \mathcal{F}} (\sum_{j \in \mathcal{C}} c_{ij} x_{ij} + f_i y_i)$.

Proof We derive the inequality by applying all conditions in the constraints of the primal and dual:

$$\sum_{p \in \mathcal{R}} \sum_{j \in \mathcal{C}^p} \alpha_j^p \leq \sum_{p \in \mathcal{R}} \sum_{j \in \mathcal{C}^p} \left(\sum_{i \in \mathcal{F}} x_{ij}^p \right) \alpha_j^p$$

$$\begin{aligned}
& (\text{because } \sum_{i \in \mathcal{F}} x_{ij}^p \geq 1 \text{ (3 [c1])}) \\
& \leq \sum_{p \in \mathcal{R}} \sum_{i \in \mathcal{F}} \sum_{j \in \mathcal{C}^p} [(\alpha_j^p - \beta_{ij}^p)x_{ij}^p + \beta_{ij}^p z_i^p] \\
& (\text{because } z_i^p \geq x_{ij}^p \text{ (3 [c2])}) \\
& \leq \sum_{p \in \mathcal{R}} \sum_{i \in \mathcal{F}} \left(\sum_{j \in \mathcal{C}^p} c_{ij} x_{ij}^p + f_i^p z_i^p \right) \\
& (\text{because } \alpha_j^p - \beta_{ij}^p \leq c_{ij} \text{ (4 [c2])}, \sum_{j \in \mathcal{C}^p} \beta_{ij}^p \leq f_i \text{ (4 [c1]) and } x_{ij}^p, \\
& \quad z_i^p \geq 0 \text{ (3 [c3])}) \\
& = \sum_{i \in \mathcal{F}} \left(\sum_{j \in \mathcal{C}} c_{ij} x_{ij} + f_i y_i \right)
\end{aligned}$$

□

According to weak duality theorem, finding a feasible solution to the primal problem (3) is transformed to that to the dual problem (4), because the latter's approximation ratio (to the optimal solution of the primal) will never exceed the former's.

4 The Algorithm

We now show how to construct an effective algorithm for solving the dual problem (4). An interesting observation is that we can extract $p \in \mathcal{R}$ in the constraints of the dual problem and this even holds for the objective function because $\alpha_j^p = 0$ when $p > r_j$. We utilize this observation to design a high-level algorithm which decomposes the problem into R subproblems and process them in the order. Specifically, all ports are categorized into groups according to their port IDs, and $\mathcal{C}^p \subset \mathcal{C} = \{j \in \mathcal{C}, r_j \geq p\}$. For the sake of simplicity, let vector $\mathbf{X}^b = \sum_{p=1}^b \mathbf{x}^p$ and $\mathbf{Y}^b = \sum_{p=1}^b \mathbf{y}^p$, $1 \leq b \leq R$, our algorithm evolves the solution from the initial stage (suppose $\mathbf{X}^0 = 0$ and $\mathbf{Y}^0 = 0$), through R phases, to \mathbf{X}^R and \mathbf{Y}^R . In each phase $p \in \mathcal{R}$, the algorithm establishes one connection for each city in \mathcal{C}^p . A facility opened in one phase at site i can be used for free by any city j in a later phase, suppose p , if this usage does not violate the constraint $X_{ij}^p \leq Y_i^p$.

The process of our algorithm is presented in [Algorithm 1](#): in the p -th phase, the solution inherited from the last phase, i.e., $(\mathbf{X}^{p-1}, \mathbf{Y}^{p-1})$, as well as \mathcal{F} and \mathcal{C}^p , is used as the input of the subroutine. Note that clients with $r_j < p$ are already fully connected and therefore not included in \mathcal{C}^p . Suppose the new opened facilities and

Algorithm 1 1.861-approximation algorithm

Input: f_i, r_j, c_{ij} for any $i \in \mathcal{F}, j \in \mathcal{C}$.

Output: x_{ij}, y_i for any $i \in \mathcal{F}, j \in \mathcal{C}$.

- (1) Initially set vector $\mathbf{X}^0 \leftarrow \mathbf{0}, \mathbf{Y}^0 \leftarrow \mathbf{0}, \mathcal{C}^1 = \mathcal{C}$ and the number of current phase $p \leftarrow 1$.
 - (2) While $p \leq R$:
 - (a) Invoke **the p -th phase algorithm** with input $(\mathbf{X}^{p-1}, \mathbf{Y}^{p-1}, \mathcal{F}, \mathcal{C}^p)$, suppose the output is $(\mathbf{X}^p, \mathbf{Y}^p)$.
 - (b) Set $p \leftarrow p + 1$.
 - (3) Set $\mathbf{x} = \mathbf{X}^R$ and $\mathbf{y} = \mathbf{Y}^R$
-

new established connections are denoted by $(\mathbf{x}^p, \mathbf{y}^p)$, then in the next phase, we have $\mathbf{X}^p = \mathbf{X}^{p-1} + \mathbf{x}^p$ and $\mathbf{Y}^p = \mathbf{Y}^{p-1} + \mathbf{y}^p$ as part of the input. The algorithm ends when all R phases are finished.

In *the p -th phase algorithm*, we use the notation of star and definition of cost efficiency. A star is composed of a facility and a group of clients that are connected with the facility. Consider the time before the new star is selected, the *cost efficiency* of a star is defined to be

$$\text{eff}(i, p, C') = \frac{f_i^p + \sum_{j \in C'} c_{i,j}}{|C'|}, \quad (5)$$

where f_i^p is the cost paid to open a facility at site i in phase p and C' the set of member clients in the star. The two items in the numerator represent the total cost of the star, and therefore, the cost efficiency of a star is actually the average payment of all member clients to establish the star. Let $U \subseteq \mathcal{C}^p$ be the set of un-fully connected clients in phase p , and $C' \subseteq U$ is a set of clients chosen by the algorithm to be connected with facility i . As an open facility can be accessed for free under certain condition, the cost paid to the facility is equal to zero if no new facility has to be opened.

Now the dual variables, i.e., α_j^p and β_{ij}^p , can be used to find the most cost-efficient star. We use the same interpretation which was first used to interpret their counterparts in *UFL* as in [14, 22]: α_j^p is the total cost (including the connection cost and the contribution to open facilities) paid by the p -th port of client j , and β_{ij}^p is the contribution received by facility i from the p -th port of client j . As such, the most cost-effective star in each iteration of the subroutine can be found in this way: if the dual variables of all unconnected clients are raised simultaneously, the most cost-effective star will be the first star (i, p, C') such that

Algorithm 2 The p -th phase algorithm

-
- (1) Let $U \subseteq \mathcal{C}^p$ be the set of not fully connected cities, initially set $U \leftarrow \mathcal{C}^p$.
 - (2) While $U \neq \emptyset$:
 - (a) Find the most cost efficient star (i, p, C') according to formulation (5).
 - (b) Open a facility at site i , if it is not already open, and establish a connection to the facility for all cities in C' .
 - (c) Set $f_i \leftarrow 0$, $U \leftarrow U \setminus C'$.
-

$$\sum_{j \in C'} \max(t - c_{ij}, 0) = f_i^p,$$

where $\alpha_j^p = t$ and $\beta_{ij}^p = \max(t - c_{ij}, 0)$.

The p -th phase algorithm opens the most cost-efficient star repeatedly until all the clients in \mathcal{C}^p are *connected* with a facility. Once a client is connected, it is removed from U ; in contrast, a facility is never removed, instead it can be reused for free under certain condition. In fact, the subroutine is very close to the MMS algorithm proposed in [14, 22] for the *UFL* problem. The difference is, here, we need to ensure the feasibility of the solution by maintaining $X_{ij}^p \leq Y_i^p$ for any $i \in \mathcal{F}$, $p \in \mathcal{R}$, $j \in \mathcal{C}^p$. For the sake of simplicity, we set $y_i^p \leftarrow 1$ when a new facility at site i is opened and $x_{ij}^p \leftarrow 1$ when a new connection between client j and facility i is established. In order to maintain the feasibility of a solution, i.e., $X_{ij}^p \leq Y_i^p$, we consider three cases for any $j \in \mathcal{C}^p$:

1. $X_{ij}^{p-1} < Y_i^{p-1}$: In this case, feasibility of the solution maintains if we set $x_{ij}^p \leftarrow 1$. There is no need to open a new facility at site i and $f_i^p = 0$. We say the facility i is eligible to be connected by client j .
2. $X_{ij}^{p-1} = Y_i^{p-1}$ and $y_i^p = 0$: In this case, we must open a new facility at site i , i.e., set $y_i^p \leftarrow 1$, in order to establish a new connection for client j and therefore, $f_i^p = f_i$. The operating cost is shared between a set of clients in C' that need to connect to facility i .
3. $X_{ij}^{p-1} = Y_i^{p-1}$ and $y_i^p = 1$: In this case, a new facility has been opened by some clients before j in C' during the p -th phase, so client j can access facility i without paying any operating cost. Feasibility of the solution maintains if we set $x_{ij}^p \leftarrow 1$ and $f_i^p = 0$.

In these three cases, only the second one involves more than one client. Suppose each facility has a list of clients that are ordered according to their connection costs to the facility. As shown in Fig. 1, the most cost efficient star will consist of a facility and a set, containing the first k cities in this order, for some k . Therefore, the algorithm can be finished efficiently in polynomial time.

Here, we use three types of events to process these cases respectively. Note that a client will contribute to opening a facility only when it has accumulated more credit (increases in time) than the connection cost to an eligible open facility, and a star

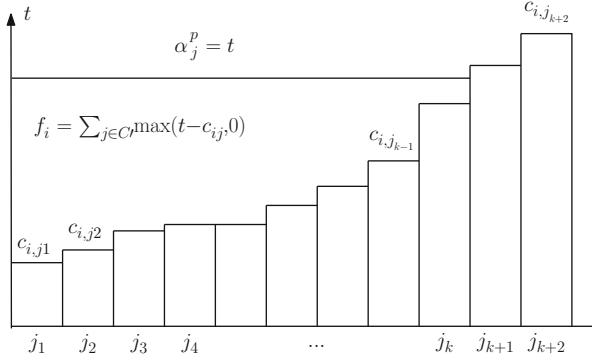


Fig. 1 Credit offers for opening a facility

Algorithm 3 Restatement of the p -th phase algorithm

- (1) At the beginning, all cities are unconnected, i.e., $t \leftarrow 0, U \leftarrow \mathcal{C}^p$. Assume city $j \in U$ has r_j ports each with some credit which increases from zero simultaneously with time before the port is connected. Set $\alpha_j^p = 0$ for any $j \notin U$.
 - (2) While $U \neq \emptyset$, increase time t until an instance of *Event-1* or *Event-2* or *Event-3* occurs. If two events occur at the same time, process them in an arbitrary order:
 - (a) *Event-1*: A city $j \in U$ has enough credit to be connected with an eligible site, suppose i , i.e., $t = c_{ij}$ and $X_{ij}^{p-1} < Y_i^{p-1}$. Set $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ in this case.
 - (b) *Event-2*: A site $i \in \mathcal{F}$ receives enough payment from cities in U to open its p -th facility, i.e., $\sum_{j \in U} \max(t - c_{ij}, 0) = f_i$. In this case, let $C' = \{j \in U : c_{ij} \leq t\}$, set $Y_i^p \leftarrow Y_i^{p-1} + 1$ and $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ for any $j \in C'$.
 - (c) *Event-3*: A city $j \in U$ has enough credit to be connected with a new opened facility, i.e., $t = c_{ij}$. Set $X_{ij}^p \leftarrow X_{ij}^{p-1} + 1$ in this case.
 - (d) For any city $j \in U$, set $\alpha_j^p \leftarrow t$ and remove city j from U if it is connected with a facility in phase p .
-

is formed only when a set of unconnected clients collectively make a contribution f_i to open a facility at site i . We restate the p -th phase algorithm based on these observations.

Remark 1 [Algorithm 1](#) is independent on the order of client set processed, e.g., we may also process clients in order $\mathcal{C}^R, \mathcal{C}^{R-1}, \dots, \mathcal{C}^1$.

5 Analysis

We assume the function of connection cost forms a metric. In order to show the performance of [Algorithm 1](#), we claim that the maximum cost ratio in each phase is bounded by a constant for any instance of the problem. Formally, let \mathcal{I} be an instance of the *FTFA* problem and $p \in \mathcal{R}$ a phase when solving the problem; we define

$$\lambda_{p,\mathcal{I}} = \max_{i \in \mathcal{F}, p \in \mathcal{R}, C' \subseteq \mathcal{C}^p} \frac{\sum_{j \in C'} \alpha_j^p}{f_i + \sum_{j \in C'} c_{ij}}$$

as the maximum cost ratio with respect to any possible star (i, p, C') .

Claim 1 The cost of the solution in each phase is equal to $\sum_{j \in \mathcal{C}_p} \alpha_j^p$, and the maximum cost ratio $\lambda_{p,\mathcal{I}}$ is bounded by a constant λ for any phase $p \in \mathcal{R}$ and any instance \mathcal{I} of the problem.

We use the inverse dual fitting technique here to analyze the approximation ratio of the high level algorithm. We do this by composing an extra instance of the problem which has the same size as the original problem but different value of facility cost and connection cost. We achieve this by scaling the facility cost and connection cost by constant λ : $f'_i \leftarrow \lambda f_i$ and $c'_{ij} \leftarrow \lambda c_{ij}$. Instead of shrinking the dual variable as in the dual-fitting [14, 18, 22] technique to achieve a feasible solution to the unscaled dual problem, we use the unshrunk dual solution which is feasible to the composed instance of the dual problem and achieve a λ -approximation ratio based on the claim. It is not hard to see that the same result can be achieved by using the dual fitting technique. However, we argue that our inverse dual fitting technique is more powerful in multifactor approximation analysis as shown in our another work [27].

Theorem 1 If the p -th phase algorithm fulfills the claim, the high level algorithm, i.e., [Algorithm 1](#), is a λ -approximation algorithm to the FTFA problem.

Proof First we check the feasibility of the solution. In the p -th phase algorithm, each phase we have $\forall i \in \mathcal{F}, j \in \mathcal{C}^p : X_{ij}^p \leq Y_i^p$. In fact, this is required by the subproblem (2) in each phase. It is not hard to see that X_{ij}^p stops increasing when $p > r_j$ because a client j is included in \mathcal{C}^p only when $p \leq r_j$ and not yet processed when $p > r_j$. Therefore, we have $\forall i \in \mathcal{F}, j \in \mathcal{C} : X_{ij}^R \leq Y_i^R$ because Y_i^p is increasing monotonously (we never close a facility). Feasibility of the solution is proved.

In order to show the cost ratio, we compose an extra instance of the problem and its feasible dual solution. Let $\beta_{ij}^p = \max(\alpha_j^p - \lambda c_{ij}, 0)$ for any $i \in \mathcal{F}, j \in \mathcal{C}, p \in \mathcal{R}$ and $C' = \{j \in \mathcal{C} : \alpha_j^p \geq \lambda c_{ij}\}$, we have $\sum_{j \in \mathcal{C}} \beta_{ij}^p = \sum_{j \in C'} \beta_{ij}^p = \sum_{j \in C'} (\alpha_j^p - \lambda c_{ij})$. According to the claim, we have

$$\sum_{j \in C'} (\alpha_j^p - \lambda c_{ij}) \leq \lambda f_i$$

for any $i \in \mathcal{F}, p \in \mathcal{R}$. That is, there exists dual variable $\beta_{ij}^p \geq 0$ such that

$$\forall p \in \mathcal{R}, i \in \mathcal{F} : \sum_{j \in \mathcal{C}} \beta_{ij}^p \leq \lambda f_i \quad (6)$$

$$\text{and } \forall p \in \mathcal{R}, i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j^p - \beta_{ij}^p \leq \lambda c_{ij}. \quad (7)$$

From the definitions of α_i and β_{ij} , apparently $\forall p \in \mathcal{R}, i \in \mathcal{F}, j \in \mathcal{C} : \alpha_j^p, \beta_{ij}^p \geq 0$.

We note that the above inequalities are exactly the constraints of the dual problem (4) after λ -factor scaling on f_i and c_{ij} . Therefore, we can compose an instance of the dual, suppose \mathcal{I}' , with facility cost $f'_i = \lambda f_i$ and connection cost $c'_{ij} = \lambda c_{ij}$. Let OPT_2 be the optimal solution to the primal problem of \mathcal{I}' , and OPT_1 the optimal solution to the primal problem of \mathcal{I} . It is clear that

$$OPT_2 = \lambda OPT_1. \quad (8)$$

From Inequalities (6) and (7), we know that (α, β) is a feasible solution to the dual problem of \mathcal{I}' . Due to the weak duality theorem, which states that the optimum of the dual problem (in the form of maximization) is no more than the optimum of the primal problem (in the form of minimization), we have

$$\sum_{j \in \mathcal{C}^p} \sum_{p \in \mathcal{R}} \alpha_j^p \leq OPT_2. \quad (9)$$

On the other hand, let SOL be the solution derived by the algorithm; we have

$$SOL = \sum_{p \in \mathcal{R}} \sum_{j \in \mathcal{C}^p} \alpha_j^p \quad (10)$$

according to the first part of the claim. Combining (8), (9), and (10), we have

$$SOL \leq \lambda OPT_1.$$

The theorem follows. \square

Remark 2 The same result can be achieved by using dual fitting, i.e., shrinking the dual solution λ times to make it “fit” to the original problem. Another interesting observation is that the greedy algorithm for the *UFL* problem, like the MMS algorithm [14, 22] or JMS algorithm [14, 18], can also be analyzed through decomposing the optimal solution into a group of stars. This is true because any solution to *UFL* can be decomposed into stars without overlap or interference. However, in the *FTFA* problem, a client is involved in multiple stars, and how to assign their costs to achieve the balance between regarding stars is a nontrivial task. Fortunately, by using the (inverse) dual-fitting technique, it provides an alternative approach to reveal the approximation ratio.

From [Algorithm 1](#), we can see that all payments are either for the connection cost or operating cost. Therefore, the first part of the claim is complied by the algorithm, and we only need to find a proper value of $\lambda \geq 1$ such that for any instance \mathcal{I} of the *FTFA* problem

$$\max_{i \in \mathcal{F}, p \in \mathcal{R}, C' \subseteq \mathcal{C}^p} \frac{\sum_{j \in C'} \alpha_j^p}{f_i + \sum_{j \in C'} c_{ij}} \leq \lambda.$$

It is clear that we only need to consider clients in $C' = \{j \in \mathcal{C}^p : \alpha_j^p \geq \lambda c_{ij}\}$ for each p . Without loss of generality, suppose there are k such clients in \mathcal{C}^p and $\alpha_1^p \leq \alpha_2^p \leq \dots \leq \alpha_k^p$. We consider some important properties of the p -th phase algorithm in order to find a proper value of λ . First, we have the following lemma on the contribution received by a facility according to Event-2 and Event-3.

Lemma 2 For any instance \mathcal{I} and phase $p \in \mathcal{R}$, $\sum_{j=h}^k \max(\alpha_h^p - c_{ij}, 0) \leq f_i$ holds for any facility $i \in \mathcal{F}$ and any client h , $1 \leq h \leq k$.

Proof Assume $\sum_{j=h}^k \max(\alpha_h^p - c_{ij}, 0) > f_i$; then a new facility at site i must be opened at the moment $t = \alpha_h^p - \epsilon$ according to Event-2 because any j with $\alpha_j^p \geq \alpha_h^p$ is still contributing to open facilities at time t . According to the assumption, there is at least one client, suppose j' , such that

$$\alpha_{j'}^p \geq \alpha_h^p, \text{ and } \alpha_{j'}^p > c_{ij'}$$

That is, $\alpha_{j'}^p > c_{ij'}$. On the other hand, j' can be connected with facility i at least at time t according to Event-3 of the algorithm which implies $\alpha_{j'}^p \leq c_{ij'}$. The contradiction directly establishes the lemma. \square

It is natural to imitate the approach proposed by Mahdian et al. [14, 22] to obtain a property regarding the triangle inequality. However, in the fault-tolerant context, this becomes more complex. In fact, we are not able to conclude that a contribution is less than the connection cost to any open facility. As shown in Fig. 2a, neither $\alpha_j^3 \leq c_{i_1 j}$ nor $\alpha_j^3 \leq c_{i_2 j}$ can be achieved even if i_1 and i_2 are opened. This is because h is already connected with facilities i_1 and i_2 before making its third contribution. Fortunately, in our algorithm, only ports of the same rank are processed in a phase, and this makes an important difference. In fact, if there are p open facilities, the p -th contribution of a client is no more than the maximum connection cost from the client to these facilities. As shown in Fig. 2b, we have $\alpha_j^3 \leq c_{i_3 j}$. Formally, we have the following lemma.

Lemma 3 For any instance \mathcal{I} and phase $p \in \mathcal{R}$, $\alpha_j^p \leq \alpha_h^p + c_{ih} + c_{ij}$ holds for any facility $i \in \mathcal{F}$ and clients h and j , $1 \leq h, j \leq k$.

Proof Assume $\alpha_j^p > \alpha_h^p$; otherwise, the lemma is obvious. Let F_1 be the set of facilities that are connected with client h at the moment when $t = \alpha_j^p - \epsilon$, we have $|F_1| = p$ because h is already connected in the p -th phase. Hence, there must exist a facility among F_1 which is not connected with j at the moment in phase p . Suppose this is a facility at site i' , we have $X_{i'j}^{p-1} < Y_{i'j}^p$. Therefore, j can be

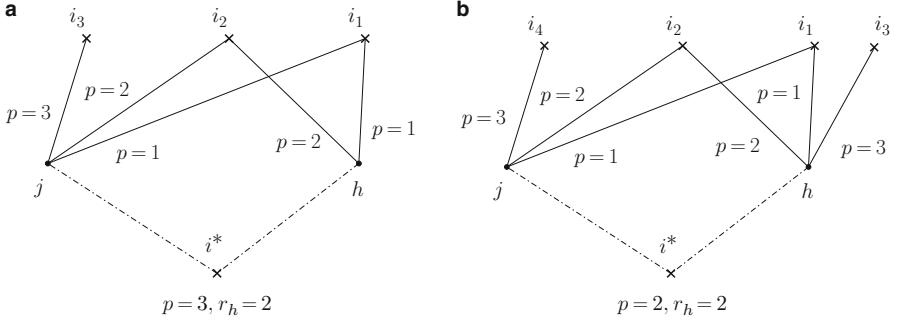


Fig. 2 Ranking of contributions. (a) $p = 3, r_h = 2$. (b) $p = 2, r_h = 2$

connected with i' without paying operating cost. Consider two cases, i.e., the facility is opened, respectively, in a previous phase and in phase p ; we have $\alpha_j^p \leq c_{i'j}$ for both cases due to Event-1 and Event-3. Further, we have $\alpha_j^p \leq c_{ij}$ if $i = i'$; otherwise, combine the triangle inequality $c_{i'j} \leq c_{i'h} + c_{ih} + c_{ih}$, and we have $\alpha_j^p \leq \alpha_h^p + c_{ij} + c_{ih}$ because $\alpha_h^p \geq c_{i'h}$ for any i' connected with client h . The lemma follows. \square

The above lemmas present some important properties of the p -th phase algorithm, and the following result turns out that they are enough to bound the ratio of the total cost of a derived solution to that of an optimal solution. Let λ_k be the maximum of the following LP:

$$\begin{aligned} & \text{maximize } \sum_{j=1}^k \alpha_j / (f + \sum_{j=1}^k d_j) \\ & \text{subjected to } \forall 1 \leq j < h \leq k : \alpha_h \leq \alpha_j + d_j + d_h \\ & \quad \forall 1 \leq h \leq k : \sum_{j=h}^k \max(\alpha_h - d_j, 0) \leq f \\ & \quad \forall 1 \leq j \leq k : \alpha_j, d_j, f \geq 0. \end{aligned} \tag{11}$$

If λ_k has an upper bound with respect to any integer k , we are able to choose this upper bound as the value of λ with respect to the claim. Formulations similar to (11) are also called factor-revealing LPs in the literature [14, 18, 21, 22].

Corollary 1 *Algorithm 1* is a 1.861-approximation algorithm for the FTFA problem.

Proof Let α_j^p be denoted by α_j , c_{ij} by d_j , and f_i by f . It is clear that Lemma 1 and 2 imply the two constraints of formulation (11). As a result,

$$\lambda \leq \sup_{k \geq 1} \{\lambda_k\}.$$

In fact, Mahdian et al. [14, 22] show that formulation (11) has an upper bound 1.861 in their analysis for the MMS algorithm. Combined with Theorem 1, the corollary follows. \square

In each phase, there are at most $n_f \cdot |\mathcal{C}^P|$ events for which the algorithm needs $n_f \cdot |\mathcal{C}^P| \log(n_f \cdot |\mathcal{C}^P|)$ time to sort these events. Considering that the algorithm runs R phases and in each phase $|\mathcal{C}^P| \leq |\mathcal{C}|$, we have the following lemma.

Lemma 4 *Time complexity of Algorithm 1 is $O(mR \log m)$, where $m = n_c n_f$.*

6 Conclusion

We have overviewed some of our recent work on the Fault-Tolerant Facility Allocation (FTFA) problem [27] which generalizes the classical facility location (FL) problem and relaxes the more challenging problem of Fault-Tolerant Facility Allocation (FTFL). For n_f sites and n_c clients with maximum number of connections R , our FTFA algorithm achieves an approximation ratio of 1.861 in time $O(mR \log m)$, where $m = n_f n_c$. Instead of restricting only one facility provided at each site as in FTFL, FTFA allows a site to hold multiple (identical) facilities for parallel access from a client. This difference makes FTFA suitable for numerous emerging applications such as content delivery networks, cloud service (virtual machine) delivery, fault-tolerant network design, and fuzzy clustering.

Suppose the downtime of facilities or links is predictable. For example, in a system where each facility needs a fraction of time to rest, the downtime ratio (percent of time to rest) is uniformly b for all facilities, and the usability required by client j is b_j (percent of time that a service is operating), and then the regarding network design problem can be modeled as an FTFA problem with r_j set as $\lceil b_j / (1 - b) \rceil$. If the downtime is unpredictable, then r_j should be set to $\lceil \log_b(1 - b_j) \rceil$. In both cases, we can apply the proposed algorithm to solve the network design problem directly. However, if facilities or links have a nonuniform downtime, the constraints on connectivity become $\sum_{i \in \mathcal{F}} (1 - b_{ij})x_{ij} \geq b_j$ for the deterministic model and $\prod_{i \in \mathcal{F}: x_{ij}=1} b_{ij} \leq 1 - b_j$ for the stochastic model. We leave these problems for future research. FTFA can also be extended into *Fault-Tolerant k-Facility Allocation* which has an extra constraint on the maximum number of open facilities as shown in our other work [27].

In our other study [27] it was shown that using factor-revealing technique we can achieve an approximation ratio of 1.61 in time $O(Rn^3)$, where $n = \max\{n_f, n_c\}$, and this ratio can be further improved to 1.52 derivable from bi-factor approximation with the help of scaling and greedy augmentation.

Recommended Reading

1. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, V. Pandit, Local search heuristic for k -median and facility location problems, in *STOC '01: Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing* (ACM, New York, 2001), pp. 21–29
2. J. Byrka, K.I. Aardal, An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem [J]. *SIAM J. Comput.* **39**(6), 2212–2231 (2010)

3. M. Charikar, S. Guha, Improved combinatorial algorithms for the facility location and k -median problems, in *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, Washington, DC, USA (IEEE Computer Society, 1999), pp. 378
4. M. Charikar, S. Guha, Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.* **34**(4), 803–824 (2005)
5. M. Charikar, S. Guha, E. Tardos, D.B. Shmoys, A constant-factor approximation algorithm for the k -median problem, in *ACM Symposium on Theory of Computing*, 1999, pp. 1–10
6. F.A. Chudak, D.B. Shmoys, Improved approximation algorithms for the uncapacitated facility location problem. *SIAM J. Comput.* **33**(1), 1–25 (2004)
7. F.A. Chudak, D.P. Williamson, Integer programming and combinatorial optimization, in *Improved Approximation Algorithms for Capacitated Facility Location Problems*. Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 1999), pp. 99–113
8. G. Cornuejols, M.L. Fisher, G.L. Nemhauser, Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Manage. Sci.* **23**(8), 789–810 (1977)
9. R.L. Francis, P.B. Mirchandani (eds.), *Discrete Location Theory* (Wiley, New York, 1990)
10. S. Guha, S. Khuller, Greedy strikes back: improved facility location algorithms. *J. Algorithms* **31**(2), 228–248 (1999)
11. S. Guha, A. Meyerson, K. Munagala, Improved algorithms for fault tolerant facility location, in *SODA '01: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, USA (Society for Industrial and Applied Mathematics, 2001), pp. 636–641
12. S. Guha, A. Meyerson, K. Munagala, A constant factor approximation algorithm for the fault-tolerant facility location problem. *J. Algorithms* **48**(2), 429–440 (2003)
13. D.S. Hochbaum, Heuristics for the fixed cost median problem. *Math. Program.* **22**(1), 148–162 (1982)
14. K. Jain, M. Mahdian, E. Markakis, A. Saberi, V.V. Vazirani, Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *J. ACM* **50**(6), 795–824 (2003)
15. K. Jain, V.V. Vazirani, Primal-dual approximation algorithms for metric facility location and k -median problems, in *IEEE Symposium on Foundations of Computer Science* (IEEE Computer Society, Los Alamitos, 1999), pp. 2–13
16. K. Jain, V.V. Vazirani, An approximation algorithm for the fault tolerant metric facility location problem, in *APPROX '00: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, London, UK (Springer, 2000), pp. 177–183
17. K. Jain, V.V. Vazirani, Approximation algorithms for metric facility location and k -median problems using the primal-dual schema and Lagrangian relaxation. *J. ACM* **48**(2), 274–296 (2001)
18. K. Jain, M. Mahdian, A. Saberi, A new greedy approach for facility location problems, in *STOC '02: Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, New York, NY, USA (ACM, 2002), pp. 731–740
19. J.-H. Lin, J.S. Vitter, e -approximations with minimum packing constraint violation, in *STOC '92: Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, New York, NY, USA (ACM, 1992), pp. 771–782
20. M. Mahdian, Y. Ye, J. Zhang, Approximation algorithms for metric facility location problems, *SIAM J. Comput.* **36**(2), 411–432 (2006)
21. R. McEliece, E. Rodemich, H. Rumsey, L. Welch, New upper bounds on the rate of a code via the delsarte-macwilliams inequalities. *IEEE Trans. Inf. Theory* **23**(2), 157–166 (1977)
22. M. Mahdian, E. Markakis, A. Saberi, V. Vazirani, A greedy facility location algorithm analyzed using dual-fitting, in *Proc. of 4th International Wrokshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, Berkeley, CA, USA (Springer, 2001), pp. 127–137
23. D.B. Shmoys, E. Tardos, K. Aardal, Approximation algorithms for facility location problems, in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, El Paso, Texas (ACM press), pp. 265–274

24. M. Sviridenko, An improved approximation algorithm for the metric uncapacitated facility location problem, in *Proceedings of the 9th International IPCO Conference on Integer Programming and Combinatorial Optimization*, London, UK (Springer, 2002), pp. 240–257
25. C. Swamy, D.B. Shmoys, Fault-tolerant facility location. ACM Trans. Algorithms **4**(4), 1–27 (2008)
26. V.V. Vazirani, *Approximation Algorithms* (Springer, Berlin, 2001)
27. S. Xu, H. Shen, The Fault-Tolerant Facility Allocation Problem, in *Proc. of 20th International Symposium*, Honolulu, Hawaii, USA (Springer, 2009), pp. 689–698

Fractional Combinatorial Optimization

Tomasz Radzik

Contents

1	Introduction	1312
2	Fractional Combinatorial Optimization: The General Case	1313
3	The Newton Method	1318
4	The Newton Method for the Linear Case	1321
5	Megiddo's Parametric Search	1328
6	Maximum Profit-to-Time Ratio Cycles	1336
7	Maximum-Mean Cycles	1338
8	Maximum Mean-Weight Cuts	1340
9	Conclusion	1349
	Cross-References	1352
	Recommended Reading	1353

Abstract

This chapter considers combinatorial optimization problems with objective functions in the form of ratios of two functions. A parametric approach to such problems is described and two main general algorithmic methods—the Newton method and Megiddo's parametric search—are explained, using in both cases the same example of computing maximum-ratio paths in acyclic graphs. Some general properties of these methods are presented, including a proof of a strongly polynomial bound on the number of iterations of the Newton method and a proof of the speedup of Megiddo's parametric search obtained by using parallel algorithms. The power of these methods is further shown in detail analyses of an algorithm for the maximum mean-weight cut problem based on the Newton method and an algorithm for the maximum profit-to-time ratio cycle problem based on Megiddo's parametric search.

T. Radzik

Department of Informatics, Kings College London, London, UK
e-mail: tomasz.radzik@kcl.ac.uk

1 Introduction

An instance of a *fractional combinatorial optimization* problem \mathcal{F} consists of a specification of a set $\mathcal{X} \subseteq \{0, 1\}^p$ and two functions $f : \mathcal{X} \rightarrow \mathbf{R}$ and $g : \mathcal{X} \rightarrow \mathbf{R}$. The objective is to

$$\mathcal{F} : \quad \text{maximize} \frac{f(\mathbf{x})}{g(\mathbf{x})} \quad \text{for } \mathbf{x} \in \mathcal{X}. \quad (1)$$

It is assumed that $f(\mathbf{x}) > 0$ for some $\mathbf{x} \in \mathcal{X}$ and $g(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathcal{X}$. The elements of the discrete domain \mathcal{X} are often called *structures*, since in concrete fractional combinatorial optimization problems, they represent combinatorial structures such as cycles, spanning trees, or other types of subgraphs. Examples of fractional combinatorial optimization include the minimum-ratio spanning-tree problem [7], the weighted maximum-mean subtree problem [5], the maximum profit-to-time cycle problem [14, 21], the maximum-mean cycle problem [15–17, 34, 38], the maximum mean-weight cut problem [50], and the fractional 0–1 knapsack problem [31]. A problem of minimizing/maximizing a fractional objective $f(\mathbf{x})/g(\mathbf{x})$ can be often solved by considering the equivalent problem of maximizing/minimizing fractional objective $(-f(\mathbf{x}))/g(\mathbf{x})$, or $(Bg(\mathbf{x}) - f(\mathbf{x}))/g(\mathbf{x})$, for a suitably large number B , if it is important to keep the numerator positive.

Fractional combinatorial optimization is a special case of (general) fractional optimization: maximize $f(\mathbf{x})/g(\mathbf{x})$ over a subset D of \mathbf{R}^n . Fractional optimization problems have been extensively studied, and many algorithms have been designed and analyzed (see, e.g., surveys [55–58, 60]). Computational methods designed for general fractional optimization can usually be adapted to fractional combinatorial optimization, but the analysis of a method for the combinatorial case may be considerably different from the analysis of the same method for the general case. For example, when general fractional optimization is considered, it is often assumed that the domain D is a compact subset of \mathbf{R}^n and functions f and g are continuous. Such assumptions clearly do not have equivalent counterparts in fractional combinatorial optimization. Computational methods for general fractional optimization converge to the optimum objective value, but they might not guarantee that this value is actually reached. In most cases, the corresponding methods for fractional combinatorial optimization do produce optimal solutions. Note that the domain \mathcal{X} is finite, so an optimal solution can be found in finite time by simply checking all structures $\mathbf{x} \in \mathcal{X}$.

In the context of optimization problems, the term “fractional” is often used to indicate nonintegral solutions. Therefore, to avoid ambiguity, “fractional optimization” is sometimes referred to using a more elaborate phrase, for example, “optimization with rational objective functions.” In this chapter, however, “fractional” consistently refers to the form of the objective function.

This chapter focuses on two main methods for fractional combinatorial optimization, the *Newton method* and *Megiddo’s parametric search method*. Both these methods, as well as most of the existing methods for general fractional optimization, are based on a reduction from fractional optimization to non-fractional, parametric optimization. The Newton method was introduced for general fractional

optimization by Dinkelbach [18], and there are a number of results concerning the fast convergence of this method and its variants (e.g., [30, 47, 54]). The analyses of the Newton method for fractional combinatorial optimization presented in this chapter are mostly based on [48, 50]. Megiddo's parametric search method [41, 42] was designed for the case of fractional combinatorial optimization when both functions f and g are linear. This method gave, for example, the first *strongly polynomial* algorithm for the maximum profit-to-time cycle problem. A strongly polynomial bound related to problem \mathcal{F} is a bound which depends polynomially on p and does not depend on any other input parameters.

This chapter is organized in the following way. [Section 2](#) describes basic properties of fractional combinatorial optimization, the reduction from fractional optimization to parametric optimization, and the binary search method. This section also introduces the maximum-ratio path problem for acyclic graphs, which is used later in the chapter to illustrate both the Newton method and Megiddo's parametric search. [Section 3](#) introduces the Newton method for fractional combinatorial optimization and presents some general results about its convergence. [Section 4](#) shows that if both functions f and g of problem \mathcal{F} are linear, then the Newton method finds an optimal solution in a strongly polynomial number of iterations, regardless of the structure of the domain \mathcal{X} . [Section 5](#) discusses in depth Megiddo's parametric search method. [Sections 6](#) and [8](#) contain two case studies. [Section 6](#) shows how Megiddo's parametric search leads to fast algorithms for the maximum profit-to-time ratio cycle problem. [Section 8](#) presents an analysis of the Newton method for the maximum mean-weight cut problem, which gives the best-known bound on the computational complexity of this problem. Most of the fastest known algorithms for fractional combinatorial optimization problems are based on either the Newton method, or Megiddo's parametric search, or the basic binary search method. The most notable exception to this rule is Karp's algorithm for the maximum-mean cycle problem [34]. This algorithm and its analysis are presented in [Sect. 7](#). [Section 9](#) contains concluding remarks, mentions other approaches, and gives suggestions for further reading.

2 Fractional Combinatorial Optimization: The General Case

For $\mathbf{x} \in \mathcal{X}$, numbers $f(\mathbf{x})$, $g(\mathbf{x})$, and $f(\mathbf{x})/g(\mathbf{x})$ are called the *cost*, the *weight*, and the *mean-weight cost* of structure \mathbf{x} . Using this terminology, problem \mathcal{F} is to compute the maximum mean-weight cost of a structure in domain \mathcal{X} and to find a structure which achieves this maximum mean-weight cost.

Consider the following problem:

$$\mathcal{P} : \quad \text{minimize } \delta \in \mathbf{R}, \quad \text{subject to } f(\mathbf{x}) - \delta g(\mathbf{x}) \leq 0, \quad \text{for all } \mathbf{x} \in \mathcal{X}.$$

A pair $(\delta^*, \mathbf{x}^*) \in \mathbf{R} \times \mathcal{X}$ is an optimal solution of problem \mathcal{P} , if and only if,

$$f(\mathbf{x}) - \delta^* g(\mathbf{x}) \leq 0 = f(\mathbf{x}^*) - \delta^* g(\mathbf{x}^*), \quad \text{for each } \mathbf{x} \in \mathcal{X}.$$

This condition is equivalent to

$$f(\mathbf{x})/g(\mathbf{x}) \leq \delta^* = f(\mathbf{x}^*)/g(\mathbf{x}^*), \quad \text{for each } \mathbf{x} \in \mathcal{X},$$

which means that δ^* is the optimum objective value and \mathbf{x}^* is an optimal solution of problem \mathcal{F} . Thus, problems \mathcal{F} and \mathcal{P} are equivalent. Many iterative methods for solving problem \mathcal{F} generate and solve a sequence of instances of the following problem, where $\delta \in \mathbf{R}$ is an additional input parameter.

$$\mathcal{P}(\delta) : \quad \text{maximize } f(\mathbf{x}) - \delta g(\mathbf{x}), \quad \text{for } \mathbf{x} \in \mathcal{X}. \quad (2)$$

Problem $\mathcal{P}(\delta)$ is called the *parametric problem* corresponding to problem \mathcal{F} and sometimes also the *non-fractional version* of problem \mathcal{F} . Let $h(\delta)$ and \mathbf{x}_δ^* denote the optimum objective value and an optimal solution of problem $\mathcal{P}(\delta)$. Then, $h(\delta) = 0$, if and only if, $(\delta, \mathbf{x}_\delta^*)$ is an optimal solution of problem \mathcal{P} , that is, if and only if, δ and \mathbf{x}_δ^* are the optimum objective value and an optimal solution of problem \mathcal{F} . This gives another equivalent formulation \mathcal{Z} of problem \mathcal{F} .

\mathcal{Z} : find $\delta \in \mathbf{R}$ such that $h(\delta) = 0$, where

$$h(\delta) = \max\{f(\mathbf{x}) - \delta g(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}. \quad (3)$$

This formulation suggests that one can try to design algorithms for problem \mathcal{F} by applying classical methods for finding a root of a function. The following properties of function h can be easily obtained from the fact that h is the maximum of a finite number of decreasing linear functions:

1. Function h is continuous on $(-\infty, +\infty)$ and strictly decreasing from $+\infty$ to $-\infty$.
2. $h(0) > 0$ (this follows from the assumption that $f(\mathbf{x}) > 0$, for some $\mathbf{x} \in \mathcal{X}$).
3. Function h has exactly one root δ^* , and $\delta^* > 0$.
4. If $\delta_1 < \delta_2 < \dots < \delta_q$ denote all values of δ for which two lines in $\{f(\mathbf{x}) - \delta g(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$ intersect, then function h is linear on each interval $[-\infty, \delta_1]$ and $[\delta_i, \delta_{i+1}]$, for $i = 1, 2, \dots, q-1$, and $[\delta_q, \infty]$.
5. Function h is convex.

Some methods for solving problem \mathcal{F} require a subroutine only for the following weaker version of problem $\mathcal{P}(\delta)$.

$$\begin{aligned} \mathcal{P}_0(\delta) : & \text{ find } \mathbf{y} \in \mathcal{X} \text{ such that} \\ & \text{sign}(f(\mathbf{y}) - \delta g(\mathbf{y})) = \text{sign}(\max\{f(\mathbf{x}) - \delta g(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}). \end{aligned}$$

That is, problem $\mathcal{P}_0(\delta)$ is only to find the sign of the objective value of problem $\mathcal{P}(\delta)$. Further discussion of the properties of computational methods for problem \mathcal{F} refers to the following three parameters:

$$\text{MAX}_f = \max\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\},$$

$$\begin{aligned}\text{MAX}_g &= \max\{g(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}, \\ \text{MIN}_g &= \min\{g(\mathbf{x}) : \mathbf{x} \in \mathcal{X}\}, \\ \text{GAP} &= \min \left\{ \left| \frac{f(\mathbf{x}')}{g(\mathbf{x}')} - \frac{f(\mathbf{x}'')}{g(\mathbf{x}'')} \right| : \mathbf{x}', \mathbf{x}'' \in \mathcal{X}, \text{ and } \frac{f(\mathbf{x}')}{g(\mathbf{x}')} \neq \frac{f(\mathbf{x}'')}{g(\mathbf{x}'')} \right\}.\end{aligned}$$

The assumptions on functions f and g imply that the optimum objective value of problem \mathcal{F} is contained in interval $(0, \text{MAX}_f/\text{MIN}_g]$. Parameter GAP is the smallest difference between two different mean-weight costs.

If the weight function g counts the number of ones, then problem \mathcal{F} is called a *uniform* fractional combinatorial optimization problem:

$$\mathcal{F}_U : \text{maximize } \frac{f(x_1, x_2, \dots, x_p)}{x_1 + x_2 + \dots + x_p}, \quad \text{for } (x_1, x_2, \dots, x_p) \in \mathcal{X}.$$

If both the cost function f and the weight function g are linear, then problem \mathcal{F} is called a *linear* fractional combinatorial optimization problem:

$$\begin{aligned}\mathcal{F}_L : \text{maximize } & \frac{a_1 x_1 + a_2 x_2 + \dots + a_p x_p}{b_1 x_1 + b_2 x_2 + \dots + b_p x_p} \\ \text{subject to } & (x_1, x_2, \dots, x_p) \in \mathcal{X}.\end{aligned}$$

An instance of problem \mathcal{F}_L consists of a specification of a set of structures $\mathcal{X} \subseteq \{0, 1\}^p$ and two real vectors $\mathbf{a} = (a_1, a_2, \dots, a_p)$ and $\mathbf{b} = (b_1, b_2, \dots, b_p)$. Throughout this chapter, the inner product $c_1 z_1 + c_2 z_2 + \dots + c_p z_p$ of two vectors $\mathbf{c} = (c_1, c_2, \dots, c_p)$ and $\mathbf{z} = (z_1, z_2, \dots, z_p)$ is denoted by $\mathbf{c}\mathbf{z}$. Thus in problem \mathcal{F}_L , the cost, the weight, and the mean-weight cost of a structure \mathbf{x} are equal to \mathbf{ax} , \mathbf{bx} , and $(\mathbf{ax})/(\mathbf{bx})$, respectively.

Maximum-Ratio Paths in Acyclic Graphs

The algorithmic methods for fractional combinatorial optimization which are considered in this chapter will be illustrated using as an example the following MaxRatioPath problem. An input instance of this problem consists of an integer $n \geq 1$, a set of edges E of an acyclic-directed graph with n vertices $1, 2, \dots, n$, an edge-cost function $\mathbf{c} : E \rightarrow \mathbf{R}$, and an edge-weight function $\mathbf{w} : E \rightarrow \mathbf{R}$. To keep this example simple, the following assumptions are made: there are no multiple edges, the vertices are numbered according to a topological sort of the graph (e.g., if $(v, u) \in E$, then $v < u$), and each vertex is reachable from vertex 1. For a path P , let $f(P) = \sum_{(v,u) \in P} \mathbf{c}(v, u)$, $g(P) = \sum_{(v,u) \in P} \mathbf{w}(v, u)$, and $f(P)/g(P)$ be the cost, the weight, and the mean-weight cost of this path. The task is to find a path from vertex 1 to vertex n which has the maximum mean-weight cost.

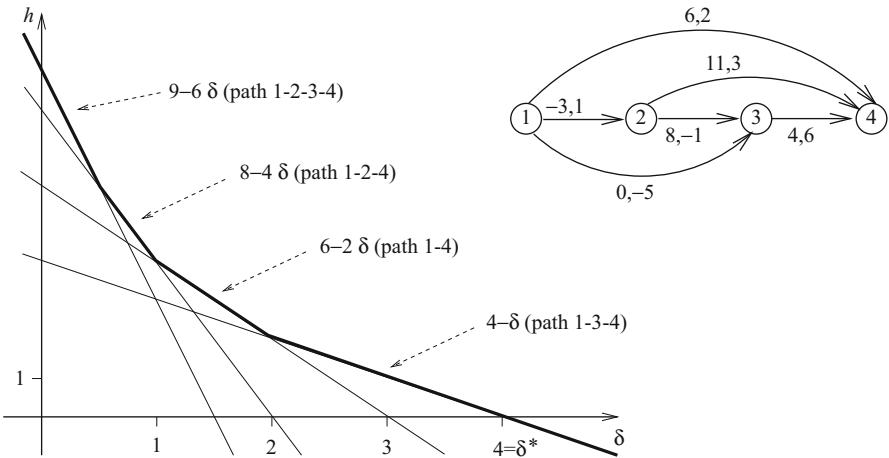


Fig. 1 An input for the MaxRatioPath problem and the corresponding function $h(\delta)$. Lines $9 - 6\delta$, $8 - 4\delta$, $6 - 2\delta$, and $4 - \delta$ correspond to the four paths from vertex 1 to vertex 4. The optimal path is 1-3-4

$$\text{MaxRatioPath} : \underset{\text{over all paths } P \subseteq E \text{ from vertex 1 to vertex } n.}{\text{maximize}} \frac{f(P)}{g(P)},$$

This is a linear fractional combinatorial optimization problem. In the terminology of the general fractional combinatorial optimization, the set of structures \mathcal{X} is here the set of characteristic vectors $\mathbf{x} \in \{0, 1\}^E$ corresponding to paths from vertex 1 to vertex n .

The parametric problem corresponding to the MaxRatioPath problem is

$$\text{MaxPath}(\delta) : \underset{\text{over all paths } P \subseteq E \text{ from vertex 1 to vertex } n.}{\text{maximize}} f(P) - \delta g(P),$$

For a fixed $\delta \in \mathbf{R}$ and a path P , $f(P) - \delta g(P) = \sum_{e \in P} (\mathbf{c}(e) - \delta \mathbf{w}(e))$, so the optimum value of problem MaxPath(δ) is the maximum cost of a path from vertex 1 to vertex n according to the edge-cost function $\mathbf{c} - \delta \mathbf{w}$. An example of an input instance of the MaxRatioPath problem and the corresponding function h are shown in Fig. 1.

In an acyclic graph, the shortest paths, as well as the longest ones, can be computed by considering the vertices according to a topological order [12, Chapter 25]. Algorithm MAXCOST described below computes the maximum cost of a path from vertex 1 to vertex n for a set of edges E and an arbitrary edge-cost function \mathbf{a} by considering the vertices from 1 to $n - 1$. While considering vertex v , the algorithm examines all edges outgoing from v . During the computation, the Boolean variable $seen[u]$ indicates whether vertex u has already been encountered.

At the end of the computation, number $d[v]$, for each $v = 1, 2, \dots, n$, is equal to the maximum cost of a path from vertex 1 to vertex v . The *predecessor* pointers form a “longest-paths” tree rooted at vertex 1.

MAXCOST(n, E, \mathbf{a})

- 1) $d[1] \leftarrow 0; seen[1] \leftarrow \text{true}$
- 2) **for** $v \leftarrow 2$ **to** n **do** $seen[v] \leftarrow \text{false}$
- 3) **for** $v \leftarrow 1$ **to** $n - 1$ **do**
- 4) **for** each u such that $(v, u) \in E$ **do**
- 5) **if** (not $seen[u]$) or $(d[v] + \mathbf{a}(v, u) - d[u] > 0)$ **then**
- 6) $d[u] \leftarrow d[v] + \mathbf{a}(v, u); predecessor[u] \leftarrow v; seen[u] \leftarrow \text{true}$
- 7) let $P = (v_1, v_2, \dots, v_k)$ where $v_1 = 1, v_k = n$, and
 $v_i = predecessor[v_{i+1}]$, for $i = 1, 2, \dots, k - 1$
- 8) **return** $d[n]$ and path P

The running time of algorithm MAXCOST is clearly $O(m)$. Thus, for each $\delta \in \mathbf{R}$, problem MaxPath(δ) can be solved in $O(m)$ time. The MaxRatioPath problem will be used later in this chapter to illustrate both the Newton method and Megiddo’s parametric search. Some details of algorithm MAXCOST may look at this point somewhat awkward, but this presentation will make it simpler to explain Megiddo’s parametric search method in Sect. 5.

The Binary Search Method

To apply the binary search method to the problem of finding the root of function h defined in (3), an algorithm \mathcal{A}_0 which for a given $\delta \in \mathbf{R}$ solves problem $\mathcal{P}_0(\delta)$ is needed. During the binary search, an interval (α, β) containing the root δ^* of function h and a structure \mathbf{x}_α such that $f(\mathbf{x}_\alpha) - \alpha g(\mathbf{x}_\alpha) > 0$ are maintained. Structure \mathbf{x}_α is a “witness” that $\delta^* > \alpha$. During one iteration of the binary search, algorithm \mathcal{A}_0 is run for the value $\delta = (\alpha + \beta)/2$. If it returns a structure \mathbf{x} such that $f(\mathbf{x}) - \delta g(\mathbf{x}) = 0$, then \mathbf{x} is an optimal solution of problem \mathcal{F} , and the computation terminates. If the returned structure \mathbf{x} is such that $f(\mathbf{x}) - \delta g(\mathbf{x}) > 0$, then δ^* must be in the interval (δ, β) , so α is set to δ and \mathbf{x}_α is set to \mathbf{x} . Otherwise δ^* is in the interval (α, δ) , so β is set to δ .

The computation proceeds until an optimal solution has been found or the desired precision has been reached. At the end of the current iteration, $\alpha < f(\mathbf{x}_\alpha)/g(\mathbf{x}_\alpha) \leq \delta^* < \beta$, so $\delta^* - f(\mathbf{x}_\alpha)/g(\mathbf{x}_\alpha) < \beta - \alpha$. Therefore, after $\lceil \log((\beta_0 - \alpha_0)/\epsilon) \rceil$ iterations, $\delta^* - f(\mathbf{x}_\alpha)/g(\mathbf{x}_\alpha) < \epsilon$, where (α_0, β_0) is the initial interval (α, β) . Throughout this chapter, the base of logarithms is 2. If the computation is continued for a sufficient number of iterations, it actually finds an optimal solution of problem \mathcal{F} . Observe that when $\delta^* - f(\mathbf{x}_\alpha)/g(\mathbf{x}_\alpha)$, which is equal to $f(\mathbf{x}^*)/g(\mathbf{x}^*) - f(\mathbf{x})/g(\mathbf{x})$ for an optimal structure \mathbf{x}^* , becomes eventually less than GAP, then $f(\mathbf{x}_\alpha)/g(\mathbf{x}_\alpha) = \delta^*$, so \mathbf{x}_α is an optimal structure. This means that the binary search finds an optimal solution of problem \mathcal{F} in at most $\lceil \log((\beta_0 - \alpha_0)/\text{GAP}) \rceil$ iterations. The initial interval (α_0, β_0) can be $(0, U/L)$, where U and L are an upper bound on MAX_f and a positive lower bound on MIN_g , respectively. For concrete fractional combinatorial optimization problems, such bounds are usually readily available. The binary search procedure

described here can be modified in a straightforward way to work with a somewhat weaker algorithm \mathcal{A}_0 , which finds only a structure \mathbf{x} with nonnegative $f(\mathbf{x}) - \delta g(\mathbf{x})$, provided that such a structure exists.

As an example, consider the special case of the MaxRatioPath problem when all edge costs and weights are integers not greater than U . For this problem, $\text{MAX}_f \leq nU$, $\text{MIN}_g \geq 1$, and $\text{GAP} \geq 1/(nU)^2$, so the binary search method gives an $O(m \log(nU))$ -time algorithm.

3 The Newton Method

The Newton method for a fractional combinatorial optimization problem \mathcal{F} , called also the Dinkelbach method [18], is an application of the classical Newton method to the problem of finding the root of function h defined in (3). To use this method, a stronger algorithm \mathcal{A}_{\max} is needed. For a given $\delta \in \mathbf{R}$, algorithm \mathcal{A}_{\max} computes $h(\delta)$ and a structure $\mathbf{x} \in \mathcal{X}$ such that $f(\mathbf{x}) - \delta g(\mathbf{x}) = h(\delta)$. That is, algorithm \mathcal{A}_{\max} solves problem $\mathcal{P}(\delta)$ for a given $\delta \in \mathbf{R}$.

The Newton method for problem \mathcal{F} is an iterative process which in each iteration generates a new, better lower estimate on the optimum objective value δ^* . During iteration i , the current estimate $\delta_i \leq \delta^*$ is considered and the following computation is performed. Algorithm \mathcal{A}_{\max} is executed for $\delta = \delta_i$ to obtain $h_i = h(\delta_i)$ and a structure $\mathbf{x}_i \in \mathcal{X}$ such that $f(\mathbf{x}_i) - \delta_i g(\mathbf{x}_i) = h_i$. If $h_i = 0$, then $\delta_i = \delta^*$ and \mathbf{x}_i is an optimal solution of problem \mathcal{F} , so the computation terminates. Otherwise a new, better estimate $\delta_{i+1} \leftarrow f(\mathbf{x}_i)/g(\mathbf{x}_i)$ is derived (observe that $\delta_i < \delta_{i+1} \leq \delta^*$, since in this case $h_i > 0$) and the algorithm proceeds to the next iteration. This process is illustrated in Fig. 2. The computation begins with $\delta_1 = 0$.

The aim is to derive bounds on the number of iterations. Let t be the index of the last iteration, or $+\infty$, if the computation does not terminate, and let $f_i = f(\mathbf{x}_i)$ and $g_i = g(\mathbf{x}_i)$, for each $1 \leq i < t + 1$. From the description of the computation,

$$h_i = f_i - \delta_i g_i, \quad \text{for each } 1 \leq i < t + 1, \quad (4)$$

$$\delta_{i+1} = \frac{f_i}{g_i}, \quad \text{for each } 1 \leq i < t. \quad (5)$$

Lemma 1 *The Newton method terminates in a finite number of iterations and*

- (a) $h_1 > h_2 > \dots > h_{t-1} > h_t = 0$,
- (b) $0 = \delta_1 < \delta_2 < \dots < \delta_{t-1} < \delta_t = \delta^*$,
- (c) $g_1 > g_2 > \dots > g_{t-1} \geq g_t$.

Proof This lemma follows immediately from the following claim. For each $1 \leq i < t + 1$,

- (a) $h_1 > h_2 > \dots > h_{i-1} > h_i \geq 0$,
- (b) $0 = \delta_1 < \delta_2 < \dots < \delta_{i-1} < \delta_i \leq \delta^*$,
- (c) $g_1 > g_2 > \dots > g_{i-1} \geq g_i$,

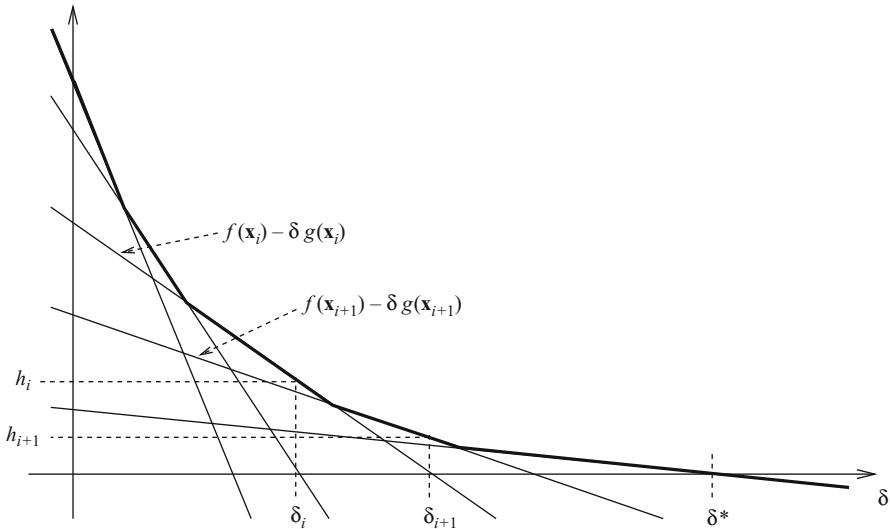


Fig. 2 The Newton method for solving $h(\delta) = 0$

and the equalities at the end of these three chains are possible only if iteration i is the last iteration in the whole computation. This claim can be proven by induction on i . For $i = 1$ the claim follows from the properties (ii) and (iii) of function h . Assume now that the claim is true for $i = j$, for some $1 \leq j < t$. Using (4) and (5),

$$\delta^* \geq \frac{f_j}{g_j} = \delta_{j+1} \quad \text{and} \quad \delta_{j+1} = \frac{f_j}{g_j} = \frac{h_j}{g_j} + \delta_j > \delta_j,$$

so (b) is true for $i = j + 1$. Since $\delta_j < \delta_{j+1} \leq \delta^*$, the monotonicity of function h implies that $h_{j+1} = h(\delta_{j+1}) < h(\delta_j) = h_j$ and $h_{j+1} = h(\delta_{j+1}) \geq h(\delta^*) = 0$, so (a) is true for $i = j + 1$. To show that also (c) is true for $i = j + 1$, the following two inequalities are used:

$$f_{j+1} - \delta_j g_{j+1} \leq \max_{\mathbf{x} \in \mathcal{X}} \{f(\mathbf{x}) - \delta_j g(\mathbf{x})\} = h_j = f_j - \delta_j g_j,$$

$$f_{j+1} - \delta_{j+1} g_{j+1} = h_{j+1} = \max_{\mathbf{x} \in \mathcal{X}} \{f(\mathbf{x}) - \delta_{j+1} g(\mathbf{x})\} \geq f_j - \delta_{j+1} g_j.$$

Subtracting these two inequalities gives $g_{j+1} \leq g_j$, and the equality is possible only if $h_{j+1} = f_j - \delta_{j+1} g_j$. Since $f_j - \delta_{j+1} g_j$ is by definition equal to 0 [see (c)], $g_{j+1} = g_j$ only if $h_{j+1} = 0$, that is, only if iteration $j + 1$ is the last iteration of the whole computation. Hence, (3) is true for $i = j + 1$. This concludes the proof of the claim.

The claim implies that all elements of sequence $(g_i)_{1 \leq i < t}$ are distinct, so t must be finite (numbers g_i are weights of structures from \mathcal{X} , so $t - 1 \leq |\mathcal{X}| < +\infty$). \square

The following lemma indicates that sequences $(h_i)_{1 \leq i \leq t}$ and $(g_i)_{1 \leq i \leq t}$ should geometrically decrease to zero. This lemma is the basic tool in all analyses of the Newton method presented in this chapter.

Lemma 2 *For each $i = 1, 2, \dots, t - 1$,*

$$\frac{h_{i+1}}{h_i} + \frac{g_{i+1}}{g_i} \leq 1. \quad (6)$$

Proof Let $1 \leq i \leq t - 1$. Structure \mathbf{x}_i maximizes $f(\mathbf{x}) - \delta_i g(\mathbf{x})$ over \mathcal{X} , so

$$\begin{aligned} h_i &= f_i - \delta_i g_i = f(\mathbf{x}_i) - \delta_i g(\mathbf{x}_i) \\ &\geq f(\mathbf{x}_{i+1}) - \delta_i g(\mathbf{x}_{i+1}) = f_{i+1} - \delta_i g_{i+1}. \end{aligned}$$

Therefore, using (4) and (5),

$$\begin{aligned} h_i &\geq f_{i+1} - \delta_i g_{i+1} \\ &= h_{i+1} + \delta_{i+1} g_{i+1} - \delta_i g_{i+1} \\ &= h_{i+1} + \frac{f_i}{g_i} g_{i+1} - \frac{h_i - f_i}{g_i} g_{i+1} \\ &= h_{i+1} + h_i \frac{g_{i+1}}{g_i}. \end{aligned}$$

This inequality immediately implies Inequality (6). \square

Lemma 2 leads to a general bound on the number of iterations of the Newton method given in [Theorem 1](#). [Theorems 2](#) and [3](#) show related bounds for special classes of fractional combinatorial optimization.

Theorem 1 *The Newton method solves problem \mathcal{F} in at most $\log(\text{MAX}_f) + \log(\text{MAX}_g) - \log(\text{MIN}_g) - \log(\text{GAP}) + 2$ iterations.*

Proof [Lemma 2](#) implies that for $i = 1, 2, \dots, t - 1$,

$$\frac{h_{i+1}g_{i+1}}{h_i g_i} \leq \frac{1}{4}. \quad (7)$$

Hence, $h_{t-1}g_{t-1} \leq (1/4^{t-2})h_1g_1$ and

$$t \leq \log_4(h_1g_1) - \log_4(h_{t-1}g_{t-1}) + 2. \quad (8)$$

Since $h_1 = \text{MAX}_f$, so

$$h_1g_1 \leq \text{MAX}_f \cdot \text{MAX}_g. \quad (9)$$

Equations (4) and (5) imply that for each $i = 1, 2, \dots, t - 1$, $\delta_{i+1} - \delta_i = h_i/g_i$. Hence,

$$h_{t-1}g_{t-1} = (\delta_t - \delta_{t-1})g_{t-1}^2 = \left(\frac{f_{t-1}}{g_{t-1}} - \frac{f_{t-2}}{g_{t-2}} \right) g_{t-1}^2 \geq \text{GAP} \cdot (\text{MIN}_g)^2. \quad (10)$$

Inequalities (8)–(10) imply the bound on the number of iterations stated in this theorem. \square

The following two theorems have been frequently rediscovered in the context of many different fractional combinatorial optimization problems.

Theorem 2 *For a linear fractional combinatorial optimization problem \mathcal{F}_L such that all input numbers a_i and b_i , $1 \leq i \leq p$, are integers not greater than U , the Newton method runs in $O(\log(pU))$ iterations.*

Proof For such a problem \mathcal{F}_L , $\text{MAX}_f \leq pU$, $\text{MAX}_g \leq pU$, $\text{MIN}_g \geq 1$, and $\text{GAP} \geq 1/(pU)^2$, so [Theorem 1](#) implies an $O(\log(pU))$ bound on the number of iterations. \square

Theorem 3 *For a uniform fractional combinatorial optimization problem \mathcal{F}_U , the Newton method runs in at most $p + 1$ iterations.*

Proof Sequence $(g_i)_{1 \leq i \leq t-1}$ is strictly decreasing. For a uniform fractional optimization problem, each number g_i is a positive integer not greater than p , so $t \leq p + 1$. \square

4 The Newton Method for the Linear Case

In this section a strongly polynomial bound on the number of iterations of the Newton method for linear fractional combinatorial optimization problem \mathcal{F}_L is derived. More precisely, it is shown that the number of iterations is $O(p^2 \log^2 p)$, using the notation introduced in [Sect. 3](#) and following the presentation in [50]. A slightly better bound of $O(p^2 \log p)$ was shown by Wang et al. [67]. Inequality (7), which is a direct consequence of [Lemma 2](#), says that the sequence $(h_i g_i)$ geometrically decreases to zero. Equations (4) and (5) imply that $h_1 g_1 = f_1 g_1$ and for $2 \leq i \leq t$,

$$h_i g_i = (f_i - \delta_i g_i) g_i = \left(f_i - \frac{f_{i-1}}{g_{i-1}} g_i \right) g_i. \quad (11)$$

For each $1 \leq i \leq t$, numbers f_i and g_i are sums of some numbers drawn from $2p$ numbers a_1, a_2, \dots, a_p and b_1, b_2, \dots, b_p . Thus, the elements of sequence $(h_i g_i)$ are created from fixed $2p$ numbers using $O(p)$ additions/subtractions and up to three multiplications/divisions. Bounds on the numbers of iterations of the Newton

method for linear fractional combinatorial optimization come from the fact that a geometric sequence whose elements are constructed in such a limited way cannot be long.

To establish intuition why such sequences cannot be long, it is useful to consider a simple special case when $b_1 \geq b_2 \geq \dots \geq b_p \geq 0$, and there exists a constant α such that for every $i = 1, 2, \dots, t-1$, $g_{i+1}/g_i \leq \alpha < 1$. That is, for this case the sequence $(g_i)_{1 \leq i \leq t}$ on its own geometrically decreases to zero. Each number g_i is equal to the sum of distinct elements drawn from the multi-set $\{b_1, b_2, \dots, b_p\}$. Obviously $g_1 \leq pb_1$. Since the sequence $(g_i)_{1 \leq i \leq t}$ decreases geometrically, then $g_k < b_1$, for some $k = O(\log p)$. This means that number b_1 is not a term in the sum g_k nor does it occur in any sum g_i , for $i \geq k$, so it can be excluded from further considerations. Since $g_k < b_1$, then $g_k \leq (p-1)b_2$, and after next $O(\log p)$ iterations, the number b_2 can be excluded, then b_3 , and so on. Therefore, the length of the sequence (g_i) is only $O(p \log p)$. (In fact, one can show that in this case the length of the sequence (g_i) is $O(p)$. See Lemma 8.)

The actual analysis of the number of iterations is more involved than what the previous paragraph may suggest, because numbers $h_i g_i$ are more “complicated” than numbers g_i (see Eq. (11)). Moreover, the sums of numbers which appear in the general case are not necessarily nonnegative. Even if all numbers a_1, a_2, \dots, a_p and b_1, b_2, \dots, b_p are positive, negative terms may appear since subtractions are used in the definition of numbers h_i . The following lemma, which gives a bound on the length of a geometrically decreasing sequence of sums of numbers, is the main tool in bounding the number of iterations of the Newton method. In the statement of this lemma, the elements of sequence $(\mathbf{y}_k \mathbf{c})_{k \geq 1}$ are constructed by adding and/or subtracting numbers drawn from the set of the components of vector \mathbf{c} .

Lemma 3 *Let $\mathbf{c} = (c_1, c_2, \dots, c_p)$ be a p -dimensional vector with nonnegative real components, and let $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q$ be vectors from $\{-1, 0, 1\}^p$. If for all $i = 1, 2, \dots, q-1$*

$$0 < \mathbf{y}_{i+1} \mathbf{c} \leq \frac{1}{2} \mathbf{y}_i \mathbf{c},$$

then $q = O(p \log p)$.

Proof Consider the following polyhedron

$$\begin{aligned} P = \{ \mathbf{z} = (z_1, z_2, \dots, z_p) \in \mathbb{R}^p : \\ (\mathbf{y}_i - 2\mathbf{y}_{i+1})\mathbf{z} \geq 0, \text{ for } i = 1, 2, \dots, q-1, \\ \mathbf{y}_q \mathbf{z} = 1, \\ z_i \geq 0, \text{ for } i = 1, 2, \dots, p \}. \end{aligned}$$

Let A and \mathbf{b} denote the coefficient matrix and the right-hand side vector of the system defining polyhedron P . This polyhedron is not empty because it contains vector $\mathbf{c}/(\mathbf{y}_q \mathbf{c})$. It follows from the polyhedral theory that there exists a vector

$\mathbf{c}^* = (c_1^*, c_2^*, \dots, c_p^*) \in P$ such that $A'\mathbf{c}^* = \mathbf{b}'$ for some nonsingular $p \times p$ submatrix A' of matrix A and a subvector \mathbf{b}' of vector \mathbf{b} (the vertices of the polyhedron are such vectors). Cramer's rule says that for each $i = 1, 2, \dots, p$,

$$c_i^* = \frac{\det A'_i}{\det A'},$$

where matrix A'_i is obtained from matrix A' by replacing the i th column with vector \mathbf{b}' .

The determinant of a $p \times p$ matrix $M = (m_{ij})$ is equal to

$$\det M = \sum_{\sigma} \varepsilon_{\sigma} m_{1,\sigma(1)} m_{2,\sigma(2)} \cdots m_{p,\sigma(p)},$$

where the summation is over all permutations σ of the set of indices $\{1, 2, \dots, p\}$ and each number ε_{σ} is equal to either 1 or -1 . Thus, $|\det M| \leq m^p p!$, where m is the maximum absolute value of an entry of matrix M . The entries of matrix A and vector \mathbf{b} , and consequently the entries of all matrices A'_i , are integers from interval $[-3, 3]$. Thus, $|\det A'_i| \leq 3^p p!$, for each $i = 1, 2, \dots, p$. Since also $|\det A'| \geq 1$, then $c_i^* \leq 3^p p!$, for each $i = 1, 2, \dots, p$, and

$$\mathbf{y}_j \mathbf{c}^* \leq \sum_{i=1}^p c_i^* \leq p 3^p p!,$$

for each $j = 1, 2, \dots, q$. Finally,

$$1 = \mathbf{y}_q \mathbf{c}^* \leq \frac{1}{2^{q-1}} \mathbf{y}_1 \mathbf{c}^* \leq \frac{1}{2^{q-1}} p 3^p p!,$$

so $q \leq \log(p 3^p p!) + 1 = O(p \log p)$. \square

The following rephrasing of [Lemma 3](#) will also be used in the further analysis of the Newton method.

Corollary 1 *Let $\mathbf{c} = (c_1, c_2, \dots, c_p) \in \mathbb{R}^p$, and let $\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_q$ be vectors from $\{0, 1\}^p$. If for all $i = 1, 2, \dots, q - 1$*

$$0 < \mathbf{y}_{i+1} \mathbf{c} \leq \frac{1}{2} \mathbf{y}_i \mathbf{c},$$

then $q = O(p \log p)$.

Two bounds on the number of iterations of the Newton method for linear fractional combinatorial optimization are shown below. The first $O(p^3 \log p)$ bound ([Theorem 4](#)) comes from a direct analysis of the sequence $(h_i g_i)$. This analysis uses

Inequality (7), Eq. (11), and Lemma 3. Then, by analyzing the sequences (h_i) and (g_i) separately, the $O(p^2 \log^2 p)$ bound is derived (Theorem 5).

Theorem 4 *The Newton method solves a linear fractional combinatorial optimization problem \mathcal{F}_L in $O(p^3 \log p)$ iterations.*

Proof For each $1 \leq i \leq t-1$, $h_i > 0$. This fact, Eq. (11), and Inequality (7) imply that for each $i = 2, 3, \dots, t-2$,

$$0 < \left(f_{i+1} - \frac{f_i}{g_i} g_{i+1} \right) g_{i+1} \leq \frac{1}{4} \left(f_i - \frac{f_{i-1}}{g_{i-1}} g_i \right) g_i,$$

and since $0 < g_i \leq g_{i-1}$, then

$$0 < f_{i+1}g_{i+1}g_i - f_i g_{i+1}^2 \leq \frac{1}{4} f_i g_i g_{i-1} - f_{i-1} g_i^2.$$

Hence, setting $s_i = f_i g_i g_{i-1} - f_{i-1} g_i^2$, the following inequalities hold for each $i = 2, 3, \dots, t-2$:

$$0 < s_{i+1} \leq \frac{1}{4} s_i.$$

For a p^3 -dimensional vector \mathbf{c} whose set of components is $\{|a_j b_k b_l| : 1 \leq j, k, l \leq p\}$, there exist p^3 -dimensional vectors $\mathbf{y}_2, \mathbf{y}_3, \dots, \mathbf{y}_{t-1}$ with components from $\{-1, 0, 1\}$ such that $s_i = \mathbf{y}_i \mathbf{c}$, for each $i = 2, 3, \dots, t-1$. The bound $t = O(p^3 \log p)$ follows now from Lemma 3. \square

The $O(p^2 \log^2 p)$ bound on the number of iterations is based on the following two lemmas.

Lemma 4 *There are at most $O(p \log p)$ iterations k such that $g_{k+1} \leq (2/3)g_k$.*

Proof Let $k_1 < k_2 < \dots < k_q$ be the indices k such that $g_{k+1} \leq (1/2)g_k$. Since sequence $(g_i)_{1 \leq i \leq t}$ is nonincreasing, then for each $i = 1, 2, \dots, q-1$,

$$0 < \mathbf{x}_{k_{i+1}} \mathbf{b} = g_{k_{i+1}} \leq g_{k_i+1} \leq \frac{1}{2} g_{k_i} = \frac{1}{2} \mathbf{x}_{k_i} \mathbf{b}.$$

Corollary 1, with $\mathbf{c} = \mathbf{b}$ and $\mathbf{y}_i = \mathbf{x}_{k_i}$ for $i = 1, 2, \dots, q$, implies that $q = O(p \log p)$. Monotonicity of sequence $(g_i)_{1 \leq i \leq t}$ further implies that there are at most $2q = O(p \log p)$ iterations k such that $g_{k+1} \leq (2/3)g_k$. \square

Lemma 5 *There are at most $O(p \log p)$ consecutive iterations k such that $g_{k+1} \geq (2/3)g_k$.*

Proof Consider a sequence of q consecutive iterations $j+1, j+2, \dots, j+q$ such that for each $k = j+1, j+2, \dots, j+q-1$,

$$g_{k+1} \geq \frac{2}{3} g_k. \quad (12)$$

The proof of this lemma is based on showing that [Corollary 1](#) can be applied to vector $\mathbf{c} = \delta_{j+q}\mathbf{b} - \mathbf{a}$ and the sequence of $q-2$ vectors $\mathbf{x}_k, k = j+1, \dots, j+q-2$. Inequalities (12) and (6) imply that for $k = j+1, j+2, \dots, j+q-1$,

$$h_{k+1} \leq \frac{1}{3} h_k. \quad (13)$$

Equations (4) and (5), and Inequalities (12) and (13), imply that the following relations hold for each $k = j+1, j+2, \dots, j+q-2$,

$$\delta_{k+2} - \delta_{k+1} = \frac{h_{k+1}}{g_{k+1}} \leq \frac{1}{2} \cdot \frac{h_k}{g_k} = \frac{1}{2}(\delta_{k+1} - \delta_k). \quad (14)$$

Inequality (14) implies

$$\begin{aligned} \delta_{j+q} - \delta_{k+1} &= (\delta_{j+q} - \delta_{j+q-1}) + (\delta_{j+q-1} - \delta_{j+q-2}) + \cdots + (\delta_{k+2} - \delta_{k+1}) \\ &\leq \frac{1}{2}(\delta_{j+q-1} - \delta_{j+q-2}) + \cdots + \frac{1}{2}(\delta_{k+1} - \delta_k) \\ &= \frac{1}{2}(\delta_{j+q-1} - \delta_k) \\ &< \frac{1}{2}(\delta_{j+q} - \delta_k). \end{aligned} \quad (15)$$

Using Eq. (5), it can be shown that $\mathbf{x}_k(\delta_{j+q}\mathbf{b} - \mathbf{a})$ is positive for each $k = j+1, j+2, \dots, j+q-2$:

$$\mathbf{x}_k(\delta_{j+q}\mathbf{b} - \mathbf{a}) = \delta_{j+q}\mathbf{b}\mathbf{x}_k - \mathbf{a}\mathbf{x}_k = \delta_{j+q}g_k - f_k = (\delta_{j+q} - \delta_{k+1})g_k > 0.$$

Thus, Inequalities (15) and the monotonicity of the sequence (g_i) imply that the following relations hold for each $k = j+1, j+2, \dots, j+q-3$:

$$\begin{aligned} 0 < \mathbf{x}_{k+1}(\delta_{j+q}\mathbf{b} - \mathbf{a}) &= (\delta_{j+q} - \delta_{k+2})g_{k+1} \leq (\delta_{j+q} - \delta_{k+2})g_k \\ &\leq \frac{1}{2}(\delta_{j+q} - \delta_{k+1})g_k = \frac{1}{2}\mathbf{x}_k(\delta_{j+q}\mathbf{b} - \mathbf{a}). \end{aligned}$$

Now [Corollary 1](#), applied to the vector $\mathbf{c} = \delta_{j+q}\mathbf{b} - \mathbf{a}$ and the sequence of $q-2$ vectors $\mathbf{x}_k, k = j+1, \dots, j+q-2$, implies the bound $q = O(p \log p)$. \square

[Lemmas 4](#) and [5](#) immediately imply the following bound on the number of iterations of the Newton method.

Theorem 5 *The Newton method solves a linear fractional combinatorial optimization problem \mathcal{F}_L in $O(p^2 \log^2 p)$ iterations.*

Strongly polynomial bounds on the number of iterations of the Newton method for problem \mathcal{F}_L are somewhat unexpected because, as the following example demonstrates, function $h(\delta)$ may consist of an exponential number of linear segments. Let $\mathbf{a} = (a_1, a_2, \dots, a_{3p})$ and $\mathbf{b} = (b_1, b_2, \dots, b_{3p})$ be vectors such that

$$a_i = \begin{cases} 2^{i-1}, & \text{for } i = 1, 2, \dots, p, \\ 0, & \text{for } i = p+1, \dots, 3p, \end{cases}$$

$$b_i = \begin{cases} 0, & \text{for } i = 1, 2, \dots, p, \\ 2^{i-p-1}, & \text{for } i = p+1, \dots, 3p. \end{cases}$$

Let $\mathcal{U} \subseteq \{0, 1\}^{3p}$ (respectively, $\mathcal{W} \subseteq \{0, 1\}^{3p}$) be the set such that a binary vector $\mathbf{x} = (x_1, \dots, x_{3p})$ belongs to \mathcal{U} (respectively, \mathcal{W}) if and only if $x_i = 0$ for each $p < i \leq 3p$ (respectively, for each $1 \leq i \leq p$). The 2^p numbers \mathbf{au} , for $\mathbf{u} \in \mathcal{U}$, are numbers $0, 1, \dots, 2^p - 1$, and the 2^{2p} numbers \mathbf{bw} , for $\mathbf{w} \in \mathcal{W}$, are numbers $0, 1, \dots, 2^{2p} - 1$. For $\mathbf{u} \in \mathcal{U}$, let $\mathbf{w}_\mathbf{u}$ denote the vector for which $\mathbf{bw}_\mathbf{u} = (\mathbf{au})^2$. Finally, let $\mathcal{X} = \{(\mathbf{u}, \mathbf{w}_\mathbf{u}) : \mathbf{u} \in \mathcal{U}, \mathbf{u} \neq \mathbf{0}\}$. For this instance $(\mathcal{X}, \mathbf{a}, \mathbf{b})$ of linear fractional combinatorial optimization, function h is equal to

$$h(\delta) = \max\{k - \delta k^2 : k = 1, 2, \dots, 2^p - 1\}$$

and consists of $2^p - 1$ linear segments.

The bound stated in [Theorem 5](#) is the best-known bound on the number of iterations of the Newton method for a general problem \mathcal{F}_L . However, when a concrete problem of type \mathcal{F}_L is considered, it often happens that special properties of this problem enable us to derive a better bound. For example, [Theorem 5](#) implies that the Newton method solves the MaxRatioPath problem in $O(m^2 \log^2 m)$ iterations and, consequently, in $O(m^3 \log^2 m)$ total time. It will be shown below, using the properties of the MaxRatioPath problem, that if the edge weights are nonnegative, then the Newton method requires actually only $O(m \log n)$ iterations. A more advanced example of such an analysis is presented in [Sect. 8](#). One more technical lemma about the Newton method for the general case is needed.

Lemma 6 *For each $1 \leq i \leq t-2$ and $\delta \geq \delta_{i+2}$,*

$$f_i - \delta g_i \leq -h_{i+1}. \tag{16}$$

Proof Since sequence (δ_i) monotonically increases and numbers g_i are positive, it is enough to show that Inequality (16) is true for each $1 \leq i \leq t-2$ and $\delta = \delta_{i+2}$. Equations (4) and (5) and Lemma 1(1) imply that for each $1 \leq i \leq t-2$,

$$\begin{aligned} h_{i+1} &= f_{i+1} - \delta_{i+1} g_{i+1} = (\delta_{i+2} - \delta_{i+1}) g_{i+1} \\ &\leq (\delta_{i+2} - \delta_{i+1}) g_i = \delta_{i+2} g_i - f_i. \end{aligned}$$

Thus Inequality (16) is true for each $1 \leq i \leq t-2$ and $\delta = \delta_{i+2}$. \square

Theorem 6 *The Newton method solves the MaxRatioPath problem in $O(m \log n)$ iterations and in $O(m^2 \log n)$ total time, if the weights of the edges are nonnegative.*

Proof Let $(n, E, \mathbf{c}, \mathbf{w})$ be an input instance of the MaxRatioPath problem. The proof uses the notation from the description of the MaxRatioPath problem in Sect. 2 and the following additional definitions. For $\delta \in \mathbf{R}$, a vertex v and an edge $(v, u) \in E$

$$\mathbf{c}_\delta(v, u) \stackrel{\text{def}}{=} (\mathbf{c} - \delta \mathbf{w})(v, u) = \mathbf{c}(v, u) - \delta \mathbf{w}(v, u),$$

$d_\delta(v) \stackrel{\text{def}}{=} \text{the maximum length of a path from vertex 1 to vertex } v$
according to the edge-cost function \mathbf{c}_δ ,

$$\bar{\mathbf{c}}_\delta(v, u) \stackrel{\text{def}}{=} \mathbf{c}_\delta(v, u) + d_\delta(v) - d_\delta(u).$$

For a subset of edges $Q \subseteq E$, $\mathbf{c}_\delta(Q)$ [respectively, $\bar{\mathbf{c}}_\delta(Q)$] denotes the sum of $\mathbf{c}_\delta(e)$ (respectively, $\bar{\mathbf{c}}_\delta(e)$) over all edges $e \in Q$. The introduced definitions imply that for each path $P \subseteq E$,

$$\mathbf{c}_\delta(P) = f(P) - \delta g(P).$$

It is also easy to verify that for each edge $(v, u) \in E$,

$$\bar{\mathbf{c}}_\delta(v, u) \leq 0 \tag{17}$$

and for each path P from vertex 1 to vertex n ,

$$\bar{\mathbf{c}}_\delta(P) = \mathbf{c}_\delta(P) - d_\delta(n). \tag{18}$$

Consider the computation of the Newton method on input $(n, E, \mathbf{c}, \mathbf{w})$. For $i = 1, 2, \dots, t$, let P_i be the path from vertex 1 to vertex n computed during iteration i . For each $1 \leq i \leq t$,

$$h_i = f(P_i) - \delta_i g(P_i) = \mathbf{c}_{\delta_i}(P_i) = d_{\delta_i}(n). \tag{19}$$

An edge e is said to be *essential* at the beginning of iteration i , if it belongs to at least one of the paths P_i, P_{i+1}, \dots, P_t . The proof of the theorem is based on showing that a sequence of $k = \lceil \log n \rceil + 1$ consecutive iterations decreases the number of essential edges at least by one. Consider a sequence of iterations $i, i+1, \dots, i+k$. Inequality (7) implies that

$$h_{i+k} g_{i+k} \leq \frac{1}{n^2} h_{i+1} g_{i+1}. \quad (20)$$

If $g_{i+k} \leq (1/n)g_i$, then let $e \in P_i$ be such an edge that

$$\mathbf{w}(e) \geq \frac{1}{|P_i|} \sum_{a \in P_i} \mathbf{w}(a) = \frac{1}{|P_i|} g(P_i) \geq \frac{1}{n-1} g_i > g_{i+k}.$$

Since sequence (g_j) is nonincreasing, then for each $j \geq i+k$, $\mathbf{w}(e) > g_j = \sum_{a \in P_j} \mathbf{w}(a)$, so edge e does not belong to path P_j . Thus, edge e is essential at the beginning of iteration i (since $e \in P_i$) but is not essential at the beginning of iteration $i+k$ (since $e \notin P_j$, for each $j \geq i+k$).

If $g_{i+k} > (1/n)g_i$, then also $g_{i+k} > (1/n)g_{i+1}$, and Inequality (20) implies that $h_{i+k} < (1/n)h_{i+1}$. This inequality and Lemma 6 imply that

$$\mathbf{c}_{\delta_{i+k}}(P_i) = f(P_i) - \delta_{i+k} g(P_i) = f_i - \delta_{i+k} g_i \leq -h_{i+1} < -nh_{i+k}.$$

Therefore, using (18) and (19),

$$\bar{\mathbf{c}}_{\delta_{i+k}}(P_i) = \mathbf{c}_{\delta_{i+k}}(P_i) - d_{\delta_{i+k}}(n) < -nh_{i+k} - h_{i+k} = -(n+1)h_{i+k}.$$

This means that there must be an edge $e \in P_i$ such that $\bar{\mathbf{c}}_{\delta_{i+k}}(e) < -h_{i+k}$. Let e be such an edge. Referring to (17) and (18), for any path P from vertex 1 to vertex n which contains edge e ,

$$\mathbf{c}_{\delta_{i+k}}(P) = \bar{\mathbf{c}}_{\delta_{i+k}}(P) + d_{\delta_{i+k}}(n) \leq \bar{\mathbf{c}}_{\delta_{i+k}}(e) + h_{i+k} < 0.$$

Hence, none of the paths P_j , for $j \geq i+k$, contain edge e , because $\delta_j \geq \delta_{i+k}$ and

$$\mathbf{c}_{\delta_{i+k}}(P_j) \geq \mathbf{c}_{\delta_j}(P_j) = h_j \geq 0.$$

Thus, edge e is essential at the beginning of iteration i but is not essential at the beginning of iteration $i+k$. \square

5 Megiddo's Parametric Search

This section presents the parametric search method for fractional combinatorial optimization proposed by Megiddo [41, 42]. The method is introduced using the MaxRatioPath problem as an example. Let $(n, E, \mathbf{c}, \mathbf{w})$ be an input instance of this

problem and let δ^* denote the optimum value for this input. See Sect. 2 for the notation concerning the MaxRatioPath problem. The maximum cost of a path from vertex 1 to vertex n according to the edge-cost function $\mathbf{c} - \delta^* \mathbf{w}$ is equal to 0. Thus, if δ^* is known and algorithm MAXCOST is run on input $(n, E, \mathbf{c} - \delta^* \mathbf{w})$, then it returns value 0 and a path P which is optimal for the input instance $(n, E, \mathbf{c}, \mathbf{w})$ of the MaxRatioPath problem.

The crucial idea of Megiddo's method is to run algorithm MAXCOST on input $(n, E, \mathbf{c} - \delta^* \mathbf{w})$ without knowing the value of δ^* . Inspecting the pseudocode of this algorithm, it should be clear that the problem with running algorithm MAXCOST for the cost function $\mathbf{c} - \delta^* \mathbf{w}$ with unknown δ^* is the comparisons performed in line 5. An oracle which gives correct outcomes of those comparisons is needed.

When algorithm MAXCOST is run for the cost function $\mathbf{c} - \delta^* \mathbf{w}$ with unknown δ^* , then at each point of the computation, each $d[v]$ is no longer a number but a function of δ^* . Fortunately, the way the variables $d[v]$ are updated in line 6 implies that they are always linear functions of δ^* . Therefore, the comparison

$$d[v] + (\mathbf{c}(v, u) - \delta^* \mathbf{w}(v, u)) - d[u] > 0?$$

performed in line 5 is always of the form “ $s - \delta^* t > 0$?” for some known numbers s and t . To find out the outcome of such a comparison, the relation between numbers s/t and δ^* has to be established. The three possibilities, $s/t < \delta^*$, $s/t = \delta^*$, and $s/t > \delta^*$, are equivalent to the optimum value of the problem MaxPath(s/t) being positive, equal to 0, or negative, respectively. Thus, establishing the relation between s/t and δ^* can be done by running algorithm MAXCOST for the cost function $\mathbf{c} - (s/t)\mathbf{w}$. That is, the required oracle can be implemented using algorithm MAXCOST itself. The pseudocode of the resulting algorithm for the MaxRatioPath problem is given below.

WEIGHTEDMAXCOST $(n, E, \mathbf{c}, \mathbf{w})$

- 1) $d[1] \leftarrow 0$; $seen[1] \leftarrow \text{true}$
- 2) **for** $v \leftarrow 2$ **to** n **do** $seen[v] \leftarrow \text{false}$
- 3) **for** $v \leftarrow 1$ **to** $n - 1$ **do**
- 4) **for** each u such that $(v, u) \in E$ **do**
- 5) let s and t be the numbers such that

$$s - \delta^* t = d[v] + \mathbf{c}(v, u) - \delta^* \mathbf{w}(v, u) - d[u]$$
- 6) **if** $t \neq 0$ **then**
- 7) $x \leftarrow$ the number returned by $\text{MAXCOST}(n, E, \mathbf{c} - (s/t)\mathbf{w})$
- 8) **if** (not $seen[u]$) or ($t = 0$ and $s > 0$) or ($tx < 0$) **then**
- 9) $d[u] \leftarrow d[v] + \mathbf{c}(v, u) - \delta^* \mathbf{w}(v, u)$
- 10) $predecessor[u] \leftarrow v$; $seen[u] \leftarrow \text{true}$
- 11) let $P = (v_1, v_2, \dots, v_k)$ where $v_1 = 1$, $v_k = n$, and
 $v_i = predecessor[v_{i+1}]$, for $i = 1, 2, \dots, k - 1$
- 12) **return** path P

Theorem 7 Algorithm WEIGHTEDMAXCOST solves the MaxRatioPath problem in $O(m^2)$ time.

Proof The computation $\text{MAXCOST}(n, E, \mathbf{c} - \delta^* \mathbf{w})$ returns an optimal path P for the instance $(n, E, \mathbf{c}, \mathbf{w})$ of the MaxRatioPath problem. Tracing, in parallel, the computations $\text{MAXCOST}(n, E, \mathbf{c} - \delta^* \mathbf{w})$ and $\text{WEIGHTEDMAXCOST}(n, E, \mathbf{c}, \mathbf{w})$ shows that the *predecessor* pointers change in exactly the same way in both computations, because the outcomes of the corresponding conditions checked in line 5 of the first algorithm and in line 8 of the second one are always the same. Thus, both computations return the same path, so the computation $\text{WEIGHTEDMAXCOST}(n, E, \mathbf{c}, \mathbf{w})$ returns an optimal path P for the instance $(n, E, \mathbf{c}, \mathbf{w})$ of the MaxRatioPath problem. The running time of algorithm WEIGHTEDMAXCOST is $O(m^2)$ because algorithm MAXCOST is executed at most m times. \square

To generalize the above idea, the notion of *linear algorithm* is needed. Let $\mathcal{Q}(\delta)$ denote a problem whose input includes a parameter $\delta \in \mathbf{R}$. An algorithm \mathcal{A} is a linear algorithm for problem $\mathcal{Q}(\delta)$, if it satisfies the following conditions:

1. For each fixed $\bar{\delta} \in \mathbf{R}$, algorithm \mathcal{A} computes a correct solution of problem $\mathcal{Q}(\bar{\delta})$.
2. The value of each arithmetic expression on each possible execution path of algorithm \mathcal{A} is a linear function of parameter δ .

Observe that algorithm WEIGHTEDMAXCOST does not actually need in line 7 an algorithm which computes a maximum-cost path. Everything works equally well if instead an algorithm is used which only finds a positive-cost path, if such a path exists. In general terms, this means that instead of considering problem $\mathcal{P}(\delta)$, it is enough to consider problem $\mathcal{P}_0(\delta)$. The following theorem summarizes the basic form of Megiddo's parametric search.

Theorem 8 *If there exists a linear algorithm \mathcal{A} for the parametric problem $\mathcal{P}_0(\delta)$ corresponding to a fractional combinatorial optimization problem \mathcal{F} which runs in time T , then problem \mathcal{F} can be solved in $O(T^2)$ time.*

Proof Run algorithm \mathcal{A} for problem $\mathcal{P}_0(\delta^*)$ with unknown δ^* . Since \mathcal{A} is a linear algorithm, each comparison encountered during the computation is equivalent to a comparison “ $s - \delta^* t > 0 ?$ ” for some numbers s and t and can be evaluated by solving problem $\mathcal{P}_0(s/t)$. If each such problem is solved by running algorithm \mathcal{A} with $\delta = s/t$, then the whole computation runs in $O(T^2)$ time. \square

A better bound than the bound stated in [Theorem 8](#) may be obtained, if possible independence of comparisons in algorithm \mathcal{A} is exploited. Consider, for example, iteration v , $1 \leq v \leq n - 1$, of the outer loop of the computation $\text{MAXCOST}(n, E, \mathbf{c} - \delta^* \mathbf{w})$. Each comparison in line 5 performed during this iteration would give exactly the same outcome if it were performed right at the beginning of the iteration, because it does not depend on any updates done during this iteration. (Note that this statement would not be true if the graph had multiple edges.) Let $s_k - \delta^* t_k$, for $k = 1, 2, \dots, \deg(v)$, be the expressions which have to be compared with 0 during this iteration, where $\deg(v)$ denotes the number of edges outgoing from vertex v . To know the outcome of these comparisons, the relation between unknown number δ^* and all numbers s_k/t_k has to be established. The relation between s_k/t_k and δ^* can be established for each k independently, by computing

$\text{MAXCOST}(n, E, \mathbf{c} - \delta\mathbf{w})$ for each $\delta = s_k/t_k$. Alternatively, the numbers s_k/t_k can first be sorted, and then, the position of δ^* with respect to the elements of the sorted sequence (s'_k/t'_k) can be established by binary search. The latter approach requires only $O(\log(\deg(v)))$ applications of algorithm MAXCOST, one application per one iteration of the binary search. The running time of sorting $\deg(v)$ numbers and $O(\log(\deg(v)))$ applications of algorithm MAXCOST is $O(\deg(v) \log(\deg(v))) + O(m \log(\deg(v))) = O(m \log(\deg(v)))$. Thus, the running time of the whole computation is

$$O\left(m \sum_{v=1}^{n-1} \log(\deg(v))\right) = O(m n \log(m/n)), \quad (21)$$

since $\sum_{v=1}^{n-1} \deg(v) = m$ and the logarithmic function is concave.

To generalize this idea, the notion of a *stage* of the computation is introduced. Consider a problem \mathcal{Q} whose input includes a parameter $\delta \in \mathbf{R}$. A stage of the computation of an algorithm for problem \mathcal{Q} is a (continual) part of the computation which has the following property. For each comparison performed during one stage, if this comparison is moved to the beginning of this stage, the outcome of the comparison does not change. Only the comparisons whose outcomes may depend on the value of parameter δ are considered here. An algorithm is said to consist of r stages, if the computation of this algorithm can be partitioned into r stages, independently of the actual input values. The theorem below distinguishes between the main algorithm \mathcal{A}_1 , which guides the whole computation, must be linear, and should ideally consist of a small number of stages, and the “subordinate” algorithm \mathcal{A}_2 , which is used to resolve individual comparisons performed by algorithm \mathcal{A}_1 . Algorithms \mathcal{A}_1 and \mathcal{A}_2 may of course be the same, but they are usually different, if they are to provide the best possible running time of the whole computation (see Sect. 6).

Theorem 9

If there exist

1. A linear algorithm \mathcal{A}_1 for the parametric problem $\mathcal{P}_0(\delta)$ corresponding to a fractional combinatorial optimization problem \mathcal{F} , which runs in time T_1 and consists of r stages with q_i comparisons during stage i , and
2. An algorithm \mathcal{A}_2 for problem $\mathcal{P}_0(\delta)$ which runs in time T_2 , then problem \mathcal{F} can be solved in $O(T_1 + T_2 \sum_{i=1}^r \log q_i)$ time.

Proof Run algorithm \mathcal{A}_1 for problem $\mathcal{P}_0(\delta^*)$ with unknown δ^* . Consider stage i of this computation. During this stage, q_i independent comparisons of the form $s - \delta^* t$ are to be resolved. Computing the outcomes of all these comparisons can be reduced to computing the relation between δ^* and q_i numbers x_1, x_2, \dots, x_{q_i} .

If these numbers are first sorted and then δ^* is located in the sorted sequence by binary search, as suggested earlier in the example, then the running time of this stage is $O(q_i \log q_i + T_2 \log q_i)$. This gives, however, only an $O(T_1 \max_i \{\log q_i\} + T_2 \sum_{i=1}^r \log q_i)$ bound on the running time of the whole computation.

To obtain the bound claimed in the statement of the theorem, observe that sorting numbers x_1, x_2, \dots, x_{q_i} can be avoided. Instead, the median of these numbers can be found in $O(q_i)$ time [4] and compared with δ^* by running algorithm \mathcal{A}_2 . The outcome of this comparison reveals the relation between δ^* and half of the numbers x_1, x_2, \dots, x_{q_i} . This process can be continued by finding the median of the remaining half. This way, the numbers x_1, x_2, \dots, x_{q_i} are eventually partitioned into the numbers smaller than δ^* and the numbers greater than δ^* . The running time of this computation is

$$O\left(\sum_{k=0}^{\log q_i} \left(\frac{q_i}{2^k} + T_2\right)\right) = O(q_i + T_2 \log q_i).$$

Summing this bound over all stages i gives the bound claimed in this theorem. \square

To minimize the bound of [Theorem 9](#) for a given fractional combinatorial optimization problem, algorithm \mathcal{A}_2 should obviously be the fastest known algorithm for the non-fractional version of this problem. To find good candidates for algorithm \mathcal{A}_1 , parallel algorithms should be considered, since such algorithms are specially designed to have a small number of stages. A parallel algorithm which runs in time T and uses P processors can be viewed as an algorithm which runs in sequential time TP and consists of T stages with at most P operations per stage. Thus, the following theorem is an immediate corollary of [Theorem 9](#).

Theorem 10 *If there exist*

1. *A linear parallel algorithm \mathcal{A}_1 for the parametric problem $\mathcal{P}_0(\delta)$ corresponding to a fractional combinatorial optimization problem \mathcal{F} , which runs in time T_1 and uses P processors, and*
2. *A (sequential) algorithm \mathcal{A}_2 for problem $\mathcal{P}_0(\delta)$ which runs in time T_2 , then problem \mathcal{F} can be solved in $O(T_1 P + T_2 T_1 \log P)$ time.*

Consider another example of fractional combinatorial optimization, the maximum-ratio spanning-tree problem. An input instance of this problem consists of an undirected connected graph $G = (V, E)$, an edge-cost function $\mathbf{c} : E \rightarrow \mathbf{R}$, and an edge-weight function $\mathbf{w} : E \rightarrow \mathbf{R}$. The cost and the weight of a spanning tree in graph G are equal to the sum of the costs and the sum of the weights of the edges of this tree, respectively. The problem is to find a spanning tree in graph G with the maximum cost-to-weight ratio or, using the introduced terminology, a spanning tree with the maximum mean-weight cost. The corresponding parametric problem $\mathcal{P}(\delta)$ is the problem of computing a maximum spanning tree for the edge-cost function $\mathbf{c} - \delta \mathbf{w}$. The maximum-cost spanning-tree problem, which is equivalent to the minimum-cost spanning-tree problem, can be solved in $T_{\text{MST}} = O(\min\{m \log \log n, m + n \log n\})$ time [22, 68], where m is the number of edges and n is the number of vertices in the input graph.

A maximum-ratio spanning tree is a maximum-cost spanning-tree for the edge-cost function $\mathbf{c} - \delta^* \mathbf{w}$. Maximum-cost spanning trees depend only on the order of the edges by their costs. Thus, to find a maximum-cost spanning-tree for the edge-cost function $\mathbf{c} - \delta^* \mathbf{w}$, it is enough to sort m numbers $\mathbf{c}(e) - \delta^* \mathbf{w}(e)$, $e \in E$. Sorting m numbers can be done in $O(\log m)$ parallel time using m processors [11]. Therefore, [Theorem 10](#) implies that the maximum-ratio spanning-tree problem can be solved in $O(m \log m + T_{\text{MST}} \log^2 m) = O(T_{\text{MST}} \log^2 m)$ time.

Going back to the general case, since the objective is maximization, then it could be reasonably expected that many subtasks appearing during the computation of algorithm \mathcal{A}_1 of [Theorem 9](#) may be of the form

$$u(\delta^*) \leftarrow \max\{u_1(\delta^*), u_2(\delta^*), \dots, u_q(\delta^*)\}, \quad (22)$$

where each u_i is a linear function of parameter δ . Two different approaches to computing such a maximum are discussed next. If the maximum (22) is calculated by comparing each u_i , for $i = 2, 3, \dots, q$, with the already known maximum of u_1 through u_{i-1} , then this computation has $q - 1$ stages with one comparison per stage, and the maximum can be computed in $O(T_2 q)$ time. Here T_2 denotes, as in [Theorem 9](#), the running time of an algorithm \mathcal{A}_2 , which is used to resolve the comparisons performed during the computation of algorithm \mathcal{A}_1 .

Computation of the maximum (22) can be organized in $\lceil \log q \rceil$ stages by scheduling the comparisons in a tournament tree. Each stage consists of $O(q)$ comparisons, so [Theorem 9](#) implies that the maximum (22) can be computed in $O(q + T_2 \log^2 q)$ time. The number of stages needed to compute the maximum can be further reduced to $O(\log \log q)$ in the following way. Follow the tournament for $O(\log \log \log q)$ stages to reduce the number of live elements to $k_1 \leq q / (\log \log q)$. Live elements are those which so far have won all their comparisons. Now perform a sequence of the following stages. During stage i , partition the set of k_i live elements into $k_i^2 / (2k_1)$ equal-size groups, and perform all pairwise comparisons in each group. This stage requires $2k_1 = O(q / \log \log q)$ comparisons and reduces the number of live elements to $k_{i+1} = k_i^2 / (2k_1)$. Thus, $k_i = k_1 / 2^{2^{i-1}-1}$, so there are only $O(\log \log q)$ such stages. Therefore, this algorithm for computing the maximum of q numbers consists of $O(q)$ comparisons partitioned into $O(\log \log q)$ stages. Using this algorithm as algorithm \mathcal{A}_1 , [Theorem 9](#) implies that the maximum (22) can be computed in $O(q + T_2 \log q \log \log q)$ time. The above $O(\log \log q)$ -stage algorithm for computing the maximum of q numbers is based on Valiant's parallel algorithm which runs in $O(\log \log q)$ time and uses q processors [65].

There is an alternative approach to computing the maximum (22). The maximum of q linear functions is a convex, piecewise linear function with at most q breakpoints. If two convex, piecewise linear functions π_1 and π_2 have q_1 and q_2 breakpoints, respectively, and these breakpoints are given as two sequences sorted by x -coordinates, then the sorted sequence of the breakpoints of the function $\max\{\pi_1, \pi_2\}$ can be computed in $O(q_1 + q_2)$ time by a straightforward merging

process. Therefore, by following the merging scheme of the merge–sort algorithm, all breakpoints of the maximum of q linear functions u_i can be computed in $O(q \log q)$ time. Finding the linear segment of function u which contains δ^* can then be done by binary search over the breakpoints of function u . Using this approach, the maximum (22) can be computed in $O(q \log q + T_2 \log q)$ time.

Consider an algorithm \mathcal{A} for a problem \mathcal{Q} whose input includes a parameter $\delta \in \mathbf{R}$. A *maxima-computing phase* of an algorithm \mathcal{A} is a (continual) part of the computation of \mathcal{A} such that all comparisons performed during this part of the computation are (implicitly) involved in computing maxima, and all these maxima are independent, that is, they can be performed in arbitrary order without changing the outcome of the whole computation. As in the definition of a stage of an algorithm, only comparisons whose outcomes may depend on the value of parameter δ are considered here. The following theorem is based on the two approaches to computing the maximum (22) described above. The *size* of a maximum of q elements is equal to q .

Theorem 11 *If there exist*

1. *A linear algorithm \mathcal{A}_1 for the parametric problem $\mathcal{P}_0(\delta)$ corresponding to a fractional combinatorial optimization problem \mathcal{F} , which runs in time T_1 and consists of r maxima-computing phases, and the total size of the maxima in phase i is at most q_i , and*
2. *An algorithm \mathcal{A}_2 for problem $\mathcal{P}_0(\delta)$ which runs in time T_2 , then problem \mathcal{F} can be solved in*
 - (i) $O(T_1 + T_2 \sum_{i=1}^r \log q_i \log \log q_i)$ *time and*
 - (ii) $O(T_1 + \sum_{i=1}^r (q_i + T_2) \log q_i)$ *time.*

Proof Run algorithm \mathcal{A}_1 for problem $\mathcal{P}_0(\delta^*)$ with unknown δ^* and consider phase i of this computation. During this phase, independent maxima of the total size at most q_i have to be computed. If these maxima are computed in parallel using the $O(\log \log q)$ -stage algorithm described above, then the running time of this phase is $O(q_i + T_2 \log q_i \log \log q_i)$, and the running time of the whole computation is as in part (i) of the theorem. If first all breakpoints of all maxima are computed and sorted together and then δ^* is located in the sorted list by binary search, then the running time of this phase is $O((q_i + T_2) \log q_i)$ and the running time of the whole computation is as in part (ii) of the theorem.

The best bound for the MaxRatioPath problem shown so far in this chapter is $O(mn \log(m/n))$ (see (21)). Using Theorem 11, an $O(mn)$ bound is now shown. This bound, however, is based on other algorithm for computing a maximum-cost path than algorithm MAXCOST. Algorithm MAXCOST considers vertices $1, 2, \dots, n - 1$, and while considering vertex v , it examines all edges outgoing from v . The following algorithm MAXCOST2 considers vertices $2, 3, \dots, n$, and while considering vertex v , it examines all edges incoming into v . This algorithm returns only the maximum cost of a path from vertex 1 to vertex n but can be easily modified so that it returns also a maximum-cost path.

```

MAXCOST2( $n, E, \mathbf{a}$ )
1)  $d[1] \leftarrow 0$ 
2) for  $v \leftarrow 2$  to  $n$  do  $d[v] \leftarrow -\infty$ 
3) for  $v \leftarrow 2$  to  $n$  do
4)    $d[v] \leftarrow \max\{d[u] + \mathbf{a}(u, v) : (u, v) \in E\}$ 
5) return  $d[n]$ 

```

Algorithm MAXCOST2 is still not quite what is needed. If [Theorem 11](#) is applied with algorithm MAXCOST2 as algorithm \mathcal{A}_1 , then the bound $O(mn \log(m/n) \log \log(m/n))$ is obtained from part (i) of the theorem and the bound $O(mn \log(m/n))$ from part (ii) of the theorem. These bounds do not improve bound (21).

By rearranging the computation of algorithm MAXCOST2, the recursive algorithm MAXCOSTRECURSIVE shown below is obtained. This algorithm considers first, recursively, the subgraph induced by vertices $u = 1, 2, \dots, k = \lfloor (n+1)/2 \rfloor$, then considers the edges $(u, v) \in E$ such that $1 \leq u \leq k < v \leq n$ (observe that all maxima of this part of the computation—lines 7–8 below—are independent), and finishes up by considering, again recursively, the subgraph induced by vertices $v = k, k+1, \dots, n$.

```

MAXCOSTRECURSIVE( $n, E, \mathbf{a}$ )
1)  $d[1] \leftarrow 0$ 
2) for  $v \leftarrow 2$  to  $n$  do  $d[v] \leftarrow -\infty$ 
3) COMPUTEMAXIMA(1,  $n$ )
4) return  $d[n]$ 
COMPUTEMAXIMA( $p, r$ )
5)  $k \leftarrow \lfloor (p+r)/2 \rfloor$ 
6) if  $k > p$  then COMPUTEMAXIMA( $p, k$ )
7) for  $v \leftarrow k+1$  to  $r$  do
8)    $d[v] \leftarrow \max\{d[v], d[u] + \mathbf{a}(u, v) : p \leq u \leq k, (u, v) \in E\}$ 
9) if  $k+1 < r$  then COMPUTEMAXIMA( $k+1, r$ )

```

Algorithm MAXCOSTRECURSIVE runs in $O(m)$ time (each edge is considered only once), and for each $k = 1, 2, \dots, \log n$, it has $r_k = n/2^k$ maxima-computing phases (assume for convenience that n is a power of 2). Each of these r_k phases considers only edges $(u, v) \in E$ such that $a+1 \leq u \leq a+2^{k-1} < v \leq a+2^k$, for some integer a , so the total size of the maxima in one such phase is at most $q_k = \min\{2^{2k}, m\}$. Therefore, part (11) of [Theorem 11](#) implies that the MaxRatioPath problem can be solved within the time bound:

$$O\left(m + \sum_{k=1}^{\log n} r_k(q_k + m) \log q_k\right) = O\left(m + \sum_{k=1}^{\log n} \frac{n}{2^k} \cdot m \cdot (2k)\right) = O(nm).$$

Bound (i) of [Theorem 11](#) gives here the same time bound. [Theorem 11](#) is used in the next section to show low time bounds for computing a maximum profit-to-time ratio cycle.

This section is concluded with a comment about implementation of Megiddo's parametric search. Throughout the whole computation of algorithm \mathcal{A}_1 , an interval (α, β) containing the optimum value δ^* should be maintained. Initially $(\alpha, \beta) = (0, +\infty)$. If during the computation the relation between a given number x and unknown δ^* has to be established, then it should be first checked if x belongs to (α, β) . If $x \notin (\alpha, \beta)$, then the relation between x and δ^* is known. If $x \in (\alpha, \beta)$, then algorithm \mathcal{A}_2 is run to find out whether $x = \delta^*$, $\delta^* \in (\alpha, x)$, or $\delta^* \in (x, \beta)$. If $x = \delta^*$, then the entire computation should be terminated, since the optimum objective value has just been found. In the other two cases, the interval (α, β) should be updated and the computation should proceed to the next iteration. Maintaining interval (α, β) does not improve the worst-case running times, but it should significantly improve the average running times.

6 Maximum Profit-to-Time Ratio Cycles

In this section we consider the following classical fractional combinatorial optimization problem, which we call the MaxRatioCycle problem. An input instance of this problem consists of a directed graph $G = (V, E)$, an edge-cost function $\mathbf{c} : E \rightarrow \mathbf{R}$, and an edge-weight function $\mathbf{w} : E \rightarrow \mathbf{R}$. Let n and m denote the number of vertices and the number of edges in graph G , respectively. As in the MaxRatioPath problem, if P is a path in G , then $f(P) = \sum_{(i,j) \in P} \mathbf{c}(i, j)$, $g(P) = \sum_{(i,j) \in P} \mathbf{w}(i, j)$, and $f(P)/g(P)$ are the cost, the weight, and the mean-weight cost of this path. The task is to find a cycle in graph G with the maximum mean-weight cost.

$$\text{MaxRatioCycle : } \text{maximize } \frac{f(\Gamma)}{g(\Gamma)} \text{ over all cycles } \Gamma \text{ in graph } G.$$

We assume that graph G is not acyclic, the weight of each cycle is positive, and there exists a cycle with positive cost. A cycle $\Gamma = (v_0, v_1, \dots, v_j = v_0)$ is simple if the vertices v_0, v_1, \dots, v_{j-1} are distinct. There are infinitely many cycles in G , but to find a cycle with the maximum mean-weight cost, we can limit the search only to simple cycles. If a cycle Γ is not simple, then its set of edges can be partitioned into a number of simple cycles $\Gamma_1, \Gamma_2, \dots, \Gamma_k$, and

$$\frac{f(\Gamma)}{g(\Gamma)} = \sum_{i=1}^k \frac{f(\Gamma_i)}{g(\Gamma)} = \sum_{i=1}^k \frac{g(\Gamma_i)}{g(\Gamma)} \cdot \frac{f(\Gamma_i)}{g(\Gamma_i)} \leq \max_{1 \leq i \leq k} \left\{ \frac{f(\Gamma_i)}{g(\Gamma_i)} \right\}.$$

The inequality holds because numbers $g(\Gamma_i)/g(\Gamma)$ are positive and sum up to 1. Therefore, there must be a simple cycle which has the maximum mean-weight cost

among all cycles. For convenience, however, instead of considering only simple cycles, we define the set of feasible solutions \mathcal{X}_G as the set of the cycles in graph G which contain at most n edges.

The MaxRatioCycle problem models the following *tramp-steamer* problem [14, 39]. A trip from a port v to a port u takes $\mathbf{w}(v, u)$ time and gives profit of $\mathbf{c}(v, u)$ units. To maximize the mean daily profit, the steamer should follow a cycle which maximizes the ratio of the total profit to the total time. The MaxRatioCycle problem is often called the *maximum profit-to-time ratio cycle problem*.

The parametric problem corresponding to the MaxRatioCycle problem is

$$\text{MaxCycle}(\delta) : \text{maximize } f(\Gamma) - \delta g(\Gamma), \text{ over all cycles } \Gamma \in \mathcal{X}_G.$$

An optimal solution of problem MaxCycle(δ) is a cycle from \mathcal{X}_G with the maximum cost according to the edge-cost function $\mathbf{c} - \delta\mathbf{w}$. To be able to directly apply the Newton method to the MaxRatioCycle problem, we need an algorithm which for given edge costs finds a cycle $\Gamma \in \mathcal{X}_G$ with the maximum cost. (Observe that restricting cycles to simple cycles only makes the problem NP-hard.) A weaker algorithm, which finds a positive-cost cycle, if there exists one, is sufficient for Megiddo's parametric search.

Let $\mathbf{a} : E \rightarrow \mathbf{R}$ be an arbitrary edge-cost function. Let $d_k(v, u)$ denote the maximum cost of a path from vertex v to vertex u containing at most k edges. If there is no path from v to u , then $d_k(v, u) = -\infty$. We have

$$d_1(v, u) = \begin{cases} \mathbf{a}(v, u), & \text{if } (v, u) \in E, \\ -\infty, & \text{if } (v, u) \notin E, \end{cases}$$

and

$$d_{2k}(v, u) = \max\{ d_k(v, u), \max_{x \in V} \{d_k(v, x) + d_k(x, u)\} \}. \quad (23)$$

To abstract from technical details, we assume that n is a power of 2. For n other than a power of 2, we could simply change the set of feasible solutions to the set of all cycles containing at most $2^{\lceil \log n \rceil}$ edges. The recursive relation (23) gives immediately an $O(n^3 \log n)$ algorithm for computing numbers $d_n(v, u)$, for each pair of vertices v and u . If during the computation the information about the numbers which define the maxima in Eq. (23) is stored, then a maximum-cost path from v to u can also be found, for each pair of vertices v and u . The maximum cost of a cycle $\Gamma \in \mathcal{X}_G$ is equal to $\max_{v \in V} \{d_n(v, v)\}$. Hence, we have an $O(n^3 \log n)$ algorithm which for a given δ solves problem MaxCycle(δ). Using this algorithm and the Newton method, we obtain an algorithm for the MaxRatioCycle problem which runs in $O(n^3 m^2 \log^3 n)$ time for arbitrary edge costs and weights (see [Theorem 5](#)) and in $O(n^3 \log n \log(nU))$ time, if all edge costs and weights are integers not greater than U (see [Theorem 2](#)). Using a similar analysis to the analysis presented for the MaxRatioPath problem in the proof of [Theorem 6](#), one can show that if the weights of the edges are nonnegative, then the Newton method solves the MaxRatioCycle problem in $O(m \log n)$ iterations and, consequently, in $O(n^3 m \log^2 n)$ time. This $O(m \log n)$ bound on the number of iterations for the

MaxRatioCycle problem, as well as for the MaxRatioPath problem, can be further improved to $O(m)$ (an example of analysis which can give such a bound is presented in Sect. 8). However, we do not include here a proof of this bound, since this bound anyway yields a slower algorithm than algorithms which can be obtained by using Megiddo's parametric search.

Let $\text{MaxCycle}_0(\delta)$ denote the problem of detecting a positive-cost cycle according to the edge-cost function $\mathbf{c} - \delta\mathbf{w}$, if there exists such a cycle. To apply Megiddo's parametric search to the MaxRatioCycle problem, we have to combine two algorithms for the parametric problem $\text{MaxCycle}_0(\delta)$: a linear algorithm \mathcal{A}_1 , which is to be run with unknown δ^* and should ideally have a small number of stages/phases, and a fast algorithm \mathcal{A}_2 used for resolving comparisons performed by the first algorithm (see Theorems 9 and 11). The problem of detecting a positive-cost cycle is equivalent to the problem of detecting a negative-cost cycle (simply negate the edge costs). All known algorithms for detecting a negative-cost cycle are actually based on single-source shortest-path algorithms. The best-known bound for shortest-path algorithms on graphs with arbitrary edge costs is $O(nm)$. This bound is achieved, for example, by the Bellman–Ford algorithm [2, 20]. Goldberg's shortest-path algorithm [23] runs in $O(m\sqrt{n} \log U)$, if the edge costs are integers greater than $-U$, for a positive number U . Thus, for arbitrary edge costs, the best-known bound on the running time of algorithm \mathcal{A}_2 is $O(nm)$. It turns out that the best candidate for algorithm \mathcal{A}_1 is the algorithm which is based on the computation of numbers $d_k(v, u)$ using the recursive relation (23). This algorithm runs in $O(n^3 \log n)$ time and consists of $O(\log n)$ maxima-computing phases, and the total size of all maxima in one phase is $O(n^3)$. Therefore, Theorem 11 implies that the MaxRatioCycle problem can be solved in $O(n^3 \log n + nm \log^2 \log \log n)$ time (part (i) of the theorem) and in $O(n^3 \log^2 n)$ time (part (ii) of the theorem).

7 Maximum-Mean Cycles

The maximum-mean cycle problem is the uniform MaxRatioCycle problem. An input instance of this problem consists of a directed graph $G = (V, E)$ and an edge-cost function $\mathbf{c} : E \rightarrow \mathbf{R}$. We want to find a maximum-mean cycle in G , that is, a cycle which has the maximum mean cost. The mean cost of a cycle Γ is equal to $(\sum_{e \in \Gamma} \mathbf{c}(e)) / |\Gamma|$. In this section we describe the $O(nm)$ algorithm for the maximum-mean cycle problem designed by Karp [34]. This algorithm is an example that there are specialized algorithms for some fractional combinatorial optimization problems, which are faster than algorithms yielded by general methods such as the Newton method and Megiddo's parametric search.

Let n and m denote the number of vertices and the number of edges in graph G . We assume that graph G is not acyclic. Let s be a vertex in graph G such that all other vertices are reachable from s . If such a vertex does not exist, we can add to graph G a new vertex s and edges (s, v) with arbitrary costs, for each $v \in V$. This modification does not change the cycles or their costs. For each vertex $v \in V$ and integer $k \geq 0$, let $d_k(v)$ be the maximum cost of a path of length k from vertex s to vertex v . If no such path exists, then $d_k(v) = -\infty$. For a subset of edges $A \subseteq E$,

$\mathbf{c}(A)$ denotes the sum $\sum_{e \in A} \mathbf{c}(e)$. Let δ^* denote the maximum mean cost of a cycle in graph G .

Theorem 12

$$\delta^* = \max \left\{ \min_{0 \leq k \leq n-1} \left\{ \frac{d_n(v) - d_k(v)}{n-k} \right\} : v \in V \text{ and } d_n(v) > -\infty \right\}. \quad (24)$$

Proof If we decrease the cost of each edge by the same amount α , then for each vertex $v \in V$ and each $k \geq 0$, $d_k(v)$ decreases by αk , and consequently, $(d_n(v) - d_k(v))/(n-k)$ decreases by α . Thus, both sides of Eq. (24) decrease by α . Therefore, it is enough to prove the theorem for the case when $\delta^* = 0$.

If $\delta^* = 0$, then Eq. (24) is equivalent to

$$0 = \max \left\{ \min_{0 \leq k \leq n-1} \{d_n(v) - d_k(v)\} : v \in V \text{ and } d_n(v) > -\infty \right\},$$

which is equivalent to

$$0 = \max \{ d_n(v) - \max_{0 \leq k \leq n-1} \{d_k(v)\} : v \in V \text{ and } d_n(v) > -\infty \}. \quad (25)$$

Let v be a vertex such that $d_n(v) > -\infty$, and let P be a path of length n and cost $d_n(v)$ from s to v . Path P is not simple (i.e., at least one vertex occurs on P more than once), so the edges of P can be partitioned into a cycle Γ and, possibly trivial, path R from s to v . Let $k = |R| \leq n-1$. Since $\delta^* = 0$, then $\mathbf{c}(\Gamma) \leq 0$, and we have

$$d_n(v) = \mathbf{c}(P) = \mathbf{c}(R) + \mathbf{c}(\Gamma) \leq \mathbf{c}(R) \leq d_k(v) \leq \max_{0 \leq k \leq n-1} \{d_k(v)\}.$$

Thus, the right-hand side of (25) is at most zero.

Now we show that the right-hand side of (25) is at least zero. Let Γ be a zero-cost cycle in graph G . Since $\delta^* = 0$, such a cycle must exist. Let v be an arbitrary vertex on cycle Γ , and let P_1 be a maximum-cost path from s to v . Such a path must exist because the graph does not have positive-cost cycles. Let P_2 be a path of length at least n which consists of path P_1 followed by some number of repetitions of cycle Γ . Since the cost of cycle Γ is equal to 0, the costs of paths P_1 and P_2 are the same, so path P_2 is a maximum-cost path from s to v . Let P_3 be the path which consists of the first n edges of path P_2 . Path P_3 is a maximum-cost path from vertex s to some vertex w , because any initial part of a maximum-cost path must be a maximum-cost path. Thus, we have

$$d_n(w) = \mathbf{c}(P_3) = \max_{0 \leq k \leq n-1} \{d_k(w)\}.$$

This means that the right-hand side of (25) is at least zero. □

Karp's algorithm for the maximum-mean cycle problem is based on [Theorem 12](#). Start with $d_0(s) = 0$ and $d_0(v) = -\infty$, for each $v \neq s$. Then in each phase $k = 1, 2, \dots, n$, numbers $d_k(v)$ for each $v \in V$ are computed using the following relation:

$$d_k(v) = \max\{d_{k-1}(u) + \mathbf{c}(u, v) : (u, v) \in E\}. \quad (26)$$

Knowing all numbers $d_k(v)$, for $v \in V$ and $0 \leq k \leq n$, δ^* can be computed in $O(n^2)$ time using Eq. (24). Let v be a vertex which gives the maximum in (24), and let P be a maximum-cost path from s to v of length n . Such a path can be constructed, if during the computation of numbers $d_k(x)$ the information about the edges which give the maxima in Eq. (26) is stored. Each cycle extracted from path P , and there must be at least one, is a maximum-mean cycle.

Karp's algorithm, if implemented in the straightforward way as described above, runs in $\Theta(nm)$ time: each edge is considered once in each of the n phases. Dasdan and Gupta [16] propose implementations of Karp's algorithm, which try to limit the number of edges considered in each phase, improving average running time. Hartmann and Orlin [28] extend Karp's algorithm to the maximum cost-to-time ratio cycles, for the case when the edge transit times (edge weights) are small integers. Orlin and Ahuja [45] show a scaling algorithm for the minimum (and maximum)-mean cycle problem with integral edge costs bounded by C , which runs in $O(\sqrt{nm} \log(nC))$ time.

8 Maximum Mean-Weight Cuts

In [Sect. 6](#) a problem was discussed (the MaxRatioCycle problem) for which the algorithms obtained by applying Megiddo's parametric search are considerably better than the algorithms yielded by the Newton method, at least for the worst-case inputs. In this section the MaxRatioCut problem is considered, which is an example of the reverse situation. Megiddo's parametric search method does not give fast algorithm for this problem, because there are no fast parallel algorithms for the maximum-flow problem, which is de facto the non-fractional version of the MaxRatioCut problem. The main aim of this section is to show a linear bound on the number of iterations of the Newton method for the MaxRatioCut problem with nonnegative edge weights. This linear bound gives the fastest known algorithm for this problem.

A network $G = (V, E, \mathbf{u}, \mathbf{d})$ is a directed, strongly connected graph with a set of vertices V , a set of edges E , a nonnegative edge-capacity function $\mathbf{u} : E \rightarrow \mathbf{R}$, and a demand function $\mathbf{d} : V \rightarrow \mathbf{R}$ such that $\sum_{v \in V} \mathbf{d}(v) = 0$. Without loss of generality, it is assumed that the set of edges is symmetric, that is, if $(x, y) \in E$, then also $(y, x) \in E$. If $\mathbf{d}(v)$ is negative, then v is a *source*—a node with supply. If $\mathbf{d}(v)$ is positive, then v is a *sink*—a node with demand. As before, n and m denote the number of vertices and the number of edges in network G , respectively.

For a subset of vertices $W \subseteq V$, $\mathbf{d}(W) \stackrel{\text{def}}{=} \sum_{v \in W} \mathbf{d}(v)$ is the net demand in W . If $S \subseteq V$, $T = V - S$, $S \neq \emptyset$, and $T \neq \emptyset$, then $\text{cut}(S, T)$ is the set of edges (x, y) such that $x \in S$ and $y \in T$. The *capacity* and the *surplus* of a cut (S, T) are defined as, respectively,

$$\mathbf{u}(S, T) \stackrel{\text{def}}{=} \sum_{e \in (S, T)} \mathbf{u}(e),$$

$$\text{surplus}(S, T) \stackrel{\text{def}}{=} \mathbf{d}(T) - \mathbf{u}(S, T).$$

If network G is viewed as a model for designing shipment of the commodity from the sites with supply (the sources) to the sites with demand (the sinks), then a positive $\text{surplus}(S, T)$ means that not all demand in T can be satisfied. The amount of the demand in T which cannot be satisfied without violating the edge capacities is at least $\text{surplus}(S, T)$.

If an *edge-weight function* $\mathbf{w} : E \rightarrow \mathbf{R}$ is given, then $g(S, T) \stackrel{\text{def}}{=} \sum_{e \in (S, T)} \mathbf{w}(e)$ and $\text{surplus}(S, T)/g(S, T)$ are the *weight* and the *mean-weight surplus* of cut (S, T) . The assumption that the underlying graph is strongly connected guarantees that each cut has at least one edge. It is further assumed, as in the general model of fractional combinatorial optimization, that the edge weights are such that the weight of each cut is positive. The MaxRatioCut problem is the problem of finding a cut with the maximum mean-weight surplus in a given network G and for a given edge-weight function \mathbf{w} . The corresponding parametric problem MaxSurplusCut(δ) is the problem of finding a maximum-surplus cut in network $G_\delta = (V, E, \mathbf{u} + \delta\mathbf{w}, \mathbf{d})$, that is, in network G with the edge-capacity function changed to function $\mathbf{u} + \delta\mathbf{w}$. The problem of finding a maximum-surplus cut in network G can be reduced in $O(m)$ time to the problem of computing a maximum flow in network G . All known algorithms for computing maximum-surplus cuts are based on maximum-flow algorithms. Let $T_{\text{MaxFlow}}(n, m)$ denote the worst-case time complexity of computing a maximum flow in a network with n vertices and m edges. The results from [24, 36] imply that

$$T_{\text{MaxFlow}}(n, m) = O(\min\{nm \log(n^2/m), nm + n^{2+\epsilon}\}), \quad (27)$$

where ϵ is an arbitrary positive constant.

The MaxMeanCut problem is the uniform MaxRatioCut problem, the special case of the MaxRatioCut problem when all edge weights are equal to 1. The MaxMeanCut and the MaxRatioCut problems appear, for example, in the context of the classical *minimum-cost flow problem*. Goldberg and Tarjan [25] showed a simple strongly polynomial iterative method for solving the minimum-cost flow problem, which is based on computation of minimum-mean cycles. Erolina and McCormick [19] (see also [51]) showed an analogous method for solving the dual problem of the minimum-cost flow problem, which is based on computation of maximum mean-surplus cuts. Wallacher [66] used minimum mean-weight cost cycles in a generalization of Goldberg and Tarjan's method. Analogously,

maximum mean-weight surplus cuts can be used in a generalization of Ervolina and McCormick's method.

Theorem 8 implies that Megiddo's parametric search solves the MaxRatioCut problem in $O((T_{\text{MaxFlow}}(n, m))^2) = O(n^2 m^2 \log^2 n)$ time. The first parallel algorithm for the maximum-flow problem is due to Schiloach and Vishkin [63]. This algorithm runs in $O(n^2 \log n)$ time and uses n processors. **Theorem 10** implies that Megiddo's parametric search based on Schiloach and Vishkin's algorithm solves the MaxRatioCut problem in $O(n^3 m \log^3 n)$ time. So far no parallel maximum-flow algorithm has been designed, which would lead to a better bound for Megiddo's parametric search for the MaxRatioCut problem than $O^*(n^3 m)$. Notation O^* () hides a poly-logarithmic factor.

Theorem 5 gives an $O(m^2 \log^2 n)$ bound on the number of iterations of the Newton method for the MaxRatioCut problem. In the remaining part of this section, it is shown that special properties of the MaxRatioCut problem, notably the "maximum flow-minimum cut" duality, imply that the number of iterations is actually only $O(m)$, provided that the edge weights are nonnegative. This bound implies that the Newton method solves the MaxRatioCut problem with nonnegative edge weights in $O(m T_{\text{MaxFlow}}(n, m)) = O(n m^2 \log n)$ time. Observe that in the case of the MaxMeanCut problem, an $O(m)$ bound on the number of iterations follows from **Theorem 3** (in this case a stronger $O(n)$ bound can be shown, see [50]). From now on it is assumed that the edge weights are nonnegative.

A function $\mathbf{r} : E \rightarrow \mathbf{R}$ is a *flow* in network G if for each edge $(x, y) \in E$,

$$\begin{aligned}\mathbf{r}(x, y) &\leq \mathbf{u}(x, y), \\ \mathbf{r}(x, y) &= -\mathbf{r}(y, x).\end{aligned}$$

For a flow \mathbf{r} , if $A \subseteq E$, then $\mathbf{r}(A) \stackrel{\text{def}}{=} \sum_{e \in A} \mathbf{r}(e)$, and if $v \in V$, then $\mathbf{r}^{(\text{in})}(v)$ denotes the net flow into vertex v :

$$\mathbf{r}^{(\text{in})}(v) \stackrel{\text{def}}{=} \sum_{(v, x) \in E} \mathbf{r}(v, x).$$

Define $\mathbf{u}_\mathbf{r} \stackrel{\text{def}}{=} \mathbf{u} - \mathbf{r}$, $\mathbf{d}_\mathbf{r} \stackrel{\text{def}}{=} \mathbf{d} - \mathbf{r}^{(\text{in})}$ and, for a cut (S, T) ,

$$\text{surplus}_\mathbf{r}(S, T) \stackrel{\text{def}}{=} \mathbf{d}_\mathbf{r}(T) - \mathbf{u}_\mathbf{r}(S, T).$$

Values $\mathbf{u}_\mathbf{r}(e)$ and $\mathbf{d}_\mathbf{r}(v)$ are commonly called the residual capacity of edge e and the residual demand at vertex v with respect to flow \mathbf{r} . A flow \mathbf{r} is a *maximum flow* if it minimizes $\sum_{v \in V} |\mathbf{d}_\mathbf{r}(v)|$. The following facts from the network-flows theory are needed. Let \mathbf{r} , \mathbf{r}_{\max} , and (S_{\max}, T_{\max}) be a flow, a maximum flow, and a maximum-surplus cut in network $G = (V, E, \mathbf{u}, \mathbf{d})$.

- F1. For each cut (S, T) , $\text{surplus}_\mathbf{r}(S, T) = \text{surplus}(S, T)$.
- F2. $\mathbf{u}_{\mathbf{r}_{\max}}(S_{\max}, T_{\max}) = 0$.

- F3. For each $T \subseteq V$, $\mathbf{d}_{\mathbf{r}_{\max}}(T) \leq \mathbf{d}_{\mathbf{r}_{\max}}(T_{\max})$.
- F4. Let a new edge-capacity function \mathbf{u}' be such that $\mathbf{u}'(e) \geq \mathbf{u}(e)$, for each edge $e \in E$. There exists a maximum flow \mathbf{r}'_{\max} in network $G' = (V, E, \mathbf{u}', \mathbf{d})$ such that,
1. For each vertex $v \in V$, $d_{\mathbf{r}'_{\max}}(v)$ and $d_{\mathbf{r}_{\max}}(v)$ have the same sign;
 2. For each edge $e \in E$, $|\mathbf{r}'_{\max}(e) - \mathbf{r}_{\max}(e)| \leq d_{\mathbf{r}_{\max}}(T_{\max})$; and
 3. For each cut (S, T) in G , $|\mathbf{r}'_{\max}(S, T) - \mathbf{r}_{\max}(S, T)| \leq d_{\mathbf{r}}(T_{\max})$.

These facts can be derived from the properties of network flows described in [12, 46, Chapter 27]. Fact F1 follows actually directly from the introduced definitions (for each flow \mathbf{r} and cut (S, T) , the differences $\mathbf{d}(T) - \mathbf{d}_{\mathbf{r}}(T)$ and $\mathbf{u}(S, T) - \mathbf{u}_{\mathbf{r}}(S, T)$ are the same). Facts F2 and F3 are closely related to the maximum-flow/minimum-cut theorem. Informally speaking, the conditions listed in Fact F4 are satisfied by a maximum flow \mathbf{r}'_{\max} in network G' which “extends” the maximum flow \mathbf{r}_{\max} in network G .

The analysis of the Newton method for the MaxRatioCut problem uses the same notation as in Sect. 3, where this method was introduced. Thus,

$$\begin{aligned} f_i &= \text{surplus}(S_i, T_i), \\ g_i &= g(S_i, T_i), \\ h_i = h(\delta_i) &= \text{surplus}(S_i, T_i) - \delta_i g(S_i, T_i) = \text{surplus}_{\delta_i}(S_i, T_i), \end{aligned} \quad (28)$$

where (S_i, T_i) is the maximum-surplus cut in network $G_{\delta_i} = (V, E, \mathbf{u} + \delta_i \mathbf{w}, \mathbf{d})$ computed in iteration i . Subscript δ will always indicate that the underlying network is network G_δ , that is, network G with the edge-capacity function changed to function $\mathbf{u} + \delta \mathbf{w}$. In particular,

$$\begin{aligned} \text{surplus}_\delta(S, T) &\stackrel{\text{def}}{=} \mathbf{d}(T) - (\mathbf{u} + \delta \mathbf{w})(S, T) \\ &= \text{surplus}(S, T) - \delta g(S, T), \end{aligned} \quad (29)$$

$$\text{surplus}_{\delta, \mathbf{r}}(S, T) \stackrel{\text{def}}{=} \mathbf{d}_{\mathbf{r}}(T) - \mathbf{u}_{\delta, \mathbf{r}}(S, T), \quad (30)$$

where \mathbf{r} is a flow in network G_δ . The analysis in this section is similar to the analysis of the Newton method for the MaxRatioPath problem presented in the proof of Theorem 6. An edge e is said to be *essential* at the beginning of iteration i , if it belongs to a cut (S_j, T_j) , for some $j \geq i$. As the computation proceeds, the number of essential edges decreases and the rate of this decrement is to be analyzed. As in the proofs of the other bounds on the number of iterations of the Newton method, Lemma 2 and its direct consequence expressed in Inequality (7) are used to measure the progress of the computation.

Let \mathbf{r}_i , for $i = 1, 2, \dots, t$ (t is the index of the last iteration), be maximum flows in networks G_{δ_i} such that for each i , all conditions of Fact F4 above are satisfied with $G = G_{\delta_i}$, $G' = G_{\delta_{i+1}}$, $\mathbf{r}_{\max} = \mathbf{r}_i$, and $\mathbf{r}'_{\max} = \mathbf{r}_{i+1}$. Using (30), Facts F1 and F2, and then (28),

$$\begin{aligned}
\mathbf{d}_{\mathbf{r}_i}(T_i) &= \text{surplus}_{\delta_i, \mathbf{r}_i}(S_i, T_i) - \mathbf{u}_{\delta_i, \mathbf{r}_i}(S_i, T_i) \\
&= \text{surplus}_{\delta_i}(S_i, T_i) \\
&= h_i.
\end{aligned} \tag{31}$$

The following lemma gives a simple, sufficient condition for an edge not to be essential.

Lemma 7 *If*

$$\mathbf{u}_{\delta_i, \mathbf{r}_i}(e) > h_i, \tag{32}$$

then edge e is not essential at the beginning of iteration i .

Proof Let (S, T) be a cut containing an edge e which satisfies Inequality (32). Using Facts F1 and F3 and Eq. (31),

$$\begin{aligned}
\text{surplus}_{\delta_i}(S, T) &= \text{surplus}_{\delta_i, \mathbf{r}_i}(S, T) = \mathbf{d}_{\mathbf{r}_i}(T) - \mathbf{u}_{\delta_i, \mathbf{r}_i}(S, T) \\
&\leq \mathbf{d}_{\mathbf{r}_i}(T_i) - \mathbf{u}_{\delta_i, \mathbf{r}_i}(S, T) = h_i - \mathbf{u}_{\delta_i, \mathbf{r}_i}(S, T) \\
&\leq h_i - \mathbf{u}_{\delta_i, \mathbf{r}_i}(e) < 0.
\end{aligned}$$

Therefore, none of the cuts (S_j, T_j) , for $j \geq i$, can contain edge e , because $\delta_j \geq \delta_i$ and

$$\text{surplus}_{\delta_i}(S_j, T_j) \geq \text{surplus}_{\delta_j}(S_j, T_j) = h_j \geq 0.$$

This means that edge e is not essential at the beginning of iteration i . \square

The proof of the $O(m)$ bound on the number of iterations shown below in **Theorem 14** involves quite a few technical details. Therefore, to highlight the main line of the argumentation, a simpler $O(m \log m)$ bound is first shown.

Theorem 13 *The Newton method solves the MaxRatioCut problem in $O(m \log m)$ iterations, if the edge weights are nonnegative.*

Proof The proof shows that a sequence of $k = \lfloor \log m \rfloor + 2$ consecutive iterations decreases the number of essential edges at least by 1. This claim immediately implies the $O(m \log m)$ bound on the number of iterations. Consider iterations $i, i+1, \dots, i+k$. Inequality (7) implies that

$$h_{i+k} g_{i+k} < \frac{1}{m^2} h_{i+1} g_{i+1}. \tag{33}$$

If $g_{i+k} < (1/m)g_i$, then, using the assumption that the edge weights are nonnegative, for some edge $e \in (S_i, T_i)$,

$$\mathbf{w}(e) \geq \frac{1}{|(S_i, T_i)|} \sum_{a \in (S_i, T_i)} \mathbf{w}(a) \geq \frac{1}{m} g_i > g_{i+k}.$$

Such an edge e cannot belong to cut (S_{i+k}, T_{i+k}) or to any cut (S_j, T_j) , for $j \geq i + k$ (since sequence (g_l) is nonincreasing), so e is essential at the beginning of iteration i but is not essential at the beginning of iteration $i + k$.

If $g_{i+k} \geq (1/m)g_i$, then also $g_{i+k} \geq (1/m)g_{i+1}$, and Inequality (33) implies that $h_{i+k} < (1/m)h_{i+1}$. Fact F3 implies that for each vertex $v \in T_i$, $d_{r_i}(v)$ is nonnegative. Hence, Fact F4(8) further implies that for each vertex $v \in T_i$, $d_{r_{i+k}}(v)$ is nonnegative, so $d_{r_{i+k}}(T_i)$ must be nonnegative as well. Therefore, using Lemma 6 and Fact F1,

$$\begin{aligned} -mh_{i+k} &> -h_{i+1} \geq f_i - \delta_{i+k} g_i \\ &= \text{surplus}(S_i, T_i) - \delta_{i+k} g(S_i, T_i) \\ &= \text{surplus}_{\delta_{i+k}}(S_i, T_i) \\ &= \text{surplus}_{\delta_{i+k}, r_{i+k}}(S_i, T_i) \\ &= \mathbf{d}_{r_{i+k}}(T_i) - \mathbf{u}_{\delta_{i+k}, r_{i+k}}(S_i, T_i) \\ &\geq -\mathbf{u}_{\delta_{i+k}, r_{i+k}}(S_i, T_i). \end{aligned} \quad (34)$$

The above inequality implies that there is an edge $e \in (S_i, T_i)$ such that $\mathbf{u}_{\delta_{i+k}, r_{i+k}}(e) > h_{i+k}$. Such an edge is essential at the beginning of iteration i , but Lemma 7 implies that it is not essential at the beginning of iteration $i + k$. \square

Theorem 14 *The Newton method solves the MaxRatioCut problem in $O(m)$ iterations, if the edge weights are nonnegative.*

Proof Lemma 2 implies that for each iteration i except the last one,

$$\frac{g_{i+1}}{g_i} \leq \frac{1}{2} \quad (35)$$

or

$$\frac{h_{i+1}}{h_i} \leq \frac{1}{2}. \quad (36)$$

In the proof of this theorem, separate bounds are derived for the number of iterations when Inequality (35) holds and for the number of iterations when Inequality (36) holds. Let i_1, i_2, \dots, i_q be the indices of the iterations for which Inequality (35) holds. Let $\mu_j = g_{i_j}$, for $j = 1, 2, \dots, q$, and let $(\alpha_k)_{k=1}^m$ be the sequence of the edge weights in nonincreasing order. Sequences $(\alpha_k)_{k=1}^m$ and $(\mu_j)_{j=1}^q$ satisfy the conditions of Lemma 8 below, so $q \leq m$.

Now an upper bound on the number of iterations when Inequality (36) holds is established. Here the argument is more involved, because numbers h_i are not

simply subsums of a fixed set of m numbers. From now on only iterations when Inequality (36) holds are considered. To avoid towering subscripts, these iterations are numbered from 1 to $t' \leq t$. Thus, “iteration i ” and subscripts i refer now to the i th iteration for which Inequality (36) holds. In the remaining of the proof, the same notation and definitions as in the proof of [Theorem 13](#) are used, but the new numbering of the iterations is taken into account.

Consider cut $(S_i, T_i) = \{e_1, e_2, \dots, e_p\}$ computed in iteration i . For technical reasons, it is assumed that there are at least $\log p + 5$ iterations following iteration i , that is, $t' \geq i + \log p + 5$. The proof of [Theorem 13](#) was based on showing that at least one edge from cut (S_i, T_i) becomes unessential by the beginning of iteration $i + \lfloor \log m \rfloor + 2$. Here, it is shown that there exists $l \geq 3$ such that at least $l - 2$ edges from cut (S_i, T_i) become unessential by the beginning of iteration $i + l + 2$. This claim immediately implies an $O(m)$ bound on the number of iterations.

Fact F4(2), Eq. (31), and Inequality (36) imply that for each edge $e \in E$ and index l such that $i + 3 \leq l \leq t'$,

$$\begin{aligned} |\mathbf{u}_{\delta_{i+3}, \mathbf{r}_{l+1}}(e) - \mathbf{u}_{\delta_{i+3}, \mathbf{r}_l}(e)| &= |\mathbf{r}_{l+1}(e) - \mathbf{r}_l(e)| \\ &\leq \mathbf{d}_{\mathbf{r}_l}(T_l) = h_l \\ &\leq \frac{1}{2^{l-i-2}} h_{i+2}. \end{aligned} \quad (37)$$

Analogously, using Fact F4(3), for each cut (S, T) and index l such that $i + 3 \leq l \leq t'$,

$$|\mathbf{u}_{\delta_{i+3}, \mathbf{r}_{l+1}}(S, T) - \mathbf{u}_{\delta_{i+3}, \mathbf{r}_l}(S, T)| \leq \frac{1}{2^{l-i-2}} h_{i+2}. \quad (38)$$

The derivations (34) for $k = 3$ and Inequality (36) give

$$\mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+3}}(S_i, T_i) \geq h_{i+1} \geq 2h_{i+2}. \quad (39)$$

Using this inequality and Inequality (38), for each index l such that $i + 4 \leq l \leq t'$,

$$\begin{aligned} \mathbf{u}_{\delta_{i+3}, \mathbf{r}_l}(S_i, T_i) &= \mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+3}}(S_i, T_i) \\ &\quad + (\mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+4}}(S, T) - \mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+3}}(S, T)) \\ &\quad + \cdots + (\mathbf{u}_{\delta_{i+3}, \mathbf{r}_l}(S, T) - \mathbf{u}_{\delta_{i+3}, \mathbf{r}_{l-1}}(S, T)) \\ &\geq \mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+3}}(S_i, T_i) - h_{i+2} \left(\frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^{l-i-3}} \right) \\ &\geq h_{i+2}. \end{aligned} \quad (40)$$

For each $l = 1, 2, \dots, t' - i - 2$ and $j = 1, 2, \dots, p$, let

$$\alpha_{l,j} = \frac{1}{h_{i+2}} \mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+2+l}}(e_j).$$

Inequalities (39) and (40) imply that matrix $(\alpha_{l,j})$ satisfies Condition 2 of 9 below (since $\sum_{j=1}^p \alpha_{l,j} = \mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+2+l}}(S_i, T_i) / h_{i+2}$). Inequality (37) implies that Condition 3 is satisfied as well. Condition 1 holds because it was assumed that $t' \geq i + \log p + 5$. Thus, Lemma 9 can be applied to matrix $(\alpha_{l,j})$. According to the definition of “good entries” in the statement of Lemma 9, if $\alpha_{l,j}$ is a good entry, then there exists $1 \leq l' \leq l$ such that $\alpha_{l',j} > 1/2^{l'}$. If $\alpha_{l',j} > 1/2^{l'}$, then

$$\mathbf{u}_{\delta_{i+2+l'}, \mathbf{r}_{i+2+l'}}(e_j) \geq \mathbf{u}_{\delta_{i+3}, \mathbf{r}_{i+2+l'}}(e_j) = h_{i+2}\alpha_{l',j} > \frac{1}{2^{l'}} h_{i+2} \geq h_{i+l'+2},$$

so Lemma 7 implies that edge e_j is unessential at the beginning of iteration $i+l+2$. Lemma 9 implies that there exists $l \geq 3$ such that there are at least $l-2$ good entries among entries $\alpha_{l,1}, \alpha_{l,2}, \dots, \alpha_{l,p}$. The $l-2$ edges corresponding to these $l-2$ entries are essential at the beginning of iteration i but become unessential by iteration $i+l+2$. \square

It remains to prove the following two technical lemmas.

Lemma 8 *Let $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_m \geq 0$ and $\mu_1 > \mu_2 > \dots > \mu_q > 0$ be such that*

1. $\mu_{j+1} \leq (1/2)\mu_j$, for $j = 1, 2, \dots, q-1$, and
2. $\mu_j \leq \sum \{\alpha_k \mid \alpha_k \leq \mu_j\}$, for $j = 1, 2, \dots, q$.

Then, $q \leq m$.

Proof Let $\bar{\alpha}_k = \alpha_k + \alpha_{k+1} + \dots + \alpha_m$. Condition 2 implies that $\mu_1 \leq \bar{\alpha}_1$, and if $\mu_j < \bar{\alpha}_m = \alpha_m$, then $\mu_j \leq 0$. Thus, all numbers μ_j lie in the interval $[\bar{\alpha}_m, \bar{\alpha}_1]$. To prove that $q \leq m$, it is shown that each of the intervals $(\bar{\alpha}_m, \bar{\alpha}_{m-1}]$ and $(\bar{\alpha}_{m-1}, \bar{\alpha}_{m-2}], \dots, (\bar{\alpha}_2, \bar{\alpha}_1]$ contains at most one element from the sequence (μ_j) . Let $\mu_j \in (\bar{\alpha}_{k+1}, \bar{\alpha}_k]$, for some $1 \leq j \leq q-1$ and $1 \leq k \leq m-1$. It is shown next that $\mu_{j+1} \leq \bar{\alpha}_{k+1}$, so $\mu_{j+1} \notin (\bar{\alpha}_{k+1}, \bar{\alpha}_k]$. If $\bar{\alpha}_{k+1} \geq (1/2)\bar{\alpha}_k$, then

$$\mu_{j+1} \leq \frac{1}{2}\mu_j \leq \frac{1}{2}\bar{\alpha}_k \leq \bar{\alpha}_{k+1}.$$

If $\bar{\alpha}_{k+1} < (1/2)\bar{\alpha}_k$, then

$$\mu_{j+1} \leq \frac{1}{2}\mu_j \leq \frac{1}{2}\bar{\alpha}_k = \bar{\alpha}_k - \frac{1}{2}\bar{\alpha}_k = \alpha_k + \bar{\alpha}_{k+1} - \frac{1}{2}\bar{\alpha}_k < \alpha_k.$$

The above inequality and Condition 2 imply that the inequality $\mu_{j+1} \leq \bar{\alpha}_{k+1}$ holds also in this case. \square

Lemma 9 *Let $(\alpha_{l,j})$ be a $q \times p$ matrix such that*

1. $q \geq \log p + 3$,
2. $\sum_{j=1}^p \alpha_{l,j} \geq 1$, for each $1 \leq l \leq q$, and
3. $|\alpha_{l+1,j} - \alpha_{l,j}| \leq 1/2^l$, for each $1 \leq l \leq q-1$ and $1 \leq j \leq p$.

If $\alpha_{l,j} > 1/2^l$, then this and all subsequent entries in column j are “good entries.” There exists l such that $3 \leq l \leq q$ and row l of the matrix contains at least $l - 2$ good entries.

Proof Condition 3 implies that if $1 \leq l' < l'' \leq q$ and $1 \leq j \leq p$, then

$$\begin{aligned}\alpha_{l'',j} &\leq \alpha_{l',j} + \frac{1}{2^{l'}} + \frac{1}{2^{l'+1}} + \cdots + \frac{1}{2^{l''-1}} \\ &< \alpha_{l',j} + \frac{1}{2^{l'-1}}.\end{aligned}\tag{41}$$

If $\alpha_{l,j}$ is the first good entry in column j and $l \geq 2$, then

$$\alpha_{l,j} \leq \alpha_{l-1,j} + \frac{1}{2^{l-1}} \leq \frac{1}{2^{l-1}} + \frac{1}{2^{l-1}} = \frac{1}{2^{l-2}}.\tag{42}$$

The first inequality above follows from Condition 3, and the second one holds because entry $\alpha_{l-1,j}$ is not a good entry.

Assume that for each $3 \leq l \leq q$, row l contains at most $l - 3$ good entries. (In particular, there are no good entries in rows 1–3.) The proof proceeds by showing that this assumption implies that the sum of the entries in the last row is less than 1, which contradicts Condition 2 of the lemma. Let j_1, j_2, \dots, j_r be the indices of all columns which have at least one good entry. Let $\alpha_{l_1,j_1}, \alpha_{l_2,j_2}, \dots, \alpha_{l_r,j_r}$ be the first good entries in these columns. Let j_1, j_2, \dots, j_r be ordered in such a way that $l_1 \leq l_2 \leq \cdots \leq l_r$. Hence, for each $k = 1, 2, \dots, r$, entries $\alpha_{l_k,j_1}, \alpha_{l_k,j_2}, \dots, \alpha_{l_k,j_k}$ are good entries, so row l_k contains at least k good entries. Row l_k contains at most $l_k - 3$ good entries (from the assumption), so

$$k \leq l_k - 3.\tag{43}$$

The sum of the entries in the last row is at most

$$\begin{aligned}&\frac{p-r}{2^q} + \alpha_{q,j_1} + \alpha_{q,j_2} + \cdots + \alpha_{q,j_r} \\ &\leq \frac{p}{2^q} + \left(\alpha_{l_1,j_1} + \frac{1}{2^{l_1-1}} \right) + \cdots + \left(\alpha_{l_r,j_r} + \frac{1}{2^{l_r-1}} \right) \\ &\leq \frac{1}{8} + \left(\frac{1}{2^{l_1-2}} + \frac{1}{2^{l_1-1}} \right) + \cdots + \left(\frac{1}{2^{l_r-2}} + \frac{1}{2^{l_r-1}} \right) \\ &= \frac{1}{8} + 6 \left(\frac{1}{2^{l_1}} + \cdots + \frac{1}{2^{l_r}} \right) \\ &\leq \frac{1}{8} + 6 \left(\frac{1}{2^4} + \frac{1}{2^5} + \cdots \right) \\ &< 1.\end{aligned}$$

The first inequality follows from Inequality (41), the second one from Condition 1 and Inequality (42), and the third one from Inequality (43). \square

9 Conclusion

The binary search, the Newton method, and Megiddo's parametric search form together a powerful set of methods for solving fractional combinatorial optimization problems. All three methods reduce a fractional problem to a sequence of instances of the corresponding non-fractional problem. Implementations of the binary search and the Newton method are straightforward once a procedure for solving the non-fractional problem is provided. Implementation of Megiddo's parametric search is a bit more complicated, because this method, unlike the other two, does not treat a procedure for the underlying non-fractional problem as a black box but has to examine its structure and modify it appropriately.

Megiddo originally proposed his parametric search method in the context of fractional combinatorial optimization problems such as the minimum-ratio spanning-tree problem and the maximum profit-to-time cycle problem [41]. Since then, however, his method has been extended to far more general optimization problems; see for example [9, 10, 27, 44, 64]. If Megiddo's parametric search method is used to obtain a fast algorithm for a given fractional combinatorial optimization problem, then the main task is to find a parallel algorithm for the underlying non-fraction problem which exhibits a right trade-off between the parallel running time and the total amount of computation performed.

The Newton method for general fractional optimization has been extensively studied since Dinkelbach [18] introduced it in 1967, and a number of modifications and extensions of this method have been proposed and analyzed [30, 47, 54]. The strongly polynomial bounds on the number of iterations in the Newton method for fractional combinatorial optimization presented in Sects. 4 and 8 come from early 1990s [49, 50], but it is still not clear how tight those bounds are. The $O(p^2 \log^2 p)$ bound shown in Theorem 5 can be improved to $O(p^2 \log p)$, as shown by Wang et al. [67], but the best-known worst-case lower bound on the number of iteration is only $\Omega(p \log p)$. The main tool used in Sect. 4 in the derivation of the $O(p^2 \log^2 p)$ bound is Lemma 3, which gives an $O(p \log p)$ bound on the length of a geometric sequence of subsums of a p -element set. The same lemma is also the basis for the $O(p^2 \log p)$ bound in [67]. Goldmann (M. Goldmann, On a subset-sum problem, 1994, personal communication) shows that the $O(p \log p)$ bound in Lemma 3 is tight, but there may be a better way of using it in the analysis of the Newton method.

For problems such as MaxRatioPath, MaxRatioCycle, and MaxRatioCut, the Newton method requires only a linear number of iterations, provided that the edge weights are nonnegative, and the proofs of these three bounds are remarkably similar to each other (compare the proofs of Theorems 6 and 13). However, a unified

framework which would generalize these three examples has not been proposed yet. Such a framework would have to be based on the “primal-dual” structure of problems such as the three problems mentioned here. Another related task is to examine if the assumption that the edge weights are nonnegative is really necessary to obtain a liner bound on the number of iterations.

The Newton method for fractional combinatorial optimization is an application of the classical Newton method to compute the root of a piecewise linear, convex function h defined in (3). The number of linear segments of function h is an obvious upper bound on the number of iterations of this method. The artificial example presented in Sect. 4 shows that function h may consist of an exponential number of linear segments, even for a linear fractional combinatorial optimization problem. The results of Gusfield [26] (upper bound) and Carstensen [6] (lower bound) imply that for the MaxRatioPath and the MaxRatioCycle problems, function h consists, in the worst case, of $n^{\Theta(\log n)}$ linear segments. Such tight bounds, however, are known only for relatively few problems. For example, no tight bounds on the complexity of function h for the MaxRatioCut problem have been reported yet.

The Newton method constructs a sequence of improving lower bounds converging to the optimum objective value, but since it does not provide upper bounds, no bound on the error is available at any given iteration. There are methods for general fractional optimization, which are based on the Newton method but also construct improving upper bounds (see, e.g., [30, 47, 54]). It is not clear yet if these methods can also lead to interesting results for fractional combinatorial optimization.

The methods for solving fractional combinatorial optimization problems rely on methods for solving the corresponding non-fractional optimization problems. The underlying non-fractional problem may however be difficult itself and might not have an efficient exact algorithm. Hashizume et al. [29] analyze Megiddo’s parametric search method for maximizing $(a_0 + \mathbf{ax})/(b_0 + \mathbf{bx})$, when the algorithm for the underlying non-fractional problem of maximizing $(\mathbf{a} - \delta\mathbf{b})\mathbf{x}$ returns only approximate solutions. They show that, under appropriate assumptions, the accuracy of the approximate solution obtained for the fractional problem is at least as good as the accuracy of the solutions computed for the non-fractional problem. As an example of their analysis, they present a fully polynomial approximation scheme for the fractional 0–1 knapsack problem. Billionet [3] presents a different way of extending Megiddo’s method to approximation algorithms, which relaxes some assumptions needed in [29] and leads to improved approximation algorithms for fractional knapsack problems.

Both Hashizume et al. [29] and Billionet [3] need for their approximation scheme for fractional optimization a good approximation algorithm for the corresponding linear optimization, which may involve negative coefficients even if all coefficients in the fractional objective function are positive (the objective function of problem (2) for a fixed value of parameter δ may have negative coefficients). This is not an issue for the (linear) knapsack problem since the items with negative values can be simply discarded, but there are linear optimization problems which admit approximation schemes only for the case when all coefficients are nonnegative. Correa et al. [13]

discusses examples of such problems and proposes an extension of Megiddo's method which is based on approximation algorithms for linear optimization with nonnegative coefficients. As an application of their method, they show improved approximation algorithms for finding a minimum average weight k -connected subgraphs.

In the binary search scheme described in Sect. 2, it may be possible to use a faster approximation algorithm $\tilde{\mathcal{A}}_0$ instead of the exact algorithm \mathcal{A}_0 while still guaranteeing that the interval (α, β) with the optimal value δ^* shrinks by a constant factor in each iteration. Examples of such approximate binary search for fractional combinatorial optimization include algorithms for the maximum mean-cut problem proposed by McCormick [40] and Iwano et al. [32] and an algorithm for the fractional assignment problem proposed by Shigeno et al. [62]. Kabadi and Aneja [33] extend the Newton method to approximation algorithms and give a strongly polynomial bound on the number of iterations.

Both the Newton method and Megiddo's parametric search require a good algorithm for solving the corresponding parametric problem (2) for any fixed value of parameter δ . The parametric problem (2), however, might not have a more convenient objective function than the original fractional problem. For example, for a fractional *cost-to-reliability ratio* problem

$$\text{CostToReliability : } \begin{aligned} & \text{minimize} && \frac{c_1x_1 + c_2x_2 + \cdots + c_px_p}{r_1^{x_1}r_2^{x_2}\cdots r_p^{x_p}}, \\ & \text{for } \mathbf{x} \in \mathcal{X} \subseteq \{0, 1\}^p, \end{aligned}$$

where each cost c_i is nonnegative and each reliability factor r_i is in $(0, 1]$, the corresponding parametric problem (2) is

$$\mathcal{P}(\delta) : \text{minimize } c_1x_1 + c_2x_2 + \cdots + c_px_p - \delta r_1^{x_1}r_2^{x_2}\cdots r_p^{x_p}, \text{ for } \mathbf{x} \in \mathcal{X}.$$

If the objective function of the above parametric problem turns out to be too difficult, then the following simpler parameterization may be more helpful:

CostToReliability(δ) :

$$\begin{aligned} & \text{minimize} && c_1x_1 + c_2x_2 + \cdots + c_px_p + \delta(d_1x_1 + d_2x_2 + \cdots + d_px_p), \\ & \text{for } \mathbf{x} \in \mathcal{X}, \end{aligned}$$

where $d_i = -\ln r_i$. It can be shown that there is a value $\hat{\delta}$ such that an optimal solution of the $\text{CostToReliability}(\hat{\delta})$ problem is an optimal solution of the CostToReliability problem. Thus, the CostToReliability problem can be solved by solving $\text{CostToReliability}(\delta)$ over the whole range of δ ($\hat{\delta}$ exists, but it is not known how $\hat{\delta}$ could be pre-calculated). This approach was used by Ahuja [1] for computing the minimum cost-to-reliability path and by Chandrasekaran et al. [8]

for computing the minimum cost-to-reliability spanning tree but may be impractical for other problems, if the parametric problem $\text{CostToReliability}(\delta)$ has a large (super-polynomial) number of solutions. Katoh [35] proposed a general approximation scheme for the CostToReliability problem, which is based on the parametric problem $\text{CostToReliability}(\delta)$.

The optimal solutions of a fractional combinatorial optimization problem of maximizing $f(\mathbf{x})/g(\mathbf{x})$, for $\mathbf{x} \in \mathcal{X} \subseteq \{0, 1\}^p\}$, belong to the *Pareto-optimal frontier* $P \subseteq \mathcal{X}$ of the bi-objective optimization problem of maximizing $f(\mathbf{x})$ and minimizing $g(\mathbf{x})$, for $\mathbf{x} \in \mathcal{X}$. A solution $\mathbf{x} \in \mathcal{X}$ belongs to P if, and only if, for each $\mathbf{y} \in \mathcal{X}$, $f(\mathbf{x}) \geq f(\mathbf{y})$ or $g(\mathbf{x}) \leq g(\mathbf{y})$. Mittal and Schulz [43] propose the following nonparametric approximation framework: compute a (suitably defined) approximate Pareto-optimal frontier $\widetilde{P} \subseteq \mathcal{X}$ for this bi-objective optimization, and return a solution $\mathbf{x} \in \widetilde{P}$ which gives the largest ratio $f(\mathbf{x})/g(\mathbf{x})$. They show sufficient conditions under which this framework leads to fully polynomial approximation schemes. This general approach can be tried whenever the objective function is a combination of several functions (a fractional objective $f(\mathbf{x})/g(\mathbf{x})$ is a simple example of a function which combines two functions $f(\mathbf{x})$ and $g(\mathbf{x})$). Mittal and Schulz apply their approach to a *sum-of-ratios problem* with a knapsack-type constraint, which arises, for example, in the context of assortment optimization in retail management [52, 53]. The sum-of-ratios optimization problems are a natural generalization of fractional optimization. (The terms “fractional optimization” and “fractional programming” are actually often used as encompassing both single-ratio and multi-ratio problems.) Schaible and Shi [59] survey sum-of-ratio optimization problems with convex subsets of R^n as feasible regions. However, there are still only few theoretical and algorithmic results for sum-of-ratio *combinatorial* optimization problems.

Although the focus of this chapter is on asymptotic bounds for the worst-case running times of methods and algorithms for fractional combinatorial optimization, it should be mentioned that theoretical research on this topic has been complemented by experimental studies which aim at evaluation of practical performance of algorithms. These include experimental studies of algorithms for the maximum-mean cycle and maximum mean-weight cycle problems [15, 17], algorithms for the fractional prize-collecting Steiner tree problem on trees [37], and parametric algorithms for the maximum mean cut problem arising in the context of seat allocation to political parties within the regions after a general election [61]. The results of experimental studies do not always match the theoretical analysis. For example, Klau et al. [37] observe that an $O(n^2)$ -time algorithm based on the Newton method consistently outperforms in their experiments an $O(n \log n)$ -time algorithm based on Megiddo’s parametric search.

Cross-References

- ▶ [Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs](#)
- ▶ [Network Optimization](#)

Recommended Reading

1. R.K. Ahuja, Minimum cost-reliability ratio path problem. *Comput. Oper. Res.* **15**, 83–89 (1988)
2. R.E. Bellman, On a routing problem. *Q. Appl. Math.* **16**, 87–90 (1958)
3. A. Billionnet, Approximation algorithms for fractional knapsack problems. *Oper. Res. Lett.* **30**(5), 336–342 (2002)
4. M. Blum, R.W. Floyd, V. Pratt, R.L. Rivest, R.E. Tarjan, Time bounds for selection. *J. Comp. Syst. Sci.* **7**(4), 448–461 (1973)
5. J. Carlson, D. Eppstein, The weighted maximum-mean subtree and other bicriterion subtree problems. *CoRR*. (2005). [abs/cs/0503023](https://arxiv.org/abs/cs/0503023)
6. P.J. Carstensen, The complexity of some problems in parametric, linear, and combinatorial programming. Ph.D. Thesis (Doctoral Thesis), Department of Mathematics, University of Michigan, Ann Arbor, 1983
7. R. Chandrasekaran, Minimum ratio spanning trees. *Networks* **7**, 335–342 (1977)
8. R. Chandrasekaran, Y.P. Aneja, K.P.K. Nair, Minimal cost-reliability ratio spanning tree. *Ann. Discrete Math.* **11**, 53–60 (1981)
9. E. Cohen, N. Megiddo, Maximizing concave functions in fixed dimension, in *Complexity in Numerical Optimization*, ed. by P. Pardalos (World Scientific, Singapore, 1993), pp. 74–87
10. E. Cohen, N. Megiddo, Strongly polynomial time and NC algorithms for detecting cycles in dynamic graphs. *J. Assoc. Comput. Mach.* **40**(4), 791–830 (1993)
11. R. Cole, Parallel merge sort. *SIAM J. Comput.* **17**(4), 770–785 (1988)
12. T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms* (MIT, Cambridge, 1990)
13. J.R. Correa, C.G. Fernandes, Y. Wakabayashi, Approximating a class of combinatorial problems with rational objective function. *Math. Program.* **124**(1–2), 255–269 (2010)
14. G.B. Dantzig, W.O. Blattner, M.R. Rao, Finding a cycle in a graph with minimum cost to time ratio with application to a ship routing problem, in *Theory of Graphs*, ed. by P. Rosentiehl (Gordon and Breach, New York, 1967), pp. 77–84
15. A. Dasdan, Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Trans. Des. Autom. Electron. Syst.* **9**, 385–418 (2004)
16. A. Dasdan, R.K. Gupta, Faster maximum and minimum mean cycle algorithms for system-performance analysis. *IEEE Trans. CAD Integr. Circ. Syst.* **17**(10), 889–899 (1998)
17. A. Dasdan, S.S. Irani, R.K. Gupta, Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems, in *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference (DAC '99)*, ed. by M.J. Irwin (ACM, New York, 1999), pp. 37–42
18. W. Dinkelbach, On nonlinear fractional programming. *Manag. Sci.* **13**, 492–498 (1967)
19. T.R. Ervolina, S.T. McCormick, Two strongly polynomial cut cancelling algorithms for minimum cost network flow. *Discrete Appl. Math.* **46**, 133–165 (1993)
20. L.R. Ford Jr., D.R. Fulkerson, *Flows in Networks*. (Princeton University Press, Princeton, 1962)
21. B. Fox, Finding minimal cost-time ratio circuits. *Oper. Res.* **17**, 546–551 (1969)
22. M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms. *J. Assoc. Comput. Mach.* **34**, 596–615 (1987)
23. A.V. Goldberg, Scaling algorithms for the shortest paths problem. *SIAM J. Comput.* **24**(3), 494–504 (1995)
24. A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem. *J. Assoc. Comput. Mach.* **35**, 921–940 (1988)
25. A.V. Goldberg, R.E. Tarjan, Finding minimum-cost circulations by canceling negative cycles. *J. Assoc. Comput. Mach.* **36**, 388–397 (1989)
26. D. Gusfield, Sensitivity analysis for combinatorial optimization. Technical Report UCB/ERL M90/22, Electronics Research Laboratory, University of California, Berkeley, May 1980
27. D. Gusfield, Parametric combinatorial computing and a problem of program module distribution. *J. Assoc. Comput. Mach.* **30**(3), 551–563 (1983)

28. M. Hartmann, J.B. Orlin, Finding minimum cost to time ratio cycles with small integral transit times. *Networks* **23**, 567–574 (1993)
29. S. Hashizume, M. Fukushima, N. Katoh, T. Ibaraki, Approximation algorithms for combinatorial fractional programming problems. *Math. Program.* **37**, 255–267 (1987)
30. T. Ibaraki, Parametric approaches to fractional programs. *Math. Program.* **26**, 345–362 (1983)
31. H. Ishii, T. Ibaraki, H. Mine, Fractional knapsack problems. *Math. Program.* **13**, 255–271 (1976)
32. K. Iwano, S. Misono, S. Tezuka, S. Fujishige, A new scaling algorithm for the maximum mean cut problem. *Algorithmica* **11**(3), 243–255 (1994)
33. S.N. Kabadi, Y.P. Aneja, An efficient, strongly polynomial, ε -approximation parametric optimization scheme. *Inform. Process. Lett.* **64**(4), 173–177 (1997)
34. R.M. Karp, A characterization of the minimum cycle mean in a digraph. *Discrete Math.* **23**, 309–311 (1978)
35. N. Katoh, A fully polynomial time approximation scheme for minimum cost-reliability ratio problems. *Discrete Appl. Math.* **35**(2), 143–155 (1992)
36. V. King, S. Rao, R. Tarjan, A faster deterministic maximum flow algorithm. *J. Algorithms* **17**(3), 447–474 (1994)
37. G.W. Klau, I. Ljubic, P. Mutzel, U. Pferschy, R. Weiskircher, The fractional prize-collecting Steiner tree problem on trees: extended abstract, in *Algorithms – ESA 2003, Proceedings of the 11th Annual European Symposium*, Budapest, 16–19 September 2003. Lecture Notes in Computer Science, vol. 2832 (Springer-Verlag, Berlin, Heidelberg, 2003), pp. 691–702
38. E.L. Lawler, Optimal cycles in doubly weighted directed linear graphs, in *Theory of Graphs*, ed. by P. Rosenthal (Gordon and Breach, New York, 1967), pp. 209–213
39. E.L. Lawler, *Combinatorial Optimization: Networks and Matroids* (Holt, Reinhart, and Winston, New York, 1976)
40. S.T. McCormick, A note on approximate binary search algorithms for mean cuts and cycles. UBC Faculty of Commerce Working Paper 92-MSC-021, University of British Columbia, Vancouver, 1992
41. N. Megiddo, Combinatorial optimization with rational objective functions. *Math. Oper. Res.* **4**, 414–424 (1979)
42. N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms. *J. Assoc. Comput. Mach.* **30**, 852–865 (1983)
43. S. Mittal, A.S. Schulz, A general framework for designing approximation schemes for combinatorial optimization problems with many objectives combined into one, in *APPROX-RANDOM*. Lecture Notes in Computer Science, vol. 5171 (Springer-Verlag, Berlin, Heidelberg, 2008), pp. 179–192
44. C. Haibt Norton, S.A. Plotkin, É. Tardos, Using separation algorithms in fixed dimension. *J. Algorithm* **13**, 79–98 (1992)
45. J.B. Orlin, R.K. Ahuja, New scaling algorithms for the assignment and minimum mean cycle problems. *Math. Program.* **54**, 41–56 (1992)
46. C.H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, 1982)
47. P.M. Pardalos, A.T. Phillips, Global optimization of fractional programs. *J. Global Opt.* **1**, 173–182 (1991)
48. T. Radzik, Newton's method for fractional combinatorial optimization, in *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science*, Pittsburgh, PA, USA (1992), pp. 659–669
49. T. Radzik, Newton's method for fractional combinatorial optimization. Technical Report STAN-CS-92-1406, Department of Computer Science, Stanford University, January 1992
50. T. Radzik, Parametric flows, weighted means of cuts, and fractional combinatorial optimization, in *Complexity in Numerical Optimization*, ed. by P. Pardalos (World Scientific, Singapore, 1993), pp. 351–386
51. T. Radzik, A. Goldberg, Tight bounds on the number of minimum-mean cycle cancellations and related results. *Algorithmica* **11**, 226–242 (1994)

52. P. Rusmevichientong, Z.-J.M. Shen, D.B. Shmoys, A PTAS for capacitated sum-of-ratios optimization. *Oper. Res. Lett.* **37**(4), 230–238 (2009)
53. P. Rusmevichientong, Z.-J.M. Shen, D.B. Shmoys, Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Oper. Res.* **58**(6), 1666–1680 (2010)
54. S. Schaible, Fractional programming 2. On Dinkelbach's algorithm. *Manag. Sci.* **22**, 868–873 (1976)
55. S. Schaible, A survey of fractional programming, in *Generalized Concavity in Optimization and Economics*, ed. by S. Schaible, W.T. Ziemba (Academic, New York, 1981), pp. 417–440
56. S. Schaible, Bibliography in fractional programming. *Z. Oper. Res.* **26**(7), 211–241 (1982)
57. S. Schaible, Fractional programming. *Z. Oper. Res.* **27**, 39–54 (1983)
58. S. Schaible, T. Ibaraki, Fractional programming. *Eur. J. Oper. Res.*, **12**(4), 325–338 (1983)
59. S. Schaible, J. Shi, Fractional programming: the sum-of-ratios case. *Optim. Methods Softw.* **18**, 219–229 (2003)
60. S. Schaible, J. Shi, Recent developments in fractional programming: single-ratio and max-min case, in *Proceedings of the 3rd International Conference on Nonlinear Analysis and Convex Analysis*, Tokyo, 25–29 August 2003 (Yokohama Publishers, Yokohama, 2004), pp. 493–506
61. P. Serafini, B. Simeone, Parametric maximum flow methods for minimax approximation of target quotas in biproportional apportionment. *Networks* **59**(2), 191–208 (2012)
62. M. Shigeno, Y. Saruwatari, T. Matsui, An algorithm for fractional assignment problems. *Discrete Appl. Math.* **56**(2–3), 333–343 (1995)
63. Y. Shiloach, U. Vishkin, An $O(n^2 \log n)$ parallel max-flow algorithm. *J. Algorithms* **3**, 128–146 (1982)
64. S. Toledo, Maximizing non-linear concave functions in fixed dimension, in *Complexity in Numerical Optimization*, ed. by P. Pardalos (World Scientific, Singapore, 1993), pp. 429–447
65. L.G. Valiant, Parallelism in comparison problems. *SIAM J. Comput.* **4**, 348–355 (1975)
66. C. Wallacher, A generalization of the minimum-mean cycle selection rule in cycle canceling algorithms. Unpublished manuscript, Institut für Angewandte Mathematik, Technische Universität Carolo-Wilhelmina, Germany, November 1989
67. Q. Wang, X. Yang, J. Zhang, A class of inverse dominant problems under weighted l_∞ norm and an improved complexity bound for Radzik's algorithm. *J. Global Optim.* **34**, 551–567 (2006)
68. A.C. Yao, An $O(|E| \log \log |V|)$ algorithm for finding minimum spanning trees. *Inform. Process. Lett.* **4**, 21–23 (1975)

Fuzzy Combinatorial Optimization Problems

Panos M. Pardalos, Emel Kizilkaya Aydogan, Feyza Gurbuz,
Ozgur Demirtas and Birce Boga Bakirli

Contents

1 Fuzzy Graph.....	1358
2 Fuzzy Linear Programming.....	1360
3 Fuzzy Integer Programming.....	1363
4 Fuzzy Spanning Trees.....	1365
5 Fuzzy Shortest Path.....	1366
6 Fuzzy Network Flows.....	1368
7 Fuzzy Minimum Cost Flow Problems.....	1369
8 Fuzzy Matching Algorithm.....	1370
8.1 Fuzzy Maximum Matching Algorithms.....	1371
8.2 Fuzzy Weighted Matching Algorithm.....	1371
9 Fuzzy Matroids.....	1380
10 Fuzzy Approximation Algorithms.....	1382
10.1 Fuzzy Set Covering.....	1382
10.2 Fuzzy Max-Cut Problem.....	1383
10.3 Fuzzy Coloring.....	1384
11 Fuzzy Knapsack Problems.....	1385
12 Fuzzy Bin Packing.....	1387

P.M. Pardalos (✉)

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: panos.pardalos@gmail.com

E.K. Aydogan • F. Gurbuz

Faculty of Engineering, Department of Industrial Engineering, Erciyes University, Kayseri,
Turkey
e-mail: ekaydogan@erciyes.edu.tr

O. Demirtas

Department of Business Administration, Erciyes University, Kayseri, Turkey
e-mail: ozgurdemirtas@hvkk.tsk.tr

B.B. Bakirli

Faculty of Engineering, Department of Industrial Engineering, Gazi University, Ankara, Turkey
e-mail: birceboga@yahoo.com

13	Fuzzy Multicommodity Flows and Edge-Disjoint Paths.....	1390
13.1	Minimal Cost Multicommodity Flow Problem.....	1392
13.2	Fuzziness in Real Transportation Problems.....	1393
13.3	MCMFP in Fuzzy Environment.....	1394
13.4	MCMFP with Uncertain Travel Cost.....	1395
14	Fuzzy Network Design Problems.....	1395
14.1	Steiner Tree Problem.....	1397
15	Fuzzy Traveling Salesman Problem.....	1398
15.1	Fuzzy Matrix to Represent TSP Solution.....	1399
16	Fuzzy Facility Location.....	1402
	Cross-References.....	1404
	Recommended Reading.....	1404

Abstract

Decision-making is an ongoing process for humankind. Many of these real-world decisions can be modeled using the problems contained in the generalized area of combinatorial optimization. Another area with recent advancements is fuzzy logic.

This chapter is concerned about the fuzzy solution approaches of combinatorial optimization problems. First section presents the fuzzy graph theory. Fuzzy linear programming and fuzzy integer programming are explained in second and third sections. Fuzzy spanning trees, fuzzy shortest path, fuzzy network flows, and the fuzzy minimum cost flow problems are introduced at Sects. 4–7. Fuzzy matching algorithm, fuzzy matroids, and fuzzy approximation algorithm are discussed in Sects. 8–10. Basic information on fuzzy knapsack and fuzzy bin-packing problems is given in Sects. 11 and 12. Fuzzy multicommodity flows and edge-disjoint paths are explained in Sect. 13 in detail with four subsections. Fuzzy network design problems and fuzzy traveling salesman problem are provided in Sects. 14 and 15 respectively. Finally, fuzzy facility location is presented in the last section.

1 Fuzzy Graph

A pair $G = (V, E)$ with $E \subseteq E(V)$ is called a graph (on V). The elements of V are the vertices of G , and those of E the edges of G . The vertex set of a graph G is denoted by V_G and its edge set by E_G . Therefore, $G = (V_G, E_G)$.

In the literature, graphs are also called simple graphs; vertices are called nodes or points; edges are called lines or links. More extensive information on graph theory can be found at [1, 2].

Fuzzy graph theory was introduced by Rosenfeld in 1975 [3], and some important definitions about this theory are as follows [3–10]:

Definition 1 A fuzzy graph G which is a pair of functions can be denoted by $G : (\sigma, \mu)$ where σ is a fuzzy subset of set V (a nonempty set) and μ is a symmetric fuzzy

relation on σ . The underlying crisp graph of $G : (\sigma, \mu)$ is denoted by $G^*(V, E)$ where $E \subseteq V \times V$. A fuzzy graph G is complete if $\mu(uv) = \sigma(u) \wedge \sigma(v)$ for all $u, v \in V$ where uv stands for the edge between u and v .

Definition 2 A fuzzy graph G denoted by $G : (\sigma, \mu)$. The degree of a vertex u is $d_{G(u)} = \sum_{u \neq v} \mu(uv)$. As $\mu(uv) > 0$ for $uv \in E$ and $\mu(uv) = 0$ for $uv \notin E$, this is equivalent to $d_{G(u)} = \sum_{uv \in E} \mu(uv)$. Then the minimum degree of G is $\delta(G) = \Lambda\{d(v)/v \in V\}$, and the maximum degree of G is $\Delta(G) = \vee\{d(v)/v \in V\}$.

Definition 3 The strength of connectedness between two vertices u and v is $\mu^\infty(u, v) = \sup\{\mu^k(u, v)/k = 1, 2, \dots\}$ where

$$\mu^k(u, v) = \sup\{\mu(uu_1) \wedge \mu(u_1u_2) \wedge \dots \wedge \mu(u_{k-1}v)/u_1, \dots, u_{k-1} \in V\}.$$

Definition 4 An edge uv is a fuzzy bridge of $G : (\sigma, \mu)$ if deletion of uv reduces the strength of connectedness between pair of vertices.

Definition 5 A vertex u is a fuzzy cut vertex of $G : (\sigma, \mu)$ if deletion of u reduces the strength of connectedness between some other pair of vertices.

Definition 6 Let $G : (\sigma, \mu)$ be a fuzzy graph such that $G^*(V, E)$ is a cycle. Then G is a fuzzy cycle if and only if there does not exist a unique edge xy such that $\mu(xy) = \vee\{\mu(uv)/(uv) > 0\}$.

Definition 7 The order of a fuzzy graph G is $O(G) = \sum_{u \in V} \sigma(u)$. The size of a fuzzy graph G is $S(G) = \sum_{uv \in E} \mu(uv)$.

Definition 8 Let $G : (\sigma, \mu)$ be a fuzzy graph on $G^*(V, E)$. If $d_G(v) = k$ for all $v \in V$, that is, if each vertex has some degree k , then G is said to be a regular fuzzy graph of degree k or a k -regular fuzzy graph. This is analogous to the definition of regular graphs in crisp graph theory.

Definition 9 Let $G : (\sigma, \mu)$ be a fuzzy graph on $G^*(V, E)$. The total degree of a vertex $u \in V$ is defined by $td_G(u) = \sum_{u \neq v} \mu(UV) + \sigma(u) = \sum_{uv \in E} \mu(UV) + \sigma(u) = d_G(u) + \sigma(u)$. If each vertex G has the same total degree k , then G is said to be a totally regular fuzzy graph of total degree k or a k -totally regular fuzzy graph.

The most important areas for the application of fuzzy graphs and fuzzy relations are logic, topology, pattern recognition, information theory, control theory, artificial intelligence, neural networks, operations research, planning, and systems analysis.

Bhattacharya [4] established some connectivity concepts regarding fuzzy cut nodes and fuzzy bridges. The author also introduced the notions of eccentricity and center.

Moreover, Bhattacharya and Suraweera [11] introduced an algorithm to find the connectivity of a pair of nodes in a fuzzy graph [4].

Furthermore, Tong and Zheng [12] presented an algorithm to find the connectivity matrix of a fuzzy graph. While, Xu [13] introduced connectivity parameters of fuzzy graphs to problems in chemical structures, Sunitha and Vijayakumar [7] focused on characterizing fuzzy trees using its unique maximum spanning tree. In addition, a sufficient condition for a node to be a fuzzy cut node is presented in [14]. Also, center problems in fuzzy graphs [15], blocks in fuzzy graphs [16], and properties of self-complementary fuzzy graphs [17] were introduced by the same authors. Therefore, using the concept of the strongest paths, they have obtained a characterization for blocks in fuzzy graphs [15]. Bhutani and Rosenfeld have focused on the concepts of strong arcs [18], fuzzy end nodes [19], and geodesics in fuzzy graphs [20]. In [17], the authors have defined the concepts of strong arcs and strong paths. They have pointed out the existence of a strong path between any two nodes of a fuzzy graph and have studied the strong arcs of a fuzzy tree. In [15], the concepts of fuzzy end nodes and multimin and locamin cycles are studied. The concept of strong arc in maximum spanning trees [21] and its applications in cluster analysis and neural networks were presented by Sameena and Sunitha [21, 22]. According to Mathew and Sunitha, there are different types of arcs in fuzzy graphs and they have obtained an arc identification procedure [14].

2 Fuzzy Linear Programming

Linear programming is one of the most widely used decision-making tools for solving real-world problems. Many researchers focus on the area of fuzzy linear programming because of the fact that the real-world situations are characterized by imprecision rather than exactness. In order to have a clear understanding, few of such researches are discussed/shown below.

Gupta and Mehlawat studied a pair of fuzzy primal–dual linear programming problems and calculate duality results using an aspiration level approach. They use an exponential membership function, which is in contrast to the earlier works that relied on a linear membership function. As the fuzzy environment causes a duality gap, they investigate how choosing the exponential membership function impacts the gap [23].

Amiria and Nasseria applied a linear ranking function to order trapezoidal fuzzy numbers. Then, they establish the dual problem of the linear programming problem with trapezoidal fuzzy variables and hence deduce some duality results. In particular, they prove that the auxiliary problem is indeed the dual of the linear programming problems with trapezoidal fuzzy variables (FVLP). Having established the dual problem, the results will then follow as natural extensions of duality results for linear programming problems with crisp data. Finally, using the results, they develop a new dual algorithm for solving the FVLP problem directly, making use of the primal simplex tableau [24].

In his paper, Ramik introduced a broad class of FLP problems firstly and defined the concepts of β -feasible and (α, β) -maximal and minimal solutions of FLP problems. The class of classical LP problems can be embedded into the class of

FLP ones. Furthermore, he defines the concept of duality and proves the weak and strong duality theorem generalizations of the classical ones for FLP problems [25].

On the other hand, Inuiguchi treats fuzzy linear programming problems with uncertain parameters whose ranges are specified as fuzzy polytopes in his study. The problem is formulated as a necessity measure optimization model. It is shown that the problem can be reduced to a semi-infinite programming problem and solved by a combination of a bisection method and a relaxation procedure. An algorithm in which the bisection method and the relaxation procedure converge simultaneously is proposed. A simple numerical example is given to illustrate the solution procedure [26].

Wua et al. present an efficient method to optimize such a linear fractional programming problem. First, some theoretical results are developed based on the properties of max-Archimedean t-norm composition. Then, the result is used to reduce the feasible domain. The problem can thus be simplified and converted into a traditional linear fractional programming problem and eventually optimized in a small search space. A numerical example is provided to illustrate the procedure [27].

In addition to above-mentioned studies, a new method to find the fuzzy optimal solution of same type of fuzzy linear programming problems is proposed by Kumar et al. It is easier to apply the proposed method, compared to the existing method in order to solve the fully fuzzy linear programming problems with equality constraints occurring in real-life situations. To illustrate, the proposed method numerical examples are solved, and the obtained results are discussed [28]. Additionally, Lotfi et al. discussed full fuzzy linear programming (FFLP) problems of which all parameters and variables are triangular fuzzy numbers. They use the concept of the symmetric triangular fuzzy number and introduce an approach to defuzzify a general fuzzy quantity [29].

By proposing the fuzzy multiobjective linear programming (FMOLP) model with triangular fuzzy numbers, Zenga et al., transformed the FMOLP model and its corresponding fuzzy goal programming (FGP) problem to crisp ones. These can be solved by the conventional programming methods. The FMOLP model was applied to crop area planning of Liang Zhou region, Gansu province of northwest China, and then the optimal cropping patterns were obtained under different water-saving levels and satisfaction grades for water resources availability of the decision-makers (DM) [30].

Further, Liang developed an interactive fuzzy multiobjective linear programming (i-FMOLP) method for solving the fuzzy multiobjective transportation problems with piecewise linear membership function. Proposed i-FMOLP method aims to simultaneously minimize the total distribution costs and the total delivery time. i-FMOLP method also has reference to fuzzy available supply and total budget at each source and fuzzy forecast demand with maximum warehouse space at each destination. Additionally, the above-mentioned method describes a systematic framework that facilitates the fuzzy decision-making process, enabling a decision-maker (DM) to interactively modify the fuzzy data and related parameters until a set of satisfactory solutions are obtained. An industrial case is presented to demonstrate the feasibility of applying such proposed method to real transportation problems [31].

Peidro et al. modeled supply chain (SC) uncertainties by fuzzy sets and develop a fuzzy linear programming model for tactical supply chain planning in a multi-echelon, multiproduct, multilevel, multi-period supply chain network in their study. In this approach, the demand, process, and supply uncertainties are jointly considered. The aim is to centralize multi-node decisions simultaneously to achieve the best use of the available resources along the time horizon so that customer demands are met at a minimum cost. With this intention, Peidro et al.'s proposal is tested by using data from a real automobile SC. Thus, fuzzy model provides the decision-maker (DM) with alternative decision plans with different degrees of satisfaction [32].

Other solutions are found by Buckley and Feuring to the fully fuzzified linear program where all the parameters and variables are fuzzy numbers in their study. First, they change the problem of maximizing a fuzzy number, the value of the objective function, into a multiobjective fuzzy linear programming problem. Then, they prove that fuzzy flexible programming can be used to explore the whole nondominated set to the multiobjective fuzzy linear program. Hence, an evolutionary algorithm is designed to solve the fuzzy flexible program, and they apply this program to two applications to generate good solutions [33].

Comparatively, Chanas and Zielinski analyzed the linear programming problem with fuzzy coefficients in the objective function. The set of nondominated (ND) solutions with respect to an assumed fuzzy preference relation, according to Orlovsky's concept, is supposed to be the solution of the problem. Special attention is paid to unfuzzy nondominated (UND) solutions (the solutions which are nondominated to the degree 1). The main results of their paper are sufficient conditions on a fuzzy preference relation which allows reducing the problem of determining UND solutions to that of determining optimal solutions of a classical linear programming problem. These solutions can thus be determined by means of classical linear programming methods [34].

Another major study is by Stanciulescu et al., modeling a multiobjective decision-making process by a multiobjective fuzzy linear programming problem with fuzzy coefficients for the objectives and the constraints. Moreover, the decision variables are linked together because they have to sum up to a constant. Most of the time, the solutions of a multiobjective fuzzy linear programming problem are crisp values. Thus, the fuzzy aspect of the decision is partly lost, and the decision-making process is constrained to crisp decisions. However, they propose a method that uses fuzzy decision variables with a joint membership function instead of crisp decision variables. First, they consider lower-bounded fuzzy decision variables that set up the lower bounds of the decision variables. Second, the method is generalized to lower-upper-bounded fuzzy decision variables that also set up the upper bounds of the decision variables. The results are closely related to the special type of the problem they are coping with, since they embed a sum constraint in the joint membership function of the fuzzy decision variables. Numerical examples are presented in order to illustrate their method [35].

By the same token, as a case study, Sadeghi and Hosseini tried to demonstrate the method of application of FLP for optimization of supply energy system in Iran.

They used FLP model comprises fuzzy coefficients for investment costs. Following the mentioned purpose, it is realized that FLP is an easy and flexible approach that can be a serious competitor for other confronting uncertainties approaches, that is, stochastic and minimax regret strategies [36].

On the other hand, Liu proposed a new kind of method for solving fuzzy linear programming problems based on the satisfaction (or fulfillment) degree of the constraints. Using a new ranking method of fuzzy numbers, the fulfillment of the constraints can be measured. Then the properties of the ranking index are discussed. With this ranking index, the decision-maker can make the constraints tight or loose based on his optimistic or pessimistic attitude and get the optimal solution from the fuzzy constraint space. The corresponding value of objective distribution function therefore can be obtained. A numerical example illustrates the merits of the approach [37].

Chen and Ko proposed fuzzy linear programming models to determine the fulfillment levels of PCs under the requirement to achieve the determined contribution levels of design requirements (DRs) for customer satisfaction. In addition, by considering the design risk, they incorporate failure modes and effect analysis (FMEA) into quality function deployment (QFD) processes, which are treated as the constraint in the models. In order to cope with the vague nature of product development processes, fuzzy approaches are used for both FMEA and QFD. The illustration of the suggested models is performed with a numerical example to indicate the applicability in practice [38].

Eventually, Katagiri et al. considered multiobjective linear programming problems with fuzzy random variables coefficients. A new decision-making model is proposed to maximize both possibility and probability, which is based on possibilistic programming and stochastic programming. An interactive algorithm is constructed to obtain a satisficing solution satisfying at least weak Pareto optimality [39].

3 Fuzzy Integer Programming

The linear programming models that have been discussed thus far all have been continuous, in the sense that decision variables are allowed to be fractional.

Often this is a realistic assumption. For instance, we might easily produce 102 3/4 gallons of a divisible good such as wine. It might also be reasonable to accept a solution giving an hourly production of automobiles at 58 1/2 if the models were based upon average hourly production, and the production had the interpretation of production rates.

At other times, however, fractional solutions are not realistic, and we must consider the optimization problem:

$$\text{Maximize} \quad \sum_{j=1}^n c_j x_j, \quad (1)$$

subject to

$$\sum_{j=1}^n a_{ij} = b_i, (i = 1, 2, \dots, m), \quad (2)$$

$$x_j \geq 0 (j = 1, 2, \dots, n), \quad (3)$$

$$x_j \text{ integer (for some or all } j = 1, 2, \dots, n). \quad (4)$$

This problem is called the (linear) integer programming problem. It is said to be a mixed-integer program when some, but not all, variables are restricted to be integer and is called a pure integer program when all decision variables are integers.

The implications of fuzzy set theory in optimal decision-making were first recognized by Bellman and Zadeh [40] and were later extended to linear programming problems with fuzzy constraints and multiple objectives by Zimmermann [41]. The latter formulation was subsequently extended to integer programming problems [42–45]. Fuzzy techniques have recently been applied to the analysis and optimization [46, 47]. Also, linear and integer linear programming are known to capture well optimization problems relevant to high-speed networks [48].

The fuzzy integer programming (FIP) methods provide another type of potentially useful approach for integer programming under uncertainty. Major shortcomings with the FIP methods are that, firstly, it may be difficult to obtain membership information for all system components in practical problems; secondly, FIP methods may lead to more complicated submodels that are computationally difficult for practical applications; and thirdly, most of the FIP solution algorithms are indirect approaches containing intermediate control variables or parameters, which are difficult to determine by certain criteria. They are thus unable to communicate uncertainty directly into the optimization processes and resulting specific solutions [49].

Tan et al. present integer programming optimization models for planning the retrofit of power plants at the regional or national level at their study. In addition to the base case (i.e., non-fuzzy or crisp) formulation, two fuzzy extensions are given to account for the inherent conflict between environmental and economic goals, as well as parametric uncertainties pertaining to the emerging carbon capture technologies. Case studies are shown to illustrate the modeling approach [50].

Asratian and Kuzjurin considered covering programs with 0–1 variables and cost function of the form $\sum_j x_j$ under the assumption that they know a pattern of coefficients in constraints, which are nonzero and only those coefficients can vary in some interval $[1, M]$ (zero elements do not change their value). As their main result, they found some sufficient conditions guaranteeing the variation of integral optimum in the average case (over all zero–nonzero patterns) is close to 1 as the number of variables tends to infinity. This means that for typical patterns the values of nonzero elements in A can vary without affecting significantly the value of the optimum of the integer program (i.e., the optimum value depends mostly on the pattern but not on the values of nonzero elements) [48].

Gharehgozli et al. present a new mixed-integer goal programming (MIGP) model for a parallel-machine scheduling problem with sequence-dependent setup times and release dates. Two objectives are considered in the model to minimize the total weighted flow time and the total weighted tardiness simultaneously. Due to the complexity of the above model and uncertainty involved in real-world scheduling problems, it is sometimes unrealistic or even impossible to acquire exact input data. Hence, they consider the parallel-machine scheduling problem with sequence-dependent setup times under the hypothesis of fuzzy processing time's knowledge and two fuzzy objectives as the MIGP model [51].

Li et al. developed a two-stage fuzzy robust integer programming (TFRIP) method for planning environmental management systems under uncertainty. Their approach integrates techniques of robust programming and two-stage stochastic programming within a mixed-integer linear programming framework. It can facilitate dynamic analysis of capacity-expansion planning for waste-management facilities within a multistage context. The TFRIP method is applied to a case study of long-term waste-management planning under uncertainty. Generated solutions for continuous and binary variables can provide desired waste-flow allocation, capacity-expansion plans with a minimized system cost, and maximized system feasibility [52].

Allahviranloo and Afandizadeh formulated an investment model to find the optimum investment steps by application of operational research science and fuzzy logic concept to model the available uncertainties. Fuzzy integer linear programming models are used to determine the optimum investment and development of a port [53].

Also, Emam studied a bi-level integer nonlinear programming problem [54] with linear or nonlinear constraints, where nonlinear objective function at each level is maximized. The bi-level integer nonlinear programming (BLI-NLP) problem can be thought as a static version of the Stackelberg game, which is used as a leader-follower game. A Stackelberg game is used by the leader, or the higher-level decision-maker (HLD), given the rational reaction of the follower, or the lower-level decision-maker (LLD). He proposed a two-planner integer model and a solution method for solving this problem. This method uses the concept of tolerance membership function and the branch-and-bound technique to develop a fuzzy max-min decision model for generating Pareto optimal solution for this problem; an illustrative numerical example is given to represent the obtained results [53].

4 Fuzzy Spanning Trees

In classical mathematical programming, the coefficients of objective functions or constraints in problems are assumed to be completely known. However, in real systems, they are uncertain than constant. In order to deal with such uncertainty, stochastic programming [55] and fuzzy programming [56] were considered. Both are useful tools for the decision-making under a stochastic environment or a fuzzy environment, respectively [57].

In parallel Gao and Lu formulated a fuzzy quadratic minimum spanning tree problem as expected value model, chance-constrained programming, and dependent-chance programming according to different decision criteria in their study. Then the crisp equivalents are derived when the fuzzy costs are characterized by trapezoidal fuzzy numbers. Furthermore, a simulation-based genetic algorithm using Prüfer number representation is designed for solving the proposed fuzzy programming models as well as their crisp equivalents, and a numerical example is displayed to illustrate the effectiveness of the genetic algorithm at the end of the study [57].

Katagiri et al. [58] investigated bottleneck spanning tree problems where each cost attached to the edge in a given graph is represented with a fuzzy random variable. The problem is to find the optimal spanning tree that maximizes a degree of possibility or necessity under some chance constraint. After transforming the problem into the deterministic equivalent one, they introduce the subproblem which has close relations to the deterministic problem. Utilizing fully the relations, they give a polynomial order algorithm for solving the deterministic problem.

Additionally, Katagiri et al. [59] studied minimum spanning tree problems where each edge weight is a fuzzy random variable. A fuzzy goal for the objective function is defined to capture the imprecise judgment of the decision-maker. They also proposed a decision-making model based on a possibilistic programming model and the expectation optimization model in stochastic programming.

5 Fuzzy Shortest Path

The shortest path problem (SPP) is one of the most fundamental and well-known combinatorial optimization problem that appears in many applications, including communications, transportation, routing, supply chain management, or models involving agents as a subproblem [60].

The problem of finding the shortest path from a specified source node to the other nodes is a fundamental matter in graph theory and one that is currently being greatly studied [61–68].

The classical problem seeks to select a path with minimum length from a finite set of paths. In real-world problems, arc lengths represent traveling time, cost, distance, or other variables. However, in practice, uncertainty cannot be avoided, and usually, the arc lengths cannot be determined precisely. For instance, on road networks, for several reasons, that is, traffic, accidents, or weather condition, arc lengths representing the vehicle travel time are subject to uncertainty. In such situations, a fuzzy shortest path problem (FSPP) seems to be more realistic and reliable.

According to Hernandes et al., an iterative algorithm assumes a generic ranking index for comparing the fuzzy numbers involved in the problem, in such a way that each time in which the decision-maker wants to solve a concrete problem(s) he/she can choose (or propose) the ranking index that best suits that problem. Hernandes et al.'s algorithm is based on the Ford–Moore–Bellman algorithm for classical graphs. In concrete, it can be applied in graphs with negative parameters,

and it can detect whether there are negative circuits. For the sake of illustrating the performance of the algorithm in the study, it has been developed using only certain order relations. However, it is not restricted at all to use these comparison relations exclusively. As the theoretical base of a decision support system's concern is to solve this kind of problems, the proposed iterative algorithm is easy to understand [69].

Tajdin et al. concerned with the design of a model and an algorithm for computing a shortest path in a network having various types of fuzzy arc lengths. Firstly, to obtain membership functions for the considered additions, they developed a new technique for the addition of various fuzzy numbers in a path using α -cuts by proposing a linear least squares model. Then, using a recently proposed distance function for comparison of fuzzy numbers, they present a dynamic programming method for finding a shortest path in the network [70].

Moazeni discussed the shortest path problem in his study. A positive fuzzy quantity is assigned to each arc as its arc length on a network. He defines an order relation between fuzzy quantities with finite supports. Then by applying Hansen's multiple labeling method together with Dijkstra's shortest path algorithm, he proposes a new algorithm for finding the set of nondominated paths with respect to the extension principle. Moreover, he shows that the only existing approach for this problem, Klein's algorithm, may lead to a dominated path in the sense of extension principle [60].

Keshavarz and Khorram concentrated on a shortest path problem on a network where arc lengths (costs) are not deterministic numbers, but imprecise ones in their study. Here, costs of the shortest path problem are fuzzy intervals with increasing membership functions, whereas the membership functions of the total cost of the shortest path are a fuzzy interval with a decreasing linear membership function. By the max-min criterion suggested in [71], the fuzzy shortest path problem can be treated as a mixed-integer nonlinear programming problem. They pointed out that this problem can be simplified into a bi-level programming problem that is very solvable. In order to solve the bi-level programming problem, they propose an efficient algorithm based on the parametric shortest path problem. An illustrative example is used to denote their algorithm [72].

In a network, the arc lengths may represent time or cost. In practical situations, it is reasonable to assume that each arc length is a discrete fuzzy set. It is called the discrete fuzzy shortest path problem. There are several methods reported to solve this kind of problem in the literature. In these methods, they can obtain either the fuzzy shortest length or the shortest path. In their study, Chuang and Kung claimed a new algorithm which can obtain both of them. The discrete fuzzy shortest length method is proposed to find the fuzzy shortest length, and the fuzzy similarity measure is utilized to get the shortest path. An illustrative example represents their proposed algorithm [62].

Ji et al. considers the shortest path problem with fuzzy arc lengths in their study. According to different decision criteria, the concepts of expected shortest path, α -shortest path, and the shortest path in fuzzy environment are originally introduced, and three types of models are formulated. In order to solve these models, a hybrid intelligent algorithm integrating simulation and genetic algorithm are asserted and numerous examples illustrate its effectiveness [73].

Okada deals with a shortest path problem on a network in which a fuzzy number, instead of a real number, is assigned to each arc length in his study. Such a problem is “ill-posed” because each arc cannot be identified as being either on the shortest path or not. Therefore, based on the possibility theory, he introduces the concept of “degree of possibility” that an arc is on the shortest path. Every pair of distinct paths from the source node to any other node is implicitly assumed to be noninteractive in the conventional approaches. This assumption is unrealistic and also involves inconsistencies. To overcome this drawback, he defines a new comparison index between the sums of fuzzy numbers by considering interactiveness among fuzzy numbers. An algorithm is presented to determine the degree of possibility for each arc on a network. This algorithm is evaluated by means of large-scale numerical examples. Consequently, this approach is found efficient even for real-world practical networks [66].

The fuzzy shortest path (SP) problem aims at providing decision-makers with the fuzzy shortest path length (FSPL) and the SP in a network with fuzzy arc lengths. In their study, Chuand and Kung represent each arc length a triangular fuzzy set and propose a new algorithm to deal with the fuzzy SP problem. First, they proposed a heuristic procedure to find the FSPL among all possible paths in a network. It is based on the idea that a crisp number is a minimum number if and only if any other number is larger than or equal to it. It owns a firm theoretic base in fuzzy sets theory and can be implemented effectively. Second, they propose a way to measure the similarity degree between the FSPL and each fuzzy path lengths. The path with the highest similarity degree is the SP. An illustrative example is added to display their proposed approach [61].

6 Fuzzy Network Flows

Network flow problems have a wide range of engineering and management applications such as analyses and design of computer networks, cable television networks, transportation systems, communication networks, project schedules, queuing systems, inventory systems, and manpower allocation [74].

In parallel, Liu and Kao [74] studied the network flow problems in that the arc lengths of the network and the objective value are fuzzy numbers. The illustrative example is on the problem of multimedia transmission over Internet.

Flows in networks provide very useful models in a number of practical contexts. The major techniques in the area revolve around celebrated max-flow min-cut theorem (MFMCT) and associated algorithms. In this context, Diamond develops analogues of the MFMCT and Karp–Edmonds algorithm for networks with fuzzy capacities and flows in their study. The principal difference between fuzzified and traditional crisp versions is that although the maximum fuzzy flow corresponds to a minimum fuzzy capacity, the latter may incorporate a number of network cuts. Preliminary results are for interval-valued flows and capacities which, in

themselves, provide robustness and estimates for flows in an uncertain environment. In turn, a fuzzy theorem and algorithm are obtained by regarding these intervals as level sets of fuzzy numbers [75].

Georgiadis considered a general class of optimization problems regarding spanning trees in directed graphs (arborescence). In his study, Georgiadis considered the following optimization problem. He assumed that edge costs take values in a set U endowed with a dyadic relation and a dyadic operation. He sought to find the directed spanning tree whose cost (with respect to the dyadic operation) is minimal with respect to the dyadic relation. Then, he provided an algorithm for solving the problem, which can be considered as generalization of Edmonds' special cases of this problem provide algorithms for the minimum sum, bottleneck and lexicographically optimal arborescence, as well as the widest-minimum sum spanning arborescence problem [76].

7 Fuzzy Minimum Cost Flow Problems

Minimal cost flow problem (MCFP) is an important problem in combinatorial optimization and network flows and has many applications in practical problems, such as communication, transportation, urban design, and job scheduling models [77].

The aim of the minimal cost flow problem (MCFP) in fuzzy nature, denoted with FMCFP, is to find the least cost of the shipment of a commodity through a capacitated network in order to satisfy imprecise concepts in supply or demand of network nodes and capacity or cost of network links [78].

Ghatee and Hashemi presented a model in which the supply and demand of nodes and the capacity and cost of edges are represented as fuzzy numbers. For easier reference, they refer to this group of problems as fully fuzzified MCFP. Hukuhara's difference and approximated multiplication are used to represent their model. Thereafter, they sort fuzzy numbers by an order using a ranking function and show that it is a total order, that is, a reflexive, antisymmetric, transitive, and complete binary relation. Utilizing the proposed ranking function, they transform the fully fuzzified MCFP into three crisp problems solvable in polynomial time [79].

Also, Ghatee and Hashemi deal with fuzzy quantities and relations in multi-objective minimum cost flow problem in their study. When t-conorms and t-conorms are available, the goal programming is applied to minimize the deviation among the multiple costs of fuzzy flows and the given targets. To obtain the most optimistic and the most pessimistic satisficing solutions of the problem, two different polynomial time algorithms are introduced by applying some network transformations. To verify the performance of this approach in actual substances, network design under fuzziness is considered, and an efficient scheme is proposed including genetic algorithm including fuzzy minimum cost flow problem [80].

Similarly, Ghatee et al. study the minimal cost flow problem (MCFP) with fuzzy link costs, say fuzzy MCFP, to understand the effect of uncertain factors in applied shipment problems. With respect to the most possible case, the worst case, and the

best case, the fuzzy MCFP can be converted into a three-objective MCFP. Applying a lexicographical ordering on the objective functions of derived problem, two efficient algorithms are provided to find the preemptive priority-based solution(s), namely, p-successive shortest path algorithm and p-network simplex algorithm. In both of them, lexicographical comparison is used which is a natural ordering in many real problems especially in hazardous material transportation where transporting through some links jeopardizes people's lives. Presented schemes maintain the network structure of the problem, and so, they are efficiently implementable. A sample network is added to illustrate the procedure and to compare the computational experiences with those of previously established works. Finally, the above-presented approach is applied to find an appropriate plan to transit hazardous materials in Khorasan roads network using annual data of accidents [81].

8 Fuzzy Matching Algorithm

Matching problem is one of the most important optimization problems. The model is widely used in practice such as pattern recognizing, system optimization, and job assignment problem. A matching of a graph is an independent subset M of edge set. In other words, a matching is a set of pairwise disjoint edges.

To explain this algorithm, first, we will explain fuzzy maximum matching algorithms. A maximum matching has as many edges in it as possible. Another version of this general problem is called the maximum weighted matching problem, and significantly more difficult will be captured in the second subsection of this section.

A maximum weighted matching problem is to choose a maximum matching in a given graph so that the sum of weight of the edges in it is at the maximum level. Nonbipartite weighted matching appears to be one of the “hardest” combinatorial optimization problems that can be solved in polynomial time.

Another version of weighted matching algorithms is a well-known problem called the assignment problem (AP). It can also be called as the minimum weight perfect matching problem in bipartite graphs. The fuzzy assignment problem (FAP) is also introduced at the Sect. 8.2.1.

There are several other inexact matching methods based on continuous optimization that have been proposed in the recent years. Among them we can cite the fuzzy graph matching (FGM) that is a simplified version of weighted graph matching (WGM) problem and based on fuzzy logic. In FGM, as the cost of matching two nodes does not depend on the matching of the other nodes of the graph, the objective function is considerably simpler than the objective function in WGM problem [82].

Medasani et al. [83] provided a relaxation approach to graph matching based on a fuzzy assignment matrix. Hence, the authors are able to derive that iterative FGM algorithm, which has a matching accurate than the graduated assignment algorithm.

Medasani and Krishnapuram [84] introduced a fuzzy approach to content-based retrieval of segmented images using fuzzy attributed relational graphs (FARGs) and an efficient FGM. In sum, the FGM algorithm is of the order $O(n^2m^2)$ and hence suitable for image retrieval.

8.1 Fuzzy Maximum Matching Algorithms

The maximum matching model is also called cardinality matching problem. An independent subset of the edges set is called a matching of a graph, and a matching with as many edges in it as possible is called a maximum matching.

Matching problems with fuzzy weights arise when the decision-maker has some vague information about some data of the real-world system. In other words, the parameters of a system can be characterized by fuzzy variables.

Essentially, Liu et al. initialized the concepts of expected maximum fuzzy weighted matching, the α -maximum fuzzy weighted matching, and the most maximum fuzzy weighted matching. According to various decision criteria, by using the credibility theory, with crisp equivalents are also given, the maximum fuzzy weighted matching problem is formulated as expected value model, chance-constrained programming, and dependent-chance programming. Furthermore, a hybrid genetic algorithm is designed to solve the proposed fuzzy programming models. Finally, a numerical example is given. The weights of the edges are trapezoidal fuzzy variables in the numerical example [85].

Liu and Gao [86] suggested concepts of the maximum random fuzzy weighted matching problem: (a) the expected maximum random fuzzy weighted matching, (b) the (α, β) -maximum random fuzzy weighted matching, and (c) the most maximum random fuzzy weighted matching. Later they formulated the maximum random fuzzy weighted matching problem as an expected value model, chance-constrained programming, and dependent-chance programming. To get the expected maximum random fuzzy weighted matching, the expected maximum random fuzzy weighted matching problem can be formulated as follows:

$$\begin{cases} \max \sum_{(i,j) \in E} x_{ij} \\ \max E[\sum_{(i,j) \in E} \xi_{ij} x_{ij}] \\ s.t. \sum_{j:(i,j) \in E} x_{ij} \leq 1, i = 1, 2, \dots, |V| \\ x_{ij} = 0 \text{ or } 1, (i, j) \in E. \end{cases} \quad (5)$$

Later a knowledge-based hybrid genetic algorithm is designed for solving the problem.

8.2 Fuzzy Weighted Matching Algorithm

Main goal of the algorithm is to find a matching in a given graph such that sums the weights of the edges at its maximum level. The sum of the weights of the edges in a matching is called the weight of matching. Sometimes, instead of a matching maximum weighted matching problem is used to find a maximum matching with maximum weight.

When the decision-maker has some vague information about some data of a real-life system, the matching model with fuzzy weighted is used. In other words, fuzzy variables are used for the parameters of a system.

In recent years, a lot of researches on matching have been conducted. A quite compact and flexible search-based artificial algorithm that makes use of relations based on representation of the graph is proposed by Balakrishnarajan and Venuganligam [87] to find all the perfect matching on a general graph.

An extension of the bipartite weighted matching problem is studied by Hsieh et al. [88]. A reduction algorithm reducing the matching problem to the assignment problem is proposed. Also, three examples are used to illustrate the process of the proposed method and its applications. As a result of the reduction to assignment problem on a larger graph, exact solution can be found at polynomial time.

Hsieh et al. [89] studied the weighted matching problem for on-line handwritten Chinese character recognition. The Hungarian method and a greedy algorithm based on the Hungarian method are proposed to solve the matching problem by a reduction algorithm. For each iteration of the greedy algorithm, a matched pair is deleted, and if the relation of their neighbors does not match, a new matching is then found by applying Hungarian method.

Lamb [90] studied the weighted matching with penalty and showed that bipartite weighted matching with penalty problem can be reduced to a bipartite maximum weight matching problem. The reduction to a maximum weight matching problem is not only easier than the reduction of Hsieh et al. [88, 89] to an AP, but also produces a smaller problem.

By using the appropriate data structures, Steiner and Yeomans [91] designed a linear time algorithm for maximum matching in convex, bipartite graphs and showed that the maximum matching problem can be efficiently transformed into an off-line minimum problem. This algorithm finds a maximum matching in less time complexity which is less than the graph size.

Kim and Wormald [92] studied the matching problem as an extension of matching problem, in random regular graphs, which are showing a random d -regular graph for even d asymptotically, almost surely (a.s.), has an edge decomposition into Hamilton cycles.

Zito [93] also studied bipartite and d -regular random graphs, and proved that with high probability, ratio between the sizes of any two maximal matchings approaches one in dense random graphs and random bipartite graphs. According to their findings, weaker bounds hold for sparse random graphs and random d -regular graphs.

For this model primarily, some preliminary knowledge of credibility theory and notations could be used. Let ξ be a fuzzy variable with membership function $\mu(x)$ and r a real number. Then, the possibility measure [94] and the necessity measure of fuzzy event $\leq \{\xi \leq r\}$ are defined as

$$\text{Pos}\{\xi \leq r\} = \sup_{x \leq r} \{\mu(x)\}, \text{Nec}\{\xi \leq r\} = 1 - \text{Pos}\{\xi > r\},$$

respectively.

The credibility measure of fuzzy event $\{\xi \leq r\}$ is defined by Liu and Liu [95] as

$$\text{Cr}\{\xi \leq r\} = \frac{1}{2}(\text{Pos}\{\xi \leq r\} + \text{Nec}\{\xi \leq r\}).$$

It can easily verified that $\text{Cr}\{\xi \leq r\} + \text{Cr}\{\xi > r\} = 1$ for any fuzzy event $\{\xi \leq r\}$. Liu [96] also presented two critical values, say α -optimistic value which was defined as

$$\xi_{\sup}(\alpha) = \sup\{r | \text{Cr}\{\xi \geq r\} \geq \alpha\}$$

and a pessimistic value which was defined as

$$\xi_{\inf}(\alpha) = \inf\{r | \text{Cr}\{\xi \leq r\} \geq \alpha\},$$

which serve as roles to rank fuzzy variables, where $\alpha \in (0, 1]$ and ξ is a fuzzy variable. The expected value of a fuzzy variable ξ is defined as

$$E[\xi] = \int_0^\infty \text{Cr}\{\xi \geq r\} dr - \int_{-\infty}^0 \text{Cr}\{\xi \leq r\} dr$$

provided that at least one of the two integrals is finite.

For more detailed properties of credibility measure, the interested reader may consult Liu [97].

For this model, it is thought that all the graphs are undirected and simple. Let $G(V, E, \xi)$ be a graph, where V , E , and ξ are the vertices set, edges set, and weights vector of the graph G . Sometimes, we employ $E(G)$ and $V(G)$ to denote the edges set and the vertices set of graph G , respectively. The degree and the set of incident edges of the vertex $v \in V$ are denoted by $dG(v)$ and $Ne[v]$, respectively. For a subset S of E or V , $G[S]$ denotes the subgraph induced by the subset S . The weight of edge $(i, j) \in E$ is denoted by a fuzzy variable ξ_{ij} , where

$$\xi_{ii} = 0 \text{ and } \xi_{ij} = \xi_{ji}, i, j = 1, 2, \dots, [V(G)].$$

Let $M \subseteq E$ be a matching of $G(V, E, \xi)$. Let

$$x_{ij} = \begin{cases} 1, & \text{if } (i, j) \in M, \\ 0, & \text{otherwise.} \end{cases}$$

Then the matching M can also be denoted by such a vector x , and the cost function of matching x can be defined as

$$f(x, \xi) = \sum_{(i, j) \in E} \xi_{ij} x_{ij'}$$

where x and ξ denote the vectors consist of x_{ij} and ξ_{ij} , $(i, j) \in E$, respectively.

Definition 10 The α -cost of $f(x, \xi)$ is defined as the cost value \bar{f} such that

$$\text{Cr}\{f(x, \xi) \geq \bar{f}\} \geq \alpha,$$

where α is a predetermined confidence level between 0 and 1.

It is clear that the cost function $f(x, \xi)$ is also a fuzzy variable when the vector ξ is a fuzzy vector. In order to rank $f(x, \xi)$, different ideas are employed in different situations. If the decision-maker wants to find a maximum weighted matching with maximum expected value of the cost function $f(x, \xi)$, then the following concept is induced.

Definition 11 A maximum matching x^* is called the expected maximum fuzzy weighted matching if

$$E[x^*] \geq E[f(x, \xi)]$$

for any maximum matching x .

In other situation, the decision-maker hopes to maximize the a-cost value \bar{f} with

$$\text{Cr}\{f(x, \xi) \geq \bar{f}\} \geq \alpha.$$

For this case, we propose the concept of a-maximum fuzzy weighted matching as follows:

Definition 12 A maximum matching x^* is called a-maximum fuzzy weighted matching if

$$\max\{\bar{f} | \text{Cr}\{f(x^*, \xi) \geq \bar{f}\} \geq \alpha\} \geq \max\{\bar{f} | \text{Cr}\{f(x, \xi) \geq \bar{f}\} \geq \alpha\}$$

for any maximum matching x , where a is a predetermined confidence level.

Furthermore, the most maximum weighted matching arises when the decision-maker gives a cost value \bar{f} and hopes that the credibility of the cost value exceeding \bar{f} will be as maximized as possible

Definition 13 A maximum matching x^* is called the most maximum fuzzy weighted matching if

$$\text{Cr}\{f(x^*, \xi) \geq \bar{f}\} \geq \text{Cr}\{f(x, \xi) \geq \bar{f}\}$$

for any maximum matching x , where \bar{f} is a predetermined cost value.

Based upon these, we will recast the maximum fuzzy weighted matching problem as multiobjective expected value model, chance-constrained programming, and dependent-chance programming, respectively.

Liu and Liu [95] initialized the fuzzy expected value model. Its main idea is to optimize the expected value of the objective function subjected to some expected constraints. Therefore, the expected maximum fuzzy weighted matching problem can be formulated as

$$\begin{cases} \max \sum_{(i,j) \in E} x_{ij} \\ \max E[\sum_{(i,j) \in E} \xi_{ij} x_{ij}] \\ s.t. \sum_{(i,j) \in E} x_{ij} \leq 1, i = 1, 2, \dots, |V|, \\ x_{ij} = 0 \text{ or } 1. \end{cases} \quad (6)$$

With respect to stochastic chance-constrained programming [98], Liu and Iwamura [99] offered the concept of fuzzy chance-constrained programming. When the decision-maker wants to optimize the critical value of the cost function subjected to some chance-constraints, the chance-constrained programming is employed. Therefore, the α -maximum fuzzy weighted matching problem can be formulated as follows:

$$\begin{aligned} & \max \sum_{(i,j) \in E} x_{ij} \\ & \max \bar{f} \\ & s.t. \text{Cr}\{\sum_{(i,j) \in E} \xi_{ij} x_{ij} \geq \bar{f}\} \geq \alpha \\ & \sum_{(i,j) \in E} x_{ij} \leq 1, i = 1, 2, \dots, |V| \\ & x_{ij} = 0 \text{ or } 1. \end{aligned} \quad (7)$$

Liu [100] proposed the dependent-chance programming in which the goal is to optimize a fuzzy event's chance in an uncertain environment, such as possibility, necessity, or credibility. According to this model, the problem to find a most maximum fuzzy weighted matching with given cost value \bar{f} can be formulated as follows:

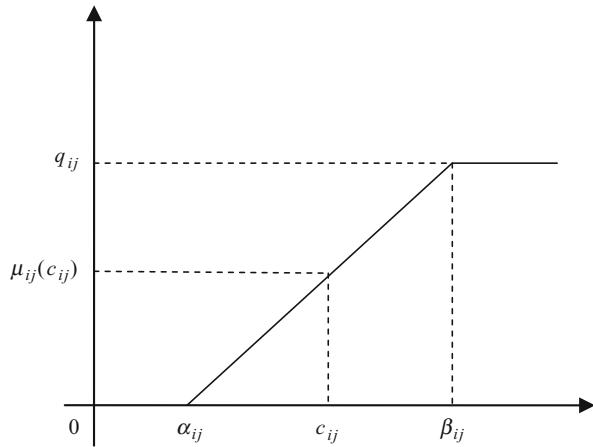
$$\begin{cases} \max \sum_{(i,j) \in E} x_{ij} \\ \max \text{Cr}\{\sum_{(i,j) \in E} \xi_{ij} x_{ij} \geq \bar{f}\} \\ s.t. \sum_{(i,j) \in E} x_{ij} \leq 1, i = 1, 2, \dots, |V|, \\ x_{ij} = 0 \text{ or } 1. \end{cases} \quad (8)$$

8.2.1 Fuzzy Assignment Problem

Assignment problem (AP) is a special type of linear programming problem and widely used in both manufacturing and service systems. AP can also be called as the minimum weight perfect matching problem in bipartite graphs.

In this model the objective is to assign n number of jobs to n number of persons at a minimum cost or time. The problem's mathematical formulation suggests that this is a 0–1 programming problem and is highly degenerate. It is applicable for transportation problems in which all the algorithms developed to find optimal solution.

Fig. 1 Membership function of c_{ij}



The AP parameters are imprecise numbers instead of fixed real numbers because time or cost for doing a job by a firm (machine or person) might vary due to a lot of reasons. Assigning men to jobs, drivers to buses, trucks to delivery routes, etc. are some examples over the past 50 years for this matter.

Here, it is aimed to find such an assignment, in which the total costs or sum of the single assignments is to become a minimum ratio [101]. The Hungarian method [102], which is the most popular algorithm for AP, carries through the algorithm directly on the $n \times n$ cost matrix $[c_{ij}]$, where c_{ij} represents the cost associated with worker i ($i = 1, 2, \dots, n$) who has performed job j ($j = 1, 2, \dots, n$) as seen in Fig. 1.

In this model, all c_{ij} 's are deterministic numbers.

The AP also could be modeled as a 0–1 integer programming problem:

$$\min \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (9)$$

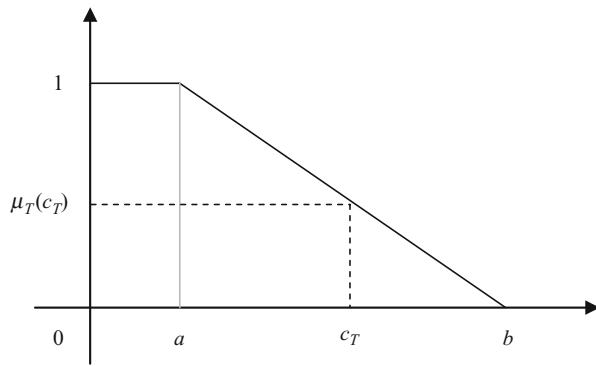
$$s.t. \sum_{j=1}^n x_{ij} = 1 \text{ for } i = 1, 2, \dots, n, \quad (10)$$

$$\sum_{i=1}^n x_{ij} = 1 \text{ for } j = 1, 2, \dots, n, \quad (11)$$

$$x_{ij} \in \{0, 1\} \text{ for } i, j = 1, 2, \dots, n. \quad (12)$$

However, many real-world application costs are not deterministic. For example, a consultant company provides different services to their customers for predetermined charges in each case. The job or case quality (may be the satisfaction of the customer or the job performance of the worker) may assume to be positively correlated to the input time or cost of the worker. Highest quality of the case is different because of the difference among individual workers. Furthermore, the manager of the company usually restricts the total labor hours or costs to a range as his fuzzy goal.

Fig. 2 Membership function of c_T



Hence, a minimum personnel cost is assumed to perform a job, and bigger spending costs to get result for higher quality until it reaches an upper bound, but it has to be thought that an increase in cost does not increase the quality. In this case, the costs are no longer deterministic numbers, and they will be denoted as c_{ij} 's, α_{ij} as the least cost associated with the worker i performing the job j and β_{ij} as the least cost associated with worker i performing the job j at the highest quality (it is assumed that $\beta_{ij} > \alpha_{ij} > 0$). Also the quality matrix $[q_{ij}]$ that represents the highest quality associated with the worker i performing the job j ($0 < q_{ij} \leq 1$) is defined. In most real cases, $0 < q_{ij} < 1$ and then c_{ij} is a subnormal fuzzy interval. The membership function of c_{ij} is defined as the linear monotone increasing function shown in Fig. 2, which suggests that any expense exceeding β_{ij} is useless since the quality (worker's job performance or customer's satisfaction) can no longer be increased at its upper limit q_{ij} condition.

$x_{ij} = 1$ is added to (13) because there is no real expense if $x_{ij} = 0$ in any feasible solution x of (12).

$$\mu_{ij}(c_{ij}) = \begin{cases} q_{ij} & \text{if } c_{ij} \geq \beta_{ij}, x_{ij} = 1, \\ \frac{q_{ij}(c_{ij} - \alpha_{ij})}{\beta_{ij} - \alpha_{ij}}, & \text{if } \alpha_{ij} \leq c_{ij} \leq \beta_{ij}, x_{ij} = 1, \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

The notation α_{ij}, β_{ij} is used to denote c_{ij} (all the α_{ij} 's form the matrix $[\alpha_{ij}]$ and β_{ij} 's form the matrix $[\beta_{ij}]$).

Matrix $[c_{ij}]$ is shown as follows:

$$[\tilde{c}_{ij}] = [\langle \alpha_{ij}, \beta_{ij} \rangle] = \begin{bmatrix} \langle \alpha_{11}, \beta_{11} \rangle & \cdots & \langle \alpha_{1n}, \beta_{1n} \rangle \\ \vdots & \ddots & \vdots \\ \langle \alpha_{n1}, \beta_{n1} \rangle & \cdots & \langle \alpha_{nn}, \beta_{nn} \rangle \end{bmatrix}. \quad (14)$$

In addition to this model, let c_T denote the total cost that is related to the job performance of the manager, and for the upper and lower bounds of the total cost,

numbers a and b are defined. The membership function of c_T is defined as the linear monotonically decreasing function in (15) and uses the notation $(a;b)$ to denote fuzzy interval c_T . The manager chooses a and b numbers as constant values. Then Werners [103] fuzzy linear programming in which taking a number less than or equal to the minimum assignment of matrix $[\alpha_{ij}]$ as a , and a number larger than or equal to the maximum assignment of matrix $[\beta_{ij}]$ as b could be used:

$$\mu_T(c_T) = \mu_T\left(\sum_{i=1}^n \sum_{j=1}^n c_{ij}x_{ij}\right) = \begin{cases} 1, & c_T \leq a \\ \frac{b - \sum_{i=1}^n \sum_{j=1}^n c_{ij}x_{ij}}{b-a}, & a \leq c_T \leq b, \\ 0, & c_T \geq b. \end{cases} \quad (15)$$

The performance of worker i , denoted as μ_i , based on the solution x will be

$$\mu_i = \mu_i(c_{ij}) \text{ for } x_{ij} = 1. \quad (16)$$

Here Bellman–Zadeh's criterion [71] was chosen that maximizes the minimum of the membership functions corresponding to which solution, to equally emphasize the performance of workers and manager, that is,

$$\max - \min_{x_{ij}} (\mu_i, \mu_T(c_T)) \quad (17)$$

or

$$\max - \min_{x_{ij}} = 1(\mu_{ij}(c_{ij}), \mu_T(c_T)), \quad (18)$$

where x_{ij} is an element of a feasible solution x of (9). Then we can represent the fuzzy AP as follows:

$$\max - \min(\mu_{ij}(c_{ij}), \mu_T(c_T))x_{ij} = 1 \quad (19)$$

$$s.t. \sum_{j=1}^n x_{ij} = 1 \text{ for } i = 1, 2, \dots, n, \quad (20)$$

$$\sum_{i=1}^n x_{ij} = 1 \text{ for } j = 1, 2, \dots, n,$$

$$x_{ij} \in \{0, 1\} \text{ for } i, j = 1, 2, \dots, n.$$

The cost of worker i performing job j is restricted to be less than or equal to β_{ij} since any expense exceeding β_{ij} is useless. By membership functions of (13) and (15), the following model (20) could be showed:

$$\max \lambda \quad (21)$$

$$s.t. \lambda x_{ij} \leq \frac{q_{ij}(c_{ij}^\lambda - \alpha_{ij})}{\beta_{ij} - \alpha_{ij}} x_{ij} \quad \text{for } i, j = 1, 2, \dots, n \quad (22)$$

$$\lambda \leq \frac{b - \sum_{i=1}^n \sum_{j=1}^n c_{ij}^\lambda x_{ij}}{b - a},$$

$$0 \leq \lambda x_{ij} \leq q_{ij} x_{ij} \quad \text{for } i = 1, 2, \dots, n, \quad (23)$$

$$0 \leq \lambda \leq 1$$

$$\sum_{j=1}^n x_{ij} = 1 \quad \text{for } i = 1, 2, \dots, n, \quad (24)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad \text{for } i = 1, 2, \dots, n, \quad (25)$$

$$c_{ij}^\lambda x_{ij} \leq \beta_{ij} x_{ij} \quad \text{for } i, j = 1, 2, \dots, n \quad (26)$$

$$x_{ij} \in \{0, 1\} \quad \text{for } i, j = 1, 2, \dots, n, \quad (27)$$

where C_{ij}^λ denotes the α -cut of c_{ij} , and $\sum_{i=1}^n \sum_{j=1}^n c_{ij}^\lambda x_{ij}$ is the corresponding total cost c_T^λ . In (26), since x_{ij} , c_{ij}^λ and λ all are decision variables, it can be treated as a mixed-integer nonlinear programming model.

In recent years, both fuzzy transportation and fuzzy assignment problems have received major attention. Many have been developed to advance numerous methodologies and applications to various decision problems. To deal with imprecision, and vagueness in real-life situations, Zadeh [104] introduced the concept of fuzzy sets.

Chen [105] presented a fuzzy assignment model that did not consider the differences of individuals. Similarly, Wang [106] solved a model by graph theory.

Dubois and Fortemps [107] proposed a flexible assignment problem, which combines with fuzzy theory, multiple criteria decision-making, and constraint-directed methodology. They survey refinements of the ordering of solutions supplied by the max-min formulation, namely, the discrimin partial ordering and the leximin complete preordering. Additionally, they have given a general algorithm which computes all maximal solutions in the sense of these relations. They also demonstrated and solved an example of fuzzy assignment problem.

On the other hand, Sakawa et al. [108] dealt with actual problems on production and work force assignment in a housing material manufacturer and a subcontract firm and formulated two-level linear and linear fractional programming problems according to profit and profitability maximization respectively. By applying interactive fuzzy programming for two-level linear and linear fractional programming problems, they derived satisfactory solutions to the problems.

Chanas et al. [109] solved transportation problems with fuzzy supply and demand values. Tada and Ishii [110] and Chanas and Kuchta [111] added an integer restriction to that fuzzy transportation problem and solved it. Also they suggested a specific definition for an optimal solution to the transportation problem using fuzzy cost coefficients and proposed an algorithm [112].

Lin and Wen [113] solved the assignment problem with fuzzy interval number costs by a labeling algorithm. The algorithm begins with primal feasibility and proceeds to obtain dual feasibility while maintaining complementary slackness until the primal optimal solution is found.

Feng and Yang [114] investigated a two-objective-cardinality assignment problem. A chance-constrained goal programming model is constructed for the problem, and tabu search algorithm based on fuzzy simulation is used to solve the problem.

Majumdar and Bhunia [115] proposed an elitist genetic algorithm with interval-valued fitness function to solve generalized assignment problem with imprecise cost/time. In their study, the coefficient of the interval-valued assignment problem's objective function has been considered as interval-valued number. Also, they represented the imprecision of cost(s)/time(s) with interval-valued numbers.

Ye and Xu [116] proposed an effective method on priority-based genetic algorithm to solve fuzzy vehicle routing assignment when there is no genetic algorithm which can give clearly procedure of solving it.

Liu and Gao [117] proposed an equilibrium optimization problem using genetic algorithm. Then they extended the assignment problem to the equilibrium multi-job assignment problem and equilibrium multi-job quadratic assignment problem.

Kumar et al. [118] proposed a new method to find the fuzzy optimal solution of fuzzy assignment problems by representing all the parameters as triangular fuzzy numbers. In their method, decision-maker takes the final results as fuzzy numbers, and if there is no uncertainty about the cost, their method gives the same result as in crisp assignment problem.

Mukherjee and Basu [119] introduced a mathematical model of the assignment problem considering the restrictions on job cost and person cost based on his/her efficiency/qualification. Under these conditions, both the cost and the restriction of qualification are considered as triangular or trapezoidal fuzzy numbers. They also considered the case where a person without certain qualification/efficiency should not be assigned a particular job.

However, Nagarajan and Solairaju [120] considered the assignment costs as imprecise numbers described by fuzzy numbers. They transformed the fuzzy assignment problem to a crisp assignment problem using Robust's ranking indices.

9 Fuzzy Matroids

In combinatorial optimization, matroid means independence structure which is a structure that generalizes linear independence in vector spaces. Many real-world problems can be defined and solved using matroids theory. Matroids are precisely the structures for which the very simple and efficient greedy algorithm works.

When we take as E the set of all edges in graph G and consider a set of edges independent if and only if it does not contain a simple cycle, this edge set is called a forest in graph theory. It can also be called the cycle matroid or graphic matroid of G ; it is usually written $M(G)$. The graphic matroid of G can also be defined as the column matroid of any oriented incidence matrix of G . A base can be called a spanning tree in G , and a circuit is a subset of edges creating a simple cycle in G .

When we want to find an object A composed of the elements of a given finite set E , a real weight $w(e)$ is associated with every element e of E , and we seek an

object A for which the total weight $w(A) = \sum_{e \in A} w(e)$ is minimal or maximal. A classical example is finding the minimum spanning tree or a shortest path in a given graph, where E is the set of edges of this graph [121].

A system is called weighted if there is a nonnegative weight $w(e)$ given for every element $e \in E$ [122].

In the matroidal combinatorial optimization problem, a nonnegative weight $w(e)$ is given for every element $e \in E$, and we seek a base B , for which the cost $F(B) = \sum_{e \in E} w_e$ is maximal (or minimal) [123].

In classical problem, it is assumed that all the weights are precisely known. However, this assumption may be a serious restriction since in many practical applications, the exact values of the weights are not known in advance. A typical example is traveling time between two cities in the shortest path problems or activity durations in scheduling problems. One of the simplest methods of modeling the imprecise weights is to define them as closed intervals. It is then assumed that the value of each weight may fall within a given range, independently on the values taken by the other weights. Classical intervals can be generalized into so-called fuzzy numbers, which are actually fuzzy intervals, and are richer in information [120].

In [121, 122, 124, 125], when all values of a weight function are closed intervals or fuzzy numbers, the optimality of solutions and the optimality of elements are characterized by means of a family of interval matroids $\{(E, I_a) : a \in [0, 1]\}$. Because of the varying nature of the world, they used fuzzy intervals to model the not precisely known element weights.

Another use of fuzzy sets is in a matroid problem solved by Goetschel and Voxman where a group of independent fuzzy sets was defined as a crisp family of fuzzy subsets of a finite set satisfying certain set of axioms [126]. Their closed regular fuzzy matroid can be shown as follows: [155] if w is a weight function on E and A is a fuzzy subset of E , it could be defined as:

$$w(A) = \sum_{A(\theta) > 0} w(e)A(e). \quad (28)$$

The question asked is whether we can find an object $B \in [0, 1]^E$ for which the total weight $w(B)$ is minimal or maximal?

For Goetschel–Voxman fuzzy matroids, we can refer to studies [127–131].

Defining a fuzzy subset of a finite set E as a subset of $E_x(0, 1]$, we obtain fuzzy matroids by fuzzifying independence spaces on $E_x(0, 1]$. Instead, by defining fuzzy subsets of E as functions from E into $[0, 1]$, we can obtain fuzzy matroids from polymatroids associated with “real” rank functions [132].

The concepts of fuzzy pre-matroid and hereditary fuzzy pre-matroid are introduced and investigated. The property “to be perfect” for hereditary fuzzy pre-matroids is also considered. It is shown that Goetschel and Voxman fuzzy matroids coincide with perfect hereditary fuzzy pre-matroids. The fuzzy rank function of a Goetschel–Voxman fuzzy matroid (E, I) was defined by a map $R : [0, 1]^E \rightarrow N$.

But in general, a Goetschel–Voxman fuzzy matroid and its fuzzy rank function are not one-to-one corresponding [133].

Also in the literature, Shi [121] proposed a closed fuzzy pre-matroid as a fuzzifying matroid and generalized it to L-fuzzy set theory. They showed that an L-fuzzifying matroid can be characterized by means of its L-fuzzifying rank function which is one-to-one corresponding with the L-fuzzifying matroid.

10 Fuzzy Approximation Algorithms

In Sect. 10.1 we focus on the quite general problem called the minimum weight set cover problem. The minimum weight edge cover problem is also a special case of the minimum weight set cover problem. In Sect. 10.2 we discuss a strongly NP-hard problem, the maximum weight cut problem. The edge coloring and the vertex coloring in planar graphs will be defined in Sect. 10.3.

10.1 Fuzzy Set Covering

Hwang et al. [134] proposed the fuzzy set-covering model which uses auxiliary 0–1 variables and a system of inequalities. Model is obtained by taking logarithm of the inequalities constraints and using the nature of the Boolean variables of this fuzzy set-covering model.

Given two subsets $I = \{1, 2, \dots, m\}$ and $J = \{1, 2, \dots, n\}$ of integers, let $\tilde{p}_j = \{(i, \mu_j(i)) : i \in I\}$ be a fuzzy subset of I and $(\mu_j(i)) \in [0, 1]$ be the membership grade of $i \in I$, using the membership function μ_j of fuzzy set p_j .

The model is proposed as

$$\min \sum_{j=1}^n c_j x_j \quad (29)$$

$$s.t. 1 - \prod_{j=1}^n (1 - \mu_j(i)) \geq \alpha, \quad i = 1, 2, \dots, m, \quad (30)$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \dots, n, \quad (31)$$

$$\text{where } x_j = \begin{cases} 1, & \text{if } \tilde{p}_j \in^*, \\ 0, & \text{otherwise.} \end{cases}$$

In this model α is a given real number (level degree) that represents the desired level (each $i \in I$ and the membership grade of i is no less than the level degree α). According to Theorem 1 in [135], Problem (P_1) can be transformed to Problem (P_2) by replacing the product of 0–1 variables in constraints (30) with auxiliary variables and a system of linear inequalities. The details can be found in [135].

$$\min \sum_{j=1}^n c_j x_j \quad (32)$$

$$s.t. \sum_{t=1}^n \mu_{j_t}(i)x_{j_t} + \sum_{k=2}^n \sum_{j_1 < j_2 < \dots < j_k} (-1)^{k+1} \left(\prod_{t=1}^k \mu_{j_t}(i) \right) y_{j_1 j_2 \dots j_t} \geq \alpha, \quad (33)$$

$$i = 1, 2, \dots, m,$$

$$2y_{j_1 j_2 \dots j_t} \leq y_{j_1 j_2 \dots j_{t-1}} + x_{j_t} \leq 1 + y_{j_1 j_2 \dots j_t}, t = 1, 2, \dots, k, \quad (34)$$

$$y_{j_1 j_2 \dots j_k} = \prod_{t=1}^k x_{j_t}, \quad (35)$$

$$x_j \in \{0, 1\}, \quad j = 1, 2, \dots, n. \quad (36)$$

The mentioned formulation is mathematically appropriate and also elegant, but solutions are difficult to find because of the fact that the number of decision variables is proportion to the number of auxiliary variables and extra constraints that usually grows exponentially [135] as the number of variables increases.

10.2 Fuzzy Max-Cut Problem

The max-cut problem is a combinatorial optimization problem and is one of the first problems proved to be NP-hard [136], because it determines the maximum cut of a graph. Its goal is to find a division of a vertex set into two parts, maximizing the sum of weights over all the edges across the two vertex subsets in a given edge-weighted graph. For an unweighted graph, the weights over the edges are equal to 1. Such a division is called a maximum cut.

This problem has many applications in various fields, that is, network design, circuit layout design [137], and data clustering [138]. The weight of the cut in the max-cut problem has real and concrete implications when applied in real life due to the fact that parameters are not so deterministic owing to the effects of many factors. For example, for network design, the weight may represent the cost of some infrastructure.

There are many optimization problems that are based on this theory, and several optimization models in stochastic models [95] and also including various fuzzy programming models [139], fuzzy shortest path problems [66], and fuzzy max-flow problems [75] have been studied. Recently, Liu [140] developed a credibility theory where the expected value criterion, the optimistic value criterion, and the credibility criterion are used as fuzzy ranking methods.

10.2.1 The Max-Cut Problem with Fuzzy Coefficients

The max-cut problem is widely used in practical applications. When applied in real life, the weights of a graph have real and concrete implications. They are often not

so deterministic. The weights on edges can be formulated as fuzzy variables and are more proper to describe the quantities in the real world since decision-makers are often faced with some uncertain situations due to the vagueness or subjective nature of these parameters.

To compute this problem with fuzzy coefficients, let $G = (V, E)$ be an undirected edge-weighted graph with nonnegative weights $\xi : E \rightarrow R^+$. A cut C of G is any nontrivial subset of V , and the weight of a cut is the sum of weights of edges crossing C and \bar{C} , where

$$\bar{C} = V - C.A. \quad (37)$$

Then a max-cut is defined as a cut of G with maximum weight. The max-cut problem's goal is to find such a cut.

Assume that the vertices in V are labeled as (v_1, v_2, \dots, v_n) and the edges in E are labeled as

$$e_{ij} = (v_i, v_j), \text{ where } n = |V|, i, j = 1, 2, \dots, n. \quad (38)$$

ξ_{ij} is the weight associated with the edge e_{ij} . We define a binary variable x_i for each vertex v_i to denote whether it is in a cut

C or not:

$$x_i = \begin{cases} 1 & \text{if } v_i \in C \\ -1 & \text{if } v_i \in V - C. \end{cases} \quad (39)$$

Then any cut of the graph G can be denoted by an n -dimensional binary vector $(x_1, x_2, \dots, x_n)^T$ over $\{1, -1\}$. Similarly, any n -dimensional binary vector x_1, x_2, \dots, x_n^T over $\{1, -1\}$ corresponds to a cut of G . So the weight of a cut $(x_1, x_2, \dots, x_n)^T$ can be denoted as

$$W(x, \xi) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \xi_{ij} (1 - x_i x_j). \quad (40)$$

Obviously, if and only if v_i and v_j respectively belong to two parts of the vertex set V ($x_i x_j = -1$), the weight of the edge (if any) linking them is considered. Let B be the set of all cuts of G . Then a cut x^* is called a maximum cut if and only if $W(x^*, \xi) \geq W(x, \xi)$ for any $x \in B$.

10.3 Fuzzy Coloring

A vertex coloring of graph G is defined as an assignment of colors to the vertices of G . So, no adjacent vertices get the same color. An edge coloring of graph G is an assignment of colors to the edges of G so that no adjacent edges have the same color.

A total coloring of a graph G is a coloring of the vertices and edges of G in such a way that no two adjacent elements have the same color. More information on coloring graphs can be found in [141].

The smallest possible total over all vertices that can occur among all colorings of G using natural numbers for the colors is called the chromatic sum of a graph G [142].

Munoz et al. [143] discussed two different approaches to the graph coloring problem of a fuzzy graph with crisp nodes and fuzzy edges. The traffic lights problem and a timetabling problem are studied according to these two different approaches. They also provided the concept of chromatic number of fuzzy graphs.

Chaudhuri and De [144] also introduced a timetabling problem called the university course timetable problem. Fuzzy genetic heuristic for this problem is discussed. There is a considerable amount of uncertainty involved in objectives which comprises of different aspects of real-life data. This uncertainty is handled by formulating some parameters using fuzzy membership functions.

The class of quasi-line graphs and more specifically fuzzy circular interval graphs gained a lot of attention recently. Eisenbrand and Niemeier [145] introduced a combinatorial algorithm that calculates weighted colorings and the weighted coloring number for fuzzy circular interval graphs efficiently.

11 Fuzzy Knapsack Problems

Knapsack problem (KP) has m different objects to choose from to put into the knapsack. Each object has a weight (w_i) and benefit (p_i). Aim of the problem is to choose the objects giving the maximum benefit without exceeding the total knapsack capacity. There are many real-life problems that can be modeled as the KP, such as cargo loading, capital budgeting, cutting stock, and economic, and network planning. Basic model for the crisp KP is as follows:

Indices

i : object indice(m)

Decision Variables

$$x_i \begin{cases} 1 & \text{if object } i \text{ lays in knapsack,} \\ 0 & \text{not in knapsack,} \end{cases}$$

Parameters

p_i : benefit of project i

w_i : cost of project i

c : capacity of knapsack not to be exceeded

Objective Function

$$\max \sum_{i=1}^m p_i x_i \text{ Benefit maximization} \quad (41)$$

Constraints

$$\sum_{i=1}^m w_i x_i \leq c \forall i \text{ size not to be exceeded} \quad (42)$$

$$x_i \in [0, 1]^m \quad \forall \text{ integer constraint} \quad (43)$$

Kuchta [146] discussed a fuzzy multiple choice KP where the total risk is minimized while staying in the budget. Each project has a risk and cost where each of these is represented with fuzzy numbers.

In Kasperski and Kulej [147], the concept of using fuzzy weights and fuzzy profits for the KP is used. This is different than the classical problem where it is assumed that all the item profits and weights are deterministic, because in practice many knapsack-type problems involve items whose weights or profits are not precisely known which is a useful extension. One of the methods of dealing with imprecision in real-world data is applying the fuzzy sets theory; they used the concept of a fuzzy interval to model the imprecise profits and the imprecise weights of items.

Lin and Yao [148] introduced a fuzzy KP where all weight coefficients are fuzzy numbers to model the imprecise weights in practical situations. They find it easier for the decision-maker to decide the value of the weight of an object as a range rather than a crisp value. After defuzzifying the fuzzy number, they use the same methodology to solve the problem as the crisp version.

On the other hand, Lin [149] provided a genetic algorithm methodology to solve imprecise weight coefficients in KPs.

In addition, Sakawa et al. [150] introduced an interactive fuzzy satisficing method for multiobjective multidimensional 0–1 KPs by using both the interactive fuzzy programming methods and genetic algorithms. By considering the imprecise nature of decision-maker (DM) judgments, fuzzy goals of the objective functions are quantified by using linear membership functions.

Chen [151] proposed a parametric programming approach to the continuous knapsack problem with fuzzy objective weights. The fuzzy maximum total return is analyzed. In this model, when the object weights are fuzzy, the maximal total profit also becomes fuzzy.

The mathematical model of the crisp continuous KP is as follows: Lin and Yao [152],

$$Z = \max \sum_{i=1}^n p_i x_i \quad (44)$$

$$s.t. \sum_{i=1}^1 w_i x_i \leq M \quad (45)$$

$$0 \leq x_i \leq 1, \quad i = 1, 2, \dots, n. \quad (46)$$

The continuous KP with fuzzy object weights is of the following form Chen [153]:

$$\tilde{Z} = \max \sum_{i=1}^n p_i x_i \quad (47)$$

$$s.t. \sum_{i=1}^l \tilde{W}_i x_i \leq M \quad (48)$$

$$0 \leq x_i \leq 1, \quad i = 1, 2, \dots, n. \quad (49)$$

For this model there are some example studies. One is Nezhad et al. [154], introducing a fuzzy capital budgeting model for selecting an optimum portfolio of projects. A real-world extension of all parameters of the projects was assumed to be imprecise.

Verdegay and Moreno's work [155] focused on the advantage of a fuzzy termination criterion providing results with low errors and with very short times with respect to both the exact and approximate solutions especially when the number of objects increases. The fuzzy-rule based systems and mathematical programming models and methods are used to solve the conventional KP.

Kato and Sakawa [156] provided a genetic algorithm with decomposition structures for large-scale multidimensional 0–1 KPs with block angular structures.

Lin [157] introduced a generalized fuzzy multiconstraint KP model, and extend it to another fuzzy multiobjective programming model. The multiconstraint 0–1 KPs are solved with imprecise weight coefficients.

Hasuike et al. [158] proposed a new model of random fuzzy 0–1 KPs where the objectives are represented with fuzzy goals and expected returns assumed as random fuzzy numbers.

Shih [159] focused on fuzzy multiobjective and multilevel KPs and proposed a solution procedure.

12 Fuzzy Bin Packing

Within a given rectangular area, finding an efficient packing has much relevance to operating systems, operations research, and manufacturing industries [160, 161]. For example, in a given time to operating systems or in steelworks, the problem is related with the allocation of resources. The task is cutting the roll of raw steel to make components of products. As these are mentioned, such can be formulated as *bin-packing (BP)* problems. BP is a problem that has long been studied by many researchers to better formulate and efficiently solve practical problems [162].

The problem is known to be NP-hard [163], while its description seems to be simple; it is very difficult to solve this problem precisely. In order to simplify the formulations of the problem, we have to concentrate on finding more efficient heuristics. Most formulations, therefore, assume *rigidity*, *orientation*, and *orthogonality* of the pieces [164, 165].

Suppose that there is a bin $B = (W, H)$ and a set of pieces $P = \{p_1, \dots, p_n\}$. We can represent each piece in the following manner: $p_i = (w_i, h_i)$ because the piece is rigid and oriented (we cannot change the size of a piece or its orientation). Letting (x_i, y_i) denote the center position of a piece p_i , we can also obtain the

height of the packing as $\max_i(y_i + h_i/2)$ because of the orthogonality assumption [166]. The bin-packing problem's goal is to minimize the height of the packing $\max_i(y_i + h_i/2)$ by choosing appropriate center positions (x_i, y_i) 's, which also satisfy the two following constraints: (1) all pieces should be contained within the bin and (2) no two pieces should overlap. By applying simple arithmetic, we can convert the two constraints as a set of inequalities.

However, assuming that the heights are rigid, that is, a constraint becomes too strong in many applications. When facing strong conflicting requirements, we should trade-off between the efficiency and the satisfaction. If we want to pay additional costs for decreasing the designer's satisfaction, then we could degrade the satisfaction because of the reduction.

However, in above-mentioned issue, we are not supposing the long-term cost such as deterioration of production quality or customer dissatisfactions. Long-term cost is due reducing the heights of pieces; thus, we call the cost as *reduction cost*. In fuzzy bin-packing problems, the model formulates the height of a piece as a fuzzy number. So, its goal is to minimize the *height cost* and the *reduction cost*. From these costs combining, the total cost emerges.

The *fuzzy bin packing problem* alleviates the rigidity constraint of the conventional bin-packing problem. In the conventional bin-packing problem, a piece p_i is represented as an ordered pair of two real numbers, (w_i, h_i) in which w_i stands for the width, and h stands for the height. In the fuzzy bin-packing problem, we represent the piece as an ordered pair of a real number and a fuzzy number, (w_i, \tilde{h}_i) .

We define the fuzzy bin-packing problem N as a tuple with three components, $N = (B, P, \Psi)$, where B is the *bin*, P is the *set of pieces*, and W is the *reduction cost function*. The bin B is a rectangle which has a limited width (W) and an unlimited height ($H = \infty$). The set of pieces P is a collection of rectangular pieces $p_i = (w_i, \tilde{h}_i)$, where h is a fuzzy number. Therefore, we can get a membership degree for a reduced height \tilde{h}_i . At the initial height h_i^0 , the membership degree is 1. This degree is called as the *satisfaction degree*. As a trade-off for reducing the height of a piece, the reduction cost function W is also given to evaluate the additional cost due to reduction. It is a function of the satisfaction degree and the initial height.

Yet, the conventional bin-packing problems' two constraints are not changed in the fuzzy bin-packing problem. Unlike in the conventional bin-packing problem, these constraints cannot be represented by a set of inequalities because the height is a fuzzy number. Therefore, we define the two constraints using the α -cut operation of fuzzy sets. The fuzzy set A is usually represented by a mapping $\mu_A : U \rightarrow [0, \dots, 1]$. The α -cut of a fuzzy set A at the given membership degree n produces a crisp set, $\alpha(A, \mu) = \{x | \mu_A(x) \geq \mu\}$.

In this model α -cut of a fuzzy number always becomes an interval because the fuzzy number is a convex and normal fuzzy set. If we have two α -cuts of two fuzzy numbers \tilde{a} and \tilde{b} , that is, if we have $\alpha(\tilde{a}, \mu_a)$ and $\alpha(\tilde{b}, \mu_b)$, α -cut can be defined as follows:

$$\begin{aligned} & [\alpha(\tilde{a}, \mu_a), \alpha(\tilde{b}, \mu_a)] \\ &= \{x | \forall a \in \alpha(\tilde{a}, \mu_a) \text{ and } \forall b \in \alpha(\tilde{b}, \mu_b)\} \end{aligned} \quad (50)$$

$$s.t. a \leq x \leq b\}. \quad (51)$$

If the below constraints satisfy these conditions, it could be stated that the piece p_i is *contained* within the bin at the satisfaction degree μ_i .

$$x_i - w_i/2 \geq 0 \text{ and } x_i + w_i/2 \leq W, \quad (52)$$

$$[\alpha(y_i - h_i^*/2, \mu_i), \alpha(y_i + h_i^*/2, \mu_i)] \bigcap (-\infty, 0) = \emptyset. \quad (53)$$

And also if below constraints satisfy the two conditions, it could be said that the two pieces p_i and p_j are *nonoverlapping* at satisfaction degree μ_i and μ_j .

$$|x_i - x_j| \geq |(w_i + w_j)/2| \quad (54)$$

$$[\alpha(y_i - \tilde{h}_i/2, \mu_i), \alpha(y_i + \tilde{h}_i/2, \mu_i)] \cap [\alpha(y_j - \tilde{h}_j/2, \mu_j), \alpha(y_j + \tilde{h}_j/2, \mu_j)] = \emptyset \quad (55)$$

It can be said that, the packing is *feasible* with the satisfaction degrees (μ_1, \dots, μ_n) , if the bin contains all the pieces and no overlapping occurs with satisfaction degrees (μ_1, \dots, μ_n) .

Because a piece's height can be reduced by sacrificing satisfaction, the objective of the fuzzy bin-packing problem is defined by minimizing the *total cost* \hat{X} for a feasible fuzzy bin-packing.

- The total cost is defined as the sum of the *height cost* X_h .
 - The *reduction cost* X_r .
 - The height cost X_h represents the maximum height of the packing, which is similar to the conventional bin-packing problem.
 - The reduction cost X_r represents the extra cost due to the decrease in satisfaction.
- From these expressions, the total cost can be formulated as

$$\hat{X} = X_h + X_r.$$

The height cost and the reduction cost can be defined in algebraic forms. Suppose a piece p_i is located at the top of packing.

The upper edge \hat{y}_i of the piece p_i can be represented as $\hat{y}_i = \max[\alpha(y_i + (\tilde{h}_i/2), \mu_i)]$.

Therefore, we can represent the height cost as follows: $X_h = (\hat{y}_i)$. Also, the *reduction cost* X_r can be defined in an algebraic form. If all the functions sum taken, we get the reduction cost of the fuzzy bin-packing as follows:

$$X_r = \sum_i \Psi(\mu_i, h_i^0),$$

where h_i^0 is the initial height of the piece and $\Psi(\mu_i, h_i^0)$ should be defined according to the problem.

13 Fuzzy Multicommodity Flows and Edge-Disjoint Paths

In optimization techniques, we have to well define the given precise data because these problems do not include inexact and unsure data. But using the fuzzy sets as a technology coping with granular information is useful [167]. Also, generally speaking, information granules are collections of entities that usually originate at the numeric level and are arranged together due to their similarity, functional adjacency, indistinguishability, coherency, or the like [168]. As a theoretical perspective, it encourages an approach to data that recognizes and exploits the knowledge present in data at various levels of resolution or scales. In this sense, it encompasses all methods which provide flexibility and adaptability in the resolution where knowledge or information is extracted and represented, see Bargiela and Pedrycz [169] for details.

While solving optimization problems, following the principle of information granulation is important [170]. Fuzzy data, which are various in the literature [79], are used as a concept of the solution of these programs. Fuzzy network solutions, for example, fuzzy shortest path problem [171] or fuzzy minimal cost flow problem [172], are limited when compared to the classical fuzzy optimization programming problems. These problems can be stated as the minimal cost multicommodity flow problem (MCMFP).

A fuzzy number is a convex normalized fuzzy set of the real line R , whose membership function is piecewise continuous. The set of fuzzy numbers on \mathbb{R} is denoted with $\mathcal{F}(\mathbb{R})$. An LR type flat fuzzy number [171] is denoted as $\tilde{a} = (a_1, a_2, a_3, a_4)_{LR}$, if

$$\mu_{\tilde{a}}(x) = \begin{cases} L\left(\frac{a_2-x}{a_2-a_1}\right), & a_1 \leq x \leq a_2, \\ 1, & a_2 \leq x \leq a_3, \\ R\left(\frac{x-a_3}{a_4-a_3}\right), & a_3 \leq x \leq a_4, \end{cases} \quad (56)$$

where the symmetric nonincreasing function $L : [0, \infty] \mapsto [0, 1]$ is the left shape function, that is, $L(0) = 1$. Naturally a right shape function $R(.)$ is similarly defined as $L(.)$ (see Fig. 3). We denote the LR type flat fuzzy numbers on real line with $\mathcal{LR}(\mathbb{R})$.

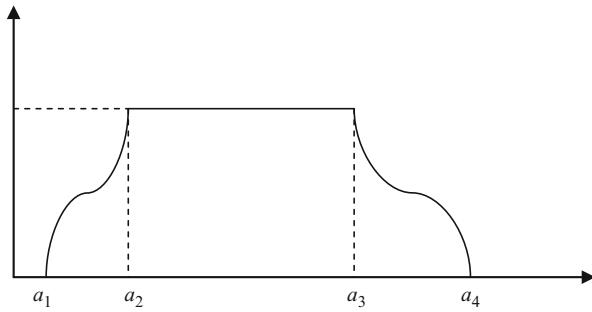
According to the extension principle, the binary operation $* \in \{+^\sim -^\sim \times^\sim \div^\sim\}$ on fuzzy number \tilde{a} and \tilde{b} with the membership functions $\mu_{\tilde{a}}(x)$ and $\mu_{\tilde{b}}(y)$ with the following:

$$\mu_{\tilde{a} * \tilde{b}}(Z) = \sup_{z=x * y} (\mu_{\tilde{a}}(x)\mu_{\tilde{b}}(y)). \quad (57)$$

Also the fuzzy scalar product of two fuzzy vectors $\tilde{x} = (\tilde{x}_1, \dots, \tilde{x}_n)$ and $\tilde{y} = (\tilde{y}_1, \dots, \tilde{y}_n)$ in $\mathcal{F}(\mathbb{R}^n)$ is defined [23] as follows:

$$\tilde{x} \times^\sim \tilde{y} = \sum_{i=1, \dots, n}^{\sim + \sim} \tilde{x}_i \tilde{x} \tilde{y}_i \quad (58)$$

Fig. 3 An *LR* type flat fuzzy number $\tilde{a} = (a_1, a_2, a_3, a_4)_{LR}$



Let $\tilde{a} = (a_1, a_2, a_3, a_4)_{LR}$ belong to $\mathcal{LR}(\mathbb{R})$. The exact formula for the extended addition and approximated formula for the extended multiplication are as follows:

$$(a_1, a_2, a_3, a_4)_{LR} \tilde{+} (b_1, b_2, b_3, b_4)_{LR} = (a_1 + b_1, a_2 + b_2, a_3 + b_3, a_4 + b_4)_{LR} \quad (59)$$

$$(a_1, a_2, a_3, a_4)_{LR} \otimes (b_1, b_2, b_3, b_4)_{LR} = (a_1.b_1, a_2.b_2, a_3.b_3, a_4.b_4)_{LR} \quad (60)$$

where $\tilde{a} \geq 0$ and $\tilde{b} \geq 0$.

LR fuzzy number could not be attained with multiplication of two *LR* fuzzy numbers. In a situation like this, interpolation methods to get a fuzzy number could be used, whose some α -cuts are equal to the multiplication of two μ -cuts of corresponding operands. The details of this idea [173] and scalar multiplication are derived as follows:

$$\lambda(a_1, a_2, a_3, a_4)_{LR} = (\lambda a_1, \lambda a_2, \lambda a_3, \lambda a_4)_{LR'}$$

where $\lambda \geq 0$.

$\tilde{b} - \tilde{a}$ is not really a difference and is rather unnatural with respect to a linear structure. For example, $\tilde{b} \tilde{+} (-1)\tilde{a}$ is not compatible with the difference in the function space. But the Hukuhara difference [174], defined as the solution for \tilde{x} in the equation $\tilde{a} \tilde{+} \tilde{x} = \tilde{b}$ if it exists, does coincide with the difference in the function space. This property justifies the application of this difference instead of the fuzzy number, $\tilde{b} \tilde{+} (-1)\tilde{a}$.

Hukuhara [174], Diamond and Korner [175], and Ghatee et al. [176] entitled as Hukuhara difference is observed to identify these differences as follows:

Definition 14 For \tilde{a} and \tilde{b} belong to $\mathcal{LR}(\mathbb{R})$ if the Hukuhara difference $\tilde{b} \ominus_H \tilde{a}$ exists, it is given by

$$[\tilde{b} \ominus_H \tilde{a}]_\alpha = \{\varsigma \in \mathbb{R} \mid [\tilde{a}]_\alpha \tilde{+} \{\varsigma\} \subseteq [\tilde{b}]_\alpha\}, \quad (61)$$

where \pm in the above equation for two sets \bar{X} and \bar{Y} means as follows:

$$\bar{X} + \bar{Y} = \{x + y | x \in \bar{X}, y \in \bar{Y}\}$$

For some cases, it can be proved that

$$(b_1, b_2, b_3, b_4)_{LR} \ominus_H (a_1, a_2, a_3, a_4)_{LR} = (c_1, c_2, c_3, c_4)_{LR'}$$

where

$$c_1 = b_1 - a_1, \quad c_2 = b_2 - a_2, \quad c_3 = b_3 - a_3, \quad c_4 = b_4 - a_4.$$

From this definition it can be said that Zadeh's difference is not true because $\tilde{a} \ominus_H \tilde{a}$ is zero.

Therefore, this operator for solving systems seems to be more accurate and reasonable results in comparison with Zadeh's difference.

Definition 15 For \tilde{a} and \tilde{b} belong to $\mathcal{LR}(\mathbb{R})$, we have:

$$\tilde{b} \ominus_H \tilde{a} = (b_1, b_2, b_3, b_4)_{LR} \ominus_H (a_1, a_2, a_3, a_4)_{LR} = (b_1 - a_1, b_2 - a_2, b_3 - a_3, b_4 - a_4)_{LR}.$$

13.1 Minimal Cost Multicommodity Flow Problem

The minimal cost multicommodity flow problem (MCMFP) which transships the flows of the commodities by minimal cost has been evaluated in this section.

The MCMFP satisfies the demand for each commodity at each vertex without violating the constraints imposed by the supply–demand and capacity.

Let N and A be the sets of nodes and links, and for a given network $G = (N, A)$: general MCMFP can be expressed:

$$\min \sum_{t=1}^T \sum_{ij \in A} c_{ij}^t x_{ij}^t, \quad (62)$$

$$\text{s.t. } \begin{cases} \sum_{t=1}^T x_{ij}^t \leq u_{ij}, \forall (i, j) \in A \\ \sum_{j:(i,j) \in A} x_{ij}^t - \sum_{j:(j,i) \in A} x_{ji}^t = b_i^t, \quad \forall i \in N = 1, \dots, T. \end{cases} \quad (63)$$

The notations are as follows:

- x_{ij}^t is a positive integral regarding the amount of flow from t th commodity which streams through link (i, j) .
- u_{ij} is the capacity of link (i, j) .
- T is the number of commodities.
- Moreover, for commodity t , c_{ij}^t and b_i^t are the unit cost of flow through link (i, j) .
- The supply or demand of node i .

If not change the variables, it can be assumed that each commodity has one resource and one customer without loss of generality [177]. Also r^t , s^t , and d^t denote the resource node, the customer node, and the travel demand for a given commodity t , respectively. Then, the following model regarding the path-flow variables is met:

$$\min \sum_{t=1}^T \sum_{p \in p^t} c_p^t f_p^t, \quad (64)$$

$$\text{s.t. } \begin{cases} \sum_{p \in p^t} f_p^t = d^t, & \forall t = 1 \dots, T, \\ x_{ij} = \sum_{t=1}^T \sum_{p \in p^t} \delta_p^{i,j} f_p^t \leq u_{ij}, & \forall (i, j) \in A, \end{cases} \quad (65)$$

where the flow of commodity t along the path p is denoted by f_p^t .

Additionally, $\delta_p = (\delta_p^{i,j})_{(i,j) \in A}$ is link-path incident vector and assigns 1 when the link (i,j) shares in path p and 0 otherwise. p^t includes all of the directed paths joining r^t and s^t .

From the first constraint, it can be said that the demand for each commodity is supplied by the path flows activated for that commodity. Including the link (i,j) , x_{ij} denotes the sum of flow of individual paths, and the other constraint shows the total flow is less than its capacity.

Equally important, vast majority of this classic model is deterministic. While this problem is influenced by social and economic interactions and is dependent on users' perceived information and granular information, one assumes the travel cost and demand to be known a priori for the entire links and nodes [177]. On the other hand, granular computing offers a landmark change from the current machine-centric to human-centric approach in information and knowledge and is an emerging conceptual and computing paradigm of information processing. It has been motivated by the urgent need for intelligent processing of empirical data that is now commonly available in vast quantities, into a humanly manageable abstract knowledge.

In other words, the granular computing's theoretical foundations involve fuzzy sets, rough sets, and random sets that linked together in a highly comprehensive treatment of this emerging paradigm [178]. In transportation concepts, due to granular information and the different provision of traffic information, the different types of transportation might be deduced. Now, we consider a case that the data of MCMFP have been exhibited with $\mathcal{LR}(\mathbb{R})$ numbers, which handle vagueness as an important category of uncertainty. So, first, we elaborate the origin of fuzziness in real problems. In the next section, two extended variants to take uncertain travel cost, and uncertain travel demand into account will be explained.

13.2 Fuzziness in Real Transportation Problems

In uncertainty modeling, transportation models adopted with fuzzy relations and quantities can contribute stochastic versions. In other words, since the classical random utility-based approach is not sufficient for uncertainty handling, some

alternate solution to probability-based models, for example, Ghatee and Hashemi [179], may be obtained with possibility-based methods. Initially Das et al. [180] applied fuzzy logic tools to transportation problems. Then, many researches are conducted particularly on this problem. From a high level categorization, in real networks, such as urban networks, we face the following three uncertainty sources:

- Unsure network topology (a network with fuzzy nodes and fuzzy links)
- Inexact travel cost (a network with fuzzy link cost)
- Imprecise travel demand (a network whose nodes include fuzzy excess or fuzzy deficit)

When the supply and demand of nodes and the capacity and cost of links are represented as fuzzy numbers, finding the lowest transportation cost of some commodities through a capacitated network is the aim of fuzzy transportation. This problem is a new branch in combinatorial optimization and network flow problems [79]. Some applications of such standpoint were presented in industries [181] such as the one we will discuss in the next section.

13.3 MCMFP in Fuzzy Environment

MCMFP is widely used in engineering and economics contexts. In urban transportation systems problems and production–distribution problems, this method emerges. Also, in designing telecommunication networks, it also has an important role. In all of aforementioned problems, finding the lowest transportation cost of some commodities through a capacitated network in order to satisfy demands at certain nodes, using available supplies at other nodes, is crucial.

Since this model is influenced by complex social and political interactions, this problem is dependent on user perception of information and nondeterministic factors of the network. Considering fuzzy numbers as the amount of supplies, demands, capacities, and costs provides a reasonable infrastructure is used to handle vagueness as an important category of uncertainty. Such an implementation of MCMFP can be applied to recognize the appropriate models of traffic assignment with intelligent agents; see, for example, [179] for details.

Ghatee and Hashemi treat with the minimal cost multi commodity flow problem (MCMFP) in the setting of fuzzy sets, by forming a coherent algorithmic framework, referred to as a fuzzy MCMFP. Given the character of granular information captured by fuzzy sets, the objective is to find multiple flows satisfying the demands of commodities, by using available supplies consuming the least possible cost. Having this aim in mind, the supply and demand of nodes may be presented linguistically; the travel cost and capacity of links can be defined under uncertainty as well. Then, to solve the problem, two efficient algorithms are motivated. In the first algorithm, they utilize fuzzy shortest paths and K-shortest paths to generate preferred and absorbing paths, and then they find the flow on them by solving a classic MCMFP. Coupled with the first, the second algorithm exhibits with fuzzy supply–demand and employs a total order on trapezoidal fuzzy numbers to reduce the fuzzy MCMFP into four classic MCMFPs [182].

13.4 MCMFP with Uncertain Travel Cost

Under traffic conditions, it is necessary to reflect a driver's perception of link travel time. For this purpose fuzzy sets of perceived link travel time are developed due the fuzziness of a driver's perception over link travel time.

There are various traffic conditions such as congestion and construction or incident, so for most frequent traffic conditions, it is necessary to construct fuzzy sets. To show these fuzzy set, Liu et al. [183] used linguistic descriptions. Obviously, if necessary, more fuzzy sets can be constructed to show other traffic conditions. Additionally, Henn [184] considered the meaning of fuzzy costs in fuzzy traffic assignment models.

In this chapter, assume that the cost, the length, or the travel time of a link (i,j) for commodity t is given by

$$\tilde{c}_{i,j}^t = (c_{(i,j),1}^t, c_{(i,j),2}^t, c_{(i,j),3}^t, c_{(i,j),4}^t) \in \mathcal{LR}(\mathbb{R}). \quad (66)$$

Then for shipping T commodities through the network, we can write:

$$\min \sum_{t=1}^T \sum_{p \in p^t} \tilde{c}_p^t f_p^t \quad (67)$$

$$s.t. \begin{cases} \sum_{p \in p^t} f_p^t = d^t, & \forall t = 1, \dots, T, \\ x_{ij} = \sum_{t=1}^T \sum_{p \in p^t} \delta_p^i f_p^t \leq u_{ij}, & \forall (i, j) \in A, \end{cases} \quad (68)$$

(69)

where $\tilde{c}_p^t = \sum_{(i,j) \in p} \tilde{c}_{i,j}^t$ for each $p \in p^t$.

This problem has an important concept that is the way of generating the absorbing and preferred paths p^t . Usually, the column generation approach [185] is used for solving the classic MCMFP. But we can take path enumeration techniques, which include a great number of algorithms with the capability of finding reasonable and preferred paths through a network, to understand p^t . In this method, the preference can be defined in terms of time, cost, distance, or some combination of these items. These reasonable paths are the base of problems consisting of path-flow variables.

14 Fuzzy Network Design Problems

In recent years, with the advance of computation evolutionary, new methods such as the optimal design of neural networks and fuzzy systems have been represented in the literature. For solving more complex real-world problems, computation evolutionary provides more robust and efficient approaches.

Network design is a combinatorial optimization problem involving optimization of several objectives such as cost, average delay, and reliability of network. Similarly, search space of the problem is huge even for small number of computers

and is intractable for nonevolutionary approaches. A network is basically a graph G of a set of nodes N and a set of links L . The links denote the connections between the nodes and are assumed to be directional.

In formulating and solving engineering or management problems, network flow models provide a rich and powerful framework. For problems such as transportation systems, communication networks, project schedules, inventory systems and man/machine allocation, there have been studies in the literature [185]. In theory and practice, this model is particularly important and it is considered in uncertain environment by many researchers [186]. Transportation models adopted with fuzzy relations and quantities can contribute stochastic versions in uncertainty modeling. More specifically, since the classical random utility-based approach is not sufficient for uncertainty handling, some alternate solution to probability-based models may be obtained with possibility-based methods; see, for example, Ghatee and Hashemi [179]. However, in order to refer their historical background, fuzzy logic tools have initially been applied to transportation problems by Das et al. [187]. Then, great attention has been focused on this problem. From a high-level categorization, in real networks, such as urban networks, there are three uncertainty sources:

- Unsure network topology (a network with fuzzy nodes and fuzzy links)
- Inexact travel cost (a network with fuzzy link cost)
- Imprecise travel demand (a network whose nodes include fuzzy excess or fuzzy deficit)

The aim of fuzzy transportation is to find the lowest transportation cost of some commodities through a capacitated network when the supply and demand of nodes, capacity, and cost of links are represented as fuzzy numbers. Given the character of granular information captured by fuzzy sets (where one could capitalize on the nonbinary character of their membership functions), such methods are capable of handling the decision-maker's risk taking. This problem is a new branch in combinatorial optimization and network flow problems [79, 179]. Some applications of such standpoint were presented in industries [188].

Furthermore, the network design problem is one of the most important problems in combinatorial optimization, which has been considered by many researchers and engineers [189]. The aim of this problem is to design a network to satisfy the demand of users who wish to travel through the network by consuming minimum construction cost. Steiner tree is the traditional variants of this problem, which have been applied to design communication and water networks [185]. In some cases, a network exists, and the management wishes to extend the network to satisfy future demand. The extension of a network can also be considered by interchanging the objectives such as cost, reliability, mobility, and accessibility. To consider such concepts, the designer needs to consider a great number of concepts. One important point in this problem is the granular information about this problem, because designing is done for a long-term period and the future statues cannot to be considered in a clear manner. It is worth reporting the sources of uncertainty in measurement of data in this problem [190]:

- The demand for transportation of each node, which is dependent on many items such as time, season, and weather conditions
- Linguistic definition of important travel

- Imperfect realization of the definition of important travel
- Nonrepresentative sampling (the sample measured may not represent the defined travels)
- Inadequate knowledge of the effects of environmental conditions on the measurement or imperfect measurement of environmental conditions.
- Personal bias in reading analogue instruments
- Finite instrument resolution or discrimination threshold
- Inexact values of measurement standards
- Inexact values of traffic constants and other parameters obtained from external sources and used in the data-reduction algorithm
- Approximations and assumptions incorporated in the measurement method and procedure
- Variations in repeated observations of travel under apparently identical conditions
- The minimum cost flow problem (MCFP) as mentioned before is a general structure in these models which provides a unified approach to many applications.

To design a network after processing these granular information, one can consider fuzzy numbers as transportation and construction costs, and with respect to these parameters, the following model may be expressed:

$$\begin{aligned}
 & \min \sum_{t=1:T}^{\mp} \sum_{(i,j) \in A}^{\mp} \tilde{c}_{ij}^t \tilde{x}_{ij}^t + \sum_{(i,j) \in A} \tilde{\mu}_{ij} z_{ij}, \quad \forall k = 1, \dots, K, \\
 & \text{s.t. } \begin{cases} \sum_{\{j:(i,j) \in A\}}^{\mp} \tilde{x}_{i,j}^t \ominus H \sum_{\{j:(j,i) \in A\}}^{\mp} \tilde{x}_{j,i}^t = \tilde{b}_i^t, & \forall i \in N, \forall t = 1, \dots, T, \\ \sum_{t=1:T}^{\mp} \tilde{x}_{(i,j),k}^t \leq \tilde{u}_{(i,j),k}, & \forall (i, j) \in A, \forall k = 1, \dots, 4 \\ 0 \leq \sum_{t=1:T}^{\mp} \tilde{x}_{ij}^t \leq \tilde{u}_{(i,j),k} z_{ij}, & \forall (i, j) \in A, \forall k = 1, \dots, 4 \end{cases}
 \end{aligned} \tag{70}$$

where \tilde{x}_{ij}^t is the fuzzy flow of commodity t and variables $z_{ij} \in \{0, 1\}$ are associated with the construction of link $(i, j) \in A$; $z_{ij} = 1$, if (i, j) belongs to the final solution; otherwise, $z_{ij} = 0$. The objective function (Eq. 70) is the sum of variable and fixed fuzzy costs. In this function c is the linear fuzzy cost associated with fuzzy flow of t th commodity through link (i, j) , and $\tilde{\mu}_{ij}$ is the fixed fuzzy cost associated with the selection of link (i, j) in the final solution. θ is a control parameter implying the trade-off between two objective functions.

14.1 Steiner Tree Problem

One of the most basic versions of finding a subgraph that optimizes some global cost function is known as the minimum weight Steiner tree [191] (or the minimum Steiner tree problem, MST) is a problem in combinatorial optimization, which may be formulated in a number of settings, with the common part being that it is required to find the shortest interconnect for a given set of objects.

Given an undirected graph with positive weights on the edges, the MST problem consists in finding a connected subgraph of minimum weight that contains a selected set of “terminal” vertices [192]. Such construction may require the inclusion of some nonterminal nodes which are called Steiner nodes. Clearly, an optimal subgraph must be a tree. Solving MST is a key component of many optimization problems involving real networks. Concrete examples are network reconstruction in biology (phylogenetic trees and regulatory subnetworks), Internet multicasting, circuit design, and power or water distribution networks design, just to mention few famous ones. MST is also a beautiful mathematical problem in itself which lies at the root of computer science being both *NP* complete [193] and difficult to approximate [194]. In physics the Steiner tree problem has similarities with many basic models such as polymers, self-avoiding walks, or transport networks (e.g., [195]) with a nontrivial interplay between local and global frustration. The Steiner tree problem has applications in circuit layout or network design. Most versions of the Steiner tree problem are NP complete.

The Steiner tree problem is superficially similar to the minimum spanning tree problem: given a set V of points (vertices), interconnect them by a network (graph) of shortest length, where the length is the sum of the lengths of all edges. Difference between the Steiner tree problem and the minimum spanning tree problem is that, in the Steiner tree problem, extra intermediate vertices and edges may be added to the graph in order to reduce the length of the spanning tree. These new vertices introduced to decrease the total length of connection are known as Steiner points or Steiner vertices. It has been proved that the resulting connection is a tree, known as the Steiner tree. There may be several Steiner trees for a given set of initial vertices.

The fuzzy steiner tree (FST) problem on a graph is the version of the classical ST problem where edge weights are defined as fuzzy numbers. Seda [196] provided a modification of the shortest paths approximation for solving FST problem. Linear triangular fuzzy numbers as the edge weights and Cheng’s centroid point fuzzy ranking method were used in his study [196].

15 Fuzzy Traveling Salesman Problem

Traveling salesman problem (TSP) is finding the shortest route for a given number of cities. It is one of the most widely studied combinatorial optimization problems [197]. Main characteristics of TSP are that every city has to be visited and no profit is associated to the cities.

TSP is one of the most widely studied combinatorial optimization problems. This has led to numerous extensions and modifications of the basic TSP.

A variant of TSP where a profit value is associated to each city and cities are selected depending on their profit is proposed in the literature. Feillet et al. [198] define these kinds of problems as TSP with profit [198].

The problem in which cities are selected to be visited depending on the profit associated with them is called traveling salesman problem with profit (TSP with profit). TSP with profit is encountered in many different situations. For instance,

it may not be possible to visit every city in a TSP application. In this kind of application, some constraints can enforce selection of cities to be visited. Balas and Martin [199] introduce the scheduling of daily operations of a steel rolling mill, which is an application of TSP with profit. This problem gives rise to a prize collecting traveling salesman problem (prize collecting TSP) with penalty terms in the objective function.

In the literature, TSP with profit is studied as a single-objective problem. The only attempt to solve TSP with profit as a biobjective problem is studied by Keller and Goodchild [200]. The main difference of biobjective approach compared to a single-objective approach is finding not only one but Pareto optimal solutions. By finding more solutions, the trade-off among them can be analyzed to make better decision. The purpose of this study is to develop a multiobjective approach for the biobjective TSP with profit in order to obtain the Pareto optimal solutions.

The TSP can be described as follows:

In the graph $G = (V, E)$, V is the set of nodes, or cities, and E is the set of edges, $E = \{(a, b) : a, b \in V\}$. The Euclidean distance between a and b is D_{ab} , supposing that $D_{ab} = D_{ba}$. The object of TSP is to find a minimal-length closed tour that visits each city once and only once, and the closed tour is also called Hamiltonian cycle. TSP has been proven to be an NP complete problem.

15.1 Fuzzy Matrix to Represent TSP Solution

When domains X, Y are finite sets, fuzzy relation can be expressed by fuzzy matrix. Suppose $X = \{X_1, X_2, \dots, X_m\}; Y = \{Y_1, Y_2, \dots, Y_n\}$, then the fuzzy relation from X to Y can be expressed as follows:

$$R = (r_{ij})_{m \times n} = \begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mn} \end{bmatrix}. \quad (71)$$

Here $r_{ij} \in [0, 1]$ represents the degree of membership of the i th element X_i in domain X and the j th element Y_j in domain Y to relation R . First we introduce the following definitions:

Definition 16 Set S is a sequence of a TSP solution,

if $S = \{S_1, S_2, \dots, S_n\}$, n is the number of the cities, $S_i(i \in \{1, 2, \dots, n\})$ is the i th node in the visit tour (TSP solution).

Definition 17 Set N is the serial number of a TSP,

if $N = \{N_1, N_2, \dots, N_n\}$, n is the number of the cities, $N_i(i \in \{1, 2, \dots, N\})$ represents the concrete node (city) in a TSP problem.

Explanation: For each element in Set S , S_i means that current tour has visited $i-1$ nodes and will visit the i th nodes. The state space can be expressed as S_s ,

$$S_s = S \times N = \{(S_i, V_j) | S_i \in S, N_j \in N\}. \quad (72)$$

The fuzzy relation R of S and N has the following meaning: for each element in the matrix R ,

$$r_{i,j} = \mu_R(S_i, N_j), (0 < r_{i,j} < 1). \quad (73)$$

μ_R is the membership function; the value of r_{ij} means the degree of membership that the i th node S_i in the feasible solution S “choose” the node with the serial number of N_j in a concrete problem.

15.1.1 Fuzzy Branch-and-Bound Algorithm

In this algorithm, each node is a particular schedule and tree structure is used in this model. The strategy of “breath first search” is used to determine the best partial schedule node to branch, the lower-bound values for completion times of whole partial schedule limits are calculated, and the node having the least value is chosen. Branching from lowest bound, the procedure is repeated at each time. The nodes having highest values of the lower bounds than the completion time of this schedule are fathomed after obtaining an order where all jobs are scheduled.

As shown below, A indicates the set of scheduled jobs. In Eq. 2, the lower-bound value of fuzzy completion time of each schedule beginning with A is calculated.

$$\tilde{AS}_{(A)} = \max \begin{cases} \tilde{\alpha}_{(A)} \oplus \sum_{i \in U} \tilde{P}_{i1} \oplus \min_{i \in U} \{\tilde{P}_{i2} + \tilde{P}_{i3}\} \\ \tilde{\beta}_{(A)} \oplus \sum_{i \in U} \tilde{P}_{i2} \oplus \min_{i \in U} \{\tilde{P}_{i3}\} \\ \tilde{\gamma}_{(A)} \oplus \sum_{i \in U} \tilde{P}_{i3} \end{cases} \quad (74)$$

In this equation:

- r_{ij} : Fuzzy operation time of i th job on j th workstation
- U : The set of jobs which are not scheduled
- $\tilde{\alpha}_A$: Completion time of the last job in schedule A on the first machine
- $\tilde{\beta}_A$: Completion time of the last job in schedule A on the second machine
- $\tilde{\gamma}_A$: Completion time of the last job in schedule A on the third machine

In this model, firstly the first machine assumed to operate first and third machine as last.

The completion times of the jobs on the machines are calculated by using Eqs. 3 and 4. If there is only one job in the set of A , then the completion times are computed by Eq. 3. If there is more than one job, then Eq. 4 is used.

$$\tilde{\alpha}_i = \tilde{P}_{i1}, \tilde{\beta}_i = \tilde{P}_{i1} \oplus \tilde{P}_{i2}, \tilde{\gamma}_i = \tilde{P}_{i1} \oplus \tilde{P}_{i2} \oplus \tilde{P}_{i3} \quad (75)$$

$$\tilde{\alpha}_{(A)} = \tilde{\alpha}_{(A-i)} \oplus \tilde{P}_{i1} \quad (76)$$

$$\tilde{\beta}_{(A)} = \max\{\tilde{\beta}_{(A-i)}, \tilde{\alpha}_{(A)}\} \oplus \tilde{P}_{i2} \quad (77)$$

$$\tilde{\gamma}_{(A)} = \max\{\tilde{\gamma}_{(A-i)}, \tilde{\beta}_{(A)}\} \oplus \tilde{P}_{i3} \quad (78)$$

After fuzzy lower-bound values for nodes are calculated, branching from the lowest bound to form new nodes for all jobs is not scheduled yet. This process is continued until all jobs are scheduled. Addition and maximum operations are fuzzy operations in this computation process.

Lee and Li's [201] approach is used for ranking in the maximum operation. They propose the use of generalized mean and standard deviation based on the probability measures of fuzzy events to rank fuzzy number. Let us assume that two fuzzy sets, where each member is a number,

$$\tilde{\alpha}_1 = \{(\mu_{x_1}, x_1), (\mu_{x_2}, x_2)\} \quad (79)$$

$$\tilde{\alpha}_2 = \{(\mu_{y_1}, y_1), (\mu_{y_2}, y_2), (\mu_{y_3}, y_3)\} \quad (80)$$

are given. The dimensions of the fuzzy numbers can be various. By using the extension principle, the addition of $\tilde{\alpha}_1$ and $\tilde{\alpha}_2$ is calculated as follows:

$$\begin{aligned} \tilde{\alpha}_1 \oplus \tilde{\alpha}_2 = & \{(\min(\mu_{x_1}, \mu_{y_1}), (x_1 + y_1)), \\ & (\min(\mu_{x_2}, \mu_{y_2}), (x_1 + y_2)), (\min(\mu_{x_1}, \mu_{y_3}), (x_1 + y_3)), \\ & (\min(\mu_{x_2}, \mu_{y_1}), (x_2 + y_1)), (\min(\mu_{x_2}, \mu_{y_2}), (x_2 + y_2)), \\ & (\min(\mu_{x_2}, \mu_{y_3}), (x_2 + y_3)) \end{aligned} \quad (81)$$

If there are more than one $(x_i + y_j)$ having the same value, then only the one having the largest membership degree is kept.

According to Agin [202], branch and bound is a powerful method capable of solving combinatorial problems with nonlinear, discontinuous, or nonmathematically defined objective functions and under several types of constraints. In branch-and-bound method, a tree structure which consists of properly connected nodes is established. Throughout the search, constraints imposed by the problem should be taken into account. Agin [202] divides these into two groups, namely, implicit and explicit constraints. In a successfully developed branch-and-bound algorithm, implicit constraints are satisfied by the manner in which the search tree is established. Explicit constraints, however, are to be considered in each step of the search. An example to implicit constraints might be given as the precedence relations, whereas explicit constraints might be exemplified by resource limitations in RCPSP [203]. A feasible solution to the problem, therefore, has to assign numerical values to the set of decision variables (e.g., start dates of all activities in RLP and RCPSP) so that both implicit and explicit constraints are satisfied.

Nodes, of which a search tree consists of, are subsets of the set of all solutions of the combinatorial problem. Branching, on the other hand, is the partitioning of any set of feasible solutions into separate subsets [202].

Branching process starts from the root node (the node in the uppermost level of the tree) which represents the set of all solutions. In some instances during the search, there might be nodes from which no branching has occurred yet. These nodes which are to be discovered further are called intermediate nodes. On the contrary to intermediate nodes which imply a partial solution, final nodes represent a complete solution. In order to reach a final node (leaf), all decisions required to establish a valid solution set have to be made. In RLP, for example, a leaf stores start dates of all noncritical activities. Obviously final nodes are located in the lowermost level of the search tree.

Two main characteristics of branch-and-bound algorithms presented by Agin [202] are branching characteristic and bounding characteristic. According to the definitions provided, branching characteristic ensures that an optimal solution is going to be reached at the end of the search since all possible combinations are going to be considered, whereas bounding characteristic implies a possibility to reach the optimal solution without visiting each node by pruning some parts of the tree.

Finally definition of the lower bound should be given since this concept is in the very heart of the branch-and-bound logic. Lower bound is a value of the objective function for all solutions included in a specific node such that none of the solutions that could be branched from that node will have a better objective function value than that bound. As this definition implies, there is no use to branch a node any further if its lower-bound value is worse than the objective function value of one of the explored final nodes (complete solutions). Objective function value of the best complete solution explored so far, that is, upper bound, is used to decide whether a node is promising or not. Obviously, upper bound at the end of the search provides the optimal solution to the problem.

According to Agin [202], a branch-and-bound algorithm might be said to consist of rules for:

1. Deciding on how to continue the search given an intermediate node (branching rule)
2. Deciding on how to calculate lower bounds on each established node
3. Deciding on the intermediate node from which to branch next
4. Recognizing when a node contains only infeasible or nonoptimal solutions
5. Recognizing optimal solutions encountered on final nodes

16 Fuzzy Facility Location

In earlier studies in location problems literature, we saw that this area firstly related to industrial contexts, referring to the supply of a single commodity from a set of potential locations, where facilities may be placed, to clients of known locations and demands, at minimum cost. Location problems consist of determining the locations of the facilities and the flows of the commodity from facilities to clients at a minimum cost.

Facility location selection that exhibits a significant impact on market share and profitability is critical and also has strategic issues in supply chain design and management [204, 205].

Solving the problems in which the demands and the costs are deterministic, many efficient methods are used. In the literature there are some examples such as the following: Akinc and Khumawala [206] used a branch- and-bound algorithm for a capacitated warehouse location problem. Dearing and Newruck [207] developed an implicit enumeration algorithm with Lagrange relaxation for a capacitated bottleneck facility location problem. Badri [208] used analytic hierarchy process with multiobjective goal programming methods for a facility location. Dupont [209] introduced a new type of facility location model with a concave global cost function and proposed a branch-and-bound algorithm for this model. For other types of methods also some examples in the literature, for example, Drezner and Hamacher [210], Ernst and Krishnamoorthy [211], Lozano et al. [212], and Misra et al. [213].

Facility location problem copes with the imprecise or qualitative nature of the linguistic assessment in real applications. Also this problem processes imprecise, uncertain, or vague data, such as the imprecise experience-based demands of the clients or the costs of operating the facilities. The imprecise nature of these data is due to the fact that the precise and complete historical data are usually unavailable in real situations. To the former, fuzzy sets and fuzzy logic [104, 214–216] are suitable methods of dealing with the linguistic assessment.

For example, Batanovi et al. [217] discussed a class of fuzzy maximum covering location problems in networks by assuming that the relative weights of demand nodes are imprecise and described by linguistic expressions and developed algorithms for choosing the best facility locations based on comparison operations on discrete fuzzy sets. Bhattacharya et al. [218] considered facilities located under multiple fuzzy criteria and proposed a fuzzy goal programming approach to deal with the problem. Ishii et al. [219] developed a location model by considering the customer satisfaction degree with respect to the distance of each customer from the facility. Kahraman et al. [220] solved facility location problems by using four different approaches to fuzzy multiattribute group decision-making, that is, fuzzy model of group decision, fuzzy synthetic evaluation, Yager's weighted goals method, and a fuzzy analytic hierarchy process.

Uncertainty is closely related to risks that include environmental and organizational. For the companies it is important to reduce risks that involves in their location decisions in order to enhance customer service and improve their business processes, to result increasing competitiveness and profitability. Uncertain parameters based on fuzzy set theory [95, 221–223] have been developed, which aim to deal with the cases of imprecise or vague data. For instance, Bhattacharya et al. [224] considered facilities located under multiple fuzzy criteria and proposed a fuzzy goal programming approach to deal with the problem. Ishii et al. [225] developed a location model by considering the satisfaction degree with respect to the distance from the facility for each customer. Assuming that demands of customers are represented as fuzzy variables, Wen and Iwamura [226] presented a continuous α -cost facility location model employing Hurwicz criterion, while

Zhou and Liu [227] presented three types of continuous capacitated location-allocation problem with different decision criteria.

Stochastic problems deal with the cases where the parameters are treated as random variables. In real applications, randomness and fuzziness may coexist in a facility location problem. Due to the fact that there are some variables such as subjective judgment, imprecise human knowledge and perception in capturing statistical data may embrace randomness and fuzziness at the same time or some data that belong to the history for the location problems may be insufficient. Therefore, the experience-based fuzzy information can be incorporated into the originally available statistic data. In both cases, there is a genuine need to deal with a hybrid uncertainty containing simultaneously randomness and fuzziness. The concept of fuzzy random variables [228–230] was introduced to quantify and deal with the phenomena in which vagueness and randomness appear at the same time. This formalism serves as a basic tool to construct a framework of decision-making models operating in a fuzzy and random environment.

Cross-References

► [Advanced Techniques for Dynamic Programming](#)

Recommended Reading

1. R. Diestel, *Graph Theory* (Springer, New York, 1997)
2. H. Korte, J. Vygen, *Combinatorial Optimization Theory and Algorithms*, 4th edn. (Springer, Berlin, 2008)
3. A. Rosenfeld, Fuzzy graphs, in *Fuzzy Sets and Their Applications*, ed. by L.A. Zadeh, K.S. Fu, M. Shimura (Academic, New York, 1975), pp. 77–95
4. P. Bhattacharya, Some remarks on fuzzy graphs. *Pattern Recognit. Lett.* **6**, 297–302 (1987)
5. F. Harary, *Graph Theory*, Indian Student Edition, (Narosa/Addison Wesley, New Delhi, 1988)
6. K.R. Bhutani, On automorphism of fuzzy graphs. *Pattern Recognit. Lett.* **12**, 413–420 (1991)
7. M.S. Sunitha, A.A. Vijayakumar, Characterization of fuzzy trees. *Inf. Sci.* **113**, 293–300 (1999)
8. J.N. Mordeson, P.S. Nair, *Fuzzy Graphs and Fuzzy Hypergraphs* (Physica, Heidelberg, 2000)
9. A. NagoorGani, M. BasheerAhamed, Order and size in fuzzy graph. *Bull. Pure Appl. Sci.* **22E**(1), 145–148 (2003)
10. A.N. Gani, K. Radha, On regular fuzzy graphs. *J. Phys. Sci.* **12**, 33–40 (2008)
11. P. Bhattacharya, F. Suraweera, An algorithm to compute the max–min powers and a property of fuzzy graphs. *Pattern Recognit. Lett.* **12**, 413–420 (1991)
12. Z. Tong, D. Zheng, An algorithm for finding the connectedness matrix of a fuzzy graph. *Congr. Numer.* **120**, 189–192 (1996)
13. J. Xu, The use of fuzzy graphs in chemical structure research, in *Fuzzy Logic in Chemistry*, ed. by D.H. Rouvry (Academic, San Diego, 1997), pp. 249–282
14. S. Mathew, M.S. Sunitha, Types of arcs in a fuzzy graph. *Inf. Sci.* **179**, 1760–1768 (2009)
15. M.S. Sunitha, A. Vijayakumar, Some metric aspects of fuzzy graphs, in *Proceedings of the Conference on Graph Connections*, ed. by R. Balakrishna, H.M. Mulder, A. Vijayakumar (CUSAT, Allied Publishers, New Delhi, 1999), pp. 111–114

16. M.S. Sunitha, A. Vijayakumar, Blocks in fuzzy graphs. *J. Fuzzy Math.* **13**(1), 13–23 (2005)
17. M.S. Sunitha, A. Vijayakumar, Complement of a fuzzy graph. *Indian J. Pure Appl. Math.* **33**(9), 1451–1464 (2002)
18. K.R. Bhutani, A. Rosenfeld, Fuzzy end nodes in fuzzy graphs. *Inf. Sci.* **152**, 323–326 (2003)
19. K.R. Bhutani, A. Rosenfeld, Geodesics in fuzzy graphs. *Electron. Notes Discret. Math.* **15**, 51–54 (2003)
20. K. Sameena, M.S. Sunitha, Strong arcs and maximum spanning trees in fuzzy graphs. *Int. J. Math. Sci.* **5**(1), 17–20 (2006)
21. K. Sameena, M.S. Sunitha, Characterisation of g-self centered fuzzy graphs. *J. Fuzzy Math.* **16**(4), 787–792 (2008)
22. K. Sameena, M.S. Sunitha, Distance in fuzzy graphs. Ph.D. Thesis, National Institute of Technology Calicut, India, 2008
23. P. Gupta, M. KumarMehlawat, Bector–Chandra type duality in fuzzy linear programming with exponential membership functions. *Fuzzy Sets Syst.* **160**, 3290–3308 (2009)
24. N. Mahdavi-Amiria, S.H. Nasseria, Duality results and a dual simplex method for linear programming problems with trapezoidal fuzzy variables. *Fuzzy Sets Syst.* **158**, 1961–1978 (2007)
25. J. Ramík, Duality in fuzzy linear programming with possibility and necessity relations. *Fuzzy Sets Syst.* **157**, 1283–1302 (2006)
26. M. Inuiguchi, Necessity measure optimization in linear programming problems with fuzzy polytopes. *Fuzzy Sets Syst.* **158**, 1882–1891 (2007)
27. Y.K. Wu, S.M. Guub, J.C. Liu, Reducing the search space of a linear fractional programming problem under fuzzy relational equations with max-Archimedean t-norm composition. *Fuzzy Sets Syst.* **159**, 3347–3359 (2008)
28. A. Kumar, J. Kaur, P. Singh, A new method for solving fully fuzzy linear programming problems. *Appl. Math. Model.* **35**, 817–823 (2011)
29. H. Lotfi, T. Allahviranloo, M.A. Jondabeh, L. Alizadeh, Solving a full fuzzy linear programming using lexicography method and fuzzy approximate solution. *F. Appl. Math. Model.* **33**, 3151–3156 (2009)
30. X. Zenga, S. Kanga, F. Li, L. Zhang, P. Guo, Fuzzy multi-objective linear programming applying to crop area planning. *Agric. Water Manage.* **98**, 134–142 (2010)
31. T. Liang, Distribution planning decisions using interactive fuzzy multi-objective linear programming. *Fuzzy Sets Syst.* **157**, 1303–1316 (2006)
32. D. Peidro, J. Mula, M. Jimenez, M. Botella, A fuzzy linear programming based approach for tactical supply chain planning in an uncertainty environment. *Eur. J. Oper. Res.* **205**, 65–80 (2010)
33. J.J. Buckley, T. Feuring, Evolutionary algorithm solution to fuzzy problems: fuzzy linear programming. *Fuzzy Sets Syst.* **109**, 35–53 (2000)
34. S. Chanas, P. Zielinski, On the equivalence of two optimization methods for fuzzy linear programming problems. *Eur. J. Oper. Res.* **121**, 56–63 (2000)
35. C. Stanciulescu, Ph. Fortemps, M. Instale, V. Wertz, Multi objective fuzzy linear programming problems with fuzzy decision variables. *Eur. J. Oper. Res.* **149**, 654–675 (2003)
36. M. Sadeghi, H.M. Hosseini, Energy supply planning in Iran by using fuzzy linear programming approach (regarding uncertainties of investment costs). *Energy Policy* **34**, 993–1003 (2006)
37. X. Liu, Measuring the satisfaction of constraints in fuzzy linear programming. *Fuzzy Sets Syst.* **122**, 263–275 (2001)
38. L.H. Chen, W.C. Ko, Fuzzy linear programming models for new product design using QFD with FMEA. *Appl. Math. Model.* **33**, 633–647 (2009)
39. H. Katagiri, M. Sakawa, K. Kato, I. Nishizaki, Interactive multi objective fuzzy random linear programming: maximization of possibility and probability. *Eur. J. Oper. Res.* **188**, 530–539 (2008)

40. R. Bellman, L.A. Zadeh, Decision making in a fuzzy environment. *Manage. Sci.* **17B**, 141–164 (1970)
41. H.J. Zimmermann, Fuzzy programming and linear programming with several objective functions. *Fuzzy Sets Syst.* **1**, 45–55 (1978)
42. F. Herrera, J.L. Verdegay, H.J. Zimmermann, Boolean programming with fuzzy constraints. *Fuzzy Sets Syst.* **55**, 285–293 (1993)
43. M.S. Osman, O.M. Saad, A.G. Hasan, Solving a special class of large-scale fuzzy multi objective integer linear programming problems. *Fuzzy Sets Syst.* **107**, 289–297 (1999)
44. J.J. Buckley, L.J. Jowers, *Monte Carlo Methods in Fuzzy Optimization* (Springer, Berlin, 2008)
45. K. Eshghi, J. Nematian, Special classes of fuzzy integer programming models with all-different constraints. *Sci. Iran. Trans. E* **16**, 1–10 (2009)
46. R.R. Tan, Using fuzzy numbers to propagate uncertainty in matrix-based LCI/LCI. *Int. J. Life Cycle Assess.* **13**, 585–593 (2008)
47. R.R. Tan, A.B. Culaba, K.B. Aviso, A fuzzy linear programming extension of the general matrix-based life cycle model. *J. Clean. Prod.* **16**, 1358–1367 (2008)
48. A.S. Asratian, N.N. Kuzjurin, Two sensitivity theorems in fuzzy integer Programming. *Discret. Appl. Math.* **134**, 129–140 (2004)
49. G.H. Huang, W. Baetz, G. Patry, Grey fuzzy integer programming: an application to regional waste management planning under uncertainty. *Socio-Econ. Plan. Sci.* **29**(1), 17–38 (1995)
50. R.R. Tan, D.K.S. Ng, D.C.Y. Foo, K.B. Aviso, Crisp and fuzzy integer programming models for optimal carbon sequestration retrofit in the power sector. *Chem. Eng. Res. Des.* **88**, 1580–1588 (2010)
51. A.H. Gharehgozli, R.T. Moghaddam, N. Zaerpour, A fuzzy-mixed-integer goal programming model for a parallel-machine scheduling problem with sequence-dependent set up times and release dates. *Robot. Comput.-Integr. Manuf.* **25**, 853–859 (2009)
52. Y.P. Li, G.H. Huang, X.H. Nie, S.L. Nie, A two-stage fuzzy robust integer programming approach for capacity planning of environmental management systems. *Eur. J. Oper. Res.* **189**, 399–420 (2008)
53. M. Allahviranloo, S. Afandizadeh, Investment optimization on port's development by fuzzy integer programming. *Eur. J. Oper. Res.* **186**, 423–434 (2008)
54. O.E. Emam, A fuzzy approach for bi-level integer non-linear programming problem. *Appl. Math. Comput.* **172**, 62–71 (2006)
55. S. Vajda, *Probabilistic Programming* (Academic, New York, 1972)
56. M. Sakawa, *Fuzzy Sets and Interactive Optimization* (Plenum, New York, 1993)
57. J. Gao, M. Lu, Fuzzy quadratic minimum spanning tree problem. *Appl. Math. Comput.* **164**, 773–788 (2005)
58. H. Katagiri, M. Sakawa, H. Ishii, Fuzzy random bottleneck spanning tree problems using possibility and necessity measures. *Eur. J. Oper. Res.* **152**, 88–95 (2004)
59. H. Katagiri, E.B. Mermri, M. Sakawa, K. Kato, A study on fuzzy random minimum spanning tree problems through possibilistic programming and the expectation optimization model, in *The 47th IEEE International Midwest Symposium on Circuits and Systems*, Hiroshima, Japan, (2004), pp. III-49–52
60. S. Moazeni, Fuzzy shortest path problem with finite fuzzy quantities. *Appl. Math. Comput.* **183**, 160–169 (2006)
61. T.N. Chuang, J.Y. Kung, The fuzzy shortest path length and the corresponding shortest path in a network. *Comput. Oper. Res.* **32**, 1409–1428 (2005)
62. T.-N. Chuang, J.Y. Kung, A new algorithm for the discrete fuzzy shortest path problem in a network. *Appl. Math. Comput.* **174**, 1660–1668 (2006)
63. F. Hernandes, A. Yamakami, Um algoritmo para o problema de caminhomínimo em grafos com custos nos arcos fuzzy, in *XV Congresso Brasileiro de Automática*, Gramado, RS, 2004
64. J.A. Moreno, J.M. Moreno, J.L. Verdegay, Fuzzy location problems on networks. *Fuzzy Sets Syst.* **142**, 393–405 (2004)

65. S.M.A. Nayem, M. Pal, Shortest path problem on a network with imprecise edge weight. *Fuzzy Optim. Decis. Mak.* **4**, 293–312 (2005)
66. S. Okada, Fuzzy shortest path problems incorporating interactivity among paths. *Fuzzy Sets Syst.* **142**(3), 335–357 (2004)
67. A. Sengupta, T.K. Pal, Solving the shortest path problem with intervals arcs. *Fuzzy Optim. Decis. Mak.* **5**, 71–89 (2006)
68. M.T. Takahashi, Contribuições para o estudo de grafos fuzzy: Teoria e algoritmos. Ph.D. Thesis, Faculdade de Engenharia Elétrica e de Computação, UNICAMP, 2004
69. F. Hernandes, M.T. Lamata, J.L. Verdegay, A. Yamakami, The shortest path problem on networks with fuzzy parameters. *Fuzzy Sets Syst.* **158**, 1561–1570 (2007)
70. A. Tajdin, I. Mahdavi, N. Mahdavi-Amiri, B. Sadeghpour-Gildeh, Computing a fuzzy shortest path in a network with mixed fuzzy arc lengths using α -cuts. *Comput. Math. Appl.* **60**, 989–1002 (2010)
71. R.E. Bellman, L.A. Zadeh, Decision-making in a fuzzy environment. *Manage. Sci.* **17B**, 141–164 (1970)
72. E. Keshavarz, E. Khorram, A fuzzy shortest path with the highest reliability. *J. Comput. Appl. Math.* **230**, 204–212 (2009)
73. X.K. Iwamura, Z. Shao, New models for shortest path problem with fuzzy arc lengths. *Appl. Math. Model.* **31**, 259–269 (2007)
74. ST. Liu, C. Kao, Network flow problems with fuzzy arc lengths. *IEEE Trans. Syst. Man Cybern. B* **34**(1), 765–769 (2004)
75. P. Diamond, A fuzzy max-flow min-cut theorem. *Fuzzy Sets Syst.* **119**, 139–148 (2001)
76. L. Georgiadis, Arborescence optimization problems solvable by Edmonds' algorithm. *Theor. Comput. Sci.* **301**, 427–437 (2003)
77. M. Ghatee, S.M. Hashemi, Generalized minimal cost flow problem in fuzzy nature: an application in bus network planning problem. *Appl. Math. Model.* **32**, 2490–2508 (2008)
78. S.M. Hashemi, M. Ghatee, E. Nasrabadi, Combinatorial algorithms for the minimum interval cost flow problem. *Appl. Math. Comput.* **175**, 1200–1216 (2006)
79. M. Ghatee, S.M. Hashemi, Ranking function-based solutions of fully fuzzified minimal cost flow problem. *Inf. Sci.* **177**, 4271–4294 (2007)
80. M. Ghatee, S.M. Hashemi, Application of fuzzy minimum cost flow problems to network design under uncertainty. *Fuzzy Sets Syst.* **160**, 3263–3289 (2009)
81. M. Ghatee, S.M. Hashemi, M. Zarepisheh, E. Khorram, Preemptive priority-based algorithms for fuzzy minimal cost flow problem: an application in hazardous materials transportation. *Comput. Ind. Eng.* **57**, 341–354 (2009)
82. D. Conte, P. Foggia, X.C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition. *Int. J. Pattern Recognit. Artif. Intell.* **18**(3), 265–298 (2004)
83. S. Medasani, R. Krishnapuram, Y.S. Choi, Graph matching by relaxation of fuzzy assignments. *IEEE Trans. Fuzzy Syst.* **9**, 173–182 (2001)
84. S. Medasani, R. Krishnapuram, A fuzzy approach to content-based image retrieval, in *FUZZ-IEEE '99: IEEE International Fuzzy Systems Conference Proceedings*, Seoul, Korea, 22–25 Aug 1999 (IEEE, Piscataway, 1999), pp. 1251–1260
85. L. Liu, Y. Li, L. Yang, The maximum fuzzy weighted matching models and hybrid genetic algorithm. *Appl. Math. Comput.* **181**, 662–674 (2006)
86. L. Liu, X. Gao, Maximum random fuzzy weighted matching models and hybrid genetic algorithm, <http://orsc.edu.cn/~xfgao/>
87. M. Balakrishnarajan, P. Venuvanaligam, An artificial intelligence approach for the generation and enumeration of perfect matching on graphs. *Comput. Math. Appl.* **29**, 115–121 (1995)
88. A. Hsieh, C. Ho, K. Fan, An extension of the bipartite weighted matching problem. *Pattern Recognit. Lett.* **16**, 347–353 (1995)
89. A. Hsieh, K. Fan, T. Fan, Bipartite weighted matching for on-line handwritten Chinese character recognition. *Pattern Recognit.* **28**, 143–151 (1995)
90. J. Lamb, A note on the weighted matching with penalty problem. *Pattern Recognit. Lett.* **19**, 261–263 (1998)

91. G. Steiner, G. Yeomans, A linear time algorithm for maximum matching in convex, bipartite graphs. *Comput. Math. Appl.* **31**, 91–96 (1996)
92. J. Kim, N. Wormal, Random matchings which induce Hamilton cycles and Hamiltonian decompositions of random regular graphs. *J. Combin. Theory B* **81**, 20–44 (2001)
93. M. Zito, Small maximal matchings in random graphs. *Theor. Comput. Sci.* **297**, 487–507 (2003)
94. D. Dubois, H. Prade, *Possibility Theory: An Approach to Computerized Processing of Uncertainty* (Plenum, New York, 1988)
95. B. Liu, Y.K. Liu, Expected value of fuzzy variable and fuzzy expected value models. *IEEE Trans. Fuzzy Syst.* **10**, 445–450 (2002)
96. B. Liu, *Theory and Practice of Uncertain Programming* (Physica, New York, 2002)
97. B. Liu, *Uncertainty Theory: An Introduction to Its Axiomatic Foundations* (Springer, Berlin, 2004)
98. A. Charnes, W.W. Copper, Chance-constrained programming. *Manage. Sci.* **6**, 73–79 (1959)
99. B. Liu, K. Iwamura, A note on chance constrained programming with fuzzy coefficients. *Fuzzy Sets Syst.* **100**, 229–233 (1998)
100. B. Liu, Dependent-chance programming with fuzzy decisions. *IEEE Trans. Fuzzy Syst.* **7**(3), 354–360 (1999)
101. H. Haken, M. Schanz, J. Starke, Treatment of combinatorial optimization problems using selection equations with cost terms. Part I. Two-dimensional assignment problems. *Phys. D* **134**, 227–241 (1999)
102. H.W. Kuhn, The Hungarian method for the assignment problem. *Nav. Res. Logist. Quart.* **2**, 253–258 (1996)
103. B. Werners, Interactive multiple objective programming subject to Klexible constraints. *Eur. J. Oper. Res.* **31**, 342–349 (1987)
104. L.A. Zadeh, Fuzzy sets. *Inf. Control* **8**, 338–353 (1965)
105. M.S. Chen, On a fuzzy assignment problem. *Tamkang J.* **22**, 407–411 (1985)
106. X. Wang, Fuzzy optimal assignment problem. *Fuzzy Math.* **3**, 101–108 (1987)
107. D. Dubois, P. Fortemps, Computing improved optimal solutions to max - min flexible constraint satisfaction problems. *Eur. J. Oper. Res.* **118**, 95–126 (1999)
108. M. Sakawa, I. Nishizaki, Y. Uemura, Interactive fuzzy programming for two-level linear and linear fractional production and assignment problems: a case study. *Eur. J. Oper. Res.* **135**, 142–157 (2001)
109. S. Chanas, W. Kolodziejczyk, A. Machaj, A fuzzy approach to the transportation problem. *Fuzzy Sets Syst.* **13**, 211–221 (1984)
110. M. Tada, H. Ishii, An integer fuzzy transportation problem. *Comput. Math. Appl.* **31**, 71–87 (1996)
111. S. Chanas, D. Kuchta, Fuzzy integer transportation problem. *Fuzzy Sets Syst.* **98**, 291–298 (1998)
112. S. Chanas, D. Kuchta, A concept of the optimal solution of the transportation problem with fuzzy cost coefficients. *Fuzzy Sets Syst.* **82**, 299–305 (1996)
113. C.J. Lin, U.P. Wen, A labeling algorithm for the fuzzy assignment problem. *Fuzzy Sets Syst.* **142**, 373–391 (2004)
114. Y. Feng, L. Yang, A two-objective fuzzy k-cardinality assignment problem. *J. Comput. Appl. Math.* **197**, 233–244 (2006)
115. J. Majumdar, A.K. Bhunia, Elitist genetic algorithm for assignment problem with imprecise goal. *Eur. J. Oper. Res.* **177**, 684–692 (2007)
116. X. Ye, J. Xu, A fuzzy vehicle routing assignment model with connection network based on priority-based genetic algorithm. *World J. Model. Simul.* **4**, 257–268 (2008)
117. L. Liu, X. Gao, Fuzzy weighted equilibrium multi-job assignment problem and genetic algorithm. *Appl. Math. Model.* **33**, 3926–3935 (2009)
118. A. Kumar, A. Gupta, A. Kaur, Method for solving fully fuzzy assignment problems using triangular fuzzy numbers. *Int. J. Comput. Inf. Eng.* **3**:4, 231–234 (2009)

119. S. Mukherjee, K. Basu, A more realistic assignment problem with fuzzy costs and fuzzy restrictions. *Adv. Fuzzy Math.* **5**(3), 395–404 (2010)
120. R. Nagarajan, A. Solairaju, Computing improved fuzzy optimal Hungarian assignment problems with fuzzy costs under Robust ranking techniques. *Int. J. Comput. Appl.* **6**(4), 6–13 (2010)
121. F.G. Shi, A new approach to the fuzzification of matroids. *Fuzzy Sets Syst.* **160**, 696–705 (2009)
122. A. Kasperski, P. Zielinski, On combinatorial optimization problems on matroids with uncertain weights. *Eur. J. Oper. Res.* **177**, 851–864 (2007)
123. A. Kasperski, P. Zielinski, A possibilistic approach to combinatorial optimization problems on fuzzy-valued matroids, in *Fuzzy Logic and Applications, 6th International Workshop, WILF, Crema, Italy*, ed. by I. Bloch, A. Petrosino, A. Tettamanzi (Springer-Verlag, Berlin, Heidelberg, 2006), pp. 46–52
124. J. Fortin, A. Kasperski, P. Zielinski, Efficient methods for computing optimality degrees of elements in fuzzy weighted matroids, in *Fuzzy Logic and Applications, 6th International Workshop, WILF, Crema, Italy*, ed. by I. Bloch, A. Petrosino, A. Tettamanzi (Springer, Berlin, Heidelberg, 2006), pp. 99–107
125. A. Kasperski, P. Zielinski, Using gradual numbers for solving fuzzy-valued combinatorial optimization problems, in *Foundations of Fuzzy Logic and Soft Computing, 12th International Fuzzy Systems Association World Congress, Cancun, Mexico*, ed. by P. Melin, O. Castillo, L.T. Aguilar, J. Kacprzyk, W. Pedrycz (Springer, Berlin, Heidelberg, 2007), pp. 656–665
126. R. Goetschel, W. Voxman, Fuzzy matroids. *Fuzzy Sets Syst.* **27**, 291–302 (1998)
127. R. Goetschel, W. Voxman, Bases of fuzzy matroids. *Fuzzy Sets Syst.* **31**, 253–261 (1989)
128. R. Goetschel, W. Voxman, Fuzzy matroids and a greedy algorithm. *Fuzzy Sets Syst.* **37**, 201–214 (1990)
129. I.-C. Hsueh, On fuzzification of matroids. *Fuzzy Sets Syst.* **53**, 319–327 (1993)
130. L.A. Novak, A comment on ‘Bases of fuzzy matroids’. *Fuzzy Sets Syst.* **87**, 251–252 (1997)
131. L.A. Novak, On Goetschel and Voxman fuzzy matroids. *Fuzzy Sets Syst.* **117**, 407–412 (2001)
132. R. Goetschel, W. Voxman, Fuzzy matroids. *Fuzzy Sets Syst.* **27**, 291–302 (1998)
133. R. Goetschel, W. Voxman, Fuzzy matroids and a greedy algorithm. *Fuzzy Sets Syst.* **37**, 201–214 (1990)
134. M.J. Hwang, C.I. Chiang, Y.H. Liu, Solving a fuzzy set covering problem. *Math. Comput. Model.* **40**(7/8), 861–865 (2004)
135. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988)
136. R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, ed. by R. Miller, J. Thatcher (Plenum, New York, 1972), pp. 85–103
137. F. Barahona, M. Grötschel, M. Junger, G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design. *Oper. Res.* **36**, 493–513 (1998)
138. J. Poland, T. Zeugmann, Clustering pairwise distances with missing data: maximum cuts versus normalized cuts. *Lect. Notes Comput. Sci.* **4265**, 197–208 (2006)
139. H.F. Wang, M.L. Wang, A fuzzy multi-objective linear programming. *Fuzzy Sets Syst.* **86**, 61–72 (1997)
140. B. Liu, *Uncertainty Theory*, 2nd edn. (Springer, Berlin, 2007)
141. P.M. Pardalos, T. Mavridou, J. Xue, The graph coloring problem: a bibliographic survey, in *Handbook of Combinatorial Optimization*, vol. 2, ed. by D.Z. Du, P.M. Pardalos (Kluwer Academic, Boston, 1998), pp. 331–395
142. C. Eslahchi, B.N. Onagh, Vertex strength of fuzzy graphs. *Int. J. Math. Math. Sci.* **2006**, 1–9 (2006)
143. S. Munoz, T. Ortuno, J. Ramirez, J. Yanez, Coloring fuzzy graphs. *Omega* **32**, 211–221 (2005)
144. A. Chaudhuri, K. De, Fuzzy genetic heuristic for university course timetable problem. *Int. J. Adv. Soft Comput. Appl.* **2**(1), 100–123 (2010)

145. F. Eisenbrand, M. Niemeier, European Conference on Combinatorics, Graph Theory and Applications, Coloring fuzzy circular interval graphs. *Electron. Notes Discret. Math.* **34**, 543–548 (2009)
146. D. Kuchta, A generalisation of an algorithm solving the fuzzy multiple choice knapsack problem. *Fuzzy Sets Syst.* **127**(2), 131–140 (2002)
147. A. Kasperski, M. Kulej, The 0–1 knapsack problem with fuzzy data. *Fuzzy Optim. Decis. Mak.* **6**, 163–172 (2007)
148. F.T. Lin, J.S. Yao, Using fuzzy number in knapsack problem. *Eur. J. Oper. Res.* **135**(1), 158–176 (2001)
149. F.T. Lin, Solving the imprecise weight coefficients Knapsack problem by genetic algorithms, in *2006 IEEE International Conference on Systems, Man, and Cybernetics*, Taipei, Taiwan, 8–11 Oct 2006
150. M. Sakawa, K. Kato, T. Shibano, An interactive fuzzy satisfying method for multiobjective multidimensional 0–1 knapsack problems through genetic algorithms, in *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya University, Japan, 20–22 May, 1996 (Institute of Electrical and Electronics Engineers, Piscataway, 1996), pp. 243–246
151. S.P. Chen, Analysis of maximum total return in the continuous knapsack problem with fuzzy object weights. *Appl. Math. Model.* **33**, 2927–2933 (2009)
152. F.T. Lin, J.S. Yao, Using fuzzy number in knapsack problem. *Eur. J. Oper. Res.* **135**(1), 158–176 (2001)
153. S.P. Chen, Analysis of maximum total return in the continuous knapsack problem with fuzzy object weights. *Appl. Math. Model.* **33**, 2927–2933 (2009)
154. S. Sadi-Nezhad, K.K. Damghani, N. Pilevari, Application of 0–1 fuzzy programming in optimum project selection. *World Acad. Sci. Eng. Technol.* **64**, 335–339 (2010)
155. J.L. Verdegay, E. Vergara-Moreno, Fuzzy termination criteria in Knapsack problem algorithms. *Mathw. Soft Comput.* **7**, 89–97 (2000)
156. K. Kato, M. Sakawa, Genetic algorithms with decomposition procedures for multidimensional 0–1 knapsack problems with block angular structures. *IEEE Trans. Syst. Man Cybern. B* **33**(3), 410–419 (2003)
157. F.T. Lin, On the generalized fuzzy multiconstraint 0–1 knapsack problem, in *Proceedings of 2006 IEEE International Conference on Fuzzy Systems Sheraton Vancouver Wall Centre Hotel*, Vancouver, BC, Canada 16–21 July 2006
158. T. Hasuike, H. Katagiri, H. Ishii, Probability maximization model of 0–1 Knapsack problem with random fuzzy variables, in *Proceedings of 2008 IEEE International Conference on Fuzzy Systems*, Hong Kong, China, 2008. FUZZ
159. H. Shih, Fuzzy approach to multilevel Knapsack problems. *Comput. Math. Appl.* **49**, 1157–1176 (2005)
160. J.D. Ullman, Complexity of sequencing problems, in *Computer and Job-Shop Scheduling Theory*, ed. by E.G. Coffman (Wiley, New York, 1975)
161. C.B. Kim, K.A. Seong, H. Lee-Kwang, J.O. Kim, Design and implementation of FEGCS. *IEEE Trans. Syst. Man Cybern. A* **28**(3) (1998)
162. B.S. Baker, E.G. Coffman, R.L. Rivest, Orthogonal packing in two dimensions. *SIAM J. Comput.* **9**, 846–855 (1980)
163. B.S. Baker, D.J. Brown, H.P. Katsef, A 5=4 algorithm for two-dimensional packing. *J. Algorithm* **2**, 348–368 (1981)
164. H. Dyckho, A typology of cutting and packing problems. *Eur. J. Oper. Res.* **44**, 145–159 (1990)
165. M. Adamowicz, A. Albano, Nesting two-dimensional shapes in rectangular modules. *Comput. Aided Des.* **8**(1), 27–33 (1976)
166. B. Chazelle, The bottom-left bin packing heuristic: an efficient implementation. *IEEE Trans. Comput.* **C-32**(8), 697–707 (1983)
167. R. Dong, W. Pedrycz, A granular time series approach to long-term forecasting and trend forecasting. *Phys. A* **387**, 3253–3270 (2008)

168. J.N. Choi, S.K. Oh, W. Pedrycz, Identification of fuzzy models using a successive tuning method with a variant identification ratio. *Fuzzy Sets Syst.* **159**, 2873–2889 (2008)
169. A. Bargiela, W. Pedrycz, *Granular Computing: An Introduction* (Kluwer Academic, Dordrecht, 2002)
170. W. Pedrycz, K. Hirota, Fuzzy vector quantization with the particle swarm optimization: a study in fuzzy granulation_degranulation information processing. *Signal Process.* **87**, 2061–2074 (2007)
171. S. Okada, T. Soper, A shortest path problem on a network with fuzzy arc lengths. *Fuzzy Sets Syst.* **109**, 129–140 (2000)
172. H.S. Shih, E.S. Lee, Fuzzy multi-level minimum cost flow problems. *Fuzzy Sets Syst.* **107**, 159–176 (1999)
173. M. Dehghan, M. Ghatee, B. Hashemi, Inverse of a fuzzy matrix of fuzzy numbers. *Int. J. Comput. Math.* (2008). doi:10.1080/00207160701874789
174. M. Hukuhara, Intégration des applications measurable dont la valeurest un compact convexe. *FunkcialajEkvacioj* **10**, 205–223 (1967)
175. P. Diamond, R. Korner, Extended fuzzy linear models and least squares estimates. *Comput. Math. Appl.* **33**, 15–32 (1997)
176. M. Ghatee, S.M. Hashemi, B. Hashemi, M. Dehghan, The solution and duality of imprecise network problems. *Comput. Math. Appl.* **55**, 2767–2790 (2008)
177. R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows* (Prentice-Hall, Englewood Cliffs, 1993)
178. A. Bargiela, W. Pedrycz, *Granular Computing: An Introduction* (Kluwer Academic, Dordrecht, 2002)
179. M. Ghatee, S.M. Hashemi, Traffic assignment model with fuzzy level of travel demand: an efficient algorithm based on quasi-Logit formulas. *Eur. J. Oper. Res.* (2008). doi:10.1016/j.ejor.2007.12.023
180. S.K. Das, A. Goswami, S.S. Alam, Multiobjective transportation problem with fuzzy interval cost, source and destination parameters. *Eur. J. Oper. Res.* **117**, 100–112 (1999)
181. P. Ekel, W. Pedrycz, R. Schinzinger, A general approach to solving a wide class of fuzzy optimization problems. *Fuzzy Sets Syst.* **97**, 49–66 (1998)
182. M. Ghatee, S. Mehdi Hashemi, Some concepts of the fuzzy multi commodity flow problem and their application in fuzzy network design. *Math. Comput. Model.* **49**, 1030–1043 (2009)
183. H. Liu, X. Ban, B. Ran, P.B. Mirchandani, A formulation and solution algorithm for fuzzy dynamic traffic assignment model. *Transp. Res. Rec.* **178** (2003)
184. V. Henn, What is the meaning of fuzzy costs in fuzzy traffic assignment models? *Transp. Res. C* **13**, 107–119 (2005)
185. R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows* (Prentice-Hall, Englewood Cliffs, 1993)
186. P.R. Bhave, R. Gupta, Optimal design of water distribution networks for fuzzy demands. *Civil Eng. Environ. Syst.* **21**, 229–245 (2004)
187. S.K. Das, A. Goswami, S.S. Alam, Multiobjective transportation problem with fuzzy interval cost, source and destination parameters. *Eur. J. Oper. Res.* **117**, 100–112 (1999)
188. P. Ekel, W. Pedrycz, R. Schinzinger, A general approach to solving a wide class of fuzzy optimization problems. *Fuzzy Sets Syst.* **97**, 49–66 (1998)
189. A.M. Costa, A survey on benders decomposition applied to fixed-charge network design problems. *Comput. Oper. Res.* **32**, 1429–1450 (2005)
190. M. Ghatee, S.M. Hashemi, Some concepts of the fuzzy multicommodity flow problem and their application in fuzzy network design. *Math. Comput. Model.* **49**, 1030–1043 (2009)
191. D.Z. Du, J.M. Smith, J.H. Rubinstein, *Advances in Steiner Trees* (Kluwer Academic, Dordrecht, 2000)
192. F.K. Hwang, D.S. Richards, P. Winter, *The Steiner Tree Problem* (North-Holland, Amsterdam, 1992)

193. R. Karp, in *Complexity of Computer Computations*, ed. by R.E. Miller, J.W. Thatcher. IBM Research Symposia Series, vol. 43 (Plenum, New York, 1972), p. 85
194. G. Robins, A. Zelikovsky, in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, San Francisco, 2000), pp. 770–779
195. M. Durand, Phys. Rev. Lett. **98**, 088701 (2007)
196. M. Seda, Fuzzy shortest paths approximation for solving the fuzzy steiner tree problem in graphs. Int. J. Math. Comput. Sci. **1**:3, 22–26 (2005)
197. G. Gutin, A. Punnen (eds.), *Traveling Salesman Problem and Its Variations* (Kluwer Academic, Dordrecht, 2002); C. Helmberg, The m-cost ATSP. Lect. Notes Comput. Sci. **1610**, 242–258 (1999)
198. D. Feillet, P. Dejax, M. Gendreau, Traveling salesman problems with profits. Transp. Sci. **39**(2), 188–205 (2005)
199. E. Balas, G. Martin, *Roll-a-Round: Software Package for Scheduling the Rounds of a Rolling Mill* (Balas and Martin Associates, Pittsburgh, 1965)
200. C.P. Keller, M. Goodchild, The multiobjective vending problem: a generalization of the traveling salesman problem. Environ. Plan. B **15**, 447–460 (1988)
201. E.S. Lee, R.J. Li, Comparison of fuzzy numbers based on probability measure of fuzzy events. Comput. Math. Appl. **15**, 887–896 (1988)
202. N. Agin, Optimum seeking with branch and bound. Manage. Sci. **13**(4), B-176–B-185 (1996)
203. E. Demeulemeester, W. Herroelen, *Project Scheduling: A Research Handbook* (Kluwer Academic, Boston, 2002)
204. S.H. Owen, M.S. Daskin, Strategic facility location: a review. Eur. J. Oper. Res. **111**(3), 423–447 (1998)
205. D.J. van der Zee, J.G.A.J. van der Vorst, A modeling framework for supply chain simulation: Opportunities for improved decision making. Decis. Sci. **36**(1), 65–95 (2005)
206. U. Akinc, B.M. Khumawala, An efficient branch and bound algorithm for the capacitated warehouse location problem. Manage. Sci. **23**(6), 585–594 (1977)
207. P.M. Dearing, F.C. Newruck, A capacitated bottleneck facility location problem. Manage. Sci. **25**(11), 1093–1104 (1979)
208. M.A. Badri, Combining the analytic hierarchy process and goal programming for global facility location-allocation problem. Int. J. Prod. Econ. **62**(3), 237–248 (1999)
209. L. Dupont, Branch and bound algorithm for a facility location problem with concave site dependent costs. Int. J. Prod. Econ. **112**(1), 245–254 (2008)
210. Z. Drezner, W. Hamacher (eds.), *Facility Location: Applications and Theory* (Springer, New York, 2004)
211. A.T. Ernst, M. Krishnamoorthy, Solution algorithms for the capacitated single allocation hub location problem. Ann. Oper. Res. **86**(1–4), 141–159 (1999)
212. S. Lozano, F. Guerrero, L. Onieva, J. Larrañeta, Kohonen maps for solving a class of location-allocation problems. Eur. J. Oper. Res. **108**(1), 106–117 (1998)
213. A. Misra, A. Roy, S.K. Das, Information-theory based optimal location management schemes for integrated multi-system wireless networks. IEEE/ACM Trans. Netw. **16**, 525–538 (2008)
214. W. Pedrycz, F. Gomide, *An Introduction to Fuzzy Sets; Analysis and Design* (MIT, Cambridge MA, 1998)
215. W. Pedrycz, F. Gomide, *Fuzzy Systems Engineering: Toward Human-Centric Computing* (Wiley, Hoboken, 2007)
216. L.A. Zadeh, Is there a need for fuzzy logic? Inf. Sci. **178**(13), 2751–2779 (2008)
217. V. Batanovic, D. Petrovic, R. Petrovic, Fuzzy logic based algorithms for maximum covering location problems. Inf. Sci. **179**(1–2), 120–129 (2009)
218. U. Bhattacharya, J.R. Rao, R.N. Tiwari, Fuzzy multi-criteria facility location problem. Fuzzy Sets Syst. **51**(3), 277–287 (1992)
219. H. Ishii, Y.L. Lee, K.Y. Yeh, Fuzzy facility location problem with preference of candidate sites. Fuzzy Sets Syst. **158**(17), 1922–1930 (2007)
220. C. Kahraman, D. Ruan, I. Dogan, Fuzzy group decision-making for facility location selection. Inf. Sci. **157**, 135–153 (2003)

221. D. Dubois, H. Prade, *Possibility Theory* (Plenum, New York, 1988)
222. W. Pedrycz, F. Gomide, *Fuzzy Systems Engineering: Toward Human-Centric Computing* (Wiley, Hoboken, 2007)
223. L.A. Zadeh, Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets Syst.* **1**(1), 3–28 (1978)
224. U. Bhattacharya, J.R. Rao, R.N. Tiwari, Fuzzy multi-criteria facility location problem. *Fuzzy Sets Syst.* **51**(3), 277–287 (1992)
225. H. Ishii, Y.L. Lee, K.Y. Yeh, Fuzzy facility location problem with preference of candidate sites. *Fuzzy Sets Syst.* **158**(17), 1922–1930 (2007)
226. M. Wen, K. Iwamura, Fuzzy facility location-allocation problem under the Hurwicz criterion. *Eur. J. Oper. Res.* **184**(2), 627–635 (2008)
227. J. Zhou, B. Liu, Modeling capacitated location-allocation problem with fuzzy demands. *Comput. Ind. Eng.* **53**(3), 454–468 (2007)
228. R. Kruse, K.D. Meyer, *Statistics with Vague Data* (D. Reidel, Dordrecht, 1987)
229. H. Kwakernaak, Fuzzy random variables—I. Definitions and theorems. *Inf. Sci.* **15**, 1–29 (1978)
230. Y.K. Liu, B. Liu, Fuzzy random variable: a scalar expected value operator. *Fuzzy Optim. Decis. Mak.* **2**, 143–160 (2003)
231. R. Goetschel, W. Voxman, Fuzzy matroids and a greedy algorithm. *Fuzzy Sets Syst.* **37**, 201–214 (1990)
232. D. Kuchta, A generalisation of an algorithm solving the fuzzy multiple choice knapsack problem. *Fuzzy Sets Syst.* **127**(2), 131–140 (2002)
233. F.T. Lin, J.S. Yao, Using fuzzy number in knapsack problem. *Eur. J. Oper. Res.* **135**(1), 158–176 (2001)
234. L.A. Zadeh, Fuzzy sets. *Inf. Control* **8**(3), 338–353 (1965)

Geometric Optimization in Wireless Networks

Weili Wu and Zhao Zhang

Contents

1	Introduction	1416
2	Localization	1416
2.1	Range-Based Localization	1417
2.2	Angle-Based Localization	1422
2.3	Connectivity-Based Localization	1424
2.4	Location Ambiguity	1425
3	Geometric Routing	1430
3.1	Two Modes of Geographic Routing	1431
3.2	Routing with Bounded Stretch	1433
3.3	Routing Based on Virtual Coordinates	1438
3.4	Greedy Embedding	1442
4	Boundary Recognition	1445
4.1	Geometric Method	1445
4.2	Statistical Method	1447
4.3	Graph Method	1449
4.4	Topological Method	1451
5	Conclusion	1452
	Cross-References	1452
	Recommended Reading	1452

W. Wu (✉)

Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA
e-mail: weiliwu@utdallas.edu; [wxw020100@utdallas.edu](mailto:wxxw020100@utdallas.edu)

Z. Zhang

College of Mathematics and System Sciences, Xinjiang University, Urumqi, Xinjiang, People's Republic of China
e-mail: hxhzz@sina.com; zhzhao@xju.edu.cn

Abstract

This chapter focuses on geometric optimization problems including localization, routing, and boundary recognition. These topics are not new in networks. But wireless networks bring a lot of new ingredients and challenges to the problems. For example, because of the lack of central control unit, limited capabilities of sensors, limited transmission range, mobility features, etc., it is desirable to achieve the global tasks by utilizing local information in a distributed way. The focus is put on the main ideas of recent methods.

1 Introduction

Geometric optimization problems such as localization and routing are classic topics in the study of networks. With the emergence of wireless networks, new challenges are brought into these problems. First, in a large-scale wireless network, it is difficult to control all the processing by a center; distributed computation is more preferred. Second, the sensors are often small and cheap, which cannot afford expensive measuring devices such as GPS, and have limited computation and storage capability. Third, sensors are often equipped with a limited power. In many cases, the battery cannot be replenished. To save energy, the transmission range cannot be too long (thus nodes can only communicate with a limited number of neighbors), and algorithms should be light weighted on the nodes. Mobility is also a negligible feature of wireless network. When some sensors move or are depleted of energy, local adjustment is preferred to restore the functionality of the network.

The focus of this chapter is on geometric optimization problems in three fields: localization, routing, and boundary recognition. Due to the immense reservoir of literatures in these fields, this chapter is far from a comprehensive survey. The focus is put on the main ideas of recent methods.

2 Localization

In a wireless network, knowing the positions of the nodes is a basis for a lot of applications, for example, reporting the site of an event, drawing a map of a region, tracking targets, and helping the routing. Since the network usually consists of a huge amount of nodes which are deployed randomly or in an ad hoc fashion, it is hard to know the locations of the nodes *a priori*. Global Positioning System (GPS) is a widely used technique for localization. However, in some applications, GPS is not preferred due to the following reasons: (a) Inaccessibility: in a region dense of foliage or other obstacles, the signals will be blocked from the GPS satellites. (b) Power consumption: long distance signal transmission depletes the battery of the nodes quickly and reduces the lifetime of the entire network. (c) Cost and size: in a wireless sensor network, sensors are usually very cheap and as small as coins, which cannot afford the equipment of a GPS. In such a circumstance, *self-positioning* in a distributed manner is more preferred.

In many applications, a small fraction of the nodes are supposed to know their positions, either through manual placement or using GPS. These nodes are called *landmarks* or *anchors*. The other nodes are called *unknowns*. Due to power restrictions, a node can only communicate with those nodes which are very near to itself. In this setting, a large number of nodes are not in direct communication with the landmarks. The problem is how to utilize local information collected at each node over multi-hops to determine the positions of all the nodes. This task is significantly complicated when the field is *anisotropic*, that is, either the nodes are distributed irregularly or there are holes in the field.

[Section 2.1](#) first introduces the basic technique *multilateration* used to determine the positions of the unknowns using local *range measurement* and some refinements on the technique. Then, the technique of *rendered path* is introduced which is used in [59] to deal with anisotropic field. [Section 2.2](#) introduces the technique used to locate the unknowns using local *angle measurement*. To further minimize the cost of network configuration (without range or angle measurement), it is desirable to locate the nodes based *only* on the connectivity information. This topic is addressed in [Sect. 2.3](#). One problem occurs in the implementation is the location ambiguity: There exist different embeddings of the nodes satisfying the same connectivity and measure requirements (see, e.g., [Fig. 3](#)). Works focusing on the ambiguity issues are introduced in [Sect. 2.4](#).

Error issues in range-based, angle-based, range-/angle-free, and multimodal algorithms were further addressed in [68]. Robust statistical methods were proposed in [61] to make a localization system attack tolerant. This survey omits addressing these works in details. The interested reader may refer to the corresponding papers.

2.1 Range-Based Localization

In range-based localization, the local information known to a node includes the list of its neighbors as well as the distances between its neighbors and itself, which can be estimated by received signal strength (RSS) or time of arrival (ToA) or time difference of arrival (TDoA).

2.1.1 Multilateration

The idea of *iterative multilateration* was proposed in [76, 77] in which position information is propagated from landmarks to the unknowns and the unknowns determine their positions by solving least square optimization problems. In [66], the propagated information is the estimated distances, which is utilized by a GPS-like mechanic to locate the unknowns.

The terminology *beacons* used in [76] refer to the landmarks initially. The localization is executed in an iterative way: Unknowns which are in the neighborhood of enough beacons can determine their positions in a so-called atomic multilateration way; as soon as an unknown knows its position, it becomes a beacon and is used to determine the positions of more neighboring unknowns. This process is called *iterative multilateration*.

The atomic multilateration is introduced in the following: If an unknown has three beacon neighbors which are not colinear, its position can be determined by

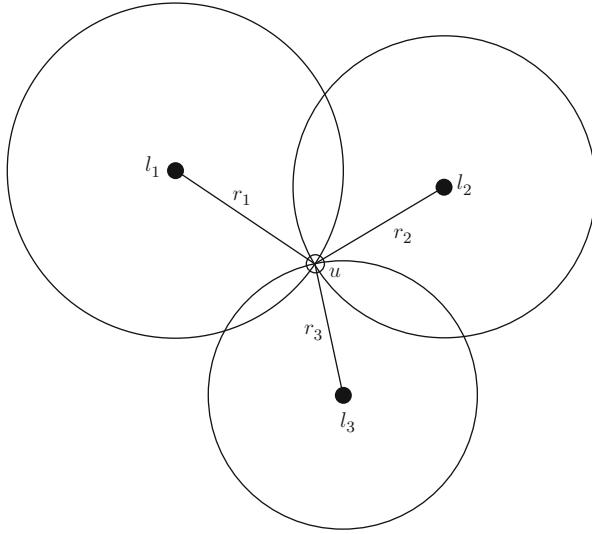


Fig. 1 Trilateration: A node which has distances r_1, r_2, r_3 to three noncollinear landmarks l_1, l_2, l_3 locates at the intersection point of the three circles C_1, C_2, C_3 , where C_i is the circle with center l_i and radius r_i

trilateration (see Fig. 1). If an unknown has four or more beacon neighbors, then due to measurement error, the position might be overdetermined. In such a situation, least square method is a natural solution: Suppose an unknown u with coordinate (x, y) has n beacon neighbors l_1, l_2, \dots, l_n , the estimated distance between u and l_i is d_i . The goal is to determine (x, y) such that

$$\sum_{i=1}^n err_i^2$$

is minimized, where

$$err_i = d_i - \sqrt{(x_i - x)^2 + (y_i - y)^2} \quad (1)$$

is the error between the measured distance and the estimated distance and (x_i, y_i) is the coordinate of beacon node l_i . This can be easily solved by linearization. In fact, the ideal situation is $err_i = 0$ for any $i = 1, 2, \dots, n$. For each i , setting $err_i = 0$ in (1) and squaring it, one obtains n equations:

$$(x_i - x)^2 + (y_i - y)^2 = d_i^2, \quad i = 1, 2, \dots, n. \quad (2)$$

Subtracting the last one from the other ones, one obtains a linear system of $n - 1$ equations:

$$2(x_n - x_i)x + 2(y_n - y_i)y = x_n^2 + y_n^2 - x_i^2 - y_i^2 - d_n^2 + d_i^2, \quad i = 1, 2, \dots, n - 1.$$

Fig. 2 An illustration of collaborative multilateration. The blacken nodes are beacon nodes, and the circled nodes are participating unknowns

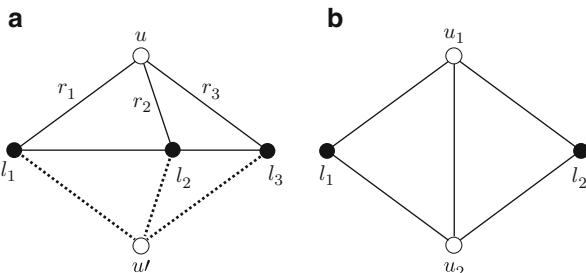
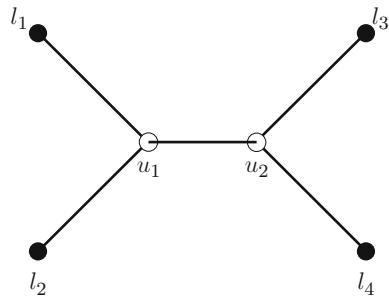


Fig. 3 Localization ambiguity

This linear system might be overdetermined, which can be solved by a classic least square method or generalized inverse in the field of numerical algebraic computation.

Based on atomic multilateration, Savvides et al. [76] proposed both a centralized way and a distributed way to realize the iteration phase of the iterative multilateration.

When there are no sufficient beacons near an unknown to determine the position, *collaborative multilateration* is used [76, 77]. The idea is illustrated using the small example in Fig. 2. For each edge, one establishes an equation as in (2). Notice that the equation corresponding to the edge u_1u_2 is related with two unknowns. There are four variables to be determined, namely, (x_1, y_1) and (x_2, y_2) (the coordinates of the two unknowns u_1 and u_2). Since there are five equations for this example, they can be determined as long as noncolinear requirement is satisfied. In general, Savvides et al. defined a node to be *participating* if it is either a beacon or an unknown with at least three participating neighbors. Thus, unknowns which are participating collaborate to determine their positions. Notice that the equation system is no longer linear; hence, optimization methods such as gradient descend or simulated annealing are used.

A drawback of the above multilateration method is the *error accumulation* due to the usage of inaccurate position information of unknowns as beacons. Another drawback is the *localization ambiguity* phenomenon. For example, if an unknown is adjacent to three beacons which are colinear, the position cannot be determined uniquely. Consider Fig. 3a for an example, both node u and u' have the same distances from beacons l_1, l_2, l_3 . Consider Fig. 3b for another example, exchanging

the locations of u_1 and u_2 , the equation system is still satisfied. The solution is not unique.

In [77], Savvides et al. further refined their method to overcome the above drawbacks. The refined method is executed in three main phases and a post-processing phase. In the first phase, nodes are grouped into so-called collaborative subtrees. The condition that a set of nodes could be assigned to a same subtree is that they can collaborate to *uniquely* determine the positions of the unknowns in the same group. This is achieved by obtaining a system of equations which is well determined or even overdetermined. For this purpose, some local conditions are imposed to exclude those nodes which violate the uniqueness if participating in a group. This phase aims to conquer the underdetermine problem (there is no sufficient information to determine the positions of the unknowns) and the localization ambiguity problem. In the second phase, *estimates* of the initial positions are obtained by a geometrical analysis on the known information including the measured distances between nodes and the positions of the beacons. In the third phase, the estimated positions are refined iteratively by least square method. Finally, in the post-processing phase, new location information is used to refine the positions of underconstrained nodes. The recursive refinement of the positions reduces the accumulated errors.

The methods proposed by Niculescu and Nath in [66] are based on the propagation of distances. The distance between two nodes which are not in direct contact with each other is estimated by propagating neighboring distance information through multi-hops. Once a node knows its estimated distances to at least three landmarks, it uses a mechanic which is similar to GPS triangulation to determine its coordinate. Three methods were proposed to estimate the distances. In *DV-distance propagation method*, based on the measured Euclidean distances between neighboring nodes, the distance between two non-neighboring nodes u, v is estimated by the length of a shortest (u, v) -path, where the length refers to the sum of the Euclidean distances along the path. In Euclidean propagation method, the Euclidean distance of a node u to a landmark l is estimated as follows. Suppose an unknown u has two neighbors u_1, u_2 which already know their estimated Euclidean distances to landmark l . Suppose u also knows the estimated Euclidean distances of uu_1, uu_2 , and u_1u_2 . Then the estimated distance between u and l can be obtained by a geometric analysis (see Fig. 4a). Clearly, for each of such a configuration, there are two possible positions for u . This localization ambiguity problem was dealt with in [66] by voting when more neighboring nodes of u have acquired their estimated distances to landmark l . The third method, *DV-hop propagation method*, proposed in [66] will be introduced in Sect. 2.3.

2.1.2 Rendered Path

The distances estimated in a hop-by-hop manner (such as the DV-distance propagation method introduced in the above) might be seriously distorted when the field is anisotropic: Shortest path may bend significantly around a hole, resulting in the estimated distances deviating largely from the true Euclidean distances.

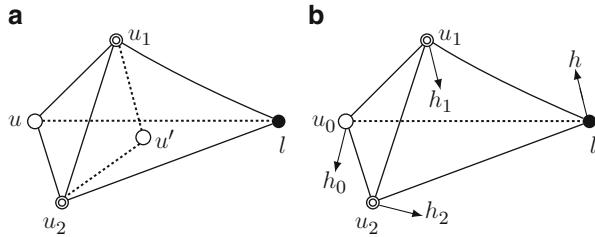


Fig. 4 The blackened nodes are landmarks. The circle nodes are unknowns. The double-circle nodes know their estimated distances in (a) or estimated orientations in (b). (a) Illustrates the estimation of the distance between u and l in Euclidean propagation method. There are two possible positions for node u . (b) Illustrates the estimation of the orientation of u from l in orientation propagation method

In [59], Li and Liu proposed the *rendered path* (REP) protocol to extract the true Euclidean information from the distances estimated by shortest paths in an anisotropic field. It was assumed that the boundaries of the holes are known in advance, and except for the holes, the rest of the network is isotropic. The main idea is to create *virtual holes* around the boundaries of holes, augment the shortest path by bypassing the virtual holes, and then extract range and angle information by comparing the shortest path with the augmented path.

First consider the basic situation when the shortest (u, v) -path intersects a convex hole at a point o (see Fig. 5a). The true Euclidean distance between u, v is

$$\|uv\| = \sqrt{\|uo\|^2 + \|vo\|^2 - 2\|uo\| \cdot \|vo\| \cdot \cos \angle uov}. \quad (3)$$

To estimate the angle $\angle uov$, a virtual hole with center o and radius r is created, and the shortest path is augmented to $uxyv$. Then

$$\angle uov = 2\pi - \frac{\|\widehat{xy}\|}{r} - \arccos \frac{r}{\|uo\|} - \arccos \frac{r}{\|vo\|}, \quad (4)$$

where $\|\widehat{xy}\|$ is the length of the arc \widehat{xy} (the shorter part along the hole centered at o). Since the lengths $\|uo\|, \|vo\|, \|ux\|, \|vy\|, \|\widehat{xy}\|$ are fairly accurate (because of the assumption of isotropy in the rest of the network excluding holes), the true Euclidean distance $\|uv\|$ can be obtained from (3) to (4).

In a general situation, a shortest path might intersect a lot of holes or intersect one hole more than once. To recognize the ways how the shortest paths meet the holes, a coloring strategy is used: Nodes on the boundaries of different holes are assigned different colors. Then a shortest path can be segmented into pieces according to the colors of the boundary nodes on it.

If a shortest path intersects a convex hole, it was proved in [59] that it intersects either at one point or along a continuous segment of the boundary. In the second case, two virtual holes centered at the tangent points were created to augment the

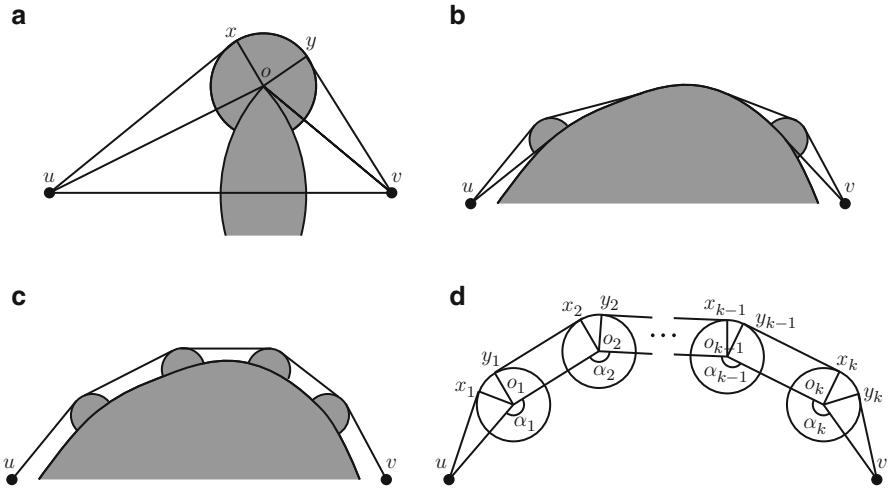


Fig. 5 Deriving true Euclidean distance using estimated distances on a shortest path

path (see Fig. 5b). If the augmented path still intersects the hole, more virtual holes were iteratively created (see Fig. 5c). By a geometrical analysis, the angles (see Fig. 5d) can be calculated by

$$\begin{aligned}\alpha_1 &= 1.5\pi - \frac{\|\widehat{x_1 y_1}\|}{r} - \arccos \frac{r}{\|u o_1\|}, \\ \alpha_i &= \pi - \frac{\|\widehat{x_i y_i}\|}{r} \text{ for } i = 2, \dots, k-1, \\ \alpha_k &= 1.5\pi - \frac{\|\widehat{x_k y_k}\|}{r} - \arccos \frac{r}{\|v o_k\|}.\end{aligned}$$

Then the true Euclidean distance $\|uv\|$ can be obtained by a simple geometrical calculation. A key observation in deriving the above formula is that the augmented path can only bend towards one direction.

In the case that a shortest path intersects more convex holes or some concave holes, the augmented path may bend towards different directions. Nevertheless, a geometrical analysis can produce a similar formula to obtain the angle information and thus the true Euclidean distance.

2.2 Angle-Based Localization

This subsection introduces the angle-based localization method used in [67]. Nodes are assumed to be able to sense the directions of the signals where they come from, using local interactions such as *angle of arrival* (AoA) or *directional antennas*.

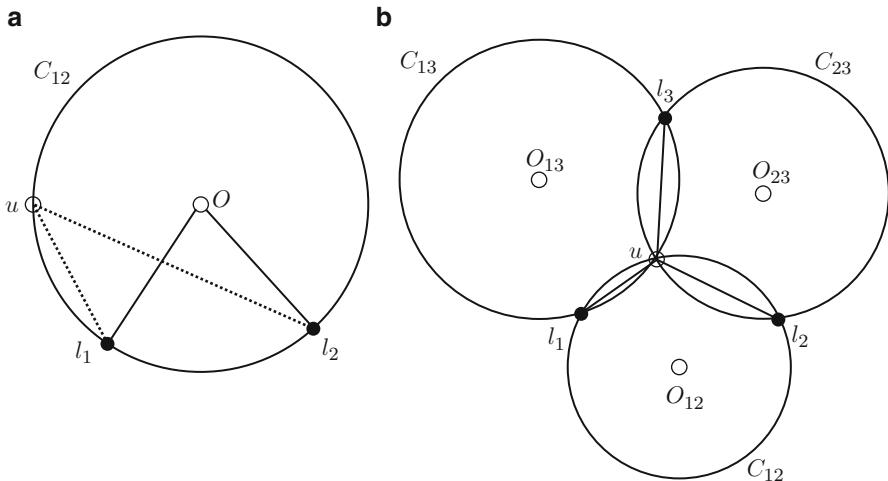


Fig. 6 Determine the position of node u by the direction information of landmarks l_1, l_2, l_3

The localization of a node using the direction information from its neighboring landmarks is called *triangulation*.

A node u which knows the directions of three of its neighboring landmarks can determine its position as shown in Fig. 6: If u sees two landmarks l_1, l_2 , then it must lie on the circle C_{12} as in Fig. 6a, where $\angle l_1 O l_2 = 2\angle l_1 u l_2$ and O is the center of the circle. Thus, if u sees three landmarks l_1, l_2, l_3 , then it is at the intersection point of the three circles C_{12}, C_{23} , and C_{13} (as in Fig. 6b).

If u sees more landmarks, a simplest method is to estimate the location by every triple of the landmarks and then place the node at the center of the estimated locations. Another method is to formulate the problem into a system of nonlinear equations, using the law of cosines, and solve it by least square method. This method is time-consuming. In [6], Betke and Gurvits proposed the idea of transforming the angle-based localization problem to the atomic multilateration problem, using the angle information to estimate the distances between the unknown and the landmarks. Representing the landmarks as complex numbers, an equation system linear in the number of landmarks was proposed in [6].

Since landmarks are only a small fraction of the nodes, most unknowns are not in direct contact with enough landmarks to determine their positions. The method proposed in [67] is based on the propagation of orientations. As long as an unknown has acquired the knowledge of its orientations from at least three landmarks, the above triangulation method is used to determine its position. This idea is similar to that in [66]. The difference is that distances are propagated in [66], while orientations are propagated in [67].

What remains to show is how to propagate the orientation information. See Fig. 4b for an illustration. Each node has a local heading h and can find out the angle between h and the direction of the signal (this angle is called the *orientation*).

Suppose node u_0 has two adjacent neighbors u_1, u_2 which know their orientations from landmark l . Then the orientation of u_0 from l (viz., $\angle h_0u_0l$) can be derived from the known orientation information $\angle h_iu_iu_j$ for $i, j = 0, 1, 2, i \neq j$ and $\angle h_iu_il, \angle hlu_i$ for $i = 1, 2$. After that, u_0 can use its orientation from l to help other neighbors to determine their orientations.

Similar to Sect. 2.1, error accumulation is a major problem of the above method. Some principles dealing with errors were discussed in [67].

2.3 Connectivity-Based Localization

In a connectivity-based localization, nodes are not equipped with any measuring devices (neither range nor angle). The problem is how to determine the positions of the unknowns based only on the connectivity information of the underlying topology of the network.

The *centroid-based method* proposed by Bulusu et al. [13] estimates the position of a node by calculating the centroid of its neighboring landmarks.

The *DV-hop-based method* proposed by Niculescu and Nath [66] first approximates the Euclidean distances by hop distances, which is called as *DV-hop propagation method*. Initially, the local information stored in a node is the list of its neighbors. Besides, landmark l_i is aware of its coordinate (x_i, y_i) . By multi-hop communication, all the nodes are informed of its *hop distances* from the landmarks. Then, each landmark l_i estimates its *average hop length* c_i by

$$c_i = \frac{\sum_j \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum_j h_{i,j}},$$

where the sum is over all the landmarks $l_j \neq l_i$ and $h_{i,j}$ is the hop distance between l_i and l_j . The values of c_i 's are flooded in a controlled manner: Once an unknown u receives a value c_i (from its nearest landmark), it ignores all the subsequent ones. Using this value, u estimates its Euclidean distances from the landmarks. That is, suppose u is h_j -hops away from landmark l_j , the estimated Euclidean distance between u and l_j is $c_i h_j$. Similar to Sect. 2.1, knowing the estimated Euclidean distances to at least three landmarks, the position of an unknown is determined by a GPS-like mechanic.

The method proposed by Shang et al. [80] is based on multidimensional scaling (MDS), which is a set of statistical techniques used in information visualization for exploring similarities in data [9]. Their algorithm takes three steps. First, a shortest path algorithm is used to estimate the distances between every pair of nodes, producing an estimated distance matrix. Then MDS is applied to the matrix, retaining the first 2 (or 3) largest eigenvalues and eigenvectors to generate a 2-dimensional (or 3-dimensional) map (the map is relative up to rotations and translations). Finally, knowing at least three absolute positions of the nodes (which

are called *seeds*), the absolute positions (i.e., the coordinates) of all the nodes are derived. Notice that without the aid of ranging measurement, the distances estimated in the first step are in fact hop distances.

Since both [66] and [80] use hop distances to approximate the true distances, these methods are only applicable to an *isotropic* field. As reported in [59], such hop distance-based methods perform poorly in an *anisotropic* field.

To deal with an anisotropic field, Shang and Ruml further refined their MDS-based method in [79]. The idea is to use MDS to construct a local map for each node based on its nearby neighbors and then merge the local maps into a global map. Since this method avoids using shortest path information between far away nodes, the distortion to the estimated distances is greatly alleviated.

The method CATL proposed in [85] also focuses on anisotropic field. The key idea is to detect *notch nodes*, where shortest paths bend from their straight-line courses. Those shortest paths passing through notch nodes are avoided or used with a low priority in estimating the distances. The detection of notch nodes is based on the observation that for a global shortest path tree, notch nodes usually have much *fatter* subtree, where *fatness* is measured by the size of the subtree up to a depth of a constant hops. It should be noted that this method can be conveniently generalized to the 3-dimensional space.

2.4 Location Ambiguity

As one has seen in Sect. 2.1, location ambiguity may occur, especially in a sparse region and with measurement noise. It is desirable that the nodes could be *uniquely localized*. Let G be the communication graph of the network. Given the positions of the landmarks and the mutual distances between neighboring nodes (distance may refer to hop distance if measurement devices are not provided), the *unique localizability* problem is to determine whether there is exactly one configuration for the unknowns which is consistent with the given information. Section 2.4.1 introduces the relationship between unique localizability and rigidity theory.

Knowing that a network is uniquely localizable is not sufficient to meet the requirement of network localization, one has to *realize* it, that is, find out the actual embedding of the graph. It was proved in [4, 22, 78] that realizing graphs on the plane is NP-hard, even when it is known that the graph is globally rigid. Some works focused on special classes of graphs which do admit an efficient realization [22, 38, 39, 64]. Some works used semidefinite programming to realize a graph [7, 81]. They will be introduced in Sect. 2.4.2.

In [57, 89], the authors asked the question of whether the knowledge of network boundaries (both the outer boundary and the inner boundaries around holes) could help to find out the true locations. The idea is to *choose* landmarks on the boundaries which can be embedded faithfully to reflect the layout of the network. Then the other nodes can be localized using trilateration based on at least three nearby landmarks.

With the layout being found fairly accurately, the danger of location ambiguity is reduced. These works will be introduced in Sect. 2.4.3.

2.4.1 Localizability and Rigidity

The *unique localizability* problem is closely related with rigidity theory [41]. A framework is a triple $\mathcal{F} = (V, E, p)$, where $G = (V, E)$ is a graph and p is an embedding of V in the d -dimensional space \mathbb{R}^d . Two frameworks $\mathcal{F}_p = (V, E, p)$ and $\mathcal{F}_q(V, E, q)$ are *equivalent* if the Euclidean distance $\|p(u)p(v)\| = \|q(u)q(v)\|$ holds for any edge $uv \in E$; they are *congruent* if one is obtained from the other by *rigid motion*, that is, translations, rotations, and reflections. A framework is *rigid* if it cannot be deformed continuously into another framework; it is *globally rigid* if any equivalent framework is congruent to it. A rigid framework might not necessarily be globally rigid (see Fig. 7). A framework \mathcal{F} is *generic* if any framework in the neighborhood of \mathcal{F} has the same rigidity as \mathcal{F} , that is, changing the locations of the points by an infinitesimal amount does not alter the rigidity of the framework. An important property is that for any two generic frameworks with the same underlying graph, they are either both rigid or both nonrigid. Thus, there is no ambiguity in defining a graph to be *rigid* (resp. *globally rigid*) if any of its generic embedding is rigid (resp. *globally rigid*).

One powerful characterization of rigidity was due to Laman.

Theorem 1 (Laman [56]) *A graph $G = (V, E)$ is rigid for \mathbb{R}^2 if and only if there is an edge subset $E' \subseteq E$ with $|E'| = 2|V| - 3$ and for any subset $E'' \subseteq E'$, $|E''| \leq 2|V(E'')| - 3$, where $V(E'')$ is the set of nodes spanned by E'' .*

Based on various formulations of Laman's Theorem, rigidity can be checked in polynomial time. For example, the Pebble Game proposed by Jacobs and Hendrickson [46] has running time $O(n^2)$, where n is the number of nodes in G .

Combinatorial characterization of global rigidity was studied in [5, 14, 43, 45]. A graph is *redundantly rigid* if the remaining graph resulted from the removal of any single edge is still rigid. Hendrickson [43] proved that a globally rigid graph in \mathbb{R}^d must be $(d + 1)$ connected and redundantly rigid. He also conjectured that the converse is true. For $d = 1$, it is trivial. For $d \geq 3$, it was disproved by Connelly [14]. The remaining case of $d = 2$ was proved to be true by Jackson and Jordán based on an inductive characterization of 3-connected graphs whose rigidity matroid is connected.

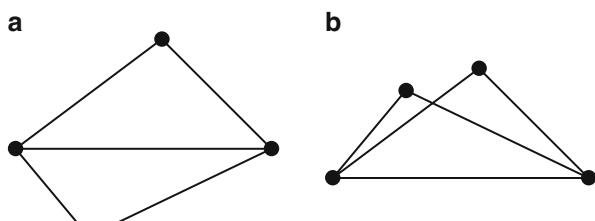


Fig. 7 A rigid but not globally rigid framework

Theorem 2 ([43, 45]) A graph G is globally rigid in \mathbb{R}^2 if and only if G is 3 connected and redundantly rigid.

The relationship between the unique localizability of a network and the global rigidity of a graph was established independently by Eren et al. [22] and Jackson and Jordán [45]. The *grounded graph* G_g is a super-graph of the communication graph G , containing extra edges between every pair of landmarks.

Theorem 3 ([22, 45]) A network G is uniquely localizable in \mathbb{R}^2 if and only if its grounded graph G_g is globally rigid.

2.4.2 Realization of Globally Rigid Graph

By Theorem 3, finding the positions of the nodes is reduced to realizing the weighted version of the grounded graph G_g , where the weight on each edge is the Euclidean distance between its two ends. The pioneer work [22] introduced a class of graphs called *trilateration graphs* and proved that they can be efficiently realized.

Definition 1 A *trilateration ordering* of a graph G in dimension d is an ordering of the nodes as u_1, u_2, \dots, u_n such that u_1, \dots, u_{d+1} form a complete subgraph K_{d+1} and for every $j \geq d + 2$, u_j has at least $d + 1$ neighbors in $\{u_1, u_2, \dots, u_{j-1}\}$. A graph is said to be a *trilateration graph in dimension d* if it admits a trilateration ordering in dimension d .

Theorem 4 ([22]) Trilateration graphs in dimension d are globally rigid in dimension d . Furthermore, they can be realized in polynomial time.

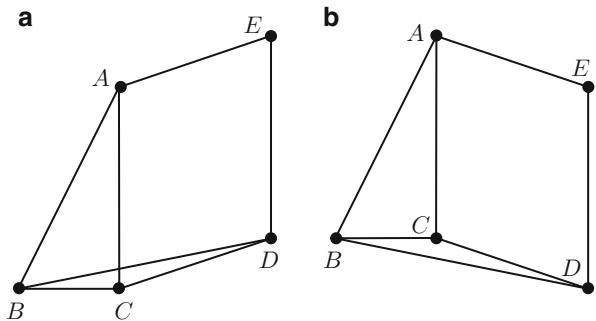
Eren et al. [22] proved that trilateration graphs exist in the real world with high probability. Let $U = \{u_1, u_2, \dots, u_n\}$ be a set of points in $[0, 1] \times [0, 1]$ generated by a two-dimensional Poisson distribution. Denote by $G_n(r)$ the graph on U such that two vertices are adjacent if and only if the distance between the two corresponding points is at most r .

Theorem 5 ([22]) If $\lim_{n \rightarrow \infty} \frac{nr^2}{\log n} > 8$, then $G_n(r)$ is a trilateration graph with a high probability. Thus, if $r \in O\left(\sqrt{\frac{\log n}{n}}\right)$, then with high probability, a realization of $G_n(r)$ can be computed in polynomial time.

In [64], Moore et al. suggested using *robust quadrilaterals* as building blocks for localization. The advantage of quadrilateral (referring to the complete graph on four vertices) is that (a) it is globally rigid itself; (b) if a quadrilateral Q has three common vertices with a globally rigid subgraph H , by pasting Q to H , one obtains a larger globally rigid graph. However, even for a globally rigid graph, location ambiguity is still possible because of measurement noise. To overcome this problem, Moore et al. introduced a measure of *robustness*. For a threshold d_{\min} which is chosen according to the measurement noise, a triangle is said to be *robust* if

$$b \sin^2 \theta > d_{\min}, \quad (5)$$

Fig. 8 Flex ambiguity.
Removing edge BD , the remaining graph can deform to another position.
Reinserting BD may keep all the distances



where θ and b are the smallest angle and the length of the shortest side of the triangle, respectively. A quadrilateral $ABCD$ can be divided into four triangles, namely, ABC , BCD , CDA , and DAB . It is *robust* if and only if all the four triangles are robust.

Theorem 6 ([64]) *If the measurement noise is normally distributed, using robust quadrilaterals as building blocks succeeds in localization with a bounded worst-case probability.*

In fact, there are two types of ambiguities in embedding a rigid graph on the plane. The *flip ambiguity* occurs when the neighbors of some node are all colinear (see Fig. 3a). The *flex ambiguity* occurs when the removal of some edge results in a nonrigid graph (see Fig. 8) or, in other words, when the graph is not redundantly rigid. By Laman's Theorem (Theorem 1), quadrilateral is redundantly rigid, and thus flex ambiguity does not occur for a graph which is constructed on quadrilaterals. Condition (5) is used to avoid flip ambiguity with a high probability.

Since trilateration graph is a stronger requirement than global rigidity, Goldenberg et al. [39] tried to relax it to *bilateration graph*, which was defined by taking $d = 1$ in Definition 1. By Theorem 4, such a graph is globally rigid on a straight line. However, it is no longer globally rigid on the plane. The way that [39] used to deal with the problem is to output all possible locations and prune the incompatible ones, which was called *finite localization*.

In practice, to uniquely localize the entire network is often unrealistic. In [38], Goldenberg et al. proposed the concept of *partially localizable network*. The focus is on the recognition of *uniquely localizable nodes*. Based on the characterization of global rigidity (Theorem 2), a sufficient condition was presented in [38]: A uniquely localizable node u must belong to a redundantly rigid 3-connected subgraph which contains three landmarks.

Biswas and Ye [7] and So and Ye [81] used semidefinite programming to find the realization. First, they modeled the network localization problem as

$$\begin{cases} \|\alpha_k - x_i\|_2^2 = \bar{d}_{ki}^2, \\ \|x_j - x_i\|_2^2 = d_{ji}^2, \end{cases} \quad (6)$$

where $\|\cdot\|_2$ is the 2-norm of vector space \mathbb{R}^d , α_k is the coordinate of landmark u_k , x_i is the coordinate of unknown node v_i , \bar{d}_{ki} is the distance between landmark u_k and unknown node v_i , and d_{ji} is the distance between two unknown nodes v_j, v_i . Equation (6) is a non-convex optimization problem. Biswas and Ye [7] and So and Ye [81] transformed it into a semidefinite programming problem (SDP) and showed that the network is globally rigid in \mathbb{R}^d if and only if the max-rank solution matrix of the SDP has rank d . Thus, as long as the network is uniquely localizable, the realization can be computed in polynomial time by solving the SDP, using, for example, the interior-point algorithm.

2.4.3 Boundary-Aided Localization

This subsection introduces the works of [57, 89] which used boundary information to help faithful localization. The main ideas of the overall algorithm will be introduced first, followed by an introduction on how to select landmarks on the boundary.

Suppose the sensors are distributed in a region \mathcal{R} , and the boundary of \mathcal{R} , denoted by $\partial\mathcal{R}$, is known, for example, by the methods introduced in Sect. 4. Use $g(u, v)$ to denote the *geodesic distance* between u, v inside \mathcal{R} , that is, the length of the shortest path avoiding obstacles. For a node u in the landmark set \mathcal{L} , the *Voronoi cell* of u is the set of points in \mathcal{R} which have u as the nearest landmark, that is,

$$V(u) = \{x \in \mathcal{R} \mid g(x, u) \leq g(x, w) \forall w \in \mathcal{L}\}.$$

The *combinatorial Delaunay complex* on \mathcal{L} is the complex $DC(\mathcal{L})$ defined on the set

$$\{L \subseteq \mathcal{L} \mid \bigcap_{u \in L} V(u) \neq \emptyset\}.$$

The dimension of a Delaunay simplex $L \subseteq \mathcal{L}$ is $\dim(L) = |L| - 1$. Hence, a landmark is a 0-dimensional simplex, a Delaunay edge is a 1-dimensional simplex, and a Delaunay triangle is a 2-dimensional simplex. Notice that in the degenerate case, $k (\geq 4)$ nodes are colinear, the complete subgraph K_k may be present in $DC(\mathcal{L})$, which is a $(k-1)$ -dimensional simplex. The *combinatorial Delaunay graph* $D(\mathcal{L})$ as in Sect. 3.3 only contains the 0-dimensional and 1-dimensional simplices. It was proved that when the landmarks are dense enough, $D(\mathcal{L})$ is rigid, but not generally globally rigid [57]. The intuitive idea is that the higher-order simplices K_k with $k \geq 3$ contained in $DC(\mathcal{L})$ will help to exclude more ambiguities. This fact was proved rigorously in [57] as long as the landmarks are sufficient dense, where *density* is measured as follows: The *inner medial axis* of \mathcal{R} is the collection of points in \mathcal{R} which have at least two nearest points on $\partial\mathcal{R}$. The *inner local feature size* of a point $x \in \partial\mathcal{R}$, denoted as $ILFS(x)$, is the distance of x to its nearest point on the inner medial axis. A set \mathcal{L} of landmarks is said to be an γ -sample, if for every point $x \in \partial\mathcal{R}$, the disk with center x and radius $\gamma \cdot ILFS(x)$ contains at least one landmark from \mathcal{L} . A *boundary cycle* is a cycle surrounding a hole or the outer boundary.

Theorem 7 ([57]) Suppose the set of landmarks \mathcal{L} is a γ -sample for some $\gamma < 1$ and each boundary cycle contains at least three landmarks. Then, the combinatorial Delaunay graph $D(\mathcal{L})$ is rigid, and the combinatorial Delaunay complex $DC(\mathcal{L})$ is globally rigid.

The algorithm in [57] is executed in five steps: (a) select landmarks on the boundary, (b) compute the landmark Voronoi diagram which is the union of the Voronoi cells, (c) extract the combinatorial Delaunay complex (CDC), (d) embed CDC by gluing the simplices in an incremental way, mass spring relaxation is used to smooth out the noise, and (e) the other nodes are localized by trilateration. It should be noted that in a wireless network with connectivity information only, distances are measured by hops. The algorithm uses hop distances to approximate geodesic distances.

The limitation of the above algorithm lies in its dependency on the quality of the detected boundary. In [89], an incremental algorithm was presented to select landmarks on the boundary without knowing the boundary in advance (the boundary is delimitated with the incremental selection of the landmarks). The algorithm does not depend on the inner local feature size (and thus the computation of the inner medial axis is not needed). In fact, the selected set of landmarks not only guarantees the global rigidity (viz., γ -sample for some $\gamma < 1$) but also guarantees a good coverage, the so-called δ -coverage defined as follows. A point in \mathcal{R} is said to be on a *Voronoi edge* if it has the same distance to two of its nearest landmarks. A point in \mathcal{R} is an *inner Voronoi vertex* if it has the same distance to at least three landmarks. For an inner Voronoi vertex x , the *Voronoi disk* $B_r(x)$ centered at x has its radius r equal to the distance from x to its nearest landmark. The combinatorial Delaunay complex $DC(\mathcal{L})$ is said to δ -cover \mathcal{R} if for every point $y \in \mathcal{R}$, there is an inner Voronoi vertex x such that y is within distance $(1 + \delta)r$ from x , where r is the radius of the Voronoi disk $B_r(x)$. Wang et al. [89] proved that if the landmarks are selected complying with the following two conditions, then both the rigidity and the coverage requirements can be satisfied:

- (a) The Voronoi edges for each landmark u are connected.
- (b) Each node in $V(u)$ is δ -covered by a Voronoi disk $B_r(x)$, where x is a Voronoi vertex with u being one of its nearest landmarks.

Based on the above two conditions, an incremental landmark selection algorithm was presented [89]. Notice that these two conditions can be checked locally.

3 Geometric Routing

In geometric routing, a widely adopted model is the unit disk graph model. Assume that two nodes can communicate with each other only when their Euclidean distance is within a constant transmission range, which is scaled to one. Then the topology underlying the geometric routing problem can be modeled as a *unit disk graph* G , in which each vertex corresponds to a node on the plane and two vertices are adjacent in G if and only if the Euclidean distance between the two nodes corresponding to them is at most one. A more general model is quasi unit disk graph. Suppose

$p : V \mapsto \mathbb{R}^2$ is a straight-line embedding of $G = (V, E)$ on the plane, $d \leq 1$ is a parameter. If the following two conditions hold

$$uv \in E \implies \|p(u)p(v)\| \leq 1, \quad (7)$$

$$\|p(u)p(v)\| \leq d \implies uv \in E, \quad (8)$$

then p is said to be a *d-quasi unit disk embedding* (*d-QUDE*) and G is called a *d-quasi unit disk graph* (*d-QUDG*). In particular, if $d = 1$, then G is exactly a unit disk graph. Notice that a quasi unit disk graph is not necessarily planar.

Geographic routing uses the geographic locations of the nodes as addresses. Every node knows its position and the positions of its neighbors. Packets are sent from their sources to their destinations through multi-hops. Every packet knows the position of its destination from the very beginning of the transmission. Most geographic routing algorithms consist of two modes: the *greedy forwarding mode* and the *recovery mode*. [Section 3.1](#) illustrates the main ideas of the two modes using the *greedy perimeter stateless routing* (GPSR) protocol proposed in [48]. A similar protocol was proposed independently by Bose et al. in [11].

In practice, the quality of the route should be taken into consideration. For example, the route found by the algorithm should not be too long compared with the optimal one; the load should be balanced which will increase the overall lifetime of the network. [Section 3.2](#) introduces some algorithms with a guarantee on the length of the route. As to the studies on load-balanced routings, the interested reader may refer to [34, 63, 70, 74, 90].

The recovery mode is needed because greedy forwarding strategy will encounter *local minimum phenomenon*, that is, the packet will get stuck because the current node is closer to the destination than any of its neighbors. This problem is especially serious when the region contains holes. Furthermore, although some algorithms of geographic routing theoretically guarantee the successful delivery of the packets, the actual delivery rate in the real network is hindered by the location inaccuracies. This motivates the design of routings based on *virtual coordinates*, which was first proposed by Rao et al. [72]. Virtual coordinate does not reflect the absolute positions of the nodes but can be chosen to increase the delivery rate of greedy strategy. [Section 3.3](#) illustrates the basic ideas of this topic by the works in [23, 65]. A related question is: is there an embedding of a graph in which the greedy forwarding strategy always succeeds? Such an embedding is called a *greedy embedding*; the studies of which will be introduced in [Sect. 3.4](#).

3.1 Two Modes of Geographic Routing

In [48], the communication graph G is taken to be the unit disk graph.

Greedy forwarding is a simple strategy to route the packets which can be realized in a distributed manner: When a packet reaches a node u , the local optimal choice for the next hop is the neighbor of u which is closest to the destination, where *closest*

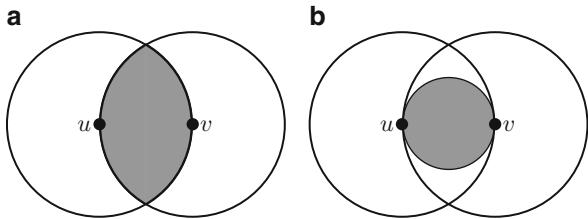
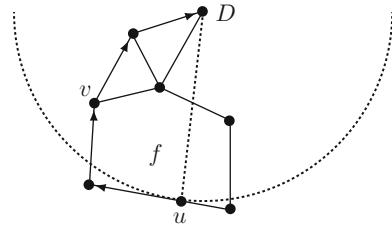
Fig. 9 (a) RNG. (b) GG

Fig. 10 u is a stuck node from which the algorithm enters the recovery mode, and from v , the algorithm goes back to the greedy forwarding mode



may be measured by different criteria. Greedy forwarding strategy works well in an evenly distributed dense network but may get stuck in a sparse network, in which there might be a case that any neighbor of u is farther from the destination than u itself. Such a node is called a *stuck node*. Such a case is called a *local minimum phenomenon*.

When a packet reaches a stuck node, the recovery mode is activated. A natural idea is to make a detour around the boundary of a *face*. To do so, a planar subgraph of the communication graph G is required, since faces are defined only for a plane graph (a plane embedding of a planar graph) which has no crossing edges.

Two planar subgraphs are used in [48]: the *relative neighborhood graph* (RNG) and the *Gabriel graph* (GG). See Fig. 9 for an illustration. Two nodes u, v are adjacent in RNG only when the shaded lune in Fig. 9a contains no other nodes, where the two circles have radius $\|uv\|$, the Euclidean distance between u and v . They are adjacent in GG only when the shaded circle in Fig. 9b contains no other nodes. Clearly, both RNG and GG are planar and RNG is a subgraph of GG. Let $G_{RNG} = G \cap RNG$ and $G_{GG} = GG \cap G$.

The recovery procedure is illustrated by Fig. 10. When the packet reaches a stuck node u , let f be the face of the plane subgraph which contains u and intersects the line uD , where D is the destination of the packet. The packet is forwarded along the boundary of f by, for example, the right-hand rule [48], until it reaches a node which is closer to D than u . From that node, the algorithm goes back to the greedy forwarding mode.

To ensure that the packet is always capable of reaching its destination as long as there exists a source-destination path in the communication graph G , an important property that the planar subgraph must satisfy is any nodes which are reachable to each other in G are also reachable to each other in the planar subgraph. G_{RNG} can

be viewed as a subgraph obtained from G by removing those edges uv for which there is a node w in the lune. Since u and v are connected by the path uvw , removing edge uv does not disconnect the graph. Thus, G_{RNG} satisfies the above property. Since G_{RNG} is a subgraph of G_{GG} , G_{GG} also satisfies the above property.

There are a lot of protocols using the same idea as the above two modes in geographic routing. The differences lie in the choice of greedy criterion and recovery mechanic. Some greedy criteria include closest to destination [11, 48, 84], nearest forward progress [44, 83], most forward within radius [84], closest direction [51], and some literatures use n -neighborhood information to find the next forwarding node [27, 82]. Some recovery mechanics include simple flooding [82], tree search [47], terminode routing [8], and face routing [48, 51].

3.2 Routing with Bounded Stretch

The work [53] is an improvement on the face routing of [51]. Notice that the face routing method used in [51] is different from that in Sect. 3.1. It never uses the greedy mode and just traverses the boundaries of those faces which intersect the line segment \overline{uv} (see Fig. 11a).

As one can see from Fig. 11a, the route found by the face routing method might be much longer than an optimal one. It is desirable that the cost of the computed

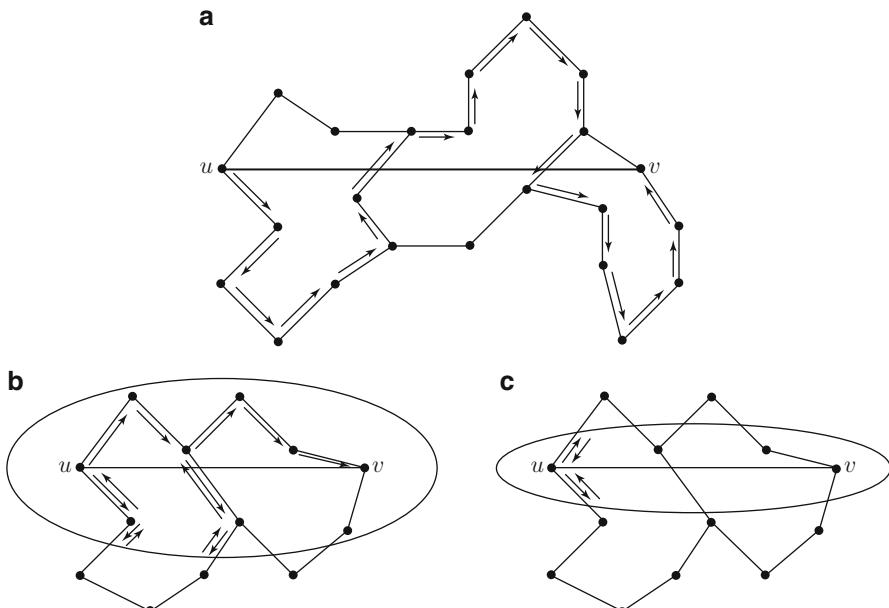
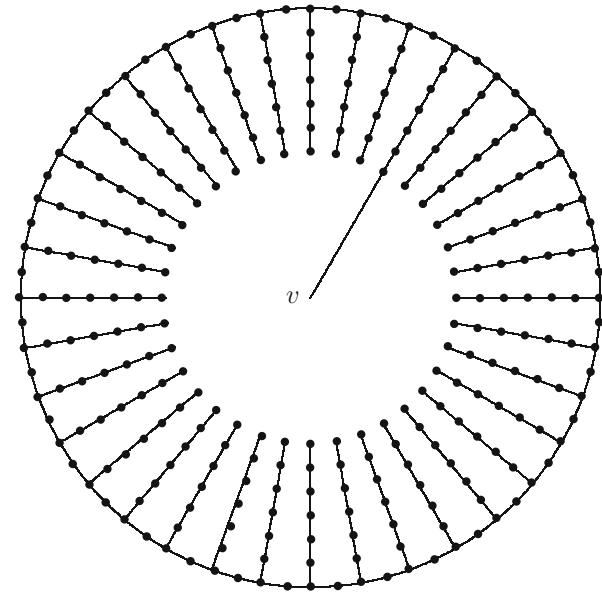


Fig. 11 (a) An illustration of face routing used in [51]. (b) An illustration of BFR. The arrows indicate the edges traversed. (c) When the ellipse is too small, BFR may get stuck

Fig. 12 An example showing a lower bound for the stretch. The route found by a local algorithm may travel all the nodes before it reaches the destination node v



route is not too large compared with the optimal cost, in other words, the *stretch* is desired to be small. In a general case, the cost c is a nondecreasing positive function defined on E . In particular, if for each edge $uv \in E$, $c(uv) = 1$, then c is the *link metric* measuring hop distances between two nodes; if $c(uv) = \|uv\|$, then c is the *Euclidean metric* measuring the sum of Euclidean distances along the path; and if $c(uv) = \|uv\|^2$, then c is the *energy metric* measuring the energy spent on a path.

In [53], Kuhn et al. constructed an example (see Fig. 12) showing that any deterministic geometric routing algorithm based on local information has a worst cost of $\Omega(c_*^2)$, where c_* is the cost of an optimal route. The same example shows that the claim is also true [53] for any randomized local algorithm in terms of expected cost. In [53, 54], local algorithms were presented producing routes with guaranteed bound $O(c_*^2)$, which are asymptotically optimal in view of the above lower bound.

All these algorithms rely on the existence of a planar subgraph whose stretch factor is bounded by a constant. Such a subgraph is called a *geometric spanner*, which will be introduced in Sect. 3.2.1. Section 3.2.2 shows how geometric spanners were used to guarantee the asymptotic optimality of the routes found by the algorithms in [53, 54].

3.2.1 Geometric Spanner

A k -spanner of graph G is a spanning subgraph H of G such that $c(P_H(uv)) \leq k \cdot c(P_G(uv))$ holds for any pair of nodes u, v , where $P_H(uv)$ and $P_G(uv)$ are the minimum cost (u, v) -paths in H and G , respectively. A k -spanner with k upper bounded by a constant is simply called a *spanner*. The parameter k is called the

stretch factor of the spanner. In particular, if c is taken to be link metric, Euclidean metric, or energy metric, the corresponding stretch factor is called *topological stretch factor*, *Euclidean stretch factor*, and *energy stretch factor*, respectively.

As one has seen in Sect. 3.1, both the relative neighborhood graph RNG and the Gabriel graph GG are planar. However, they are not good spanners [20]. In fact, Bose et al. [12] proved that the topological stretch factor can be $\Omega(\sqrt{n})$ for GG and $\Theta(n)$ for RNG, while the Euclidean stretch factor is $\Theta(\sqrt{n})$ for GG and $\Theta(n)$ for RNG.

The following lemma shows that the subgraph G_{GG} is a good spanner with respect to energy metric:

Theorem 8 ([53]) *The energy stretch factor of G_{GG} is 1.*

In [53], Kuhn et al. assumed an $O(1)$ -model (also known as *civilized graph* or λ -*precision graph*), meaning that there is a constant d_0 such that the Euclidean distance between any pair of nodes is at least d_0 . It is easy to see that in an $O(1)$ -model, the three commonly used metrics are equivalent.

Lemma 1 ([53]) *For the $O(1)$ -model, the link metric c_l , Euclidean metric c_d , and energy metrics c_e are equivalent up to a constant depending on d_0 .*

The next corollary follows directly from Theorem 8 and Lemma 1.

Corollary 1 ([53]) *For an $O(1)$ -model, G_{GG} is a spanner with respect to link, Euclidean, and energy metrics.*

A *bounded degree graph* is a graph whose maximum degree is upper bounded by a constant. Kuhn et al. [54] proved that in a bounded degree unit disk graph, any nondecreasing cost functions are equivalent up to constant factors. Then by Theorem 8, one has the following result:

Corollary 2 ([54]) *For a bounded degree unit disk graph G , G_{GG} is a spanner with respect to any nondecreasing cost functions.*

The Delaunay triangulation is known to be a planar spanner in terms of Euclidean metric [18, 49]. However, it cannot be used directly in the setting of wireless network, since a Delaunay edge may be longer than the transmission range. To overcome this problem, [36, 60] introduced the *restricted Delaunay graph* (RDG), which is obtained from the Delaunay triangulation by removing all the edges of length longer than 1. Let $G_{RDG} = RDG \cap G$. Gao et al. [36] and Li et al. [60] proved that G_{RDG} is a good spanner with respect to Euclidean metric.

Theorem 9 ([36, 60]) *The Euclidean stretch factor of G_{RDG} is $(1 + \sqrt{5}/2)\pi \approx 5.08$.*

However, the topological stretch factor of G_{RDG} might be as large as $\Theta(n)$. The reason is that although two nodes u, v are *visible* to each other on the plane, if the

network is very dense, then one has to cross a lot of Delaunay triangles to go from u to v .

If G has a *constant density*, meaning that every unit disk contains at most $O(1)$ nodes, then G_{RDG} is also a topological spanner.

Theorem 10 ([36]) *For a unit disk graph G which has constant density, the topological stretch factor of G_{RDG} is bounded by a constant.*

3.2.2 Routing Algorithms with Bounded Stretch

The algorithms in [53, 54] are improvements of the face routing algorithm in [51] and thus were named as *adaptive face routing* (AFR) and *GOAFR⁺*, respectively. The results in [53] are valid for all the three commonly used metrics, namely, the link, Euclidean, and energy cost. The limitation is that it assumes an $O(1)$ -model. This assumption is very crucial to their theoretical analysis. Later, this assumption was dropped off in [54] by using a clustering method.

Recall that face routing needs a planar subgraph to provide faces to be routed around. In [53], the planar subgraph of G is taken to be G_{GG} .

The idea of AFR is to restrict the route within an ellipse \mathcal{E} with foci u (the source) and v (the destination), that is, \mathcal{E} is the locus of all the points x with $\|xu\| + \|xv\| = \hat{c}$, where \hat{c} is a constant.

The basic procedure of AFR is called *bounded face routing* (BFR) (see Fig. 11b for an illustration). For a fixed \hat{c} , let $\mathcal{E}_{\hat{c}}$ be the ellipse corresponding to \hat{c} . While exploring a face F , one begins with any one of the two directions. There are two possibilities, one either goes around the *whole* boundary or comes across the boundary of $\mathcal{E}_{\hat{c}}$. In the latter case, one goes back to explore the other direction of the face until the boundary of $\mathcal{E}_{\hat{c}}$ is met again. Among all the intersection points of the line segment \overline{uv} and the *traversed* boundaries of face F , let x be the one which is nearest to v . The next face to be explored is the one incident with x which intersects line segment \overline{xv} . Kuhn et al. proved the following result:

Theorem 11 ([53]) *For an $O(1)$ -model, let \tilde{c} be the cost of a minimum cost (u, v) -path in G_{GG} . If \hat{c} is an upper bound for \tilde{c} , then BFR succeeds in finding a (u, v) -path with cost at most $O(\hat{c}^2)$.*

To fully understand the key points in the method, the sketch of the proof of Theorem 11 is addressed in the following: In view of Lemma 1, it suffices to deal with the link metric. Notice that with the assumption that $\hat{c} \geq \tilde{c}$, the shortest (u, v) -path of G_{GG} must lie completely in \mathcal{E} , since any path through a point outside of $\mathcal{E}_{\hat{c}}$ will have length greater than $\hat{c} \geq \tilde{c}$ by the definition of ellipse. This guarantees that a (u, v) -path lying within $\mathcal{E}_{\hat{c}}$ can be found by BFR. Furthermore, the $O(1)$ -model assumption implies that the disks with centers at the explored nodes and radius $d_0/2$ are mutually disjoint. Combining this with the observation that the length of the semimajor axis of $\mathcal{E}_{\hat{c}}$ is $\hat{c}/2$, the number of explored nodes inside $\mathcal{E}_{\hat{c}}$ is upper bounded by

$$\frac{\pi(\frac{\hat{c}}{2} + \frac{d_0}{2})^2}{\pi(\frac{d_0}{2})^2} = O(\hat{c}^2). \quad (9)$$

Since G_{GG} is a planar graph, the number of edges is linear in the number of nodes (it is well known that $m \leq 3n - 6$, where m, n are the number of edges and the number of nodes in a planar graph). Hence, there are at most $O(\hat{c}^2)$ edges within \mathcal{E} . Finally, by noticing that each edge is traversed at most four times in BFR, the theorem follows.

Since \tilde{c} is not known a priori, BFR might get stuck if the ellipse is too small (see Fig. 11c). In such a case, the algorithm enlarges \hat{c} to obtain a larger ellipse and restarts BFR. By adaptively adjusting \hat{c} , the algorithm obtains an estimate of \tilde{c} such that $\tilde{c} \leq \hat{c} \leq 2\tilde{c}$. For this \hat{c} , BFR produces a route with cost at most $O(\tilde{c}^2)$ and thus $O(c_*^2)$ by Corollary 1.

From the above analysis, it can be seen that the $\mathcal{O}(1)$ -model assumption is crucial to obtain the asymptotic optimality of AFR because of the following two reasons: First, the planar subgraph providing the faces must be a spanner of G . This is guaranteed by Corollary 1. Second, the number of edges explored must be bounded. This is guaranteed by inequality (9). A natural question is: besides $\mathcal{O}(1)$ -model, are there any other graphs supporting these two properties? In view of Theorems 9 and 10, if Euclidean metric or link metric is used, a graph G with constant density and its subgraph G_{RDG} will do. In view of Corollary 2, a bounded degree graph and its subgraph G_{GG} might be another candidate with respect to any nondecreasing cost function (however, some more efforts have to be made to adapt the algorithm).

Another idea is based on the observation that in AFR, the whole boundary of a face must be explored before the algorithm moves to the next face, which seems to be a waste of time. On the other hand, greedy forwarding strategy is averagely more efficient than face routing [55]. This is particularly true for dense graphs, since a packet can be forwarded to a visible node instead of going through a lot of small hops. So, the idea is an *early fallback to greedy mode*.

The algorithm $GOAFR^+$ is the combination of the above ideas and the bounded face routing. Face routing is only used when encountering a local minimum. Two counters p and q were introduced, counting the number of explored nodes which are closer and not closer to the destination than the starting node, respectively. If $p > \sigma q$, where σ is a constant (taken to be 1/100 in [54]), the algorithm advances to the explored node which is closest to the destination. Kuhn et al. proved that $GOAFR^+$ is asymptotically optimal for bounded degree unit disk graphs with respect to any nondecreasing cost function.

Theorem 12 ([54]) *Let G be a bounded degree unit disk graph. If u, v are connected in G , then $GOAFR^+$ finds a route with cost $O(c_*^2)$.*

To extend $GOAFR^+$ to deal with the general (unbounded degree) unit disk graphs, a clustering strategy is used. A routing backbone graph G_{BG} is constructed such that $V(G_{BG})$ is a connected dominating set of G and G_{BG} has bounded degree. Such G_{BG} can be constructed in a distributed way, for example, by the algorithms

in [1, 35, 87]. Every *ordinary node* $u \in V(G) - V(G_{BG})$ is assigned to one of its dominators $h(u) \in V(G_{BG})$ (viewed as joining the cluster with cluster head $h(u)$). These assignments enlarge G_{BG} to the *clustered backbone graph* G_{CBG} . It is known that G_{CBG} is a spanner of G with respect to link metric [87]. A packet with source u and destination v is first sent to $h(u)$, then routed through G_{BG} to $h(v)$, and finally sent to v . Using *GOAFR⁺* on G_{BG} , one obtains an asymptotic optimal route for G_{CBG} and thus G . This result also holds for any *linearly bounded cost function*, defined to be a function c for which there exists an $m > 0$ such that $c(d) \geq m \cdot d$ for any $d \in [0, 1]$. Both link metric and Euclidean metric are linearly bounded cost functions. Nondecreasing functions which are not linearly bounded are called *superlinear cost functions*. The energy metric is an example. An example was constructed in [54] showing that there are no geometric routing algorithms whose cost is bounded above by a function of c_* . To sum up:

Theorem 13 ([54]) *For any linearly bounded cost function, there is a distributed algorithm finding a route of cost at most $O(c_*^2)$, while for superlinear cost function, such an algorithm does not exist.*

3.3 Routing Based on Virtual Coordinates

In a virtual coordinate system, a set of landmarks serve as the references, the nodes are named not by their absolute positions but by their relations with the landmarks. The idea of using virtual coordinate in large networks first originated in [86], in which Tsuchiya proposed a hierarchical landmark-based scheme to generate names for the nodes. The work [72] made use of virtual coordinate to guide the greedy forwarding. The focus was on how to generate the virtual coordinate system in the absence of location information. Some nodes on the boundary are detected to serve as the landmarks. These nodes are localized by least square method based on their mutual distances. Any other node iteratively updates its virtual coordinate by putting itself at the center of its neighbors' current coordinates.

Section 3.3.1 first introduces the algorithm GLIDER proposed by Fang et al. [23] to convey the main idea of routing algorithms based on a virtual coordinate system. Then some related works will be introduced.

The choice of landmarks is crucial to the performance of the routing algorithms. Section 3.3.2 introduces the work in [65] which proposed a distributed randomized choice of landmarks and an associated routing algorithm GLDR such that the stretch factor of the routing is bounded for continuous domain.

3.3.1 GLIDER

The concern of GLIDER is how to efficiently route in a network with irregular shapes such as long narrow corridors or big holes. The routing is divided into two hierarchical levels: the *global routing* and the *local routing*. In the global routing, *combinatorial Delaunay graph* (CDG) on a set of nodes called *landmarks* is constructed to capture the global topology of the network, which is used to guide

the general orientation of the routing such that the influence of harsh topological regions can be alleviated. CDG divides the plane into *tiles*. The local routing in a tile or between two tiles follows the greedy forwarding strategy. By choosing the landmarks appropriately, each tile will have a good topology such that the hop distances approximate the Euclidean distances fairly accurate and greedy forwarding may work well.

Use $d(u, v)$ to denote the hop distance between nodes u and v in the communication graph G . The combinatorial Delaunay graph is the discrete extension of the classic (continuous) Delaunay triangulation.

Definition 2 For the communication graph $G = (V, E)$ and a set of landmarks $\mathcal{L} \subseteq V$, the *Voronoi cell* $V(u)$ at a node $u \in \mathcal{L}$ is defined to be

$$V(u) = \{v \in V \mid d(v, u) \leq d(v, w) \forall w \in \mathcal{L}\}.$$

Use $v_1 \sim v_2$ to stand for $v_1 = v_2$, or v_1 is adjacent with v_2 in G . The *combinatorial Delaunay graph* is the graph $D(\mathcal{L})$ with vertex set \mathcal{L} ; two nodes $u_1, u_2 \in \mathcal{L}$ are adjacent in $D(\mathcal{L})$ if and only if there exist nodes $v_1 \in V(u_1)$ and $v_2 \in V(u_2)$ such that $v_1 \sim v_2$. The nodes v_1, v_2 are said to be *witnesses* to the edge u_1u_2 .

By the above definition, it is easy to see that for each node $v \in V(u)$, the shortest (u, v) -path is completely contained in $V(u)$. As a consequence, the subgraph of G induced by the nodes in $V(u)$ is connected. Thus Voronoi cells partition the sensor field into connected *tiles*. The following theorem shows that $D(\mathcal{L})$ is indeed a compact high-level atlas of G , which can be used to guide the general direction of the routing.

Theorem 14 ([23]) *If G is connected, then $D(\mathcal{L})$ is connected.*

The local routing is guided by a virtual coordinate system which was called *local landmark coordinate system*. To better convey the idea, Fang et al. [23] first introduced a continuous version of the system. Let u_1, u_2, \dots, u_k be k landmarks. For a point x , let $B_i(x) = \|x - u_i\|^2$ be the squared Euclidean distance between x and u_i , $B(x) = (B_1(x), B_2(x), \dots, B_k(x))$, and $\bar{B}(x) = \frac{1}{k} \sum_{i=1}^k B_i(x)^2$. The *centered landmark-distance coordinate* $C(x) = (B_1(x) - \bar{B}(x), B_2(x) - \bar{B}(x), \dots, B_k(x) - \bar{B}(x))$. The *modified virtual distance function* is $md(x, y) = \|C(x) - C(y)\|_2^2$, where $\|\cdot\|_2$ is the 2-norm of the vector space \mathbb{R}^k . In *gradient descent*, a node forwards the packet to its neighbor which is closest to the destination, where *closeness* is measured by md . Fang et al. proved that gradient descent works well for the continuous field.

Theorem 15 ([23]) *In the continuous plane, the gradient descent is guaranteed to converge to its destination as long as there are at least three noncollinear landmarks.*

The discrete local landmark coordinate system is an analog of the above, replacing Euclidean distance by hop distance. For the nodes in a Voronoi cell $V(v)$,

the landmarks used in establishing the local coordinate system are v itself and the neighbors of v in $D(\mathcal{L})$.

For a packet to be sent from node v to node v' , suppose $v \in V(u)$ and $v' \in V(u')$. By [Theorem 14](#), there is an (u, u') -path $u_1 u_2 \dots u_k$ in $D(\mathcal{L})$. The packet is sent by gradient descent from tile to tile. In the first tile $V(u_1)$, the destination of the gradient descent is u_2 . As soon as the packet hits the boundary of $V(u_2)$, the destination is switched to u_3 . The process continues until the packet hits $V(u_k)$, and the destination is v' in the last tile.

In GLIDER, the atlas $D(\mathcal{L})$ is stored at every node, which is unrealistic in wireless sensor networks because of the limited storage capacity of the sensors. In [\[32, 33\]](#), Funke and Milosavljević modified GLIDER into a local algorithm. There are two main modifications. The first modification chooses the landmarks to be a *k-hop maximal independent set* for some fixed parameter k , that is, every pair of landmarks are at least $(k + 1)$ -hops away from each other and every other node is within k -hops of at least one landmark. The advantage of such a choice is twofold: first, it makes the tiles small (the diameter of each cell is at most $2k$), so flooding a tile is relatively cheap, and second, it results in a bounded maximum degree of $D(\mathcal{L})$, which further results in a low storage load on each node. The second modification is that instead of finding a path in $D(\mathcal{L})$, the packet is forwarded from one tile to another in a greedy way with respect to the locations of the landmarks. To be more precise, suppose the packet is currently in tile $V(u)$, the algorithm searches the neighbors of u in $D(\mathcal{L})$ to find a landmark v such that v is closer to the landmark of the destination tile than u . Then the packet is forwarded from $V(u)$ to $V(v)$. Such a strategy eliminates the need to store the whole atlas at every node. It should be noted that different from [Sect. 3.1](#), the greedy strategy here is used on $D(\mathcal{L})$ instead of on the communication graph G . To avoid stuck at local minimum, face routing strategy is used, again on $D(\mathcal{L})$ instead of on G . Notice that $D(\mathcal{L})$ is not planar in general. So, for the purpose of face routing, [\[32, 33\]](#) constructed a planar subgraph of $D(\mathcal{L})$ called *combinatorial Delaunay map* (CDM). For a landmark u and a node v , v is called an *u-vertex* if u has the smallest ID in v 's nearest landmarks. Two landmarks u_1, u_2 are adjacent in CDM if and only if there is an (u_1, u_2) -path in G whose 1-neighborhood consists only of u_1 -vertices and u_2 -vertices, and going from u_1 to u_2 along the path, all u_1 -vertices precede u_2 -vertices. Funke and Milosavljević [\[33\]](#) proved that CDM is indeed planar for quasi unit disk graphs.

Theorem 16 ([33]) *The combinatorial Delaunay map is planar for any d-quasi unit disk graph with $1/\sqrt{2} \leq d \leq 1$.*

The Beacon vector routing (BVR) algorithm proposed in [\[29\]](#) uses greedy forwarding strategy based on a distance function in a virtual coordinate system as follows: Suppose $\mathcal{L} = \{u_1, u_2, \dots, u_m\}$ is the set of landmarks which are chosen randomly. The coordinate of a node v is the vector $c(v) = (d(v, u_1), d(v, u_2), \dots, d(v, u_m))$, where the i -th component $c(v)_i = d(v, u_i)$ is the hop distance between v and u_i . Suppose D is the destination. For a parameter k , let $C_k(D)$ be the index set of k nearest landmarks to D . The distance function on node x is defined as

$$\delta_k(x, D) = M\delta_k^+(x, D) + \delta_k^-(x, D),$$

where M is a sufficiently large constant, and

$$\begin{aligned}\delta_k^+(x, D) &= \sum_{i \in C_k(D)} \max\{c(x)_i - c(D)_i, 0\}, \\ \delta_k^-(x, D) &= \sum_{i \in C_k(D)} \max\{c(D)_i - c(x)_i, 0\}.\end{aligned}$$

The packet with destination D which is currently at node v is forwarded to a node x which minimizes $\delta_k(x, D)$. Intuitively, among the k nearest landmarks of D , those which are nearer to D exerts a *pulling* force and the remaining ones exert a *pushing* force.

3.3.2 Choice of Landmarks

In [65], Nguyen et al. proposed a distributed randomized method to choose the landmarks and an associated routing algorithm called greedy landmark descent routing (GLDR). In the continuous domain, a constant stretch factor can be guaranteed. The performance in the discrete setting was shown by simulation.

The region was assumed to be *approximately regular* in the following sense: For a positive integer r and a node u , the r -hop neighborhood of u is the set of nodes at most r -hops away from u . For two real numbers $1 \leq a \leq b$, a node distribution is said to be (a, b) -regular if for any real number $r \in \mathbb{R}$, the number of nodes in the $\lfloor r \rfloor$ -hop neighborhood of any node is between ar^2 and br^2 .

The idea of the landmark selection is to guarantee both a good coverage and a good separation, which can be realized by selecting an r -hop maximal independent set (r -MIS). Since the sequential selection of r -MIS is time-consuming and the totally randomized parallel selection chooses too many redundant nodes, a method was proposed to balance the time and the size as follows: Each node u generates a random number p_u uniformly in $[0, 1]$. If it does not receive any claim of landmark at the end of Kp_u unit time, where a unit time is the time needed for a packet to be sent from one node to its neighbor, then it claims itself to be a landmark. Nguyen et al. [65] proved that the expected number of landmarks produced by the above method is upper bounded in a local area.

Theorem 17 ([65]) *For an (a, b) -regular distribution, the expected number of landmarks in any r -hop neighborhood is upper bounded by $\frac{9br^2}{K} + 16b - 1$.*

In particular, if K is taken to be $K = \Theta(r^2)$, then the upper bound is a constant.

The outline of GLDR is as follows: The virtual coordinate of each node is addressed by a subset (not all) of its nearest landmarks. These landmarks are called *addressing landmarks*. Suppose u is to send a packet to v . It first selects the landmark l from the addressing landmarks of v such that the ratio $d(u, l)/d(v, l)$ is maximized. Then the packet is routed through a shortest path towards l until it reaches a node z with $d(z, l) = d(v, l)$. If z is not v itself, the process repeats.

To deal with the case near boundary, an *Augmented GLDR* was proposed by *virtually expanding* the region and using the *mirror images*.

In the continuous domain (in which every point on the plane can serve as a relay node, the distance is the Euclidean distance, and the shortest path is just along a straight line), the addressing landmarks can be chosen such that the stretch factor is bounded by a constant. Suppose u, v are the source and the destination, respectively, and l is the current addressing landmark chosen as in the above paragraph. Let u' be the point on the line segment ul such that $\|lu'\| = \|lv\|$. The *one-step stretch factor* of routing from u to v using landmark l is defined as $s = \|uu'\| / (\|uv\| - \|u'v\|)$. Denote by R the minimum distance between the destination v and its addressing landmarks. Let $f(s)$ be a function on s defined as follows:

$$f(s) = \frac{4}{1 + p - \sqrt{q}}, \quad p = 1 - \frac{1}{s^2}, \quad z = 1 - \frac{1}{s},$$

$$q = (1 - p^2) + 2p^2(1 - z^2) + 2p\sqrt{(1 - z^2)(1 - p^2z^2)}.$$

Theorem 18 ([65]) *Suppose the domain is continuous. If $R > 2r$, then f^{-1} exists and the one-step stretch factor $s \leq f^{-1}(R/r)$. If R is large enough compared to r , then augmented GLDR successfully delivers the packet through a path with bounded stretch factor. In particular, if $R \geq 320r$ and $s < 2.5$, then the stretch factor of augmented GLDR is at most $7s/(5 - 2s)$.*

Based on the above theorem, the addressing landmarks can be chosen according to the requirement of the stretch factor, and combining with [Theorem 17](#), the expected number of the addressing landmarks for each node can be bounded by a constant.

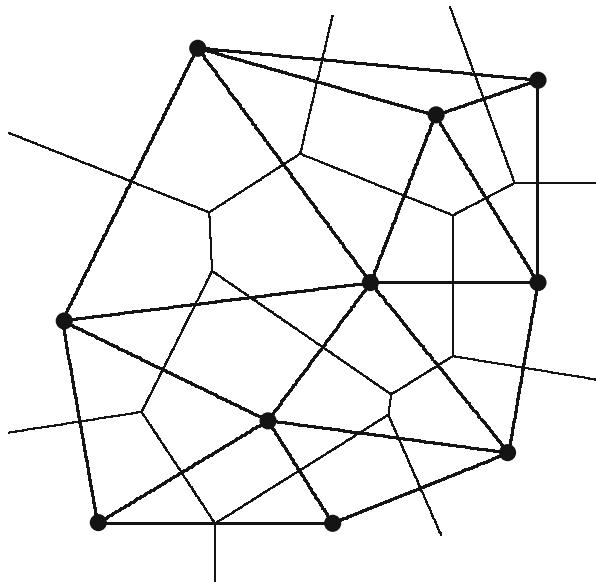
3.4 Greedy Embedding

The success of a greedy routing depends not only on the greedy criterion but also on the embedding. If the embedding is such that for any two distinct nodes u and v , node u has a neighbor which is nearer to v than u is, then greedy forwarding always succeeds in delivering packets, where *nearness* depends on the greedy criterion. Such an embedding is called a *greedy embedding* or a *greedy drawing*. Some literatures also use the terminologies *support the greedy routing* and *defeat the greedy routing* to indicate whether the greedy strategy always succeeds.

In the *Euclidean greedy routing*, the *nearness* is measured by the Euclidean distance to the destination. An embedding which supports Euclidean greedy routing is the Delaunay triangulation.

Delaunay triangulation is similar to Delaunay graph in [Sect. 2.4.3](#). The difference is that Euclidean distance is used here. Let V be a set of nodes on the plane. For a node $u \in V$, the *Voronoi cell* $V(u)$ is the convex region containing all the points on the plane which are closer to u than to any of the other nodes in V , that is,

Fig. 13 The *thick lines* form the Delaunay triangulation. The *thin lines* indicate the Voronoi diagram



$V(u) = \{p \in \mathbb{R}^2 \mid \|pu\| \leq \|pv\|, \forall v \in V, v \neq u\}$, where $\|pu\|$ is the Euclidean distance between p and u . The *Voronoi diagram* partitions the whole plane into Voronoi cells. *Delaunay triangulation* is the dual graph of the Voronoi diagram (see Fig. 13). It is well known that as long as no four nodes are cocircular, the Delaunay triangulation is well defined. It is also well known that the circumcircle of any Delaunay triangulation contains no node of V inside [71]. In fact, this property is often used as an equivalent definition of Delaunay triangulation, which can be generalized to higher-dimensional space.

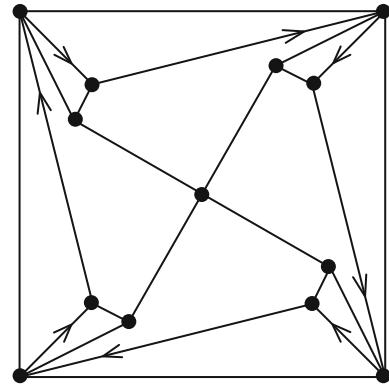
Theorem 19 ([10]) *Delaunay triangulation supports Euclidean greedy routing.*

The following result was first conjectured by Papadimitriou and Ratajczak [69]: In [19], Dhandapani discovered that any planar triangulation has an Euclidean greedy embedding. The proof is probabilistic. A constructive algorithm was given by Angelini et al. in [2]. The conjecture was completely solved by Leighton and Moitra [58].

Theorem 20 ([58, 69]) *Any planar 3-connected graph supports Euclidean greedy routing.*

In *compass routing*, the *nearness* is measured by the angle $\angle wuv$, where u is the current location of the packet, v is the destination, and w is a neighbor of u . Compass routing cannot guarantee the delivery either. Consider Fig. 14, there might be cases that the route goes infinitely around a loop. In fact, it was proved by Bose and Morin [10] that in any triangulation which defeats the compass routing, the packet must be trapped in a cycle. However, compass routing will never be defeated by

Fig. 14 An illustration of loop in compass routing. The arrows indicate the route



Delaunay triangulation [51]. Bose and Morin [10] generalized this result to regular triangulation. A *regular triangulation* is obtained by projecting the faces of the convex hull of a 3-dimensional polytope onto the plane. Delaunay triangulation is a special case of regular triangulation when the nodes of the polytope are all on a paraboloid.

Theorem 21 ([10]) *Any regular triangulation supports the compass routing.*

Suppose the current location of the packet is u and the destination is v , let w_1 (resp. w_2) be the neighbor of u which minimizes the clockwise (resp. counterclockwise) angle $\angle vuw_1$ (resp. $\angle vuw_2$). In the *randomized compass routing*, the packet is forwarded to one of w_1 and w_2 with the same probability. A triangulation is said to defeat a randomized routing if there exists a pair of source-destination nodes u, v such that the probability of successful delivery is zero in finite number of steps.

Theorem 22 ([10]) *No triangulation defeats the randomized compass routing.*

In [50], Kleinberg proved that any connected graph has an embedding in the *hyperbolic space* such that greedy strategy always succeeds with respect to the hyperbolic distance. However, for the applications in wireless networks, embeddings in the 2- or 3-dimensional Euclidean space are more interesting.

Despite of the elegance of the above results, the storage load is heavy [21]. *Succinct greedy embedding* with low load on the nodes was studied only very recently. Some related works may be found in [3, 40, 42].

For the study of greedy embeddings in a region with holes, discrete *Ricci flows* are used in [73]. The idea is to establish a *conformal map* from the region \mathcal{R} to a domain \mathcal{D} such that every hole of \mathcal{R} is mapped to a circle in \mathcal{D} (notice that greedy strategy never gets stuck at a circle).

In [28], Flury et al. presented an algorithm to find a greedy embedding of a graph in $\mathbb{R}^{O(\log^3 n)}$ with stretch factor $O(\log n)$; each coordinate of a node uses $O(\log n)$ bits. For *combinatorial unit disk graph* (unit disk graph without geometric information), the embedding can be made into $\mathbb{R}^{O(\log^2 n)}$ with a constant stretch factor.

4 Boundary Recognition

Boundary recognition (both outer boundary and inner boundary) is important for keeping track of nodes entering or leaving a region and communication between the inside and the outside. It is also important for efficient algorithms in wireless networks. As one has seen, a greedy forwarding strategy encounters a failure particularly around a *hole*, a void region in which there are no sufficient sensors. Holes are a common phenomenon in the real-world wireless sensor network. Even when the sensors are distributed uniformly at random, there might be holes due to irregular environment (such as dense vegetation or high buildings blocking direct communications) or depletion of sensor powers. Detecting holes helps with efficient routing. Furthermore, detecting holes can identify regions of interests to the users. For example, if a sensor marks itself to be unavailable when the local temperature is above a preestablished critical value, then detecting holes helps to forecast a burst of fire. If a sensor marks itself to be unavailable if the power is below a threshold, then detecting holes helps to identify regions in which new sensors should be deployed to improve the connectivity. Neglecting the existence of holes may arouse serious problems in the network. In fact, since nodes on the boundaries are usually loaded with more traffic, their energy depletes more rapidly, and thus a small hole tends to grow larger and larger.

In boundary recognition, one has to first make clear *what is a boundary*? Some authors defined boundaries by using *face boundaries* [24] and *chordless cycles* [52]. In [25, 26], boundaries are implicitly defined by identifying those nodes which are more likely to near the boundaries. One problem about the definitions is that, provided with connectivity information *only*, holes might depend on the embedding of the graph. The paper [75] contains a detailed discussion on these definitions and their limitations. To overcome this problem, Saukh et al. [75] proposed the definition of *generalized boundary* which contains nodes on a boundary of at least one straight-line embedding.

To recognize the boundaries, various methods were proposed, including geometric methods [24], statistical methods [25, 26], graph methods [30, 31, 52, 75], and topological methods [88]. They are introduced in the following.

4.1 Geometric Method

The first paper on boundary detection is [24], in which Fang et al. defined holes in a way that the greedy forwarding can get stuck only at a node on the boundary of a hole. This subsection introduces their method and its application in geometric routing.

The strict mathematical definition of a hole given in [24] is closely related with *stuck nodes*, the nodes at which greedy forwarding strategy fails. There are two types of stuck nodes: weakly stuck nodes and strongly stuck nodes.

Definition 3 ([24]) A node u is a *weakly stuck node* (resp. *strongly stuck node*) if there exists a node (resp. location) p outside of u 's transmission range such that no neighbor of u is closer to p than u itself.

Notice that the only difference in the above definitions is that p is a *node* for a weakly stuck node and is a *location* for a strongly stuck node. A weakly stuck node is always a strongly stuck node. Fang et al. defined two kinds of holes in terms of weakly stuck nodes and strongly stuck nodes, respectively.

As one has seen in Sect. 3.4, Delaunay triangulation supports compass routing. But the problem is that some Delaunay edges might be longer than 1, which is not available in the unit disk graph. To overcome this problem, [24] used the *restricted Delaunay graph* (RDG) as in Sect. 3.2.1. A node which is adjacent with a Delaunay edge longer than 1 might be a potential stuck node. The first kind of hole, called *weak-hole*, is defined to be a face of RDG with degree at least 4 (the degree of a face is the number of edges on the boundary of the face).

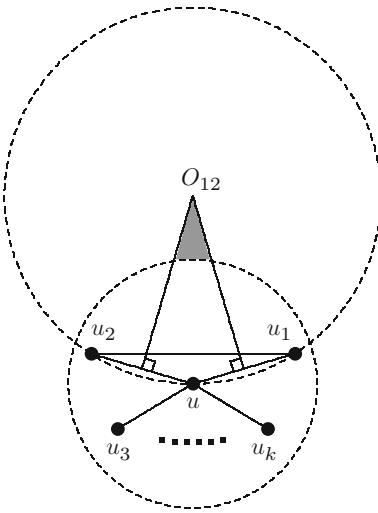
Theorem 23 ([24]) All the weakly stuck nodes lie on the boundaries of the weak-holes.

By Theorem 23, one can detect all the weakly stuck nodes and their corresponding weak-holes by computing the Delaunay triangulation and then removing all the edges of length greater than one. However, the computation is heavy [62, 71]. So Fang et al. continued to give another definition of hole using strongly stuck nodes and pseudo Jordan curve (recall that Jordan curve is a non-self-intersecting continuous loop in the plane).

Definition 4 ([24]) A directed polygonal curve C is a *pseudo Jordan curve* if by duplicating nodes into copies with infinitesimal perturbation, each copy corresponding to one occurrence of the node on C , the revised polygonal curve is a Jordan curve. A *strong-hole* is a region bounded by a pseudo Jordan curve, which is called the *boundary* of the strong-hole, such that all the strongly stuck nodes are on the boundaries of all the strong-holes.

In [24], the authors proposed a *Tent Rule* to detect strongly stuck nodes. Suppose a node u has k neighbors u_1, u_2, \dots, u_k in the unit disk graph, ordered counterclockwise. Let us first consider the question of when there can exist a location p as in Definition 3 in $\text{Cone}(u_1uu_2)$, the cone spanned by radials uu_1 and uu_2 . See Fig. 15 for an illustration. For $i = 1, 2$, draw a perpendicular bisector l_i of the line segment uu_i . Suppose l_1, l_2 intersect at point $O_{1,2}$. Then $O_{1,2}$ is the center of the circumcircle of triangle Δu_1uu_2 . A location p in $\text{Cone}(u_1uu_2)$ which is outside of the transmission range of u and is closer to u than to u_1 and u_2 can only lie in the shaded area of Fig. 15. So, if the distance $\|uO_{1,2}\| > 1$, then u is a potential strongly stuck node. In such a case, $\angle u_1uu_2$ is called a *stuck angle*. Notice that all the information needed to find a stuck angle at a node is its 1-hop neighborhood, which is completely local.

Fig. 15 An illustration of Tent Rule and stuck angle



Theorem 24 ([24]) *If u is a strongly stuck node, then there exists a stuck angle at u .*

Theorem 24 provides a necessary condition to guarantee that all the strongly stuck nodes can be found using Tent Rule.

Having found a strongly stuck node u , the strong-hole containing u can be found by the algorithm BoundHole proposed in [24], which can be implemented locally by using only angles between adjacent edges.

With the holes detected by BoundHole, geometric routing can be done as follows: When the greedy forwarding fails at a stuck node u , u must be on the boundary of a hole (by Theorem 23 or Definition 4). After that, recovery mode is activated and the packet is routed along the boundary of the hole, say, by the right-hand rule. When the packet gets to a node v which is closer to the destination D than u , the greedy forwarding mode is resumed. When D is outside the hole, such a node v always exists. When D is inside the hole, the packet might go around the boundary of the hole back to u without coming across any node closer to D than u . As long as D is reachable from u , there must be a path from u to D which intersects the boundary of the hole, say, edge $x'y'$ on the path intersects edge xy on the boundary of the hole. One of x', y' , say x' , must be a 1-hop neighbor of x or y in the unit disk graph. In this case, a *restricted flooding mode* is initiated, in which every node on the boundary of the hole sends the packet to all its 1-hop neighbors. In particular, x' can receive the packet and forward it to D .

4.2 Statistical Method

Assuming a uniform random distribution of sensors, Fekete et al. proposed statistical methods to recognize nodes near the boundaries [25, 26].

The method used in [25] is based on the intuition that nodes on the boundaries have a much lower average degree than those in the interior. Suppose A is the whole region containing n nodes. Use $|A|$ to denote the Lebesgue measure of A . Under the assumption of uniform distribution, a node u in the *interior* has an estimated neighborhood size of

$$\mu = (n - 1) \frac{|B_R(u)|}{|A|},$$

where R is the transmission range of the nodes and $B_R(u)$ is the disk with center u and radius R . Given a constant $\alpha < 1$, nodes with degree less than the threshold $\alpha\mu$ claim themselves to be boundary nodes. Notice that μ is not known a priori since the region A is exactly what needs to be determined. The performance of the method depends on the estimation of μ and the choice of α . In [25], μ is estimated based on a node degree histogram, and α is computed by sampling possible values of α and keeping track of the number of connected boundary components.

The method proposed in [26] utilizes centrality measures which are commonly used in social networks. The idea is that the centrality of nodes on the boundary is much lower than those in the interior. There are two major classes of centralities: local and global. In fact, degree used in [25] is one type of measures of local centrality. Eigenvalues of a matrix is an example of global centrality. Three centrality measures were used in [26]. The idea is based on the intuition that more shortest paths will pass through interior nodes than boundary nodes. Use $\sigma_{uv}(w)$ to denote the number of shortest (u, v) -paths containing node w . The *stress centrality* at node w is defined as

$$st(w) = \sum_{u, v \in V} \sigma_{uv}(w),$$

where V is the set of nodes in the region. Let σ_{uv} be the number of shortest (u, v) -paths. Assume that a packet with source u and destination v chooses any shortest (u, v) -path with the same probability. Then the probability that node w is used to transmit the packet is $\sigma_{uv}(w)/\sigma_{uv}$. The *betweenness centrality* at node w is defined as

$$betw(w) = \sum_{u, v \in V} \sigma_{uv}(v)/\sigma_{uv}.$$

To make the estimation localized, one restricts the attention to those nodes which are nearby. The *restricted stress centrality* at node w is defined as

$$st_r(w) = \sum_{u, v \in N_\delta(w)} \sigma_{uv}(w),$$

where $N_\delta(w)$ is the set of nodes at most δ -hops away from w . Boundary nodes are distinguished by a properly set threshold. For example, for $\delta = 1$, any node with $st_r(w) \leq \theta \binom{\mu}{2}$ claims itself to be a boundary node, where μ is the estimated average degree and θ is a parameter playing the same threshold role as α in [25]. It was

proved in [26] that if the density is sufficiently large, nodes are classified correctly with a high probability using the restricted stress centrality with $\delta = 1$.

The major limitation of the above two methods is the assumption on the sensor distribution and the high density of the sensors.

4.3 Graph Method

This subsection introduces two kinds of graph-based methods in identifying the boundaries.

4.3.1 Contour-Based Method

The methods proposed by Funke and Klein in [30, 31] are based on the observation that contours are broken at boundaries.

Unlike the continuous field, contour has no definition in wireless sensor networks since the distances are counted by hops in the communication graph (which is a unit disk graph G in [30, 31]). Funke proposed isoset which resembles contour. For a node u (called *beacon* in [30] or *seed* in [31]) and a fixed integer k , the set $\mathcal{I}(k) = \{v \mid d_{\text{hop}}(u, v) = k\}$ is called an *isoset of level k* , where $d_{\text{hop}}(u, v)$ is the hop distance between u and v in G . Suppose the subgraph of G induced by $\mathcal{I}(k)$ has t -connected components $C_1(k), C_2(k), \dots, C_t(k)$. If $t \geq 2$, the isoset $\mathcal{I}(k)$ is *broken* and holes might exist between the connected components. The *ends* of each $C_i(k)$ indicate the locations of the holes, and the ends can be identified as the nodes in $C_i(k)$ having the largest *radius* in $C_i(k)$ (the radius of a node w in a connected graph H is the hop distance between w and the node in H which is farthest from w). The ends are *marked* as near boundaries.

The above method is not local. To overcome it, more seeds are chosen, and for each seed s , only the isosets in a *bounded* neighborhood around s are examined, namely, the isosets $\mathcal{I}(k)$ with $h_{\min} \leq k \leq h_{\max}$. By choosing the seed set to be a maximal independent set, the running time can be bounded. In fact, it was proved in [31] that with h_{\max} being a constant, the running time is $O(n + m)$, where n and m are the number of vertices and the number of edges of the unit disk graph, respectively. Funke and Klein [31] also provided a theoretical guarantee under the assumption that the holes are all circles which are not too small, the distances between different holes are large enough, and the density of sensors in the non-hole area is high enough. To be more precise, they defined an ε -good sensor distribution and proved the following theorem. A set \mathcal{S} of sensors is ε -good if for every point $p \in \mathbb{R}^2$ which is not in a hole, there is a sensor $s \in \mathcal{S}$ with $\|sp\| \leq \varepsilon$.

Theorem 25 ([31]) *For circular holes with radius at least $r_{\text{hole}} = 72$, if the minimum distance between holes is at least $\Delta_{\text{hole}} = 18$, the sensor distribution is ε -good for $\varepsilon = 1/64$, and by taking $h_{\min} = 4$ and $h_{\max} = 8$, Funke and Klein's algorithm marks for each boundary point a sensor node within distance at most 4.8, and every marked sensor node has a boundary point within distance 2.8.*

4.3.2 Identifying Inner Nodes

The methods used in [52] and [75] aim to recognize *inner nodes*. All the other nodes are considered to be on the boundaries.

The graph model considered in these two papers is quasi unit disk graph (QUDG) (see Sect. 3.3 for the definition). It was assumed that $d \geq 1/\sqrt{2}$ in [52] and [75]. This value is crucial to the geometrical analyses, as can be seen from the following lemma:

Lemma 2 ([52]) *Let $G = (V, E)$ be a d -QUDG with $d \geq 1/\sqrt{2}$, u, v, x, y be four distinct nodes in V with $uv, xy \in E$. If the two straight-line segments $p(u)p(v)$ and $p(x)p(y)$ intersect, then at least one of the four edges ux, uy, vx, vy is in E .*

A cycle $C = u_0u_1\dots u_{k-1}u_0$ is *chordless* if $u_iu_j \notin E$ for any i, j with $|i - j| > 1$, where the operation “ $-$ ” is modulo k . The embedding $p(C)$ of a chordless cycle C divides the plane into an infinite face and at least one finite face (there might be more than one finite faces if C is self-intersecting). For a node set U , $N(U)$ is the set of nodes which are adjacent with at least one node in U , and $N[U] = N(U) \cup U$ is the closed neighborhood of U . The following result is a corollary of Lemma 2:

Corollary 3 ([52]) *Let $G = (V, E)$ be a d -QUDG with $d \geq 1/\sqrt{2}$, C be a chordless cycle in G , and U be a connected node set of G . If $N[C] \cap U = \emptyset$, then either U is completely contained in the inside of C or completely contained in the outside of C .*

Denote by $fit_d(k)$ the size of a maximum independent set I which can be placed inside a chordless k -cycle in a d -QUDE such that $I \cap N[C] = \emptyset$. Based on the above property, the following result is easy to see, which provides a criterion for a connected set of nodes to lie outside of a chordless cycle.

Lemma 3 ([52]) *Under the assumption of Corollary 3, if U has an independent node subset I with $|I| > fit_d(|C|)$, then U is completely outside of C .*

To decide whether there is a node inside of a chordless cycle is much more complicated. For this purpose, [52] proposed a structure called *m-flower* and proved that for a flower, there is a node inside of its related chordless cycle.

The algorithm in [52] looks for a family of node sets $\mathcal{U} = \{U_i\}$ and a family of chordless cycles $\mathcal{C} = \{C_b\}$ such that the nodes in $\bigcup_i U_i$ are guaranteed to lie inside of the cycles. Such a pair $(\mathcal{C}, \mathcal{U})$ is called a *feasible geometry description* (FGD). The goal is to find an FGD with as many inside nodes as possible (recall that the remaining nodes are regarded as boundary nodes; hence, the boundary might be clearer with more inner nodes identified). The algorithm starts from some FGD and then iteratively extends it to another FGD with more inside nodes. The initial FGD is found by detecting a flower. As explained in the previous paragraph, this provides the initial \mathcal{I} and \mathcal{C} . Then, the FGD is enlarged by the so-called augmenting cycles.

The success of the above method depends on the existence of a flower. Graph G is called *locally ε -dense* in a region $A \subseteq \mathbb{R}^2$, if every ε -disk contained in A contains a node of G , that is, for any point $x \in A$ with $B_\varepsilon(x) \subseteq A$, there is a node $u \in V(G)$ with $\|ux\| \leq \varepsilon$.

Lemma 4 ([52]) *For $d = 1$ and $0 < \varepsilon < \frac{3}{2} - \sqrt{2}$, if there exists a point $x \in \mathbb{R}^2$ such that G is locally ε -dense in the disk $B_3(x)$, then G contains a 4-flower.*

Notice that the above lemma insures the existence of a flower only when the network is dense enough, which cannot be guaranteed in a real-world network. To overcome this problem, Saukh et al. [75] proposed a generic classes of *patterns* to ensure some node inside of the pattern, which makes the algorithm in [75] works much better in a sparse network.

4.4 Topological Method

Homology, being a rigorous mathematical method for detecting and categorizing holes, steps naturally into the way of hole recognition in wireless sensor networks. The task is how to adapt it to suit the discrete settings and the special requirements of wireless networks.

The method used in [88] is based on the observation that holes incur a *void* in a shortest path tree. The rough idea is as follows: The *flow* of a shortest path tree T forks near a hole and meets again after going along opposite borderlines of the hole. The nodes where the tree flow meets again were called *cuts*. By tracking their *least common ancestor* (LCA) on the tree (which is far away because of the hole) will trap the hole. To be more concise, Wang et al. [88] defined a pair of *neighboring* nodes u, v to be a *cut pair*, if:

- (a) The hop distances $d_{hop}(u, w)$ and $d_{hop}(v, w)$ are above a threshold δ_1 , where $w = LCA(u, v)$ is the least common ancestor of u, v on T .
- (b) $\max\{d_{hop}(x_u, x_v)\}$ is above a threshold δ_2 , where the maximum is taken over all node pairs (x_u, x_v) with x_u being a node on the (u, w) -path on T and x_v being a node on the (v, w) -path on T .

The first condition indicates that the LCA is far away from the cut pair. The second condition indicates how *fat* is the hole that one wants to detect. Topologically, cut is a counterpart of *cut locus* [15] in the continuous setting (roughly speaking, the cut locus of a point x in a manifold is the set of points for which there are multiple minimizing geodesics connecting them from p). It should be noted that the communication graph is not required to be a quasi unit disk graph in [88].

The problem considered in [16, 17, 37] is different from the above boundary recognition problem. The question asked by Ghrist et al. is: assuming disk sensing range of the sensors, can the domain be covered without holes? The problem is abstracted into a Čech complex, and the detection of sensing holes is equivalent

to determining whether the domain boundary circle is null-homologous in the Čech complex. Since this computation requires accurate locations of the nodes and a global coordinate system, which is not suitable for the setting of wireless network, Ghrist et al. continued to find a sufficient but not necessary condition to detect sensing holes using Vietoris-Rips complex, based only on the pairwise communication data.

5 Conclusion

This chapter addresses three classes of geometric optimization problems in wireless sensor networks: localization, routing, and boundary recognition. The survey outlines the main ideas of recent methods for handling the new challenges brought into these problems by large-scale sensor wireless transmission. A number of successful approaches are covered to overcome difficulties in the geometric optimization due to the absence of central control unit, limited transmission range, limited computation and storage capability, etc.

Acknowledgements This work is supported by National Natural Science Foundation of China (61222201) and Specialized Research Fund for the Doctoral Program of Higher Education (20126501110001). It is also supported by National Science Foundation of the USA under grants 1020 CNS0831579 and CCF0728851.

Cross-References

► [Efficient Algorithms for Geometric Shortest Path Query Problems](#).

Recommended Reading

1. K. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, O. Fieder, Geometric spanners for wireless ad hoc networks, *IEEE Trans. Parallel Distrib. Syst.* **14**(5), 408–421 (2003)
2. P. Angelini, F. Frati, L. Grilli, An algorithm to construct greedy drawings of triangulations, in *Proceedings of the 16th International Symposium on Graph Drawing*, Heraklion, Crete, Greece, 2008, pp. 26–37
3. P. Angelini, G.D. Battista, F. Frati, Succinct greedy drawings do not always exist, in *Proceedings of the 17th International Symposium on Graph Drawing*, Chicago, IL, USA, 2009. Lecture Notes in Computer Science, vol. 5849, 2010, pp. 171–182
4. J. Aspnes, D. Goldenberg, Y.R. Yang, On the computational complexity of sensor network localization, in *The First International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS)*, Turku, Finland, 2004, pp. 32–44
5. A.R. Berg, T. Jordán, A proof of connelly’s conjecture on 3-connected circuits of the rigidity matroid. *J. Comb. Theory B* **88**(1), 77–97 (2003)
6. M. Betke, L. Gurvits, Mobile robot localization using landmarks, in *IEEE International Conference on Robotics and Automation*, San Diego, CA, USA, May 1994, vol. 2, pp. 135–142

7. P. Biswas, Y. Ye, Semidefinite programming for ad hoc wireless sensor network localization, in *Proceedings of the Third International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, USA, 2004, pp. 46–54
8. L. Blazević, L. Buttyán, S. Capkun, S. Giordano, H. Hubaux, J.L. Boudec, Self-organization in mobile ad hoc networks: the approach of terminodes. *IEEE Commun. Mag.* **39**, 166–175 (2001)
9. I. Borg, P. Groenen, *Modern Multidimensional Scaling, Theory and Applications* (Springer, New York, 1997)
10. P. Bose, P. Morin, Online routing in triangulations, in *Proceedings of the 10th International Symposium on Algorithms and Computation, ISAAC'99*, London, UK, 1999, pp. 113–122. *SIAM J. Comput.* **33**(4), 937–951 (2004)
11. P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks. *Wirel. Netw.* **7**(6), 609–616 (2001)
12. P. Bose, L. Devroye, W. Evans, D. Kirkpatrick, On the spanning ratio of Gabriel graphs and β -skeletons, in *LATIN 2002: Theoretical Informatics*. Lecture Notes in Computer Science, vol. 2286/2002 (Springer, Berlin/New York, 2002), pp. 77–97
13. N. Bulusu, J. Heidemann, D. Estrin, GPS-less low-cost outdoor localization for very small devices. *IEEE Pers. Commun.* **7**(5), 28–34 (2000)
14. R. Connelly, On generic global rigidity, in *Applied Geometry and Discrete Mathematics*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 4 (American Mathematical Society, Providence, 1991), pp. 147–155
15. M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry: Algorithms and Applications* (Springer, Berlin, 1997)
16. V. de Silva, R. Ghrist, Coverage in sensor networks via persistent homology. *Algebr. Geom. Topol.* **7**, 339–358 (2007)
17. V. de Silva, R. Ghrist, Homological sensor networks. *Not. Am. Math. Soc.* **54**(1), 10–17 (2007)
18. D.P. Dobkin, S.J. Friedman, K.J. Supowit, Delaunay graphs are almost as good as complete graphs. *Discret. Comput. Geom.* **5**(4), 399–407 (1990)
19. R. Dhandapani, Greedy drawings of triangulations, in *SODA'08: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, USA, 2008, pp. 102–111
20. D. Eppstein, Spanning trees and spanners, in *Handbook of Computational Geometry*, ed. by J.-R. Sack, J. Urrutia (North-Holland, Amsterdam, 2000), pp. 425–461
21. D. Eppstein, M.T. Goodrich, Succinct greedy graph drawing in the hyperbolic plane, in *Proceedings of the 16th International Symposium on Graph Drawing*, Heraklion, Crete, Greece, 2008, pp. 14–25
22. T. Eren, D. Goldenberg, W. Whitley, Y. Yang, S. Morse, B. Anderson, P. Belhumeur, Rigidity, computation, and randomization of network localization, in *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04)*, Hong Kong, China, March 2004, vol. 4, pp. 2673–2684
23. Q. Fang, J. Gao, L. Guibas, V. de Silva, L. Zhang, GLIDER: gradient landmark-based distributed routing for sensor networks, in *Proceedings of the 24th Conference of the IEEE Communication Society (INFOCOM)*, Miami, FL, USA, March 2005, vol. 1, pp. 339–350
24. Q. Fang, J. Gao, L. Guibas, Locating and bypassing routing holes in sensor networks. *Mob. Netw. Appl.* **11**, 187–200 (2006)
25. S.P. Fekete, A. Kröller, D. Pfisterer, S. Fischer, C. Buschmann, Neighborhood-based topology recognition in sensor networks, in *ALGOSENSORS*, Turku, Finland. Lecture Notes in Computer Science, vol. 3121 (Springer, 2004), pp. 123–136
26. S.P. Fekete, M. Kaufmann, A. Kröller, N. Lehmann, A new approach for boundary recognition in geometric sensor networks, in *Proceedings 17th Canadian Conference on Computational Geometry*, University of Windsor, Ontario, Canada, 2005, pp. 82–85
27. G.G. Finn, Routing and addressing problems in large metropolitan-scale internetworks. Technical report ISU/RR-87–180, ISI, March 1987

28. R. Flury, S. Pemmaraju, R. Wattenhofer, Greedy routing with bounded stretch, in *Proceedings of the 28th Annual IEEE Conference on Computer Communications (INFOCOM)*, Rio de Janeiro, Brazil, April 2009
29. R. Fonseca, S. Ratnasamy, J. Zhao, C.T. Ee, D. Culler, S. Shenker, I. Stoica, Beacon vector routing: scalable point-to-point routing in wireless sensor networks, in *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, USA, May 2005, pp. 329–342
30. S. Funke, Topological hole detection in wireless sensor networks and its applications, in *DIALM-POMC'05: Proceedings of the 2005 Joint Workshop on Foundations of Mobile Computing*, Cologne, Germany, 2005, pp. 44–53
31. S. Funke, C. Klein, Hole detection or: “how much geometry hides in connectivity?” in *SCG'06: Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, Sedona, AZ, USA, 2006, pp. 377–385
32. S. Funke, N. Milosavljević, Guaranteed-delivery geographic routing under uncertain node locations, in *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, Anchorage, AK, USA, May 2007, pp. 1244–1252
33. S. Funke, N. Milosavljević, Network sketching or: “how much geometry hides in connectivity? – part II”, in *SODA'07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, USA, 2007, pp. 958–967
34. J. Gao, L. Zhang, Load balanced short path routing in wireless networks. *IEEE Trans. Parallel Distrib. Syst.* Special Issue on Localized Communications **17**(4), 377–388 (2006)
35. J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, A. Zhu, Discrete mobile centers, in *Proceedings of 17-th Annual Symposium on Computational Geometry (SCG)*, Medford, MA, USA (ACM, 2001), pp. 188–196
36. J. Gao, L.J. Guibas, J. Hershberger, L. Zhang, A. Zhu, Geometric spanners for routing in mobile networks. *IEEE J. Sel. Areas Commun.* Special issue on Wireless Ad Hoc Networks **23**(1), 174–185 (2005)
37. R. Ghrist, A. Muhammad, Coverage and hole-detection in sensor networks via homology, in *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks (IPSN'05)*, Los Angeles, CA, USA, 2005, pp. 254–260
38. D. Goldenberg, A. Krishnamurthy, W. Maness, Y.R. Yang, A. Young, A.S. Morse, A. Savvides, B. Anderson, Network localization in partially localizable networks, in *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'05)*, Miami, FL, USA, March 2005, vol. 1, pp. 313–326
39. D.K. Goldenberg, P. Bihler, Y.R. Yang, M. Cao, J. Fang, A.S. Morse, B.D.O. Anderson, Localization in sparse networks using sweeps, in *MobiCom'06: Proceedings of the 12th Annual International Conference on Mobile Computing and Networking* (ACM, New York, 2006), pp. 110–121
40. M.T. Goodrich, D. Strash, Succinct greedy geometric routing in the Euclidean plane, in *ISAAC 2009*, Honolulu, HI, USA, 2009. Lecture Notes in Computer Science, vol. 5878, pp. 781–791
41. J.E. Graver, B. Servatius, H. Servatius, *Combinatorial Rigidity*. Graduate Studies in Mathematics (American Mathematical Society, Providence, 1993)
42. X. He, H. Zhang, On succinct convex greedy drawing of 3-connected plane graphs, in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, USA, Jan 2011, pp. 1477–1486
43. B. Hendrickson, Conditions for unique graph realizations. *SIAM J. Comput.* **21**(1), 65–84 (1992)
44. T. Hou, V. Li, Transmission range control in multihop packet radio networks. *IEEE Trans. Commun.* **34**(1), 38–44 (1986)
45. B. Jackson, T. Jordán, Connected rigidity matroids and unique realizations of graphs. *J. Comb. Theory Ser. B* **94**(1), 1–29 (2005)
46. D.J. Jacobs, B. Hendrickson, An algorithm for two-dimensional rigidity percolation: the pebble game. *J. Comput. Phys.* **137**(2), 346–365 (1997)
47. R. Jain, A. Puri, R. Sengupta, Geographical routing using partial information for wireless ad hoc networks. *IEEE Pers. Commun.* **8**(1), 48–57 (2001)

48. B. Karp, H. Kung, GPSR: greedy perimeter stateless routing for wireless networks, in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Boston, MA, USA, 2000, pp. 243–254
49. J.M. Keil, C.A. Gutwin, The Delaunay triangulation closely approximates the complete Euclidean graph, in *Proceedings of the International Workshop Algorithms Data Structures*, Ottawa, Canada, 1989. Lecture Notes in Computer Science Series, vol. 382, pp. 47–56
50. R. Kleinberg, Geographic routing using hyperbolic space, in *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, Anchorage, AK, USA, 2007, pp. 1902–1909
51. E. Kranakis, H. Singh, J. Urrutia, Compass routing on geometric networks, in *Proceedings of the 11th Canadian Conference on Computational Geometry*, Vancouver, BC, Canada, 1999, pp. 51–54
52. A. Kröller, S.P. Fekete, D. Pfisterer, S. Fischer, Deterministic boundary recognition and topology extraction for large sensor networks, in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, Miami, FL, USA, 2006, pp. 1000–1009
53. F. Kuhn, R. Wattenhofer, A. Zollinger, Asymptotically optimal geometric mobile ad-hoc routing, in *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, Atlanta, GA, USA, 2002, pp. 24–33
54. F. Kuhn, R. Wattenhofer, Y. Zhang, A. Zollinger, Geometric ad-hoc routing: of theory and practice, in *Proceedings of the 22nd ACM International Symposium on the Principles of Distributed Computing (PODC)*, Boston, MA, USA, 2003, pp. 63–72
55. F. Kuhn, R. Wattenhofer, A. Zollinger, Worst-case optimal and average-case efficient geometric ad-hoc routing, in *Proceedings of the 4-th ACM International Symposium on Mobile Ad-Hoc Networking and Computing (MobiHoc)*, Annapolis, MD, USA, 2003
56. G. Laman, On graphs and rigidity of plane skeletal structures. *J. Eng. Math.* **4**(4), 331–340 (1970)
57. S. Lederer, Y. Wang, J. Gao, Connectivity-based localization of large scale sensor networks with complex shape, in *Proceedings of the 27th Annual IEEE Conference on Computer Communications (INFOCOM'08)*, Phoenix, AZ, USA, May 2008, pp. 789–797. ACM Trans. Sens. Netw. **5**(4), 31:1–31:32 (2009)
58. T. Leighton, A. Moitra, Some results on greedy embeddings in metric spaces, in *Proceedings of the 49th IEEE Annual Symposium on Foundations of Computer Science*, Las Vegas, NV, USA, Oct 2008, pp. 337–346. *Discret. Comput. Geom.* **44**, 686–705 (2010)
59. M. Li, Y. Liu, Rendered path: range-free localization in anisotropic sensor networks with holes, in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking, MobiCom07* (ACM, New York, 2007), pp. 51–62
60. X.-Y. Li, G. Calinescu, P.-J. Wan, Distributed construction of a planar spanner and routing for ad hoc wireless networks, in *INFOCOM'02*, New York, USA, 2002, pp. 1268–1277
61. Z. Li, W. Trappe, Y. Zhang, B. Nath, Robust statistical methods for securing wireless localization in sensor networks, in *IPSN'05: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks* (IEEE, Piscataway, 2005), pp. 91–98
62. J. Liebeherr, M. Nahas, W. Si, Application-layer multicasting with delaunay triangulation overlays. *IEEE J. Sel. Areas Commun.* **20**(8), 1472–1488 (2002)
63. A. Mei, J. Stefa, Routing in outer space: fair traffic load in multi-hop wireless networks, in *MobiHoc'08: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing* (ACM, New York, 2008), pp. 23–32
64. D. Moore, J. Leonard, D. Rus, S. Teller, Robust distributed network localization with noisy range measurements, in *Sensys'04: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems* (ACM, New York, 2004), pp. 50–61
65. A. Nguyen, N. Milosavljevic, Q. Fang, J. Gao, L.J. Guibas, Landmark selection and greedy landmark-descent routing for sensor networks, in *Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM'07)*, Anchorage, AK, USA, May 2007, pp. 661–669
66. D. Niculescu, B. Nath, Ad hoc positioning system (APS), in *IEEE GLOBECOM*, San Antonio, TX, USA, 2001, pp. 2926–2931

67. D. Niculescu, B. Nath, Ad hoc positioning system (APS) using AOA, in *IEEE INFOCOM*, San Francisco, CA, USA, March 2003, vol. 22, pp. 1734–1743
68. D. Niculescu, B. Nath, Error characteristics of ad hoc positioning systems (APS), in *MobiHoc'04: Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Tokyo, Japan, 2004, pp. 20–30
69. C.H. Papadimitriou, D. Ratajczak, On a conjecture related to geometric routing. *Theor. Comput. Sci.* **344**(1), 3–14 (2005)
70. L. Popa, A. Rostamizadeh, R. Karp, C. Papadimitriou, I. Stoica, Balancing traffic load in wireless networks with curveball routing, in *MobiHoc'07: Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing* (ACM, New York, 2007), pp. 170–179
71. F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction* (Springer, New York, 1985)
72. A. Rao, C. Papadimitriou, S. Shenker, I. Stoica, Geographic routing without location information, in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, San Diego, CA, USA, 2003, pp. 96–08
73. R. Sarkar, X. Yin, J. Gao, F. Luo, X.D. Gu, Greedy routing with guaranteed delivery using ricci flows, in *Proceedings of the 8th International Symposium on Information Processing in Sensor Networks (IPSN'09)*, San Francisco, CA, USA, April 2009, pp. 97–108
74. R. Sarkar, W. Zeng, J. Gao, X.D. Gu, Covering space for in-network sensor data storage, in *Proceedings of the 9th International Symposium on Information Processing in Sensor Networks (IPSN'10)*, Stockholm, Sweden, April 2010, pp. 232–243
75. O. Saukh, R. Sauter, M. Gauger, P.J. Marrón, On boundary recognition without location information in wireless sensor networks. *ACM Trans. Sens. Netw.* **6**, 20:1–20:35 (2010)
76. A. Savvides, C.-C. Han, M.B. Strivastava, Dynamic fine-grained localization in ad-hoc networks of sensors, in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom 2001)*, Rome, Italy (ACM, 2001), pp. 166–1779
77. A. Savvides, H. Park, M.B. Strivastava, The n -hop multilateration primitive for node localization problems. *Mob. Netw. Appl.* **8**(4), 443–451 (2003)
78. J.B. Saxe, Embeddability of weighted graphs in k -space is strongly NP-hard, in *Proceedings of the 17th Allerton Conference in Communications, Control and Computing*, Monticello, IL, 1979, pp. 480–489
79. Y. Shang, W. Rum, Improved MDS-based localization, in *Proceedings of IEEE NOFOCOM*, Hong Kong, China, 2004
80. Y. Shang, W. Rum, Y. Zhang, M.P.J. Fromherz, Localization from mere connectivity, in *MobiHoc03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Annapolis, MD, USA, 2003, pp. 201–212
81. A.M.-C. So, Y. Ye, Theory of semidefinite programming for sensor network localization, in *SODA'05: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Vancouver, BC, Canada, 2005, pp. 405–414
82. I. Stojmenovic, X. Lin, Loop-free hybrid single-path/flooding routing algorithms with guaranteed delivery for wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **12**(10), 1023–1032 (2001)
83. I. Stojmenovic, X. Lin, Power-aware localized routing in wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **12**(11), 1122–1133, (2001)
84. H. Takagi, L. Kleinrock, Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Trans. Commun.* **32**(3), 246–257 (1984)
85. G. Tan, H. Jiang, S.Z. Zhang, A.-M. Kermarrec, Connectivity-based and anchor-free localization in large-scale 2d/3d sensor networks, in *Proceedings of the Eleventh ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc10* (ACM, New York, 2010), pp. 191–200

86. P.F. Tsuchiya, The landmark hierarchy: a new hierarchy for routing in very large networks, in *SIGCOMM'88: Symposium proceedings on Communications Architectures and Protocols*, Stanford, CA, USA, 1988, pp. 35–42
87. Y. Wang, X.-Y. Li, Geometric spanners for wireless ad hoc networks, in *Proceedings of the 22-nd International Conference on Distributed Computing Systems (ICDCS)*, Vienna, Austria, 2002
88. Y. Wang, J. Gao, J.S.B. Mitchell, Boundary recognition in sensor networks by topological methods, in *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, Los Angeles, CA, USA, Sept 2006, pp. 122–133
89. Y. Wang, S. Lederer, J. Gao, Connectivity-based sensor network localization with incremental Delaunay refinement method, in *INFOCOM'09*, Rio de Janeiro, Brazil, 2009, pp. 2401–2409
90. X. Yu, X. Ban, R. Sarkar, W. Zeng, X.D. Gu, J. Gao, Spherical representation and polyhedron routing for load balancing in wireless sensor networks, in *Proceedings of 30th Annual IEEE Conference on Computer Communications (INFOCOM'11)*, Shanghai, China, April 2011

Gradient-Constrained Minimum Interconnection Networks

Marcus Brazil and Marcus G. Volz

Contents

1	Introduction	1460
2	Motivation and Background.....	1460
2.1	Underground Mining Networks.....	1462
2.2	Network Optimization Applied to Underground Mine Design.....	1463
2.3	Minkowski Spaces and the Gradient Metric.....	1464
2.4	The Fermat-Weber Problem.....	1466
2.5	The Steiner Tree Problem.....	1470
2.6	The Gilbert Arborescence Problem.....	1471
3	The Two-Dimensional Gradient-Constrained Steiner Problem.....	1472
3.1	Local Minimality and Exact Solutions.....	1474
3.2	Computational Complexity.....	1479
4	The Three-Dimensional Gradient-Constrained Steiner Problem.....	1481
4.1	Properties of Steiner Points.....	1481
4.2	Local Minimality for a Fixed Topology.....	1485
4.3	Algorithms and Software Solutions.....	1491
5	The Gilbert Problem in Minkowski Spaces.....	1491
6	The Two-Dimensional Gradient-Constrained Gilbert Problem.....	1494
6.1	Edge Vectors and Fundamental Properties of MGAs.....	1494
6.2	A Classification of Steiner Points.....	1497
7	The Three-Dimensional Gradient-Constrained Gilbert Problem.....	1501
7.1	Classification of Steiner Points of Degree at Most Four.....	1501
7.2	Maximum Degree of Steiner Points.....	1505
8	Conclusion.....	1507
	Cross-References.....	1507
	Recommended Reading.....	1507

M. Brazil (✉)

Department of Electrical and Electronic Engineering, The University of Melbourne, Parkville,
VIC, Australia

e-mail: brazil@unimelb.edu.au

M.G. Volz

TSG Consulting, Melbourne, VIC, Australia

e-mail: marcus.volz@tsgconsulting.com.au

Abstract

In two- or three-dimensional space, an embedded network is called *gradient constrained* if the absolute gradient of any differentiable point on the edges in the network is no more than a given value m . This chapter gives an overview of the properties of gradient-constrained networks that interconnect a given set of points and that minimize some property of the network, such as the total length of edges. The focus is particularly on geometric Steiner and Gilbert networks. These networks are of interest both mathematically and for their applications to the design of underground mines.

1 Introduction

In two- or three-dimensional space, an embedded network is called *gradient constrained* if the absolute gradient (with respect to a vertical axis) of any differentiable point on the edges in the network is no more than a given value m . This chapter is an overview of the properties of interconnection networks that are minimum under such a constraint. The sorts of networks that will be considered here are geometric Steiner and Gilbert networks. Both of these involve constructing a network that interconnects a given set of points (known as *terminals*) and that minimizes some property of the network, such as the total length of edges. Unlike minimum spanning trees, Steiner and Gilbert networks may contain nodes other than the terminals; these extra nodes are referred to as *Steiner points*. The challenge in optimizing these networks lies in determining both the locations of the Steiner points and the graph topology of the network.

Gradient-constrained minimum interconnection networks are interesting both from a mathematical point of view and because of their potential applications. Mathematically, they lie between Euclidean networks and fixed orientation networks. In practical terms, they have an important role in the efficient design of underground mines. Both of these aspects will be explored in the course of this chapter.

The structure of this chapter is as follows. Section 2 outlines the motivation to this problem, particularly in terms of its application to the design of underground mines, and presents some of the required mathematical background such as properties of Minkowski spaces and the Fermat-Weber problem. Sections 3 and 4 focus on the gradient-constrained Steiner problem (in two and three dimensions) where the objective is to minimize the total edge length of the network. In Sects. 5–7, the gradient-constrained Gilbert problem is introduced; this involves designing the network so as to minimize the cost associated with a flow on the edges.

2 Motivation and Background

One of the main motivations for studying gradient-constrained interconnection problems lies in their application to the design of underground mines. Mining is a significant industry worldwide. Reducing the cost of mining operations is an

important issue for mining companies in an extremely competitive marketplace. For many years, there have been well-developed methods for modeling and optimizing the operation of open-cut mines. Commercial optimization mine planning software packages have been developed based on a technique developed in 1965 by Lerchs and Grossmann [46]. These packages have revolutionized the design of open-pit mines, allowing mining engineers and planners to manipulate and visualize data associated with optimal designs.

Until recently, comprehensive optimization tools for the design and planning of underground mining operations have not been available to the mining industry. The design process is typically facilitated by planning CAD software packages (see, e.g., [31]) which require high levels of user intuition and experience. The usefulness of these programs lies in their ability to efficiently generate a range of potential designs, from which the “best” feasible solution can be selected and refined. However, since these programs do not harness the benefits of optimization, there is no guarantee that the final solution is even close to optimal.

A number of research groups, however, are now in the process of designing and developing optimization tools for the design of underground mines. An example is the access optimization tool, *Decline Optimisation Tool* (DOT), which has been developed at The University of Melbourne. This application efficiently determines a near-optimal (where optimal means least cost) underground mining network servicing a given set of points associated with an ore body [19].

Several constraints are imposed on the geometry of the tunnels in the network, to ensure that the tunnels are navigable by haulage trucks. One such constraint is that the slope (steepness or grade) of a tunnel cannot be too large, since the maximum grade at which haulage vehicles can operate is typically 1:7. A network satisfying this constraint is called *gradient constrained*.

The first mathematical study of such networks was conducted by the Network Research Group at The University of Melbourne, who initially studied them in a vertical plane [10] and later in three dimensions [13]. An algorithm for computing least-cost gradient-constrained networks was implemented into a software application [16], called *Underground Network Optimiser* (UNO). Although this application does not account for other practical constraints, such as truck turning circle navigability, UNO is especially useful for efficiently determining optimal layouts for large-scale mining networks where the effects of constraints other than the gradient constraint on the optimal layout of the mine are not significant.

Initially, research on gradient-constrained networks focused on networks which minimize length. While length-minimizing networks optimize the total development (or infrastructure) cost associated with an underground mine, they do not account for the effects of haulage on the optimal design of the mine. Haulage costs can have a significant impact on the structure and layout of an underground mine, and considerable savings can be achieved by including both cost components in the objective function. A network facilitating flows between given pairs of vertices in the network is called a *flow-dependent network*. In the mining context, these networks can incorporate both development and haulage costs into the objective function.

This section looks first at the nature of the underground mining problem, and the application of gradient-constrained interconnection networks to this problem, and then introduces some of the main mathematical ideas required for this problem.

2.1 Underground Mining Networks

This background on mining is drawn primarily from [2] and [8]. An underground mine consists of a series of interconnecting tunnels, ore passes (near-vertical chutes down which ore is dropped), and vertical shafts (used to hoist ore up to the surface). Its purpose is to allow extraction of ore containing valuable minerals (such as gold, silver, lead, zinc, and copper) from underground locations to a predetermined surface portal (or breakout point), from where it is transported to a processing mill.

Ore zones (or stopes) are identified by geological tests such as surface and infill drilling. From this information, mining engineers can determine suitable draw points, which are the locations on the boundary of each stope from which the ore is accessed. The ore is then excavated using one of a number of possible mining methods (such as stoping, caving, room, and pillar) and is transported to the surface via large haulage trucks.

Given that these laden trucks must be able to traverse the ramps, the following important constraints must be imposed on the mine:

- Ramps must have absolute gradient not greater than some constant m . This constant is typically in the range 1:9–1:7 depending on equipment specifications.
- Ramps must satisfy a minimum turning radius (typically in the range 15–30 m), so that they are navigable by the trucks.
- Ramps must avoid certain no-go regions such as the interior of an ore body or other ramps in the mine.

A set of tunnels satisfying these constraints and interconnecting the draw points and a point at the surface can be modeled by a mathematical network T , where the draw points correspond to fixed vertices and the tunnels correspond to edges. The cost C associated with a network with a set E of edges is typically modeled by a function of the form

$$C(T) = \sum_{e \in E} (d + ht_e)l_e$$

where d is a development cost rate (typically \$4,000/m), h is a haulage cost rate (typically \$0.75/t.km), t_e is the total quantity of ore to be transported along an edge e over the life of the mine, and l_e is the length of e . The first term $\sum dl_e$ can be viewed as the total development cost of the mine and the second component $\sum ht_e l_e$ as the total haulage cost associated with the mine over its life. If h is set to zero, then the cost of the network is proportional to its length, and a network minimizing this objective function is a *length-minimizing network*.

2.2 Network Optimization Applied to Underground Mine Design

This section is a brief survey of the use of network theory in underground mine design. A number of the papers mentioned here will be discussed in more detail later in the chapter.

Lizotte and Elbrond [47] reported an early application of network theory to some underground mine layout problems but limited their work to network techniques in a horizontal plane. Brimberg et al. [24] considered a network model to minimize the cost of an underground mine with a vertical hoisting shaft connected to ore deposits by a series of horizontal tunnels. A heuristic method for a more general version of this problem was developed in [36], using fuzzy set theory.

Lee first raised the more general problem of setting up and optimizing a three-dimensional network modeling the access infrastructure costs of an underground mine layout in a 1989 presentation at the NATO Workshop on Topological Network Design, Denmark. Since then, a significant body of research has been developed in this area, some of which has been implemented into the aforementioned software products, UNO and DOT. Much of this work has been conducted by the Network Research Group at The University of Melbourne. The following is a summary of this research to date:

- A network optimization model for minimizing development and haulage costs in underground mines was established. The possibility of modeling the haulage cost of an edge as a function of the slope of the edge was also facilitated in the cost function, and conditions under which the cost function is convex were discussed. Application of the model was demonstrated via a prototype network interconnecting draw points and a vertical hoisting shaft and several case studies in which the model was applied to mines at Pajingo and Kanowna Belle [2, 8, 11, 14, 17].
- The gradient-constrained Steiner problem (in a vertical plane) was introduced. This problem asks for a shortest network interconnecting a set of points, called *terminals*, in a vertical plane such that no part of the network has absolute gradient greater than a given positive constant m satisfying $m \leq 1$. Geometric structures of these networks were used to construct a finite algorithm for solving the problem which, when suitably discretized, was shown to be NP-complete [10].
- The study of the gradient-constrained Steiner problem was extended to three dimensions. The *gradient metric* was introduced to measure the lengths of edges in gradient-constrained networks, and the terms *flat*, *maximum*, and *bent* were introduced to label edges whose respective gradients are less than, equal to, or greater than m . It was shown that a Steiner point, which is a vertex in the network not among the given set of terminals, has either three or four incident edges, and Steiner points were classified in terms of the labelings of their incident edges. The UNO software product was concurrently developed with this work [13].
- The terms *labeled minimal* and *locally minimal* were introduced to describe Steiner points for which a label-preserving perturbation and, respectively, *any*

perturbation of the Steiner point cannot shorten the length of the network. Properties of labeled-minimal Steiner points were studied, and necessary and sufficient conditions for Steiner points to be locally minimal were obtained. A formula for computing labeled-minimal degree 3 Steiner points was derived, and an algorithm for computing locally minimal Steiner points was developed [21, 61].

- The *Steiner ratio* for gradient-constrained networks connecting three points in three dimensions was investigated. The Steiner ratio is the smallest ratio of the length of a Steiner tree to the length of a minimum spanning tree when the two networks interconnect the same set of given points. It was shown that this minimum ratio for gradient-constrained networks tends to 0.75 as the value of m tends to zero [50, 51].
- A method for optimizing *declines* in underground mines was developed, a decline being a gradient-constrained, turning-circle-constrained path connecting level access points to a surface portal while avoiding certain no-go regions. A procedure for finding an optimal decline was implemented into the DOT software product, which was applied to several case studies including a decline servicing the Jandam gold mine at Pajingo and an access decline at Olympic Dam. The software was further developed to handle networks of declines [15, 16, 19, 20, 62].
- The other area that has been receiving more attention in recent years is that of understanding the effect of weighted edges, associated with flows, on the optimal design of gradient-constrained networks. The principal aim of this research is to obtain a comprehensive understanding of minimum-cost gradient-constrained flow-dependent networks. Of particular interest are the geometric structures possible in such networks. The geometric properties of optimal networks play a crucial role in improving the efficiency of heuristic algorithms for computing gradient-constrained networks [66–68].

2.3 Minkowski Spaces and the Gradient Metric

An important insight into understanding gradient-constrained networks in Euclidean space is that if the distance between two points is measured as the minimum length of a gradient-constrained path between the two points, then this measure forms a metric. For the purpose of determining lengths of edges, the network can be viewed as being embedded in a normed or Minkowski space. This subsection briefly reviews some of the relevant properties of Minkowski spaces, and defines the gradient metric, for measuring the lengths of edges in gradient-constrained networks. See [63] for a more comprehensive introduction to Minkowski geometry.

A *Minkowski space* (or *finite-dimensional Banach space*) is \mathbb{R}^n endowed with a *norm* $\|\cdot\|$, which is a function $\|\cdot\| : \mathbb{R}^n \rightarrow \mathbb{R}$ that satisfies:

- $\|x\| \geq 0$ for all $x \in \mathbb{R}^n$, $\|x\| = 0$ only if $x = 0$.
- $\|\alpha x\| = |\alpha| \|x\|$ for all $\alpha \in \mathbb{R}$ and $x \in \mathbb{R}^n$.

- $\|x + y\| \leq \|x\| + \|y\|$.

Informally, a norm provides a way of measuring the lengths of vectors in \mathbb{R}^n . A common norm is the ℓ_p norm

$$\|x\|_p = \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

where $p \geq 1$ is a real number and the x_i are the components of x . The value $p = 2$ gives the familiar Euclidean norm, denoted throughout this chapter by $|\cdot|$, and $p = 1$ gives the rectilinear norm, also called the Manhattan, taxicab, or ℓ_1 norm.

The *unit ball* associated with a given Minkowski space X is the set $B = \{x \in X : \|x\| \leq 1\}$. It is a centrally symmetric, closed, and convex subset of X with origin o at its center. A unit ball is called *smooth* if its boundary, $\text{bd}(B)$, has no sharp (nondifferentiable) points and *strictly convex* if $\text{bd}(B)$ contains no straight line segments. A *supporting hyperplane* of B is a hyperplane in X touching the boundary of B such that the interior of B lies on one side of the supporting hyperplane. An *exposed face* of B is the intersection of B with a supporting hyperplane.

It is possible to measure the length of edges in a gradient-constrained network in terms of a metric known as the gradient metric, first introduced in [13]. This means that these networks can be thought of as being in a Minkowski space. Formal definitions are given below for the three-dimensional case, but these results also generalize easily to other dimensions.

Let x_p, y_p, z_p denote the Cartesian coordinates of a point p in three-dimensional space. Assume that the z -axis is vertical. Then, the *gradient* of an edge pq is defined here to be the absolute value of the slope from p to q and is denoted by $g(pq)$. That is,

$$g(pq) = \frac{|z_q - z_p|}{\sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}}.$$

Suppose pq is an edge of a minimum gradient-constrained network embedded in Euclidean space, where the maximum permitted gradient m is given. If $g(pq) \leq m$, then pq is a straight line joining p and q and is referred to as a *straight edge*. However, if $g(pq) > m$, then pq cannot be represented as a straight line without violating the gradient constraint, but it can be represented by a zigzag line joining p and q with each segment having gradient m . Such edges are referred to as *bent edges*.

The lengths of edges in a gradient-constrained tree can be measured in a special metric, called the *gradient metric*. Suppose o is the origin and $p = (x_p, y_p, z_p)$ is a point in space. Define the *vertical metric* of the line op to be $|op|_v = cz_p$ where c is a given constant. Then the gradient metric can be defined in terms of the Euclidean and vertical metrics. Suppose m is the maximum gradient allowed in gradient-constrained trees. The length of op in the gradient metric is defined to be

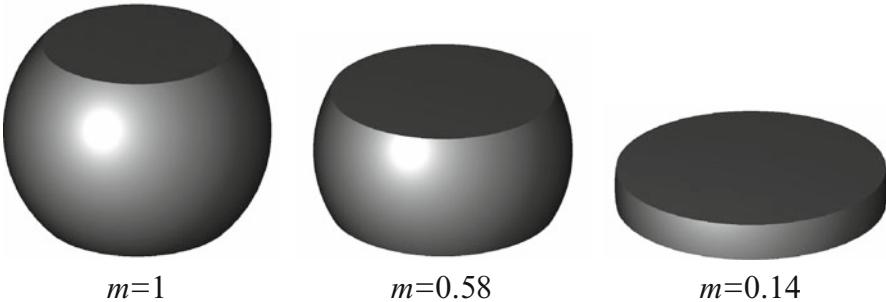


Fig. 1 Unit ball in gradient-constrained three-space for selected maximum gradients

$$|pq|_g = \begin{cases} |pq| = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2 + (z_q - z_p)^2}, & \text{if } g(pq) \leq m; \\ |pq|_v = \sqrt{1 + m^{-2}}|z_q - z_p|, & \text{if } g(pq) \geq m. \end{cases}$$

It is easily checked that this defines a metric, as it is the maximum of two metrics. Note that $|op| \leq |op|_g$ and the gradient metric is convex though it is not strictly convex.

In gradient-constrained three-space, three examples of the unit ball B_g are shown in Fig. 1 for $m = 1$, 0.58 , and 0.14 .

2.4 The Fermat-Weber Problem

The types of interconnection problems considered in this chapter are those that allow for the possibility of adding extra nodes anywhere in the space in order to minimize the length (or, more generally, cost) of the interconnection network. The simplest and most famous example of such a problem is the Fermat problem, which asks one to find a point such that the sum of its distances from a set of given points is a minimum. The more general version of this problem is called the *Fermat-Weber problem*; this problem asks for a point, called a *Fermat-Weber point*, minimizing the sum of weighted distances to a set of given points. The weights are specified by given values w_i associated with each of the given points p_i . A brief exposition of the historical background to this problem is given by Kuhn [42]. A more detailed account is included in a paper of Kupitz and Martini [44], and another good general reference is [27].

Although special instances of the Fermat-Weber problem can be solved with simple geometric constructions, in general for $k \geq 4$, the problem cannot be solved by exact methods [3]. Thus, significant research has been undertaken to develop numerical procedures for finding Fermat-Weber points. A famous iterative method for the Fermat-Weber problem (outlined below) was posed in 1937 by Andre Weiszfeld [74]. This algorithm was independently discovered by Kuhn and Kuenne [43] in 1962.

Let $N = \{p_1, \dots, p_k\}$ be the set of given points in \mathbb{R}^n with respective positive weights w_1, \dots, w_k . Let \mathbf{u}_i denote the unit vector from a point $x_0 \in \mathbb{R}^n$ to p_i , $i = 1, \dots, k$. The first step of the algorithm checks whether x_0 coincides with a given point. If not, x_0 is determined iteratively.

Weiszfeld's Algorithm

1. If there exists a point $p_j \in N$ such that

$$\left| \sum_{i=1, i \neq j}^k w_i \mathbf{u}_i \right| \leq w_j,$$

then p_j is a Fermat-Weber point.

2. Otherwise, select an error estimate ϵ .
3. Select an initial point $x^{(0)} \in \text{int}(\text{conv}(N))$.
4. For $\kappa = 0, 1, \dots$ do

$$x^{(\kappa+1)} := \frac{\sum_{p_i \in N} \frac{w_i p_i}{|p_i - x^{(\kappa)}|}}{\sum_{p_i \in N} \frac{w_i}{|p_i - x^{(\kappa)}|}}.$$

5. Stop when $|x^{(\kappa)} - x^{(\kappa+1)}| \leq \epsilon$.
-

The following example demonstrates the application of Weiszfeld's algorithm and is illustrated in Fig. 2. Let $p_1 = (0, 1)$, $p_2 = (2, 0)$, and $p_3 = (2, 2)$ be given points in the plane with respective weights $w_1 = 1$, $w_2 = 1$, and $w_3 = 1.65$. By testing the collapse condition, it can be shown that $x_0 \neq p_1, p_2, p_3$. Selecting $\epsilon = 0.001$ and $x^{(0)} = (0.25, 1)$, the path of the iteration point converges to x_0 , as shown in the figure.

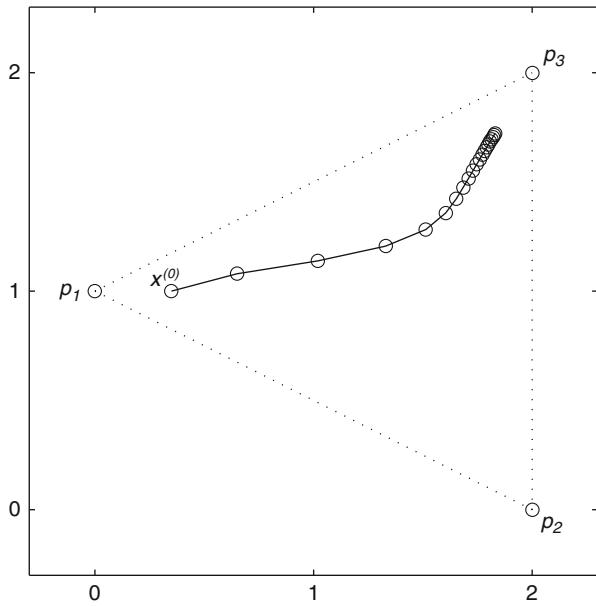
In 1995, Brimberg [23], building on the work of Kuhn [41] and Chandrasekaren et al. [25], proved that Weiszfeld's algorithm converges to the unique optimal solution for all but a denumerable set of starting points if and only if the convex hull of the given points is of dimension n .

The Fermat-Weber point can be viewed as the central node of a star network with links from the Fermat-Weber point to each of the given nodes. This network is then a minimum cost interconnection network with this star topology. In this context, one can define the *gradient-constrained Fermat-Weber problem*, introduced by Volz in [66] and [68], which is the Fermat-Weber problem with the added condition that links are measured using the gradient metric. The problem is formulated as follows:

- *Given:* A set $N = \{p_1, \dots, p_k\}$ of points in Euclidean space with respective positive weights w_1, \dots, w_k and a gradient bound m satisfying $0 < m \leq 1$
- *Find:* A point x_0 minimizing the sum of weighted distances from x_0 to the points in N such that each distance is the minimum length of a piecewise smooth curve whose gradient at each differentiable point is no more than m

In the mining context, p_1, \dots, p_k represent draw points with associated tonnages t_1, \dots, t_k . Material is hauled from the draw points via independent paths to the

Fig. 2 Iterations of Weiszfeld's algorithm for a three-point Fermat-Weber problem



Fermat-Weber point, x_0 , which might represent, for instance, the base of a vertical hoisting shaft. The total cost of the mine can be minimized by positioning the facility at an optimal location.

The weight on the path from p_i to x_0 is given by $w_i = d + ht_i$, where d and h are development and haulage cost rates, respectively. Since the t_i can take on any value, there are no restrictions on the values of the weights associated with the points in N . This is in contrast to the Gilbert arborescence problem described in the next section, in which edge weights in the network satisfy certain implicit conditions resulting from the flow structure in the network.

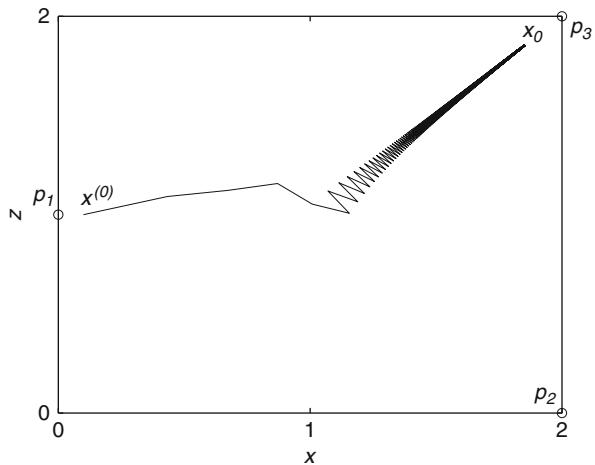
Numerical tests have shown that the Weiszfeld algorithm, described above, does not lend itself well to the gradient-constrained problem. In [66] and [68], Volz has proposed a new iterative scheme for finding a point x_0 minimizing

$$f(x) = \sum_{i=1}^k |p_i - x|_g.$$

Standard gradient descent procedures, such as those discussed in [6], cannot be easily applied here due to the fact that $f(x)$ is not everywhere differentiable. The problem for nondifferentiable convex functions, however, can be solved by the *subgradient method*, described in [57] and [5], which minimizes $f(x)$ using the iteration

$$x^{(\kappa+1)} = x^{(\kappa)} + s_\kappa \mathbf{v}^{(\kappa)}$$

Fig. 3 Iterations of the subgradient method with diminishing step sizes for a three-point gradient-constrained Fermat-Weber problem



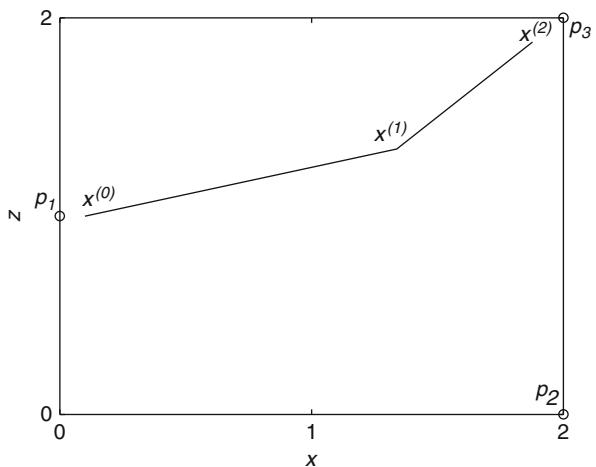
where $x^{(\kappa)}$ is the κ th iterate, $\mathbf{v}^{(\kappa)}$ is any subgradient of f at $x^{(\kappa)}$, and s_κ is the κ th step size. For diminishing step sizes satisfying $\lim_{\kappa \rightarrow \infty} s_\kappa = 0$, $\sum_{\kappa=1}^{\infty} s_\kappa = \infty$, the algorithm is guaranteed to converge to the optimal value.

For the gradient-constrained problem, use of the subgradient method may be problematic, despite guaranteed convergence. The convergence can be extremely slow due to the gradient being almost perpendicular to the direction toward the minimum. This problem is demonstrated by the following example.

Consider three points in a vertical plane: $p_1 = (0, 1)$, $p_2 = (2, 0)$, and $p_3 = (2, 2)$, as shown in Fig. 3. The three points have respective weights $w_1 = 1$, $w_2 = 1$, and $w_3 = 1.94$. The maximum allowable gradient is $m = 1$. The exact solution, which can in this case be determined exactly by simple calculus, is $x_0 = (1.9, 1.9)$. Starting at $x^{(0)} = (0.1, 1)$ and using an initial step size $s_0 = 0.345$, consider the use of the subgradient method where step sizes diminish at each iterate κ according to $s_\kappa = s_0/\sqrt{\kappa}$. The iterative path, shown in the figure, shows that convergence is very slow due to extreme oscillation about the line through p_3 with gradient m . At each iteration, the gradient is almost perpendicular to the direction toward the minimum. In fact, after 50,000 iterations, $x^{(\kappa)}$ is still noticeably distant from x_0 . If s_κ is chosen to minimize $f(x^{(\kappa)} + s_\kappa \mathbf{v}^{(\kappa)})$ (i.e., an exact line search is adopted), then $x^{(\kappa)}$ converges to the nonstationary point $(1.3385, 1.3385)$ after one iteration, due to the optimum step size diminishing to zero too early.

Volz [66] and Volz et al. [68] propose a new method that overcomes these problems by exploiting the fact that f is differentiable everywhere except in the set S of points x in \mathbb{R}^3 for which $p_j x$ is an m -edge for any $p_j \in N$. The method uses exact directional minimization at each iteration and considers a second search direction at maximum gradient when $x^{(\kappa)}$ converges to a (possibly non-minimal) point in some neighborhood of S . This notion of searching in an m -edge direction was first employed in an algorithm for finding gradient-constrained minimum Steiner trees in [16]. The procedure for the gradient-constrained Fermat-Weber

Fig. 4 Iterations of a new descent method for a three-point gradient-constrained Fermat-Weber problem



problem is outlined in [66, 68], where it is also shown that the method guarantees rapid convergence in dimensions 2 and 3.

Applying this method to the previous example results in the sequence converging to the exact solution in two iterations, $\{(0.1, 1), (1.3385, 1.3385), (1.9, 1.9)\}$ (Fig. 4).

2.5 The Steiner Tree Problem

The *minimum Steiner tree problem* asks for a shortest network spanning a given set of nodes, usually referred to as *terminals*, in a given metric space. It differs from the minimum spanning tree problem in that additional nodes, referred to as *Steiner points*, can be included to create an interconnection network that is shorter than would otherwise be possible. This is a fundamental problem in physical network design optimization, and has numerous applications, including the design of telecommunications or transport networks for the problem in the Euclidean plane (the l_2 metric), and the design of microchips for the problem in the rectilinear plane (the l_1 metric) [40].

The terminology used for the minimum Steiner tree problem in this chapter is largely based on that used in [40]. Let T be a network interconnecting a set $N = \{p_1, \dots, p_n\}$ of points, called *terminals*, in a Minkowski space. A node in T which is not a terminal is called a *Steiner point*. In a minimum network, it can always be assumed that the degree of a Steiner point is at least three. Let $G(T)$ denote the *topology* of T , that is, $G(T)$ represents the graph structure of T but not the embedding of the Steiner points. Then $G(T)$ for a shortest network T is necessarily a tree, since if a cycle exists, the length of T can be reduced by deleting an edge in the cycle. A network with a tree topology is called a *tree*, its links are called

edges, and its nodes are called *vertices*. An edge connecting two vertices a, b in T is denoted by ab and its length by $\|a - b\|$.

The *splitting* of a vertex is the operation of disconnecting two edges av, bv from a vertex v and connecting a, b, v to a newly created Steiner point. Furthermore, though the positions of terminals are fixed, Steiner points can be subjected to arbitrarily small movements provided the resulting network is still connected. Such movements are called *perturbations* and are useful for examining whether the length of a network is minimal.

A *Steiner tree* is a tree whose length cannot be shortened by a small perturbation of one or more of its Steiner points, even when splitting is allowed. By convexity, a Steiner tree is a minimum-length tree for its given topology. A *minimum Steiner tree* is a shortest tree among all Steiner trees. For many Minkowski spaces, bounds are known for the maximum possible degree of a Steiner point in a Steiner tree, giving useful restrictions on the possible topology of an SMT. For example, in Euclidean space of any dimension, every Steiner point in an ST has degree 3. Given a set N of terminals, the *Steiner problem* (or *minimum Steiner tree problem*) asks for a minimum Steiner tree spanning N .

The gradient-constrained Steiner problem is studied in more detail in [Sects. 3](#) and [4](#).

2.6 The Gilbert Arborescence Problem

In 1967, Gilbert [34] proposed a generalization of the SMT problem whereby symmetric nonnegative *flows* are assigned between each pair of terminals. The aim is to find a least cost network interconnecting the terminals, where each edge has an associated total flow such that the flow conditions between terminals are satisfied, and Steiner points satisfy Kirchhoff's rule (i.e., the net incoming and outgoing flows at each Steiner point are equal). The cost of an edge is its length multiplied by a nonnegative *weight*. The weight is determined by a given function of the total flow being routed through that edge, where the function satisfies a number of conditions. The *Gilbert network problem* (GNP) asks for a minimum-cost network spanning a given set of terminals with given flow demands and a given weight function.

A variation on this problem that will be shown to be a special case of the GNP occurs when the terminals consist of n sources and a unique sink, and all flows not between a source and the sink are zero. This problem is of intrinsic interest as a natural restriction of the GNP; it is also of interest for its many applications to areas such as drainage networks [45], gas pipelines [4], and underground mining networks [11].

If the weight function is a concave, increasing function of the flow, then the resulting minimum network has a tree topology and provides a directed path from each source to the sink (this is discussed in more detail in [Sect. 5](#)). Such a network can be called an *arborescence*, and this special case of the GNP is referred to as the *Gilbert arborescence problem*. Traditionally, the term “arborescence” has been used to describe a rooted tree providing directed paths from the unique root (source) to a

given set of sinks. Here, it is used for the case where the flow directions are reversed, that is, flow is from n sources to a unique sink. It is clear, however, that the resulting weights for the two problems are equivalent; hence, it is reasonable to also use the term “arborescence” for the latter case. Moreover, if one takes the sum of these two cases and rescales the flows (dividing flows in each direction by 2), then again the weights for the total flow on each edge will be the same as in the previous two cases. This justifies the earlier claim that the Gilbert arborescence problem can be treated as a special case of the GNP. It will be convenient, however, for the remainder of this chapter to treat arborescences as networks with a unique sink.

A gradient-constrained version of the Gilbert arborescence problem is studied in Sects. 6 and 7. In three-space, the simplest version of this problem can be formulated as follows:

- *Given:* A set of points $N = \{p_1, \dots, p_k\}$ in Euclidean three-space, where p_1, \dots, p_{k-1} are sources with respective positive flows t_1, \dots, t_{k-1} and p_k is a unique sink, a gradient bound satisfying $0 < m \leq 1$, and positive constants d and h
- *Find:* A minimum-cost network T interconnecting N which provides flow paths from the sources to the sink, such that the cost of an edge e in T is given by $(d + ht_e)l_e$, where t_e is the total flow through e and l_e is the length of e under the gradient metric

A network satisfying the above conditions is called a *minimum Gilbert arborescence* (MGA). Vertices in T not in N are called *Steiner points*, and a Steiner point is a Fermat-Weber point with respect to its adjacent vertices in T . A key difference between the two problems is that a Steiner point s in a Gilbert arborescence has *source edges*, which are directed into s , and a *unique sink edge*, directed outward from s , whereas all incident paths connected to a Fermat-Weber point are directed toward that point.

3 The Two-Dimensional Gradient-Constrained Steiner Problem

The simplest version of a gradient-constrained interconnection problem, beyond that of constructing a minimum spanning tree, is the planar Steiner problem. This problem asks for a shortest network interconnecting a given set of points (referred to as *terminals*) lying on a vertical plane in Euclidean space and operating under a gradient constraint. Unlike a spanning tree network, this network can contain nodes other than the given terminals. It will be shown in Sect. 3.2 that the inclusion of these extra variable nodes makes this an NP-hard problem.

In formal terms, the *two-dimensional gradient-constrained Steiner problem* can be stated as follows:

- *Given:* A set of points $N = \{p_1, \dots, p_k\}$ lying on a vertical plane \mathcal{P} in Euclidean three-space and a gradient bound satisfying $0 < m \leq 1$
- *Find:* A network T interconnecting N of minimum total length, where the length of each edge e is computed using the gradient metric (with gradient bound m)

As usual, the nodes of T that are not terminals are referred to as *Steiner points* and are assumed to have degree at least 3. A network T , solving this problem, is referred to as an *m -constrained minimum Steiner tree*.

From an applications point of view, this problem is interesting, not only as a specific example of the three-dimensional problem but also for potential applications in its own right. For example, consider the following problem. Imagine an exterior wall of a large multistory warehouse, or other building, with entrances at given positions on the wall. Suppose one is required to join the entrances externally by designing an interconnecting system of ramps whose gradients are sufficiently small to allow trolleys and carts to be wheeled along them, or alternatively to allow disabled access. By modeling the entrances as points and the ramps as line segments, the problem of finding the shortest system of ramps is equivalent to solving the planar version of the gradient-constrained Steiner problem.

The precise formulation of the above problem deserves a few comments. The reason that the plane \mathcal{P} is restricted to being a vertical plane is that it can then be assumed that the network T lies on \mathcal{P} . This follows from the observation that the projection of T onto \mathcal{P} does not increase the length of any link of T (since the vertical displacement between the endpoints remains the same and the horizontal displacement does not increase). This observation is no longer true in general if \mathcal{P} is not restricted to being vertical. For \mathcal{P} vertical, the problem, although set in Euclidean three-space, can be assumed to be two-dimensional.

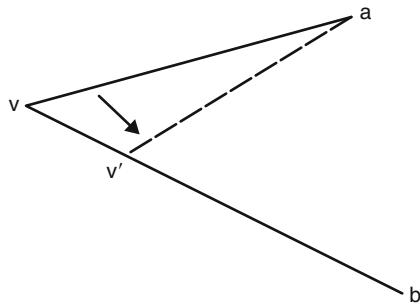
The other important restriction on the problem is that the gradient bound m satisfies $m \leq 1$. Let $\alpha = \arctan m$ be the angle of an edge of maximum gradient from the horizontal. The condition is equivalent to saying that $\alpha \leq \pi/4$. This condition is imposed largely for convenience. In particular, it immediately gives the following local property of T at any node.

Lemma 1 *Let T be an m -constrained minimal Steiner tree (for $0 < m \leq 1$) in a vertical plane \mathcal{P} . Then the angle at which any two links of T meet at a vertex is at least 2α .*

Proof Suppose, on the contrary, that there exists a node v of T such that two of the links incident with v , say, av and bv , meet at an angle less than 2α . Clearly, at least one of the two links, say, av , has gradient of absolute value strictly less than m . The length of T can now be reduced by moving the point where this link connects a to bv away from v , toward b to a new point v' such that either av' has absolute gradient m or $v' = b$ (whichever occurs first), as illustrated in Fig. 5. The shorter tree is still gradient constrained, contradicting the minimality of T . \square

This constraint on the gradient is usually easily met in practice in applications such as ramp design or underground mine access infrastructure where the typical maximum allowable gradient is generally in the range of $1/9$ – $1/7$. It follows from Lemma 1 that all nodes of T have degree at most 4 and hence that all Steiner points of T have degree 3 or 4. Indeed, the lemma has the following easy consequence, giving local geometric properties at each Steiner point.

Fig. 5 Replacing av by av' decreases the length of T



Corollary 1 Let s be a Steiner point of an m -constrained minimum Steiner tree T . Let L_s be the vertical line through s . If s is of degree 3, then s has two incident links, one of gradient m and the other of gradient $-m$, lying on one side of L_s , and a third incident link lying on the other side of L_s . If s is of degree 4, then s has two incident links, one of gradient m and the other of gradient $-m$, lying on each side of L_s .

3.1 Local Minimality and Exact Solutions

Like most other planar Steiner problems in normed spaces, the two-dimensional gradient-constrained Steiner problem is NP-hard, meaning that there is almost certainly no polynomial time algorithm for constructing an optimal solution. Despite this, the problem of finding relatively efficient exact algorithms for planar Steiner problems under a range of different norms has attracted considerable interest and success. The most effective and scalable algorithms are those based on the so-called Geosteiner approach [71, 72].

The key to the Geosteiner approach is to view a minimum Steiner tree as a union of *full Steiner trees* (FST's), which are minimum Steiner trees where all terminals are leaves and all Steiner points are interior nodes. The algorithm involves two distinct phases: a *generation phase* which generates a set of FSTs guaranteed to include those used in a minimum Steiner tree; and a *concatenation phase* which determines how to optimally combine the FSTs in the given set into a minimum Steiner tree. The concatenation phase is independent of the underlying metric and has been efficiently solved, using a branch-and-cut algorithm, by Warme [69] and Warme et al. [70, 71]. The success of the Geosteiner approach depends on keeping the number of FSTs constructed in the generation phase as small as possible. This relies on finding effective *pruning* techniques. The idea is to build the FSTs in a bottom-up manner and at each step to apply a range of geometric tests which will allow whole families of potential FSTs to be eliminated. An example of such a test is the “lune test,” which basically states that terminals cannot be too close to long edges in a minimum Steiner tree. For the Euclidean metric, a suite of pruning techniques, based on the strong local geometric restrictions on minimum Steiner trees, have been developed which keep the number of generated FSTs relatively small, for

randomly generated terminal sets. For other metrics, such as the rectilinear metric and the λ -geometry metrics (where the network is restricted to using λ equally spaced orientations, see [7]), the efficiency of the generation phase depends not only on good pruning techniques but also on the characterization of FSTs in terms of canonical forms. These canonical forms (established for general λ -geometry metrics in [18]) strongly restrict the number of FSTs that need to be considered and give a simple bottom-up method of building FSTs out of “half FSTs”; for more details, see Nielsen et al. [49]. In practice, for any fixed λ , the resulting number of generated FSTs is almost linear, and the size of the largest generated FST is bounded by a constant. Geosteiner has been used to solve randomly generated problem instances with 1,000 terminals in less than a hour (for $\lambda \leq 8$), and a single problem instance with 10,000 terminals has been solved in less than 48 h for $\lambda = 4$.

The Geosteiner approach has not been developed for the two-dimensional gradient-constrained Steiner problem, probably because the main applied interest is in the three-dimensional version of the problem, where the generation phase of Geosteiner appears to be much less effective. But the two-dimensional problem is amenable to the same general method as there are very strong structural and geometric constraints on the form of locally minimal FSTs. The theory behind this was developed in [10], and the main properties will now be discussed in the remainder of this section.

It is useful to divide the study of the structure of two-dimensional gradient-constrained minimal FSTs into two distinct cases, the first where all edges in the full tree have absolute gradient $\geq m$ and the second where there is at least one edge of gradient k where $|k| < m$. Each of these two cases will be considered in turn.

Case 1: All Edges Have Absolute Gradient $\geq m$

Recall that any edge e of absolute gradient greater than m with endpoints a and b can be embedded in the Euclidean plane as a path from a to b composed of two straight line segments, each of gradient m or $-m$, such that the Euclidean length of this path is equal to the length of e under the gradient metric. Such edges are referred to as a *bent* edges and the internal point where a bent edge changes gradient as the *corner-point* of the edge. Also, a straight line segment in a gradient-constrained minimal FST, each of whose endpoints is either a node or corner-point and which has no nodes in its relative interior, is referred to simply as a *segment*. Note that the angle between the two segments at a corner-point is 2α . This viewpoint of the FSTs being embedded in the Euclidean plane allows properties of Euclidean geometry to be employed in the study of these networks.

Let T be a two-dimensional gradient-constrained minimal FST such that every segment in T has absolute gradient m . This is very similar to the situation for rectilinear Steiner trees. Indeed, the characterization of the structure of a full rectilinear minimal Steiner tree does not use, in any essential way, the fact that the two available directions are perpendicular. It follows that by defining substructures of T in line with the standard definitions used in rectilinear Steiner trees, it is

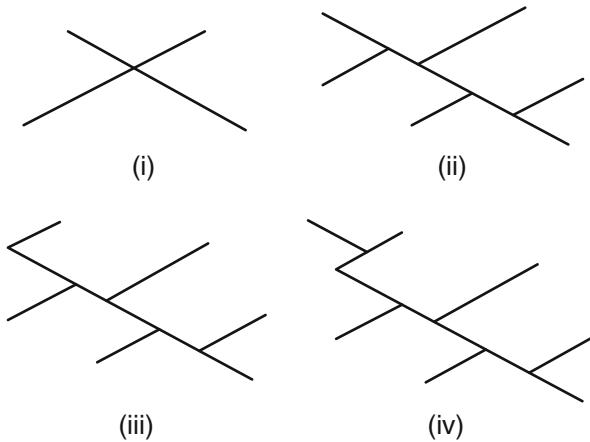


Fig. 6 The four canonical forms for full components given in [Theorem 1](#)

possible to prove a structure theorem giving a canonical form for T identical to that which has been proved for rectilinear Steiner trees. The definitions for substructures of T imitate the corresponding definitions for rectilinear trees used in [40], which are based largely on the geometric characterizations developed in [52].

A *complete line* of T is defined to be a sequence of one or more adjacent collinear segments of maximal length; it does not have terminals in its relative interior (since T is full), but it may have terminals as endpoints. A corner-point is incident to exactly two complete lines, one of gradient m and the other of gradient $-m$, referred to as the *legs* of the *corner*. If the remaining two endpoints of the legs are terminals, the corner is a *complete corner*. A cross-point is a degree 4 Steiner point of T , and the two complete lines passing through a cross-point are said to form a *cross*. Finally, given a complete line L of T , let $C(L)$ denote the set of segments (not in L) which are incident to L at any endpoints of L that are corners. Then a set A of *alternating segments incident to L* is defined to be the set of segments of T such that:

1. A contains all segments not in L but incident to L at vertices other than the endpoints of L (all of which, it follows, have a gradient which is minus the gradient of L).
2. Each element of A intersects L at a Steiner point of degree 3.
3. No two segments in $A \cup C(L)$ incident to endpoints of a single segment of L lie on the same side of L .

By performing a series of local length-preserving transformations on T , each of which either changes the embedding of a bent edge or “slides” an edge whose two endpoints are both Steiner points (see [40] or [52] for the formal definitions of these transformations in the rectilinear setting), one can obtain the following characterization of a canonical form for T , as illustrated in [Fig. 6](#). This theorem, in the rectilinear setting, was originally due to Hwang [37].

Theorem 1 Let N be a set of n terminals such that $n \geq 2$. Suppose every m -constrained minimal Steiner tree on N is full and contains only segments of absolute gradient m . Then there exists an m -constrained minimal Steiner tree on N that either:

- (i) Is a cross; or
- (ii) Consists of a single complete line with $n - 2$ alternating segments; or
- (iii) Consists of a complete corner with $n - 2$ alternating segments incident to a single leg; or
- (iv) Consists of a complete corner with $n - 3$ alternating segments incident to a one leg and a single alternating segment incident to the other leg

Note that the first form in [Theorem 1](#), a cross, only occurs if $n = 4$.

The consequence of this theorem is that for any terminal set N , there exists an m -constrained minimal Steiner tree on N such that every full component either has one of the forms listed in the previous theorem or contains an edge with absolute gradient strictly less than m . The generation of all candidate FSTs with all segments having absolute gradient m (as part of a Geosteiner approach) can be done efficiently in practice using the same techniques that have been developed for rectilinear minimum Steiner trees by Zachariasen [77].

Case 2: There Exists an Edge with Absolute Gradient $< m$

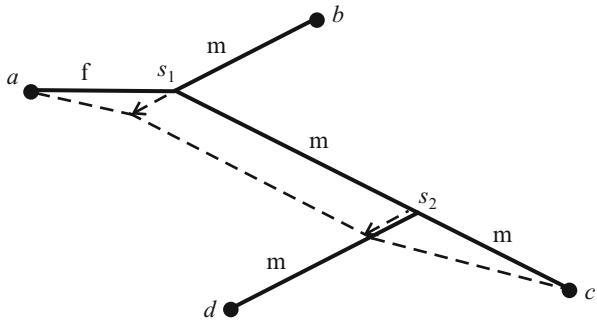
For this case, let T be a two-dimensional gradient-constrained minimal FST such that T contains a segment with gradient k where $|k| < m$. Such a segment is said to have *intermediate* gradient. Clearly, the segment must be a straight edge since segments at a corner have absolute gradient m . Furthermore, if a is an endpoint of an edge in T of intermediate gradient, then all edges of T incident with a are straight. This is an easy consequence of [Lemma 1](#), since if there is a bent edge at a , then there exists an embedding of that edge such that two of the segments incident with a have an angle between them of less than 2α , leading to a contradiction to the minimality of T by the argument in the proof of [Lemma 1](#). By a similar argument, it is also clear that a has degree at most three.

A key property of T is that every edge in T with intermediate gradient has the same gradient. More specifically, the following structure theorem holds.

Theorem 2 Let T be a full m -constrained minimal Steiner tree containing an edge of gradient k where $|k| < m$. Then every Steiner point of T is adjacent to three straight edges, one of gradient m , one of gradient $-m$, and one of gradient k .

The idea of the proof of this theorem, which appears in [10], is to show that if there are two adjacent Steiner points in T that do not satisfy the theorem, then there exists a simultaneous perturbation of both Steiner points that strictly reduces the length of T . This is illustrated in [Fig. 7](#). Suppose T is the m -constrained tree with terminals a, b, c, d as shown in the figure. Here, the points a, b, c, d all lie in a vertical plane, and all edges have gradient m apart from a single edge of intermediate

Fig. 7 A non-minimum tree where each Steiner point is locally minimal



gradient, labeled “f” in the figure. Each of the two Steiner points satisfies the conditions of [Corollary 1](#), and it is straightforward to confirm that each Steiner point is locally minimal. In other words, no perturbation of s_1 or s_2 (which fixes the other Steiner point) can reduce the length of T . However, consider the 2-point perturbation \mathbf{v} shown in the figure, where s_1 is perturbed along the line through s_1b away from b , s_2 is perturbed along the line through s_2d toward d , and s_1s_2 remains an m-edge. If L represents Euclidean length, then clearly under this perturbation, $\dot{L}(s_1b \cup s_1s_2 \cup s_2d)_{\mathbf{v}} = 0$, but it is easy to see that $\dot{L}(s_1a \cup s_2c)_{\mathbf{v}} < 0$ and hence that \mathbf{v} is a strictly length-reducing perturbation. A similar argument always applies unless s_1a and s_2c are parallel, that is, have identical gradient.

This idea that the directions of edges at a single Steiner point must propagate throughout the full component if there is no length-reducing perturbation for any pair of Steiner points is a concept that has been shown to also apply to metrics other than the gradient metric. It is well known that it is true for all smooth metrics [\[29\]](#). More recently, it has been shown that it also applies for all metrics where the unit ball is a simple polygon whose vertices lie on the Euclidean unit circle. Such a metric is referred to as a *fixed orientation* metric; the terminology comes from the observation in [\[75\]](#) and [\[76\]](#) that if D is the set of orientations (or *legal directions*) corresponding to the vertices of the polygon, then for any pair of points in the Euclidean plane, there exists a path composed of a pair of line segments with adjacent legal directions so that the Euclidean length of the path equals the distance between the points under the fixed orientation metric. In a minimal FST under such a metric containing at least two Steiner points, each Steiner point has degree 3, and hence the incident edges at that Steiner point use a set of up to six legal directions in their representations as Euclidean embeddings. This set of legal directions is known as the *direction set* of the Steiner point. It has been shown in [\[18\]](#) (for λ -geometry Steiner trees) and in [\[9\]](#) (for Steiner trees under more general fixed orientation metrics) that for any set of terminals, there exists a minimum Steiner tree such that in each full component all Steiner points use only a single direction set, and hence the full component itself uses at most six legal directions. In a similar manner to the proof of [Theorem 2](#), the proofs of these results are based on showing

that if two adjacent Steiner points have different direction sets then there exists a length-reducing perturbation on the FST.

For fixed orientation metrics, the usefulness of knowing that an FST uses only a single direction set lies in the fact that there are only a relatively small number of direction sets that need to be considered; in fact, if there are σ legal orientations (or equivalently σ vertices on the polygonal unit ball), then the number of candidate direction sets that need to be considered is $\Theta(\sigma)$, and these can be identified in $\Theta(\sigma)$ time [9]. For the gradient metric, however, the “direction set” at each Steiner point includes an unknown intermediate direction k which belongs to the uncountable set $(-m, m)$. This potential difficulty was resolved in [10] where it was shown that for any m -constrained minimal FST, there is a simple formula for computing the gradient of the intermediate edge in terms of the coordinates of the terminals and information about the topology of the tree. The correct statement of this result requires the following definition.

A full Steiner topology, each of whose edges has been assigned with the labels m , $-m$, or k with the property that the three edges meeting at a Steiner point all have different labels, is referred to as a *labeled Steiner topology*. A full m -constrained minimal Steiner tree is said to have a given labeled topology if there exists a real number k with $|k| < m$ such that the gradient of each edge corresponds to the label on that edge. It can be shown, using the threshold technique developed by Hwang and Weng [39], that m -constrained minimum Steiner trees with any labeled topology can occur.

Theorem 3 *Let T be a full m -constrained minimal Steiner tree containing an edge of gradient k where $|k| < m$ at every Steiner point. Then given the terminal set and labeled Steiner topology of T , the gradient k can be computed in linear time. Hence, the locations of all Steiner points of T can also be computed in linear time.*

The details of how to compute k and the correctness of the result are given in [10]. The locating of the Steiner points is then a simple recursive procedure based on determining (via the labeled Steiner topology) the position of each point that has two neighboring nodes in known locations.

3.2 Computational Complexity

The properties discussed in the previous section essentially give an efficient way of solving the two-dimensional gradient-constrained Steiner problem for a given Steiner topology. In practice, it should be possible to use these properties as part of a Geosteinertype algorithm to solve the general problem in a scalable way for most randomly distributed sets of terminals. In this section, however, it will be shown that the general problem is NP-complete, which suggests that there will always be families of instances for which any exact solution algorithm is unable to scale efficiently.

This result is not surprising, given that both the rectilinear and Euclidean Steiner problems are NP-hard, as shown in [32] and [33]. Indeed, it is possible to prove a somewhat stronger result, namely, that the (discretized) two-dimensional gradient-constrained Steiner problem is NP-complete even when the terminals are restricted to lying on two vertical lines. This restriction is an interesting one as it has been shown by Aho et al. [1] that imposing it on the rectilinear Steiner problem allows one to find a linear time algorithm, whereas Rubinstein et al. [55] have shown that the discretized Euclidean Steiner problem is still NP-complete under this restriction.

The formal decision version of the problem is as follows:

The Vertical Line Gradient-Constrained Steiner Problem

- *Instance:* A set of points $N = \{p_1, \dots, p_k\}$ contained in two vertical lines, a gradient bound m satisfying $0 < m \leq 1$ and a real number l .
- *Question:* Is there an m -constrained Steiner tree T with terminals N and length at most l ?

As discussed in [33], there are difficulties in describing the complexity of problems such as this caused by the necessity of computing with irrational numbers. A method of circumventing these difficulties is to introduce the discretized Euclidean metric \mathbf{d}' , where $\mathbf{d}'(a, b) = \lceil \mathbf{d}_E(a, b) \rceil$. For the discretized vertical line gradient-constrained Steiner problem, the vertices of the tree are restricted to points with integer coordinates, and the lengths of edges in the tree (or the lengths of segments, in the case of bent edges) are replaced by their discretized Euclidean lengths.

Theorem 4 *The discretized vertical line gradient-constrained Steiner problem is NP-complete.*

This theorem was first proved in [10] using a similar strategy to that devised for the Euclidean Steiner problem in [55]. A somewhat simpler, more constructive proof of both of these complexity results was given in [12].

The basic strategy of the proof is to show that solving a specific instance of the discretized vertical line gradient-constrained Steiner problem depends on being able to solve the subset sum problem, that is, the problem of deciding for any given subset $S = \{d_1, \dots, d_n\}$ of integers and integer s whether there exists a subset $J \subseteq S$ such that $\sum_J d_i = s$. The subset sum problem is known to be NP-complete, from which the theorem follows. The key to the construction of the instance encoding the subset sum problem is to place closely spaced triples of terminals on one of the vertical lines in such a way that in a gradient-constrained minimal Steiner tree, each triple will have a Steiner point adjacent to two of the terminals, and there will be a bent edge connecting the third terminal to one of that pair. For each triple, there is a choice as to which pair of terminals has an adjacent Steiner point. The locations of all terminals are contrived so that in the minimum Steiner tree, all edges with intermediate gradient are horizontal, and finding a choice of connection to each triple that makes all intermediate gradient edges horizontal is equivalent to solving the subset sum problem. This construction can easily be done in polynomial time from any instance of the subset sum problem, which completes the proof. The details of the construction and proof of correctness can be found in [12].

4 The Three-Dimensional Gradient-Constrained Steiner Problem

The *three-dimensional gradient-constrained Steiner problem* can be stated as follows:

- *Given:* A set of points $N = \{p_1, \dots, p_k\}$ in Euclidean three-space and a gradient bound satisfying $0 < m \leq 1$
- *Find:* A network T interconnecting N of minimum total length, where the length of each edge e is computed using the gradient metric (with gradient bound m)

A tree solving the three-dimensional gradient-constrained Steiner problem for some given set of terminals will be referred to as a *gradient-constrained minimum Steiner tree*.

Moving from two to three dimensions significantly increases the inherent difficulty in solving Steiner problems. The three-dimensional Euclidean Steiner tree problem, for example, is known to be considerably more difficult than the planar version. Some of the basic properties of such trees have been studied in [35] and [58]. The three-dimensional gradient-constrained Steiner problem is even more complicated. Despite the high levels of complexity involved in solving the Steiner problem, it is nevertheless important to study the properties of exact minimum Steiner trees. In higher-dimensional cases, such properties have proved crucial in the development of approximation algorithms. For example, Smith's approximation algorithm for d -dimensional Steiner trees in [58] uses the fact that all angles at Steiner points in exact minimum Steiner trees are $2\pi/3$.

This section is a survey of the most important properties of gradient-constrained minimum Steiner trees, beginning with the fundamental properties of their Steiner points.

4.1 Properties of Steiner Points

The results in this section are mostly from [13]. Let T be a gradient-constrained minimum Steiner tree. It can be assumed that all edges of T are straight lines whose lengths are given by the gradient metric. Let $|T|_g$ be the sum of the lengths of all edges in T . An edge pq in T is called an *f-edge*, *m-edge*, or *b-edge* if pq is labeled “f” ($g(pq) < m$), “m” ($g(pq) = m$), or “b” ($g(pq) > m$) respectively. The label of an edge can be thought of as indicating which of the two metrics that compose the gradient metric is “active” for that edge, with an “m” label indicating that both metrics hold simultaneously.

A very useful tool for understanding properties of Steiner points in a gradient-constrained minimum Steiner tree is the variational method, which was developed by Rubinstein et al. in [54]. The variational argument in the Steiner tree problem is as follows: for a minimum tree T , the directional derivative of $|T|_g$ is greater than or equal to zero when its Steiner points are perturbed in any directions. Note that under an arbitrarily small perturbation, the only edges which can change labeling are

m-edges. If no m-edges change their labeling under a given perturbation, then it is easily checked that the variation is reversible, and hence the directional derivative is strictly zero. Suppose $e = sa$ is an edge in T and s is a Steiner point which is perturbed to s' in direction \mathbf{u} . Let $\dot{e}_{\mathbf{u}}$ (or simply \dot{e} if \mathbf{u} is known) denote the directional derivative of the length of e . It is easy to show the following lemma and corollary:

Lemma 2

- (i) If e is an f-edge, then $\dot{e}_{\mathbf{u}} = -\cos(\angle ass')$.
- (ii) If e is a b-edge, then $\dot{e}_{\mathbf{u}} = -\cos(\angle zss')\sqrt{1+m^{-2}}$ where z is a point on the vertical line through s such that $\angle asz \leq \pi/2$.
- (iii) If e is an m-edge, then $\dot{e}_{\mathbf{u}}$ is equal to either $-\cos(\angle ass')$ or $-\cos(\angle zss')\sqrt{1+m^{-2}}$, depending on whether $g(s'a) \leq m$ or $g(s'a) > m$.

Corollary 2

- (i) When s moves horizontally, the length of e does not change if e is a b-edge or if e is an m-edge and becomes a b-edge in the move.
- (ii) When s moves vertically to the same side (or the opposite side) of the horizontal plane through s as a , e becomes shorter (or, respectively, longer) regardless of the gradient of e .

From Lemma 2, it is clear that as in the Euclidean metric, the directional derivative of e in the gradient metric is determined only by the direction of \mathbf{u} and is independent of the length of sa .

Because an f-edge is still an f-edge under a small perturbation, Lemma 2 gives the following easy corollary:

Lemma 3 Any f-edge of a Steiner point s meets other edges incident to s at an angle no less than $\pi/2$.

An important application of the variational argument is that of splitting a small angle in a nonoptimal tree. If a Steiner point s of T is perturbed in direction \mathbf{u} , denote the directional derivative of $|T|_g$ by $\dot{T}_{\mathbf{u}}$ or just \dot{T} . Now, suppose s is a Steiner point joining a, b , and c in a Euclidean Steiner tree $T = sa \cup sb \cup sc$. If $\angle acb \geq 2\pi/3$, then s collapses into c . On the other hand, if $\angle acb < 2\pi/3$ but $s = c$, then T cannot be minimum by the following variational argument: let s_1 be a point on the bisector \mathbf{u} of $\angle asb$ and close to s , then $\angle ass_1 = \angle bss_1 < \pi/3$ and $\dot{T} = 1 - \cos(\angle ass_1) - \cos(\angle bss_1) < 0$. In general, this process is referred to as *splitting* the angle at s in the subtree $sa \cup sb$ along a vector \mathbf{u} .

An example of the application of this technique to gradient-constrained minimum Steiner trees comes from understanding when a Steiner point in such a tree can have an incident b-edge. For any point p , denote the horizontal plane through p by \mathcal{H}_p . For simplicity, a point or an edge will be said to be above (or below) p if it is above (or below) \mathcal{H}_p . It is also useful to follow the convention of saying that two edges incident with s lie on the same side of \mathcal{H}_s if they lie in the same closed half-space determined by \mathcal{H}_s ; that is, this includes the possibility that one or both edges lie on \mathcal{H}_s . The two edges are said to be on different sides of \mathcal{H}_s only if their interiors lie in different open half-spaces (determined by \mathcal{H}_s).

Lemma 4 *If sa is a b-edge in a gradient-constrained minimum Steiner tree T , then no other edge incident with s lies on the same side of \mathcal{H}_s as a . Moreover, all other edges incident with s are m-edges.*

Proof Let sb be another edge of T incident with s . Let \mathbf{u} be a vector with gradient m , perturbing s to s' , such that s' lies on the same side of \mathcal{H}_s as a and the angle $\theta = \angle s'sb$ is as small as possible. Applying the variational argument to T by splitting the angle $\angle asb$ at s along \mathbf{u} , note that $\dot{T} = \dot{s}b_{\mathbf{u}}$. If sb is a b-edge lying on the same side of \mathcal{H}_s as a , then $\dot{T} < 0$ by Lemma 2 (ii). If, on the other hand, sb is an m-edge or f-edge on the same side of \mathcal{H}_s as a , or sb is an f-edge on the opposite side of \mathcal{H}_s to a , then $\theta < \pi/2$ (since $m \leq 1$ in the latter case). So again, $\dot{T} < 0$ by Lemma 2. In each case, there is a contradiction to the minimality of T .

Finally, if sb is a b-edge lying on the opposite side of \mathcal{H}_s to a , then s has only two incident edges by the above argument and s is not a Steiner point. \square

A close analysis, using a similar approach, for any Steiner point in T , gives the following lemma, which is proved in [13].

Lemma 5 *If s is a Steiner point in a gradient-constrained minimum Steiner tree T , then there are at most two incident edges lying strictly above (or below) \mathcal{H}_s . Consequently, the degree of any Steiner point is either three or four.*

Suppose s is a degree 3 Steiner point in T with two incident edges sa, sb lying on one side of \mathcal{H}_s and the third sc lying on the other side of \mathcal{H}_s . (This includes the possibility that all three edges lie on \mathcal{H}_s .) Let g_a, g_b, g_c denote the respective labels of these edges. Then the labeling of this degree 3 Steiner point is denoted by (g_ag_b/g_c) . By symmetry, it can be assumed that a and b both lie on or above \mathcal{H}_s . The theorem below shows that of the 18 possible distinct labelings for a degree 3 point, only a small number can occur in a gradient-constrained minimum Steiner tree.

Theorem 5 *If s is a degree 3 Steiner point in a gradient-constrained minimum Steiner tree T , then up to symmetry, there are five feasibly optimal labelings: (ff/f), (ff/m), (fm/m), (mm/m), and (mm/b).*

Proof By Lemmas 4 and 5, the only possible labelings of s , other than those listed in the statement of the theorem, are (mm/f) and (fm/f). So suppose, contrary to the theorem, there is only one edge, the f-edge sc , lying below s , and there is an m-edge, say, sa , lying above s . Because $g(sa) > g(sc)$, sa shrinks strictly faster than sc stretches when s moves vertically upward. Since the third edge, sb , lies on the same side of \mathcal{H}_s as a and $\dot{s}b \leq 0$ under this move, it follows that $\dot{T} < 0$, contradicting the minimality of T . \square

Now, suppose s is a degree 4 Steiner point in T with two edges sa, sb lying on one side of \mathcal{H}_s and the other two edges sc, sd lying on the other side of \mathcal{H}_s . Let g_a, g_b, g_c, g_d be the respective labels of these edges. Then the labeling of s is denoted (g_ag_b/g_cg_d) .

For degree 4 Steiner points, it can be shown, by a somewhat more sophisticated use of the variational approach, that there is only one feasibly optimal labeling. The proof of this result is highly technical and again can be found in [13]. Note that here a stronger bound on the maximum gradient m is required, but as with the previous bound, it is one that is usually easily satisfied in applications.

Theorem 6 *If s is a degree 4 Steiner point in a gradient-constrained minimum Steiner tree T and if $m < 0.38$, then the labeling of s is (mm/mm).*

By [Theorems 5](#) and [6](#), there are six possible labelings at a Steiner point when $m < 0.38$. The two theorems below use the variational argument to show that for each labeling, it is possible to uniquely locate the Steiner point s in the gradient-constrained minimum Steiner tree T in terms of the adjacent vertices of T . Suppose the edges incident with s are sa, sb, sc (or sa, sb, sc, sd if s is of degree 4), where s has labeling (g_ag_b/g_c) (or (g_ag_b/g_cg_d) if s is of degree 4). For any point p , let \mathcal{C}_p denote the cone generated by rotating a line through p with gradient m about the vertical line through p . So \mathcal{C}_p is a right vertical cone whose vertex is p .

The results for a Steiner point of degree 3 are as follows.

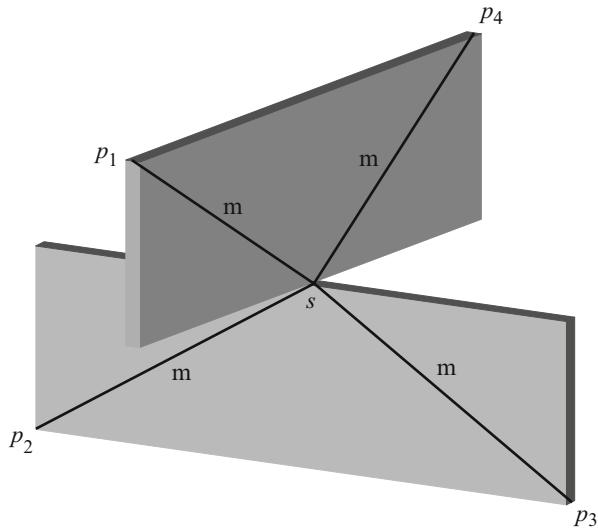
Theorem 7 *Let s be a degree 3 Steiner point in a gradient-constrained minimum Steiner tree T .*

- (i) *If s has labeling (ff/f), then s is the point on the plane through a, b, c such that $\angle asb = \angle bsc = \angle csa = 2\pi/3$.*
- (ii) *If s has labeling (mm/b), then s is a point at the intersection of $\mathcal{C}_a, \mathcal{C}_b$ and the vertical plane through b, c .*
- (iii) *If s has labeling (mm/m), then s is a point at which the three cones $\mathcal{C}_a, \mathcal{C}_b$, and \mathcal{C}_c all intersect, and $\angle asb \geq \arccos(-1 + m^2)/2 \geq \pi/2$.*
- (iv) *Suppose s has labeling (ff/m), and let \mathbf{v} be a horizontal vector tangent to \mathcal{C}_c at s . Then s is a point on \mathcal{C}_c such that $\angle(\mathbf{v}, \vec{sa}) = \angle(-\mathbf{v}, \vec{sb})$ and $\cos \angle(\vec{cs}, \vec{sa}) + \cos \angle(\vec{cs}, \vec{sb}) = 1$.*
- (v) *If s has labeling (fm/m), then s is a point lying on $\mathcal{C}_b \cap \mathcal{C}_c$, which is an ellipse, such that sa is perpendicular to this ellipse.*

Proof If the labeling at s is (ff/f), then it is clear that the location of s is the same as in the unconstrained three-dimensional Steiner problem, proving Case (i). For the remaining labelings, one can employ the variational argument. Since any vector in space can be decomposed into a sum of three noncoplanar components, to prove s can be a Steiner point of a minimum tree T , by the variational argument, it is sufficient to show that $\dot{T} \geq 0$ for perturbations of s in three noncoplanar directions. For Case (ii), it follows from [Lemma 2](#) that $\dot{T} \geq 0$ when s moves either up or down or in any horizontal direction. This proves Case (ii). The other cases can be proved by similar arguments; see [13] for details. \square

Finally, consider the case where s has degree 4. If, in this case, the edges incident with s can be partitioned into two pairs, such that each pair lies in a vertical plane through s and has the property that the projections of the two edges onto \mathcal{H}_s meet

Fig. 8 A degree 4 Steiner point with bi-vertically coplanar incident edges



at an angle of π at s , then the edges of s are said to be *bi-vertically coplanar* (see Fig. 8). Note that if two m -edges incident with s lie on opposite sides of \mathcal{H}_s and lie in the same vertical plane, then those edges are collinear. The other, noncollinear possibility is illustrated in Fig. 8. The following theorem not only shows how to construct a Steiner point of degree 4 but also shows that the existence of such a point requires a very specific configuration for the neighboring nodes. The proof is given in [13].

Theorem 8 *Let s be a degree 4 Steiner point (with labeling (mm/mm)) in a gradient-constrained minimum Steiner tree T . Then s is a point at the intersection of the four cones $\mathcal{C}_a, \mathcal{C}_b, \mathcal{C}_c, \mathcal{C}_d$, and furthermore the edges incident with s are bi-vertically coplanar.*

This work has been further extended in [21], where it is shown that a degree 3 or degree 4 locally minimal Steiner point in a gradient-constrained tree in 3-space is unique with respect to its adjacent nodes, even though the gradient metric is not strictly convex. More recently, Thomas and Weng [61] have derived explicit formulae for computing minimal Steiner points with given labelings in a gradient-constrained tree.

4.2 Local Minimality for a Fixed Topology

The characterization of Steiner points in the previous section provides a useful first step in solving the three-dimensional gradient-constrained Steiner problem with a given topology. For a fixed topology, the two-dimensional Euclidean Steiner problem has an efficient polynomial-time solution based on the methods of Melzak [48],

and similarly, it should be possible to efficiently solve the two-dimensional gradient-constrained Steiner problem using the methods in Sect. 3.2. In three-dimensional space, the problem of solving the Euclidean Steiner problem for a fixed topology is much more difficult. It has been shown in [56] that even on four terminals, a minimum Euclidean Steiner tree in 3-space cannot generally be solved by radicals. The same arguments also apply to minimum gradient-constrained Steiner trees in 3-space. This suggests that iterative methods are required to find good approximations of the solution for the general problem.

To solve any minimization problem, one can attempt a descent method. This works well for the problem at hand if the gradient constraint is relaxed. In particular, it is well known that for the analogous problem in d -dimensional Euclidean space, the network is minimal (for a given topology) if and only if there is no arbitrarily small perturbation of a single Steiner point that reduces the length of the network (see, e.g., [58]). This will be expressed by saying that if the network is minimal with respect to one-point moves, then it is minimal, or, more briefly, that one-point moves suffice.

For the gradient-constrained minimum Steiner tree problem, given terminals and a topology, one can start with an initial nonoptimal tree and then iteratively move toward a global minimum by moving Steiner points. By the convexity of the gradient metric, it follows that such a strategy will work, in theory, if one can determine the appropriate descent directions in which to move the Steiner points. This involves finding length-reducing perturbations of the positions of the Steiner points. One-point moves do not suffice for this problem, essentially due to the non-differentiability of the metric. This poses a difficulty algorithmically, and it helps to know which simultaneous multi-point moves might be required. One of the key results from [22] shows that, apart from a few easily classified exceptions, one-point moves and two-point moves suffice; that is, if the tree is not minimal for the given topology, then there exists a length reducing perturbation that fixes all but one or two Steiner points. This makes the task of finding suitable length-reducing perturbations much simpler.

There are two situations where one-point moves or two-point moves may not suffice, that is, where a consideration of all possible perturbations of one or two Steiner points is not sufficient for finding a descent direction from a non-minimal configuration. The first is where the tree contains an *extreme point*, which is defined to be a degree 3 Steiner point whose incident edges all have the maximum permitted gradient and all lie in a single vertical plane. In this case, a special length-reducing perturbation involving arbitrarily many points may be required; however, these perturbations are easily described. The other situation is where the tree degenerates to include zero-length edges. This case is very difficult to deal with in theory but does not seem to cause a problem in practical applications. Both of these situations are described in more detail later in this section.

The results in this subsection mainly come from [22]. Before discussing these results in a little more detail, it is important to be precise about what is meant here by a perturbation and the change of length under perturbations. A perturbation \mathbf{v} on T is a collection of vectors in three-dimensional Euclidean space, one for

each Steiner point of T . It is assumed that all Steiner points smoothly move with velocities specified by a given perturbation, resulting in a change of the length of T .

Perturbations are used to examine arbitrarily small alterations in the positions of the Steiner points of T and the consequential change in the total length of T . Note that the rate of change in length of T depends only on the directions and relative lengths of the vectors at the Steiner points – the length of each individual vector can be assumed to be arbitrarily small.

If T consists of a single edge uw , then the length of T with respect to the gradient metric is denoted by $L(T) = L(uw)$, and the derivative of this length with respect to a perturbation \mathbf{v} is denoted by $\dot{L}(uw)_{\mathbf{v}}$. It is also useful to define the active metric for a perturbation. Suppose $L(uw) \neq 0$. If uw is a b-edge or an f-edge, then the active metric for \mathbf{v} is the same as that of the edge. If uw is an m-edge, then the active metric for \mathbf{v} is defined to be b if the derivative of the gradient of uw is ≥ 0 and f if the derivative of the gradient is ≤ 0 (and hence both b and f if the derivative of the gradient equals 0). This is equivalent to saying that the active metric for the perturbation is the same as the metric active on the projection of the image of uw (after the perturbation has been applied) onto the vertical plane containing uw .

Note that although a perturbation can deactivate a metric on an m-edge, it cannot activate an inactive metric, since the perturbations are considered to be arbitrarily small.

For a larger tree T with edge set E , the derivative of $L(T)$ with respect to \mathbf{v} is defined to be

$$\dot{L}(T)_{\mathbf{v}} = \sum_{uw \in E} \dot{L}(uw)_{\mathbf{v}|_{\{u,w\}}},$$

where, as usual, $\mathbf{v}|_{\{u,w\}}$ is the perturbation that acts in the same way as \mathbf{v} on u and w and acts trivially on all other nodes of T . Since the terminals of a Steiner tree have fixed predetermined positions, it is assumed that all perturbations act trivially on terminals.

The convexity of the gradient metric immediately implies the following result.

Lemma 6 *Let T be a gradient-constrained Steiner tree with topology t . Then T is minimum with respect to t if and only if $\dot{L}(T)_{\mathbf{v}} \geq 0$ for every perturbation \mathbf{v} .*

As was illustrated in Fig. 7, Sect. 3.1, the existence of a perturbation \mathbf{v} on the Steiner points of T which decreases $L(T)$ does not imply that there exists a 1-point perturbation that decreases the length of T . It is helpful to examine why, in the example in Fig. 7, \mathbf{v} cannot be decomposed into 1-point perturbations on the original tree. Let \mathbf{v}_1 be the action of this perturbation on s_1 and \mathbf{v}_2 the action on s_2 . The question is why $\dot{L}(T)_{\mathbf{v}} \neq \dot{L}(T)_{\mathbf{v}_1} + \dot{L}(T)_{\mathbf{v}_2}$. The reason lies in the fact that the only active metric of s_1s_2 is the f-metric for \mathbf{v}_1 and the b-metric for \mathbf{v}_2 .

If, in the above example, s_1s_2 had a common active metric for both \mathbf{v}_1 and \mathbf{v}_2 , then there would exist a length-reducing 1-point perturbation. This would have occurred, for example, if the label of the edge s_1s_2 had been f or b.

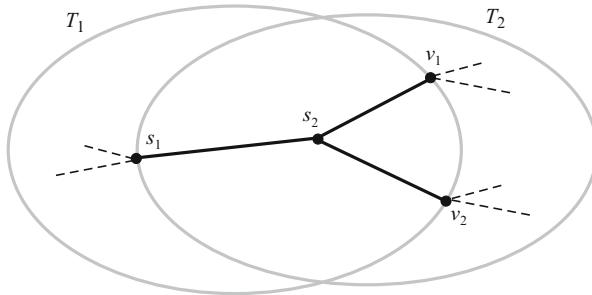


Fig. 9 The ellipses indicate the two subtrees T_1 and T_2 used for determining whether or not the edge s_1s_2 is decomposing with respect to s_2

Hence, it is evident that it is m-edges which cause difficulties when trying to decompose a length-reducing perturbation into a sum of 1-point perturbations because of the possibility of active metrics being changed in the decomposition. However, if 2-point perturbations are included in the decomposition, the problem of m-edges can often be overcome.

Let T be a full gradient-constrained Steiner tree with no zero-length edges. Suppose T has an interior node s_2 , whose three adjacent nodes are s_1 , v_1 , and v_2 (each of which may or may not be a Steiner point). Let U_1 and U_2 be the two connected components of $T - s_1s_2$ containing s_1 and s_2 , respectively. Let $T_1 = T|_{V(U_1) \cup \{s_2, v_1, v_2\}}$ and let $T_2 = T|_{V(U_2) \cup \{s_1\}}$. This is illustrated in Fig. 9. Note that s_2 is an interior node of both T_1 and T_2 . It is possible that a perturbation \mathbf{v} may decrease $L(T)$ but increase both $L(T_1)$ and $L(T_2)$. The edge s_1s_2 is said to be *decomposing* with respect to s_2 if for any perturbation \mathbf{v} acting on the interior nodes of T , there exist perturbations \mathbf{v}_1 and \mathbf{v}_2 acting on the interior nodes of T_1 and T_2 , respectively, such that

$$\dot{L}(T)\mathbf{v} = \dot{L}(T)\mathbf{v}_1 + \dot{L}(T)\mathbf{v}_2.$$

Furthermore, s_1s_2 is said to be *properly decomposing* with respect to s_2 if s_1s_2 is decomposing with respect to s_2 and $T_1 \neq T$ and $T_2 \neq T$ (i.e., s_1 is not a terminal, and at least one of v_1 and v_2 is not a terminal). The following lemma shows that if all interior edges in T are properly decomposing, then only 1- and 2-point perturbations need to be considered, when attempting to determine whether or not T is minimal.

Lemma 7 *Suppose every interior edge of full gradient-constrained Steiner tree T is properly decomposing with respect to at least one of its endpoints. If T is minimal with respect to all 1- and 2-point perturbations, then T is minimal.*

Proof Suppose, on the contrary, that T is not minimal. Then, by Lemma 6, there exists a perturbation \mathbf{v} on T , such that $\dot{L}(T)\mathbf{v} < 0$. The result follows by a straightforward induction argument on the number of Steiner points acted on (non-trivially) by \mathbf{v} , giving the existence of a length reducing 1- or 2-point perturbation. \square

Define a degree 3 Steiner point s in T to be an *extreme point* if s is labeled (mm/m) and the three edges incident with s all lie in a vertical plane.

Theorem 9 *Let T be a full gradient-constrained Steiner tree, minimal with respect to 1- and 2-point perturbations, and containing no extreme points or zero-length edges. Then T is minimal (with respect to its topology).*

Proof A geometric argument, given in [22], shows that if T contains no extreme points, then each interior edge of T is decomposing with respect to both endpoints. If T contains at least three Steiner points, then each interior edge e is incident with a Steiner point s_e which is adjacent to a third Steiner point (not an endpoint of e); thus e is properly decomposing with respect to s_e . Hence, by Lemma 7, T is minimal. On the other hand, if T contains less than three Steiner points, then T is minimal by assumption. \square

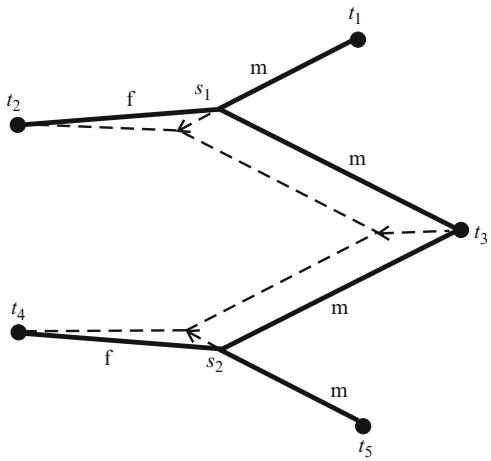
Let T_{ext} denote the subnetwork of T induced by extreme points. In other words, T_{ext} is a forest whose nodes are the extreme points of T and whose edges are the edges of T whose endpoints are both extreme. Note that each connected component of T_{ext} lies in a vertical plane. In order to understand the structure of T if it contains extreme points, one can define a class of easily identified perturbations known as backbone moves and show that T is minimal if and only if none of these moves is length reducing.

Definition Let P be a connected component of T_{ext} with node set $V(P)$, containing $k - 2$ nodes ($k \geq 3$). It is straightforward to show that P is a path in a vertical plane \mathcal{P} whose nodes are collinear. Let s_1 and s_k be the two nodes of T not in P that are neighbors of the two endpoints of P and are collinear with P (or are the collinear neighbors of the single node of P if $k - 2 = 1$). Then $T|_{V(P) \cup \{s_1, s_k\}}$ is defined to be the *backbone* of P (i.e., the backbone is the union of P and the two extra collinear edges of T incident with the endpoints of P). A *k -point backbone move* is a perturbation of $V(P) \cup \{s_1, s_k\}$ restricted to \mathcal{P} such that the active metric on all edges in or incident with P remains m; each node of $V(P)$ is perturbed an appropriate relative distance directly toward or away from its neighboring terminals. Hence the backbone remains collinear, if the perturbation is applied.

Note that a k -point backbone move is determined entirely by its action on s_1 and s_k . Finding a length-reducing k -point backbone move is no more complex than finding a length-reducing 2-point perturbation. Furthermore, such backbone moves can be shown to exist in some cases where there are no length-reducing 1- or 2-point perturbations.

A careful geometric argument, given in [22], shows that for any non-minimal tree containing no zero-length edges, there exists either a 1- or 2-point perturbation or a k -point backbone move that is length reducing.

Fig. 10 A degenerate tree minimal with respect to 1- and 2-point perturbations but not with respect to 3-point perturbations



Theorem 10 Let T be a gradient-constrained Steiner tree, minimal with respect to 1- and 2-point perturbations, and containing no zero-length edges. If T is not minimal then there exists a length-reducing k -point backbone move on T , for some $k \geq 3$.

The remaining case to consider is where T contains zero-length edges. There are a number of reasons why it is sensible to permit zero-length edges for a given topology: it ensures a minimum network exists for each topology, it is necessary to guarantee compactness of the configuration space for a fixed topology, and it means that all topologies can essentially be dealt with by considering only full topologies. For instance, some minimal Steiner trees have a Steiner vertex of degree 4, and this can be considered instead as a degenerate example of a Steiner tree whose topology has all Steiner points of degree 3 but where two such points have an edge of length zero joining them.

However, if the restriction on no zero-length edges is relaxed, then [Theorem 10](#) no longer holds. This is illustrated by the following example.

Consider a gradient-constrained tree T in the vertical plane with terminals t_1, t_2, t_3, t_4 , and t_5 , as shown in [Fig. 10](#). The tree T contains two Steiner points s_1 and s_2 such that s_1t_2 is an f-edge sloping downward from s_1 , s_2t_4 is an f-edge sloping upward from s_2 , and the remaining edges are m-edges. Now T can be viewed as a full (but degenerate) tree containing a third Steiner point s adjacent to s_1, s_2 , and t_3 , such that st_3 is a zero-length edge. It is easy to see that T is minimal with respect to 1- and 2-point moves. However, there is a 3-point move, moving s_1 along the line through s_1t_1 away from t_1 , moving s_2 along the line through s_2t_5 away from t_5 , and moving s away from t_3 so that s_1s and s_2s remain m-edges that reduces the length of T as the f-edges come closer to being parallel. This move is indicated in the figure in broken lines.

This example does not rely on s being extreme. It is possible to move some of the terminals out of the vertical plane, without changing the edge labels, so that this

problem persists. It is also possible to construct more complicated examples that require the simultaneous perturbation of arbitrarily many Steiner points (not in a vertical plane) in order to reduce the length of the tree.

4.3 Algorithms and Software Solutions

The importance of solving the three-dimensional gradient-constrained Steiner problem for a fixed topology is that it forms a key step toward the solution of the general problem of finding the minimum network joining a given set of nodes, without specifying the network topology. From the example at the end of the previous subsection, it appears that for the case of zero-length edges, characterizing the constraints on Steiner trees that are minimal with respect to the gradient metric, and having a given topology, is a difficult and probably intractable problem. However, this is not necessarily a major setback for constructing an effective approximation algorithm that seeks to find the minimum gradient-constrained Steiner tree. An example of such a software tool is UNO (Underground Network Optimiser), which makes use of the theoretical results in the previous two subsections. A description of an early version of UNO is given in [11], and a computational study using UNO is presented in [62].

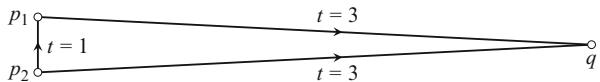
The problem of finding the best topology is in general NP-complete, since this is true for the same problem restricted to the horizontal plane. Hence, any practical algorithm requires a method of changing topologies, or at least considering many different topologies, that is unlikely to be highly efficient. UNO incorporates this aspect by including some random steps. By running for some time and repeatedly randomly encountering approximately minimal trees, the algorithm occasionally finds trees that are reasonably close to the minimal tree, even in the case of the example illustrated in Fig. 10. A sensible descent method (using the results in the previous section characterizing minima with no zero-length edges) then brings the algorithm even closer to the true global minimum. The characterization of local minima described in the previous section appears to be enough for the algorithm to perform well in practical cases.

In UNO, the problem of finding a “good” topology, for instance, the topology of the minimum tree, is tackled via a heuristic method based on simulated annealing and a genetic algorithm. In this way, the problem is reduced to one of efficiently minimizing the cost for a single topology, using descent methods.

5 The Gilbert Problem in Minkowski Spaces

Gilbert [34] proposed a generalization of the Euclidean Steiner problem incorporating into the network cost a flow between terminals. This section considers the extension of this concept to networks in general Minkowski spaces. Much of the discussion here is based on [67].

Fig. 11 An example where split routing is cheaper



Let T be a network interconnecting a set $N = \{p_1, \dots, p_n\}$ of n terminals in a Minkowski space. For each pair p_i, p_j , $i \neq j$ of terminals, a nonnegative flow $t_{ij} = t_{ji}$ is given. The cost of an edge e in T is $w(t_e)l_e$, where l_e is the length of e , t_e is the total flow being routed through e , and $w(\cdot)$ is a unit cost weight function defined on $[0, \infty)$ satisfying

$$w(0) \geq 0 \quad \text{and} \quad w(t) > 0 \text{ for all } t > 0, \quad (1)$$

$$w(t_2) \geq w(t_1) \quad \text{for all } t_2 > t_1 \geq 0, \quad (2)$$

$$w(t_1 + t_2) \leq w(t_1) + w(t_2) \quad \text{for all } t_1, t_2 > 0, \quad (3)$$

$$w(\cdot) \text{ is a concave function.} \quad (4)$$

That the function w is concave means by definition that $-w$ is convex. A network with a cost function satisfying Conditions (1)–(3) is called a *Gilbert network*. For a given edge e in T , $w(t_e)$ is called the *weight* of e and is also denoted simply by w_e . The *total cost* of a Gilbert network T is the sum of all edge costs, that is,

$$C(T) = \sum_{e \in E} w(t_e)l_e$$

where E is the set of all edges in T . A Gilbert network T is a *minimum Gilbert network*, if T has the minimum cost of all Gilbert networks spanning the same point set N , with the same flow demands t_{ij} and the same cost function $w(\cdot)$. By the arguments of [26], a minimum Gilbert network always exists in a Minkowski space when Conditions (1)–(4) are assumed for the weight function.

Conditions (1)–(3) above ensure that the weight function is nonnegative, non-decreasing, and triangular, respectively. These are natural conditions for most applications. Unfortunately, the first three conditions alone do not guarantee that a minimum Gilbert network is a tree. This can be seen by considering the following example of a Gilbert network problem with two sources and one sink in the Euclidean plane, where there exists a split-route flow that has a lower cost than any arborescence.

For this example, assume there are two sources p_1, p_2 and a sink q which are the vertices of a triangle $\Delta p_1 p_2 q$ with edge lengths $\|p_1 - p_2\|_2 = 1$ and $\|p_1 - q\|_2 = \|p_2 - q\|_2 = 10$, as illustrated in Fig. 11. The tonnages at p_1 and p_2 are 2 and 4, respectively. The weight function is $w(t) = \lceil (3t + 1)/2 \rceil$, that is, $(3t + 1)/2$ rounded up to the nearest integer. This function is positive, nondecreasing, and triangular, but not concave. For the example, only the following values need to be considered:

Fig. 12 The minimum Gilbert arborescence



t	1	2	3	4	6
w(t)	2	4	5	7	10

Routing 1 unit of the flow from p_2 via p_1 to q (and all other flow directly to q) gives a Gilbert network (Fig. 11) of total cost

$$w(1)\|p_1 - p_2\|_2 + w(3)\|p_1 - q\|_2 + w(3)\|p_2 - q\|_2 = 102.$$

For a Gilbert arborescence, the flows from p_1 and p_2 to q are routed via some variable point s as in Fig. 12. A minimum Gilbert arborescence can be calculated by using the weighted Melzak algorithm as described in [34] to correctly locate s . The calculation, which involves a straightforward geometric calculation (see [22]), results in a total cost of $\sqrt{9982.5} + 7\sqrt{3890.25} = 102.074\dots$. This shows that split routing can be necessary when the weight function is not concave.

For the remainder of the chapter, it will be assumed that the weight function w is concave in addition to the other three conditions (in fact, the weight function will generally be linear). When all of Conditions (1)–(4) apply and there is a single sink, it has been shown in [34] and [60] that there always exists a minimum Gilbert network that is a Gilbert arborescence. This means that for the remainder of this chapter, it is only necessary, without loss of generality, to consider MGAs. (Note that in [26], Condition (3), the *triangular condition*, was incorrectly interpreted as concavity of the cost function.)

The *Gilbert network problem* is to find a minimum Gilbert network for a given terminal set N , flows t_{ij} , and cost function $w(\cdot)$. Since its introduction in [34], various aspects of the Gilbert network problem have been studied, with an emphasis generally on discovering geometric properties of minimum Gilbert networks (see [26, 60, 64, 65] and [67]). As in the Steiner problem, additional vertices can be added to create a Gilbert network whose cost is less than would otherwise be possible, and these additional points are again called *Steiner points*. A Steiner point s in T is called *locally minimal* if a perturbation of s does not reduce the cost of T . A Gilbert network is called *locally minimal* if no perturbation of the Steiner points reduces the cost of T .

One of the main results established in [67] is that in a smooth Minkowski plane, that is, a two-dimensional Minkowski space where the given norm is differentiable at any $x \neq o$, every Steiner point in an MGA has degree 3. In Sect. 6, however, it will be shown that for the gradient-constrained Gilbert problem (where the corresponding Minkowski space is not smooth) degree 4 Steiner points can occur.

6 The Two-Dimensional Gradient-Constrained Gilbert Problem

The *Gilbert arborescence problem* asks for a minimum-cost flow-dependent network interconnecting given sources and a unique sink. The gradient-constrained version of the problem in a vertical plane is called the *two-dimensional gradient-constrained Gilbert arborescence problem* and is formulated as follows:

- *Given:* A set of points $N = \{p_1, \dots, p_k\}$ in a vertical plane in Euclidean space, where p_1, \dots, p_{k-1} are sources with respective positive flows t_1, \dots, t_{k-1} and p_k is a unique sink, a gradient bound satisfying $0 < m \leq 1$, and positive constants d and h
- *Find:* A minimum-cost network T interconnecting N which provides flow paths from the sources to the sink, such that the cost of an edge e in T is given by $(d + ht_e)l_e$, where t_e is the total flow through e and l_e is the length of e under the gradient metric

A network satisfying the above conditions is called a *gradient-constrained minimum Gilbert arborescence* (gradient-constrained MGA, for short). Vertices in T not in N are, as before, called *Steiner points*, and as was discussed in Sect. 5, a Steiner point is a Fermat-Weber point with respect to its adjacent vertices in T . The quantity $d + ht_e$ represents the *weight* associated with the edge e .

As in the case of the Steiner problem, there exists a gradient-constrained MGA lying in the vertical plane containing the sources and sink; it will be assumed that the gradient-constrained MGA T lies in this vertical plane.

6.1 Edge Vectors and Fundamental Properties of MGAs

As discussed in Sect. 2.3, Euclidean space equipped with the norm associated with the gradient metric is an example of a Minkowski space. Recall that in a Minkowski space X , the *unit ball* is defined to be the set $B = \{x \in X : \|x\| \leq 1\}$. The *dual space* of X , denoted by X^* , is the vector space of all linear functionals on X , a linear functional being a mapping from X into \mathbb{R}^+ . Identifying X and X^* with \mathbb{R}^n , the *dual ball* B^* is the polar body of B , that is,

$$B^* = \{y : \langle x, y \rangle \leq 1, \quad \forall x \in B\}$$

where $\langle \cdot, \cdot \rangle$ is the dot product.

In a gradient-constrained vertical plane, the unit ball B_g and dual ball B_g^* are as shown in Fig. 13a, b. Note that the diagonal lines (shown dashed) have gradient m and that the shapes of the unit ball and dual ball depend on the value of m .

For any Minkowski space X , a *norming functional* of $x \in X$ is a $\phi \in X^*$ such that $\|\phi\| = 1$ and $\phi(x) = \|x\|$. The set of norming functionals of x is denoted by ∂x . By the Hahn-Banach separation theorem [53], each nonzero $x \in X$ has an associated norming functional. Each ∂x corresponds to an *exposed face* of B^*

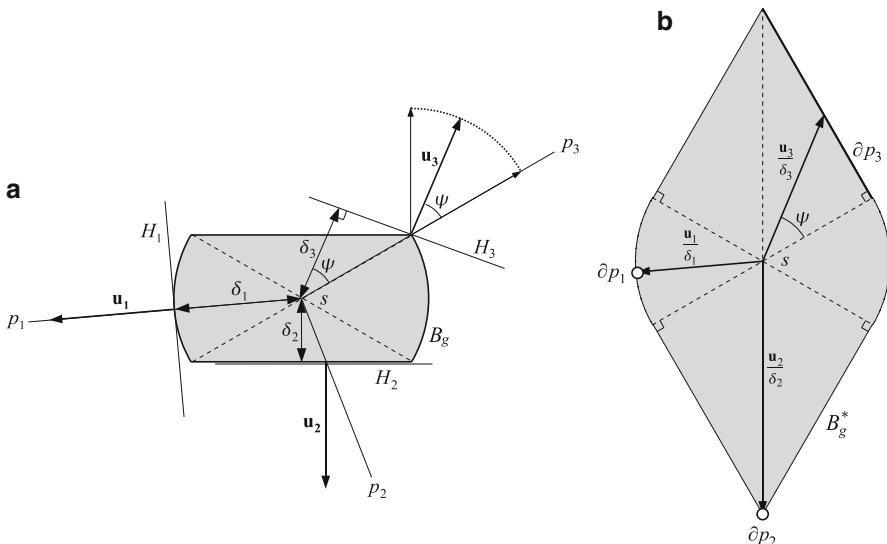


Fig. 13 Edge vectors and outward normal vectors for f-, b- and m-edges in the gradient metric. (a) The unit ball. (b) The dual ball

(i.e., the intersection of B^* with a supporting hyperplane). Furthermore, a norming functional is equivalent to a vector $\frac{\mathbf{u}}{\delta}$, where \mathbf{u} is an outward unit vector normal to the hyperplane supporting $s + B_g$ at the point where the ray from s through p intersects B_g and δ is the distance from s to this hyperplane. These vectors are referred to as *edge vectors*. Edge vectors are important for establishing an equilibrium condition at Steiner points.

In the gradient metric for a vertical plane, the following lemma characterizes vectors for edges connecting a fixed point and a Steiner point (refer to Fig. 13).

Lemma 8 Let ps be an edge with an associated weight w in a gradient-constrained MGA connecting a terminal p and a Steiner point s . Then an edge vector \mathbf{v} corresponding to an element of $w\partial|p-s|_g = w\partial(p-s)$ is computed as follows:

- If sp is an f-edge, \mathbf{v} points from s to p and has length w .
- If sp is a b-edge, then \mathbf{v} points vertically up or down, if p is above or below s , respectively, and has length $w\sqrt{1+m^{-2}}$.
- If sp is an m-edge, then any \mathbf{v} can be selected from the continuum of vectors having all possible directions between an m-edge and a vertical edge. The length of \mathbf{v} is given by $w/\cos\psi$, where ψ is the angle that \mathbf{v} makes with the m-edge.

Note that the possible vectors \mathbf{v} associated with an m-edge all point to one of the straight line segments on the boundary of B^* and the union of all these vectors forms a region bounded by a right-angled triangle.

Edge vectors, along with the variational argument, provide tools for analyzing the basic properties of Steiner points in MGAs. An optimal Steiner point s in a

gradient-constrained MGA is necessarily a Fermat-Weber point with respect to its adjacent points p_1, \dots, p_k . This leads to the following lemma, which follows from the results of Durier and Michelot [30].

Lemma 9 *Let T be a gradient-constrained MGA, and let s be a Steiner point in T with incident edges $p_1s, \dots, p_{k-1}s, sp_k$ having respective weights w_1, \dots, w_k . If s is locally minimal with respect to its adjacent vertices, then there exist edge vectors $\mathbf{v}_1, \dots, \mathbf{v}_k$, computed by Lemma 8, for which*

$$\sum_{i=1}^k \mathbf{v}_i = \mathbf{0}$$

This is referred to as the *equilibrium condition*. Note that the equilibrium condition is necessary but not sufficient for s to be locally minimal with respect to its adjacent vertices. This is in contrast to Fermat-Weber points in the gradient-constrained problem, for which the equilibrium condition is both necessary and sufficient.

The next results require the following definition.

Definition Let T be a gradient-constrained MGA in a vertical plane, and let s be a Steiner point in T . Then the horizontal and vertical lines passing through s are denoted by \mathcal{H}_s and \mathcal{V}_s , respectively.

The local properties of any Steiner point s in a gradient-constrained MGA are contained in the following two theorems and their corollary. These results were established and proved in [66], using the variational argument. Note that these properties depend on the gradient of the sink edge.

Theorem 11 *Let T be a gradient-constrained MGA in a vertical plane, and let s be a Steiner point in T with incident sink edge sp_k . If sp_k is a b-edge, then the degree of s is three, and the two source edges are m-edges on the opposite side of \mathcal{H}_s to sp_k .*

Theorem 12 *Let T be a gradient-constrained MGA in a vertical plane, and let s be a Steiner point in T with incident sink edge sp_k , where sp_k is not a b-edge. Let L denote the infinite line passing through s, p_k , and let H^+, H^- denote the two open half-planes into which L divides the vertical plane, such that H^+ is the upper half-plane if and only if the slope of L is positive. Then:*

1. *The Steiner point has at most one incident edge in H^+ , and this edge is either an f-edge or an m-edge on the opposite side of \mathcal{V}_s to sp_k .*
2. *If sp_k is an f-edge, s has no incident edges on L and either one or two incident edges in L^- . In the latter case, one of the two edges must be an m-edge on the same side of \mathcal{V}_s as sp_k , and the other must be either an f-edge or an m-edge on the opposite side of \mathcal{V}_s .*

3. If sp_k is an *m-edge*, s has at most one incident edge on L , which is necessarily an *m-edge*, at most one incident edge in L^- , which is necessarily an *m-edge* on the same side of \mathcal{V}_s as sp_k , and at most one incident edge in L^+ , which is either an *f-edge* or an *m-edge*.

These two theorems imply the following results.

Corollary 3 Let T be a gradient-constrained MGA in a vertical plane and let s be a Steiner point in T . Then:

1. The Steiner point s has at most one incident *b-edge*.
2. If the incident sink edge to s is an *m-edge*, then s has at most one incident *f-edge*.
3. The degree of s is either three or four.

6.2 A Classification of Steiner Points

Unlike minimum Steiner trees, gradient-constrained MGAs in a vertical plane do not appear to have a strong geometric structure or a canonical form. However, there are further structural restrictions at Steiner points (beyond those in [Corollary 3](#)) that have the potential to be exploited as part of an exact or heuristic algorithm. Steiner points can be classified in terms of the labels of the incident edges, in a similar way as was done in [Sect. 4.1](#). This section outlines a classification, both for degree 3 and 4 Steiner points. The results are proved in [66], using methods of subdifferential calculus and Minkowski sums – see [53] for some background on these topics.

6.2.1 Degree 3 Steiner Points

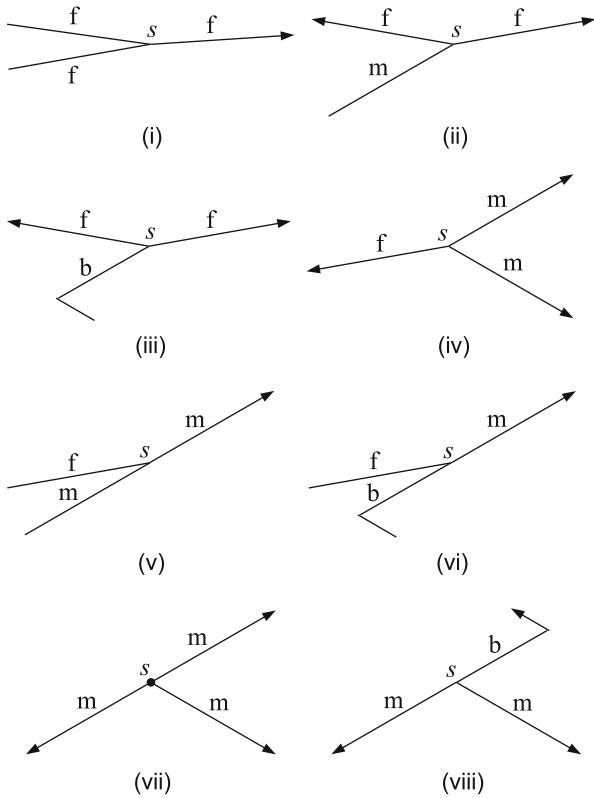
Let T be a gradient-constrained MGA in a vertical plane, and let s be a degree 3 Steiner point in T , with incident edges p_1s, p_2s, sp_3 , where p_1s, p_2s are source edges with positive flows t_1, t_2 and sp_3 is the sink edge. The incident edges p_1s, p_2s, sp_3 have respective weights $w_1 = d + ht_1$, $w_2 = d + ht_2$ and $w_3 = d + h(t_1 + t_2)$.

Let g_1, g_2, g_3 denote the respective *labels* of p_1s, p_2s, sp_3 . Then, here, the *labeling* of s is denoted $(g_i g_j g_k)$, where $i, j, k \in \{1, 2, 3\}$. In [Sect. 4.1](#), a ‘/’ sign was used to distinguish between edges lying above and below the horizontal plane \mathcal{H}_s through a Steiner point s in a gradient-constrained SMT. The situation is more complex for gradient-constrained MGAs, and distinguishing between edges above and below \mathcal{H}_s becomes less useful. Hence, in this section, there is no ‘/’ sign in the labeling. The convention employed here is that the edge labels are stated in order of increasing gradient, that is, “f” followed by “m” followed by “b.”

Since each $g_i \in \{f, m, b\}$, there are ten possible labelings of s to examine:

$$\begin{array}{lll} (\text{fff}) & (\text{ffm}) & (\text{ffb}) \\ (\text{fmm}) & (\text{fmb}) & (\text{fbm}) \\ (\text{mmm}) & (\text{mmb}) & (\text{mbm}) \\ (\text{bbb}) & & \end{array}$$

Fig. 14 Feasibly optimal labelings for degree 3 Steiner points



A labeling that can occur in a gradient-constrained MGA is called *feasibly optimal*. Note that for gradient-constrained minimum Steiner trees in a vertical plane, only three of these labelings are feasibly optimal by Corollary 1, namely, (fmm), (mmm), and (mbb). For gradient-constrained MGAs, however, additional labelings are possible.

Three of the labelings, those involving more than one “b” label, can immediately be eliminated as not being feasibly optimal by Corollary 3. It is shown in [66] that all the other labelings can occur but with some restrictions on the labeling of the sink edge.

Theorem 13 Let T be a gradient-constrained MGA in a vertical plane, and let s be a degree 3 Steiner point in T . Then s has seven feasibly optimal labelings: (fff), (ffm), (ffb), (fmm), (fmb), (mmm), and (mbb).

The labeling (fmm) has two feasibly optimal geometries, and hence there are eight possible configurations for s , as shown in Fig. 14. In the figure, edges which can feasibly be sink edges are indicated by arrows.

It is worth noting that, of the eight possible geometries for degree 3 Steiner points in gradient-constrained MGAs in a vertical plane, the three configurations (iv), (vii),

and (viii) are the only labelings in which any edge can feasibly be the sink edge. Also note that these are the only three feasibly optimal labelings for Steiner points in gradient-constrained SMTs in a vertical plane.

6.2.2 Degree 4 Steiner Points

Suppose s is a degree 4 Steiner point in a gradient-constrained MGA T in a vertical plane with four edges $p_{1s}, p_{2s}, p_{3s}, sp_4$, where p_{1s}, p_{2s}, p_{3s} are source edges with respective positive flows t_1, t_2, t_3 and sp_4 is the sink edge. Let g_1, g_2, g_3, g_4 be the respective labels of these edges. Then the labeling of s is denoted by $(g_1g_2g_3g_4)$. There are 15 potential labelings to examine:

$$\begin{array}{lll} (\text{ffff}) & (\text{fffm}) & (\text{fffb}) \\ (\text{ffmm}) & (\text{ffmb}) & (\text{ffbb}) \\ (\text{fmmm}) & (\text{fmm}) & (\text{fmbb}) \\ (\text{fbbb}) & (\text{mmmm}) & (\text{mmbb}) \\ (\text{mmbb}) & (\text{mbbb}) & (\text{bbbb}) \end{array}$$

For gradient-constrained SMTs in a vertical plane, only one of these labelings is feasibly optimal, namely, (mmmm) , which forms a *cross* (therefore, a gradient-constrained vertical plane is an example of an *X-plane* discussed in [59]). Again, additional labelings are possible for gradient-constrained MGAs in a vertical plane. Nevertheless, most of the labelings can be eliminated as not being feasibly optimal.

Theorem 14 *Let T be a gradient-constrained MGA in a vertical plane and let s be a Steiner point in T . Then the labeling of s is not $(\text{ffff}), (\text{fffb}), (\text{ffmb}), (\text{ffbb}), (\text{fmmb}), (\text{fmbb}), (\text{fbbb}), (\text{mmmm}), (\text{mmbb}), (\text{mbbb}),$ or (bbbb) .*

Theorem 15 *Let T be a gradient-constrained MGA in a vertical plane and let s be a degree 4 Steiner point in T . Then the following three labelings are feasibly optimal for s : $(\text{ffmm}), (\text{fmmm}),$ and (mmmm) .*

Examples have been constructed of Steiner points with each of these labelings, and the optimality of each case has been proved in [66]. Feasible configurations for degree 4 Steiner points are shown in Fig. 15. In the figure, edges which can feasibly be sink edges are indicated by arrows.

Whether or not the labeling (fffm) is feasibly optimal remains open to question. If the labeling is feasible, then one of the f-edges is the sink edge sp_4 , the m-edge is on the same side of \mathcal{V}_s as sp_4 (and on the opposite side of \mathcal{H}_s to sp_4), and the two remaining f-edges are on the opposite side of \mathcal{V}_s to sp_4 .

There is a sense in which the feasibly optimal labeling (fmmm) is nongeneric.

Definition Let T be a gradient-constrained MGA, and let s be a Steiner point in T . Then the labeling of s is said to be *unstable* if an arbitrarily small change in one of the edge weights causes the labeling to change.

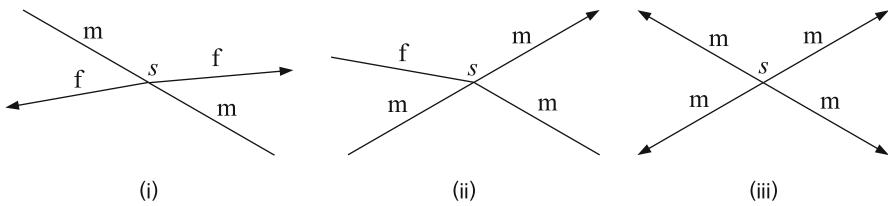


Fig. 15 Feasibly optimal labelings for degree 4 Steiner points. Edges which can feasibly be sink edges are indicated by arrows

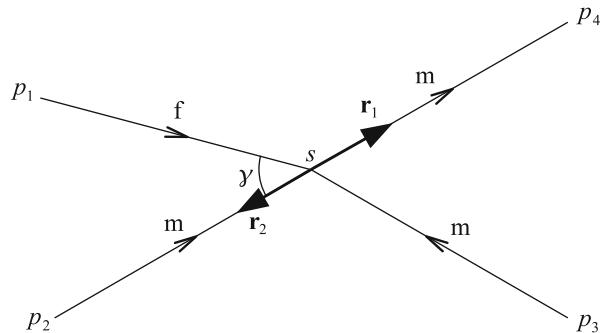


Fig. 16 Steiner point with labeling (fmmp)

Theorem 16 Let T be a gradient-constrained MGA in a vertical plane and let s be a Steiner point in T . Then the labeling (fmmp) is unstable.

The situation is illustrated in Fig. 16. If s is perturbed in direction \mathbf{r}_1 , then the variation is greater than or equal to zero when

$$t_1 \leq \frac{d}{h} \frac{\cos \gamma}{1 - \cos \gamma}.$$

Similarly, if s is perturbed in direction \mathbf{r}_2 , then the variation is greater than or equal to zero when

$$t_1 \geq \frac{d}{h} \frac{\cos \gamma}{1 - \cos \gamma}.$$

Therefore, for the variation to be greater than or equal to zero, it must be exactly zero, and this can only occur when

$$t_1 = \frac{d}{h} \frac{\cos \gamma}{1 - \cos \gamma}.$$

An arbitrarily small change in t_1 will cause the labeling of s to change, and hence Theorem 16 follows.

7 The Three-Dimensional Gradient-Constrained Gilbert Problem

The previous two sections introduced the gradient-constrained Gilbert arborescence problem and gradient-constrained minimum Gilbert arborescences (MGAs) lying in a vertical plane. While the vertical plane problem has a number of potential applications, such as designing the haulage infrastructure for underground mines where the access points to the ore bodies roughly lie in a vertical plane, most mining networks are genuinely three-dimensional. The aim of this section is to establish fundamental properties for gradient-constrained MGAs in three dimensions.

Here, a classification of degree 3 and 4 Steiner points in gradient-constrained MGAs in three-space is presented. As before, the classification of a Steiner point is in terms of the *labels* of its incident edges, where a label indicates whether the gradient of the straight line segment connecting the endpoints of an edge is less than, equal to, or greater than m . However, unlike gradient-constrained Steiner minimum trees or MGAs in the plane, there is no upper bound on the degree of a Steiner point in a gradient-constrained MGA in three dimensions. This issue is addressed in Sect. 7.2.

Let \mathcal{H}_s denote the horizontal plane passing through a point s . The other notation used in this section is identical to that in Sect. 6. The results in this section are mainly from [66].

7.1 Classification of Steiner Points of Degree at Most Four

As before, Steiner points of degree 3 and 4 are considered separately.

7.1.1 Degree Three Steiner Points

In [13], it was shown that there are five feasibly optimal labelings for degree 3 Steiner points in gradient-constrained SMTs in \mathbb{R}^3 . They are (fff), (ffm), (fmm), (mmm), and (mmn). These and additional labelings are possible for gradient-constrained MGAs in three-space.

Let T be a gradient-constrained MGA in three-space, and let s be a Steiner point in T . By Theorem 13, the labelings (fff), (ffm), (ffb), (fmm), (fmb), (mmm), and (mmn) are feasibly optimal for gradient-constrained MGAs in a vertical plane. Since the vertical plane case is a special case of the three-dimensional case, these seven labelings are also feasibly optimal in gradient-constrained MGAs in three-space. The remaining labelings are (ffb), (mbb), and (bbb). It can be shown, by a similar argument to that used in proving Lemma 4, that if s has an incident b-edge, then it has no other incident b- or m-edges on the side of \mathcal{H}_s containing the b-edge. Hence, labelings (mbb) and (bbb) are not feasibly optimal, since two edges with respective labels m,b or b,b cannot coexist on the same side of \mathcal{H}_s . The labeling (ffb) is not feasibly optimal by noting that there is no horizontal vector component to counterbalance that of the f-edge vector (since the two b-edge vectors

are vertical). Hence the equilibrium condition cannot be satisfied. This implies the following theorem.

Theorem 17 *Let T be a gradient-constrained MGA in three-space, and let s be a degree 3 Steiner point in T . Then s has seven feasibly optimal labelings: (fff), (ffm), (ffb), (fmm), (fmb), (mmm), and (mmb).*

Steiner points with these labelings are shown in Fig. 17, where the m-cones associated with the Steiner point are included to distinguish between f-, m-, and b-edges in three-space. Note that in three-space, for the labelings (fff), (ffm), (fmm), (mmm), and (mmb), any of the three edges can potentially be the sink edge. However, it can be shown that for the labelings (ffb) and (fmb), the sink edge must be labeled “f” and “m,” respectively.

7.1.2 Degree 4 Steiner Points

In Sect. 4.1 (and [13]), it was shown that the labeling (mmmm) is the only feasibly optimal labeling for degree 4 Steiner points in gradient-constrained SMTs in three-space. For this labeling, two of the m-edges must lie above \mathcal{H}_s and the other two below. Moreover, recall that the four edges incident to s are necessarily *bi-vertically coplanar*, meaning the two m-edges above \mathcal{H}_s lie in the same vertical plane and the two m-edges below \mathcal{H}_s lie in the same vertical plane (Fig. 8). When the two planes align to form a single vertical plane, then the four m-edges form a *cross*, a structure that is feasibly optimal for gradient-constrained MGAs in a vertical plane, as discussed in the previous section.

Additional labelings are possible for degree 4 Steiner points in gradient-constrained MGAs in three-space and are listed in the theorem below. As before, the potential minimality of these labelings can be proved by direct construction and checking for each construction that certain necessary and sufficient conditions are satisfied.

Theorem 18 *Let T be a gradient-constrained MGA in three-space, and let s be a Steiner point in T . Then the labelings (ffff), (ffmm), (fmmp), and (mmmm) are feasibly optimal.*

Specific examples of Steiner points with each of these feasible configurations for degree 4 Steiner points are shown in Fig. 18. For each of the two labelings (ffmm) and (fmmp), two distinct configurations of edges are shown. In all cases, the sink edge is indicated by arrows. The m-cones (generated by taking a line through s with gradient m and rotating it 360° about the vertical line through s) are shown to help distinguish between f-, m-, and b-edges. Note that in general, the sink edge can have any of the available labels except for the labeling (ffff), in which case the sink edge must be an f-edge.

The question now is whether the list in Theorem 18 is complete. As before, a b-edge incident with s cannot lie on the same side of \mathcal{H}_s as another incident b- or m-edge, which immediately eliminates the labelings (fmbb), (fbhb), (mmbb), (mbbb),

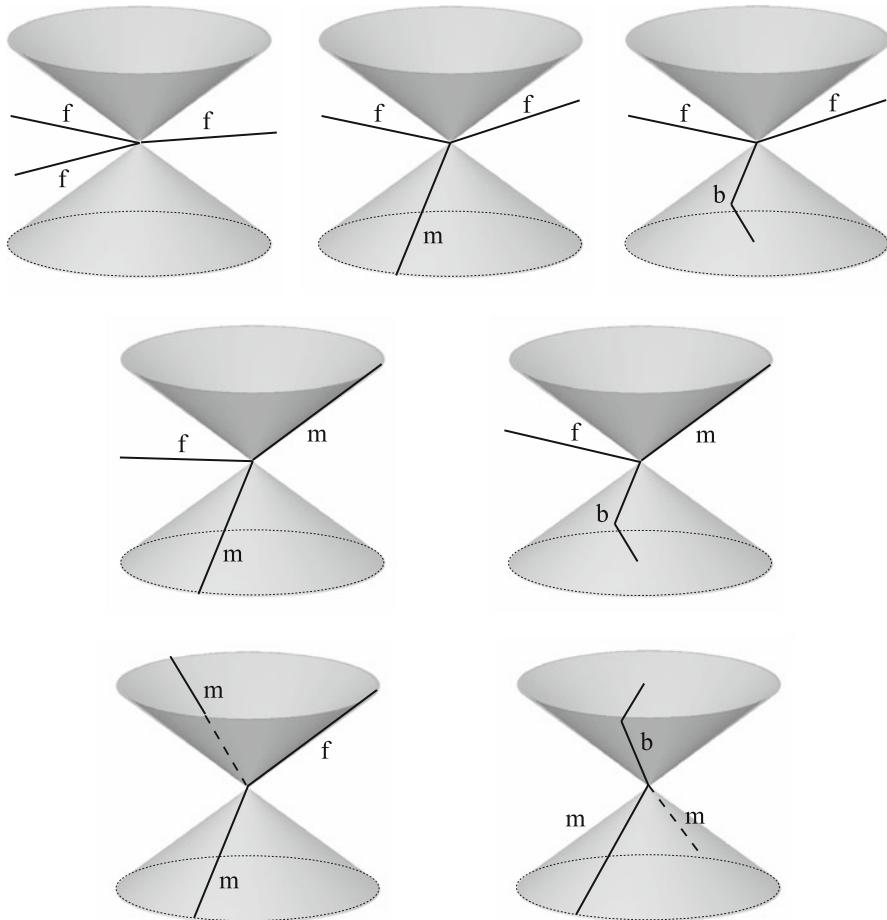


Fig. 17 Feasibly optimal labelings for degree 3 Steiner points in three-space

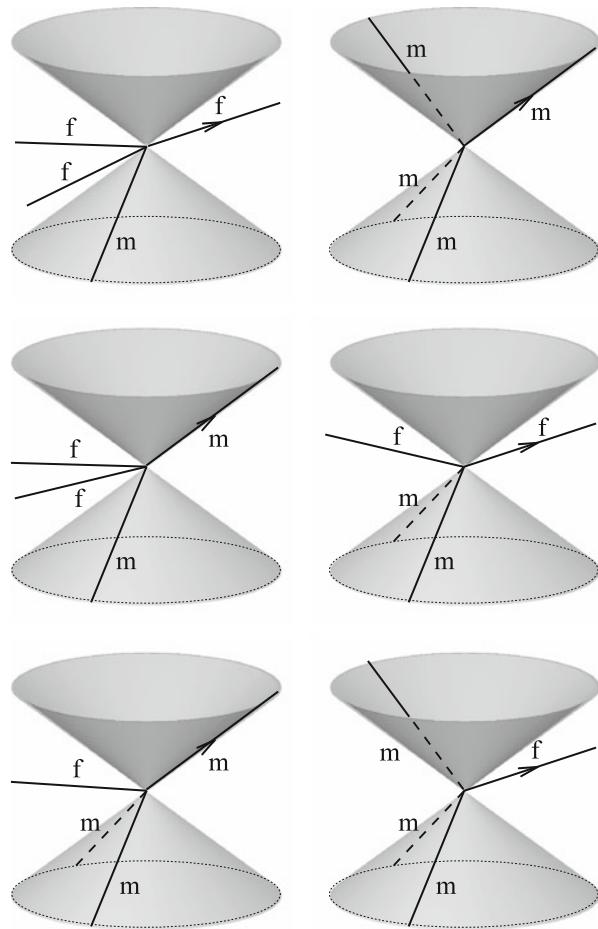
and (bbbb). A similar argument also shows that (ffffb) is never feasibly optimal. In [67], it is shown that all Steiner points in Euclidean MGAs in three-space have degree 3. It follows that the labeling (ffff) is not feasibly optimal.

The remaining labelings not accounted for are (fffb), (ffmb), (fmmb), and (mmmb). The following conjecture, if true, would prove that none of these four labelings are feasibly optimal.

Conjecture 1 *Let T be a gradient-constrained MGA in three-space, and let s be a Steiner point in T . If the degree of s is greater than 3, then s has no incident b-edges.*

There is strong evidence to suggest that this conjecture is true. The main support for this belief is that, in three-space, a b-edge path from a source to a sink can

Fig. 18 Feasibly optimal labelings for degree 4 Steiner points in three-space



be embedded in many different ways. Numerical experiments indicate that by embedding such a path using at least three zigzag components, other edges can attach to different corner-points on the zigzag so as to reduce the cost of the network.

To demonstrate this, consider the following example.

Let $p_1 = (-0.707, 0.707, 0)$, $p_2 = (-0.707, -0.707, 0)$, and $p_3 = (1, 0, -0.5)$ be sources with respective flows $t_1 = 1$, $t_2 = 1$, and $t_3 = 100$, and let $p_4 = (1, 0, 0.5)$ be the sink. The cost parameters are $d = h = 1$ and the maximum gradient is $m = 0.5$. It can be verified that the network shown in Fig. 19, in which the Steiner point s has labeling (ffmm), is a gradient-constrained MGA.

If p_3 is perturbed by ϵ in the negative x -direction, it can be observed that the topology suddenly changes to the one shown in Fig. 20 (in which $\epsilon = 0.1$). The path from p_3 to p_4 now has more space such that it has three straight-line segments instead of two. Consequently, the edges incident to p_1 and p_2 can connect to different corner-points on the path to reduce the cost of the network. This behavior

Fig. 19 Steiner point with labeling (ffmm)

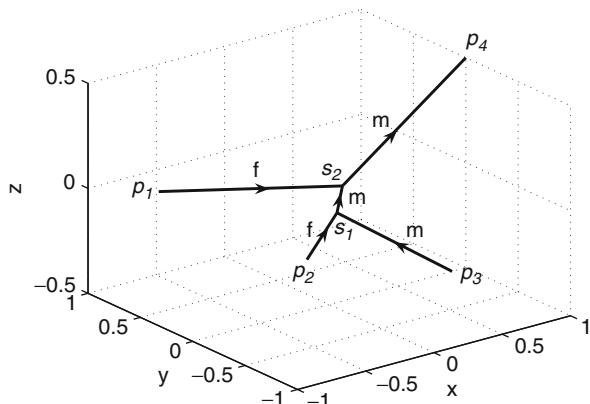
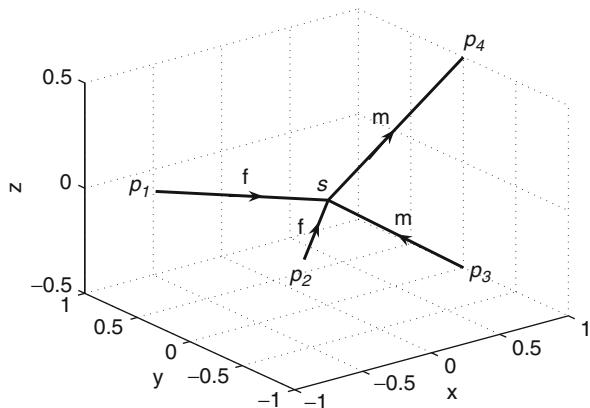


Fig. 20 Change in topology resulting from a perturbation of a terminal

has been frequently observed when attempting to construct MGAs with the above labelings.

7.2 Maximum Degree of Steiner Points

The degree of a Steiner point in a gradient-constrained Steiner minimum tree (SMT) in three dimensions is either three or four, subject to some constraints on m (see Lemma 5). It has been shown by Cox in [26] that in general there is no upper bound on the degree of a Steiner point in a minimum Gilbert network. The Steiner problem is a special case of the Gilbert arborescence problem, which is itself a special case of the Gilbert network problem. This prompts the question as to whether, for a Steiner point in a gradient-constrained MGA in three dimensions, the degree is limited to three or four as for gradient-constrained SMTs, is unbounded as for general minimum Gilbert networks, or is bounded above by some finite integer value greater than four.

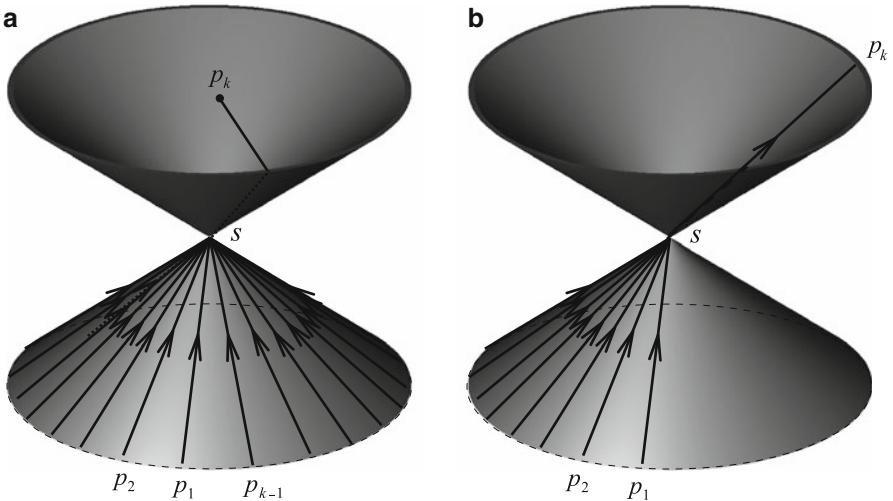


Fig. 21 Large-degree Steiner points, where the sink edge is (a) a b-edge and (b) an m-edge

Resolving this issue is far from trivial. For example, at first it may appear that the configuration illustrated in Fig. 21a, in which the Steiner point s has $k - 1$ incident source m-edges, where k is arbitrarily large, and the sink edge is a b-edge, could potentially be minimum. In this network, each path from p_i , $i = 1, \dots, k - 1$ to p_k is a geodesic under the gradient metric. Therefore, the flow cost component cannot be improved.

It is shown in [66], however, that in this case the development cost can be improved and that in fact the following result holds.

Theorem 19 *Let T be a gradient-constrained MGA in three-space and let s be a Steiner point in T with incident sink edge sp_k . If sp_k is a b-edge, then s has degree 3, and the two source edges are vertically coplanar m-edges on the opposite side of \mathcal{H}_s to the sink edge.*

Suppose, however, that the network in Fig. 21a is changed so that the sink edge is now an m-edge instead of a b-edge (Fig. 21b). The paths from p_i , $i = 1, \dots, k - 1$ to p_k are still geodesics under the gradient metric, so the flow cost cannot be improved. However, it can be shown in this case that if the flow costs are very large compared to the fixed cost, then it is possible for the network to have the shortest length among all networks connecting p_1, \dots, p_k for which the paths from the sources to the sink are geodesics. This implies that the development costs are minimum and leads to the following result.

Theorem 20 *There is no upper bound on the degree of a Steiner point in a gradient-constrained MGA in three dimensions.*

Using the UNO software product, many examples of high-degree Steiner points satisfying the properties described above have been found.

8 Conclusion

This chapter has described the theory and applications of minimum gradient-constrained interconnection networks. The focus has been on two problems: the geometric Steiner problem, which asks for a network interconnecting a given set of points of minimum total length, and the Gilbert arborescence problem, which asks for a minimum-cost flow-dependent network interconnecting a set of given sources and a unique sink. These problems have been studied in the setting of gradient-constrained space, which is a type of Minkowski space with several intriguing properties. The aim is to understand the local geometric and topological structure of these networks in order to develop efficient exact and heuristic algorithms.

Acknowledgements The research for and writing of this chapter was supported by an ARC Linkage Grant. We gratefully acknowledge the contributions of our collaborators to much of the theory described in this chapter, particularly Professor Doreen Thomas at The University of Melbourne.

Cross-References

- ▶ [Network Optimization](#)
- ▶ [Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work](#)
- ▶ [Steiner Minimum Trees in \$E^3\$: Theory, Algorithms, and Applications](#)

Recommended Reading

1. A.V. Aho, M.R. Garey, F.K. Hwang, Rectilinear Steiner trees: efficient special-case algorithms. *Networks* **7**, 37–58 (1977)
2. C. Alford, M. Brazil, D.H. Lee, Optimisation in underground mining, in *Handbook on Operations Research in Natural Resources*, ed. by A. Weintraub et al. (Springer, New York, 2007), pp. 561–577
3. C. Bajaj, The algebraic degree of geometric optimization problems. *Discret. Comput. Geom.* **3**(2), 177–191 (1988)
4. S. Bhaskaran, F.J.M. Salzborn, Optimal design of gas pipeline networks. *J. Oper. Res. Soc.* **30**(12), 1047–1060 (1979)
5. S. Boyd, A. Mutapcic, Subgradient methods, lecture notes of EE364b, Stanford University, Winter Quarter, 2006–2007 (Available at http://www.stanford.edu/class/ee364b/lectures/subgrad_method.notes.pdf)
6. S. Boyd, L. Vandenberghe, *Convex Optimization* (Cambridge University Press, New York, 2004)
7. M. Brazil, Steiner minimum trees in uniform orientation metrics, in *Steiner Trees in Industry*, ed. by X. Cheng, D.-Z. Du (Kluwer Academic Publishers, Dordrecht/Boston/London, 2001), pp. 1–28

8. M. Brazil, D.A. Thomas, Network optimization for the design of underground mines. *Networks* **49**(1), 40–50 (2007)
9. M. Brazil, M. Zachariasen, Steiner trees for fixed orientation metrics. *J. Glob. Optim.* **43**, 141–169 (2009)
10. M. Brazil, D.A. Thomas, J.F. Weng, Gradient-constrained minimal Steiner trees, in *Network Design: Connectivity and Facilities Location*, ed. by P.M. Pardalos, D.-Z. Du DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 40 (American Mathematical Society, Providence, 1998), pp. 23–38
11. M. Brazil, D.H. Lee, J.H. Rubinstein, D.A. Thomas, J.F. Weng, N.C. Wormald, Network optimisation of underground mine design. *Australas. Inst. Min. Metall. Proc.* **305**(1), 57–65 (2000)
12. M. Brazil, D.A. Thomas, J.F. Weng, On the complexity of the Steiner problem. *J. Comb. Optim.* **4**, 187–195 (2000)
13. M. Brazil, J.H. Rubinstein, D.A. Thomas, J.F. Weng, N.C. Wormald, Gradient-constrained minimum networks. I. Fundamentals. *J. Glob. Optim.* **21**(2), 139–155 (2001)
14. M. Brazil, D.H. Lee, J.H. Rubinstein, D.A. Thomas, J.F. Weng, N.C. Wormald, A network model to optimise cost in underground mine design. *Trans. S. Afr. Inst. Electr. Eng.* **93**(2), 97–103 (2002)
15. M. Brazil, D.H. Lee, M. van Leuven, J.H. Rubinstein, D.A. Thomas, N.C. Wormald, Optimising declines in underground mines. *Min. Technol.* **112**, 164–170 (2003)
16. M. Brazil, D.H. Lee, J.H. Rubinstein, D.A. Thomas, J.F. Weng, N.C. Wormald, Optimisation in the design of underground mine access, in *Orebody Modelling and Strategic Mine Planning: Uncertainty and Risk Management*, ed. by R. Dimitrakopoulos. Spectrum Series, vol. 14 (Australasian Institute of Mining and Metallurgy, Carlton, 2005), pp. 121–124
17. M. Brazil, D.A. Thomas, J.F. Weng, J.H. Rubinstein, D.H. Lee, Cost optimisation for underground mining networks. *Optim. Eng.* **6**(2), 241–256 (2005)
18. M. Brazil, D.A. Thomas, J.F. Weng, M. Zachariasen, Canonical forms and algorithms for Steiner trees in uniform orientation metrics. *Algorithmica* **44**, 281–300 (2006)
19. M. Brazil, P.A. Grossman, D.H. Lee, J.H. Rubinstein, D.A. Thomas, N.C. Wormald, Constrained path optimisation for underground mine layout, in *International Conference of Applied and Engineering Mathematics*, London, July 2007, pp. 856–861
20. M. Brazil, P.A. Grossman, D.H. Lee, J.H. Rubinstein, D.A. Thomas, N.C. Wormald, Decline design in underground mines using constrained path optimisation. *Min. Technol.* **117**, 93–99 (2008)
21. M. Brazil, D.A. Thomas, J.F. Weng, Gradient-constrained minimum networks. II. Labelled or locally minimal Steiner points. *J. Glob. Optim.* **42**, 23–37 (2008)
22. M. Brazil, J.H. Rubinstein, D.A. Thomas, J.F. Weng, N.C. Wormald, Gradient-constrained minimum networks, III. Fixed topology. *J. Optim. Theory Appl.* **155**(1), 336–354 (2012).
23. J. Brimberg, The Fermat-Weber location problem revisited. *Math. Program.* **71**(1), 71–76 (1995)
24. J. Brimberg, H. Juel, A. Schöbel, Linear facility location in three dimensions—models and solution methods. *Oper. Res.* **50**(6), 1050–1057 (2002)
25. R. Chandrasekaran, A. Tamir, Open questions concerning Weiszfeld's algorithm for the Fermat-Weber location problem. *Math. Program.* **44**, 293–295 (1989)
26. C.L. Cox, Flow-dependent networks: existence and behavior at Steiner points. *Networks* **31**(3), 149–156 (1998)
27. Z. Drezner, H.W. Hamacher (eds.), *Facility Location: Applications and Theory* (Springer, Berlin, 2004)
28. D.-Z. Du, F.K. Hwang, Reducing the Steiner problem in a normed space. *SIAM J. Comput.* **21**(6), 1001–1007 (1992)
29. D.-Z. Du, B. Gao, R.L. Graham, Z.-C. Liu, P.-J. Wan, Minimum Steiner trees in normed planes. *Discret. Comput. Geom.* **9**(1), 351–370 (1993)
30. R. Durier, C. Michelot, Geometrical properties of the Fermat-Weber problem. *Eur. J. Oper. Res.* **20**(3), 332–343 (1985)

31. G.A. Ferguson, Mining engineers toolkit. Camborne Sch. Mines Assoc. J. 10–13 (2000)
32. M.R. Garey, D.S. Johnson, The rectilinear Steiner tree problem is NP-complete. SIAM J. Appl. Math. **32**(4), 826–834 (1977)
33. M.R. Garey, R.L. Graham, D.S. Johnson, The complexity of computing Steiner minimal trees. SIAM J. Appl. Math. **32**(4), 835–859 (1977)
34. E.N. Gilbert, Minimum cost communication networks. Bell Syst. Tech. J. **46**, 2209–2227 (1967)
35. E.N. Gilbert, H.O. Pollak, Steiner minimal trees. SIAM J. Appl. Math. **16**(1), 1–29 (1968)
36. Z. Gligoric, C. Beljic, V. Simeunovic, Shaft location selection at deep multiple orebody deposit by using fuzzy TOPSIS method and network optimization. Expert Syst. Appl. **37**(2), 1408–1418 (2010)
37. F.K. Hwang, On Steiner minimal trees with rectilinear distance. SIAM J. Appl. Math. **30**, 104–114 (1976)
38. F.K. Hwang, A primer of the Euclidean Steiner problem. Ann. Oper. Res. **33**, 73–84 (1991)
39. F.K. Hwang, J.F. Weng, The shortest network under a given topology. J. Algorithms **13**(3), 468–488 (1992)
40. F.K. Hwang, D.S. Richards, P. Winter, *The Steiner Tree Problem*. Annals of Discrete Mathematics, vol. 53 (Elsevier Science Publishers, North Holland, Amsterdam, 1992)
41. H.W. Kuhn, A note on Fermat’s problem. J. Math. Program. **4**(1), 98–107 (1973)
42. H.W. Kuhn, “Steiner’s” problem revisited, in *Studies in Optimization*, ed. by G.B. Dantzig, B.C. Eaves. Studies in Mathematics, vol. 10 (Mathematical Association of America, Washington, 1974), pp. 52–70
43. H.W. Kuhn, R.E. Kuenne, An efficient algorithm for the numerical solution of the generalized Weber problem in spatial economics. J. Reg. Sci. **4**(2), 21–33 (1962)
44. Y.S. Kupitz, H. Martini, Geometric aspects of the generalized Fermat-Torricelli problem, in *Intuitive Geometry*, ed. by I. Bárány, K. Boroczky. Bolyai Society Mathematical Studies, vol. 6 (Janos Bolyai Mathematical Society, Budapest, 1997), pp. 55–127
45. D.H. Lee, Low cost drainage networks. Networks, **6**, 351–371 (1976)
46. H. Lerchs, I.F. Grossmann, Optimum design of open-pit mines. Trans. Can. Inst. Min. Metall. **LXVIII**, 17–24 (1965)
47. Y. Lizotte, J. Elbrond, Optimal layout of underground mining levels. Can. Inst. Min. Metall. Petrol. Bull. **78**(873), 41–48 (1985)
48. Z.A. Melzak, On the problem of Steiner. Can. Math. Bull. **4**, 143–148 (1961)
49. B.K. Nielsen, P. Winter, M. Zachariasen, An exact algorithm for the uniformly-oriented Steiner tree problem, in *Proceedings of the 10th European Symposium on Algorithms. Lecture Notes in Computer Science*, vol. 2461 (Springer, Berlin, 2002), pp. 760–772
50. K. Prendergast, Steiner ratio for gradient constrained networks. Ph.D. dissertation, Department of Electrical and Electronic Engineering, The University of Melbourne, Australia, 2006
51. K. Prendergast, D.A. Thomas, J.F. Weng, Optimum Steiner ratio for gradient-constrained networks connecting three points in 3-space, part I. Networks **53**(2), 212–220 (2009)
52. D.S. Richards, J.S. Salowe, A simple proof of Hwang’s theorem for rectilinear Steiner minimal trees. Ann. Oper. Res. **33**, 549–556 (1991)
53. R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, 1970)
54. J.H. Rubinstein, D.A. Thomas, A variational approach to the Steiner network problem. Ann. Oper. Res. **33**, 481–499 (1991)
55. J.H. Rubinstein, D.A. Thomas, N.C. Wormald, Steiner trees for terminals constrained to curves. SIAM J. Discret. Math. **10**, 1–17 (1997)
56. J.H. Rubinstein, D.A. Thomas, J.F. Weng, Minimum networks for four points in space. Geom. Dedic. **93**, 571–70 (2002)
57. N.Z. Shor, *Minimization Methods for Non-differentiable Functions* (Springer, Berlin/New York, 1985)
58. W.D. Smith, How to find minimal trees in Euclidean d -space. Algorithmica **7**, 137–177 (1992)
59. K.J. Swanepoel, The local Steiner problem in normed planes. Networks **36**, 104–113 (2000)

60. D.A. Thomas, J.F. Weng, Minimum cost flow-dependent communication networks. *Networks* **48**(1), 39–46 (2006)
61. D.A. Thomas, J.F. Weng, Gradient-constrained minimum networks: an algorithm for computing Steiner points. *J. Discret. Optim.* **7**, 21–31 (2010)
62. D.A. Thomas, M. Brazil, D.H. Lee, N.C. Wormald, Network modelling of underground mine layout: two case studies. *Int. Trans. Oper. Res.* **14**(2), 143–158 (2007)
63. A.C. Thompson, *Minkowski Geometry*. Encyclopedia of Mathematics and Its Applications, vol. 63 (Cambridge University Press, Cambridge/New York, 1996)
64. D. Trietsch, Minimal Euclidean networks with flow dependent costs — the generalized Steiner case, May 1985, discussion paper no. 655, Center for Mathematical Studies in Economics and Management Science, Northwestern University, Evanston, IL, 1985
65. D. Trietsch, J.F. Weng, Pseudo-Gilbert-Steiner trees. *Networks* **33**, 175–178 (1999)
66. M.G. Volz, Gradient-constrained flow-dependent networks for underground mine design. PhD thesis, Department of Electrical and Electronic Engineering, The University of Melbourne, 2008
67. M.G. Volz, M. Brazil, C.J. Ras, K.J. Swanepoel, D.A. Thomas, The Gilbert arborescence problem Networks (accepted, May 2012)
68. M.G. Volz, M. Brazil, D.A. Thomas, The gradient-constrained Fermat-Weber problem Networks (submitted 2011)
69. D.M. Warme, Spanning trees in hypergraphs with applications to Steiner trees. PhD thesis, Computer Science Department, The University of Virginia, 1998
70. D.M. Warme, P. Winter, M. Zachariasen, Exact solutions to large-scale plane Steiner tree problems, in *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore (1999), pp. 49–53
71. D.M. Warme, P. Winter, M. Zachariasen, Exact algorithms for plane Steiner tree problems: a computational study, in *Advances in Steiner Trees*, ed. by D.Z. Du, J.M. Smith, J.H. Rubenstein (Kluwer Academic Publishers, Boston, 2000), pp. 81–116
72. D.M. Warme, P. Winter, M. Zachariasen, GeoSteiner 3.1. Department of Computer Science, University of Copenhagen (DIKU) (2001), <http://www.diku.dk/geosteiner/>
73. A. Weber, *Über den Standort der Industrien* (Verlag J.C.B. Mohr, Tubingen, 1909), (Translation by C. J. Friedrich *Theory of the Location of Industries* (University of Chicago Press, Chicago, 1929))
74. E. Weiszfeld, Sur le point pour lequel la somme des distances de n points donnés est minimum. *Tohoku Math. J.* **43**, 355–386 (1937)
75. P. Widmayer, Y.F. Yu, C.K. Wong, Distance problems in computational geometry with fixed orientations, in *Proceedings of the Symposium on Computational Geometry*, Baltimore, MD (1985), pp. 186–195
76. P. Widmayer, Y.F. Yu, C.K. Wong, On some distance problems in fixed orientations. *SIAM J. Comput.* **16**(4), 728–746 (1987)
77. M. Zachariasen, Rectilinear full Steiner tree generation. *Networks* **338**, 125–143 (1999)

Graph Searching and Related Problems

Anthony Bonato and Boting Yang

Contents

1	Introduction	1512
2	Searching Undirected Graphs	1513
2.1	Robber Is Invisible and Active	1514
2.2	Robber Is Visible or Lazy	1517
3	Searching Digraphs	1519
3.1	Robber Is Visible or Lazy	1519
3.2	Robber Is Invisible and Active	1523
4	Other Searching Games	1526
4.1	Mixed Searching with Multiple Robbers	1526
4.2	Nondeterministic Graph Searching	1527
4.3	Fast Searching	1528
4.4	Graph Guarding	1529
4.5	Minimum Cost Searching	1530
4.6	Pebbling	1531
5	Open Problems on Searching Games	1532
6	Cop-Win Graphs	1535
7	Higher Cop Number	1538
7.1	Upper Bounds	1538
7.2	Lower Bounds	1540
8	Graph Classes	1542
9	Algorithmic Results	1544
10	Random Graphs	1545
10.1	Constant p	1545
10.2	The Dense Case	1546
10.3	The Sparse Case and Zigzag Theorem	1547

A. Bonato ()

Department of Mathematics, Ryerson University, Toronto, ON, Canada
e-mail: abonato@ryerson.ca

B. Yang

Department of Computer Science, University of Regina, Regina, SK, Canada
e-mail: boting@cs.uregina.ca

11	Infinite Graphs.....	1548
11.1	Cop Density.....	1549
11.2	Chordal Graphs.....	1550
11.3	Large Families of Cop-Win Graphs.....	1550
12	A Dozen Problems on Cops and Robbers.....	1551
13	Conclusion.....	1552
	Recommended Reading.....	1553

Abstract

Suppose that there is a robber hiding on vertices or along edges of a graph or digraph. Graph searching is concerned with finding the minimum number of searchers required to capture the robber. Major results of graph searching problems are surveyed, focusing on algorithmic, structural, and probabilistic aspects of the field.

1 Introduction

Graph searching is a hot topic in mathematics and computer science now, as it leads to a wealth of beautiful mathematics and since it provides mathematical models for many real-world problems such as eliminating a computer virus in a network, computer games, or even counterterrorism. In all searching problems, there is a notion of *searchers* (or *cops*) trying to capture some *robber* (or *intruder* or *fugitive*). A basic optimization question here is: What is the fewest number of searchers required to capture the robber? There are many graph searching problems motivated by applied problems or inspired by some theoretical issues in computer science and discrete mathematics. These problems are defined by (among other things) the class of graphs (e.g., undirected graphs, digraphs, or hypergraphs), the actions of searchers and robbers, conditions of captures, speed, or visibility.

Graph searching has a variety of names in the literature, such as Cops and Robbers games and pursuit-evasion problems. The reader is referred to survey papers [4, 21, 60, 62, 74] for an outline of the many graph searching models. A recent book by Bonato and Nowakowski [29] covers all aspects of Cops and Robbers games.

In this chapter, a broad overview of graph searching is given, focusing on the most important results. The first part considers the so-called searching games where the robber may occupy an edge or vertex and the robber can usually move at high speeds at any time (but not always, depending on the model). The second part considers so-called *Cops and Robbers* games where the cops and robber occupy only vertices and they move alternatively to their neighbors. Searching games are often related to various width-type parameters, while there

¹Research was supported in part by NSERC, MITACS, and Ryerson University.

²Research was supported in part by NSERC and MITACS.

is no such connection in Cops and Robbers. Owing to space constraints, proofs are omitted (however, references are given for all results). The results presented here span probabilistic, algorithmic, and structural results. As such, the chapter is not entirely self-contained; the reader can find most of the relevant background in the books [29, 49, 134]. Note that this chapter is the first reference to give a comprehensive overview of both searching and Cops and Robbers games (although searching is discussed briefly in [29]). The chapter finishes each part with a dozen problems and conjectures which are arguably the most important ones in the field of graph searching.

Let $G = (V, E)$ ($D = (V, E)$) denote a graph (or digraph) with vertex set V and edge set E . Use uv to denote an edge in a graph with two end vertices u and v and (u, v) to denote a directed edge in a digraph with tail u and head v . All graphs and digraphs considered in this chapter are finite and simple, unless otherwise stated.

Part 1. Searching Games

The moniker *searching* is used for the case when the robber can move at great speed at any time or at least when a searcher approaches him (not just move to a neighbor in his turn, as is the case in Cops and Robbers).

Searching has been linked in the literature to pathwidth and vertex separation of a graph [52, 89], to pebbling (and hence, to computer memory usage) [89], to assuring privacy when using bugged channels [58], to VLSI (i.e., very large-scale integrated) circuit design [56], and to motion planning of multiple robots [127]. A large number of graph searching games have been introduced. We will focus here on some of the basic models.

Many graph searching games are closely related to graph width parameters, such as treewidth, pathwidth, and cutwidth. They provide some interpretation of width parameters, and often they can give a deeper insight into the graph structures and produce efficient algorithms. Examples include the proof of a min-max theorem on treewidth [122], the linear time algorithm for computing pathwidth of a tree [52, 87], the polynomial time algorithm for computing branchwidth of a planar graph [123], the linear time algorithm for computing cutwidth of a tree [97], and the computation of the topological bandwidth [40, 98].

Part 1 of the chapter surveys searching games on undirected graphs and digraphs. There are four sections in this part. Section 2 deals with the undirected graph searching games in which the robber can move at high speeds along any searcher-free path. Section 3 deals with the digraph searching games in which the robber usually moves at high speeds along a searcher-free directed path (depending on the game setting). Section 4 deals with more recently introduced searching games. As with the end of Part 2, Part 1 closes with a dozen of open problems in the area.

2 Searching Undirected Graphs

The first searching model was introduced by Parsons [109], after being approached by the author of [36] (a paper dealing with finding a spelunker lost in a system of caves). Let G be a connected, undirected graph embedded in \mathbb{R}^3 with the Euclidean

distance metric such that no pair of edges intersect at a point that is not a common endpoint. Imagine there is a robber in G who can be located at any point of G , that is, anywhere along an edge or at a vertex. We want to capture the robber using the minimum number of searchers. For each positive integer k , let $\mathcal{C}_k(G)$ be the set of all families $F = \{f_1, f_2, \dots, f_k\}$ of continuous functions $f_i : [0, \infty) \rightarrow G$. A *continuous search strategy* for G is a family $F \in \mathcal{C}_k(G)$ such that for every continuous function $h : [0, \infty) \rightarrow G$ (corresponding to the robber), there is a $t_h \in [0, \infty)$ and $f_i \in F$ satisfying $h(t_h) = f_i(t_h)$. We say that f_i captures the robber at time t_h . The *continuous search number* of a graph G is the smallest k such that there exists a search strategy for G in $\mathcal{C}_k(G)$. There are some contexts for which Parsons searching is the model required. For example, in the case of searching for someone lost or a robber hiding in a cave system, the robber and the searchers move continuously. But the notion of Parsons searching presents certain difficulties because the model involves the action of continuous functions defined on the nonnegative real numbers. In this survey, only discrete versions of Parsons searching are considered.

2.1 Robber Is Invisible and Active

When the robber is invisible and active, there are three basic searching games: edge searching, node searching, and mixed searching, which involve placing some restriction on searchers, but placing no restrictions on the robber. In these search games, the discrete time intervals (or time-steps) are introduced. Initially, G contains a robber who is located at a vertex in G , and G does not contain any searchers. Each searcher has no information on the whereabouts of the robber (i.e., robber is *invisible*), but the robber has complete knowledge of the location of all searchers. The goal of the searchers is to capture the robber, and the goal of the robber is to avoid being captured. The robber always chooses the best strategy so that he evades capture. Suppose the game starts at time t_0 and the robber is captured at time t_N and the search time is divided into N intervals $(t_0, t_1]$, $(t_1, t_2]$, \dots , $(t_{N-1}, t_N]$ such that in each interval $(t_i, t_{i+1}]$ (also called *step*), exactly one searcher performs one action: placing, removing, or sliding. The robber can move from a vertex x to a vertex y in G at any time in the interval (t_0, t_N) if there exists a path between x and y which contains no searcher (i.e., robber is *active*).

In the *edge search game* introduced by Megiddo et al. [100], there are three actions for searchers: placing a searcher on a vertex, removing a searcher from a vertex, and sliding a searcher along an edge from one end to the other. The robber is captured if a searcher and the robber occupy the same vertex on G .

In the *node search game* introduced by Kirousis and Papadimitriou [89], there are two actions for searchers: placing a searcher on a vertex and removing a searcher from a vertex. The robber is captured if a searcher and the robber occupy the same vertex of G or the robber is on an edge whose endpoints are both occupied by searchers.

In the *mixed search game* introduced by Bienstock and Seymour [22], searchers have the same actions as those in the edge search game. The robber is captured if a searcher and the robber occupy the same vertex on G or the robber is on an edge whose endpoints are both occupied by searchers.

The *edge search number* of G , denoted by $\text{es}(G)$, is the smallest positive integer k such that k searchers can capture the robber. Analogously, define the *node search number* of G (written $\text{ns}(G)$) and *mixed search number* of G (written $\text{ms}(G)$). The following theorem demonstrates the relationships between search numbers.

Theorem 1 ([22, 89]) *If G is a connected graph, then the following inequalities hold:*

- (1) $\text{ns}(G) - 1 \leq \text{es}(G) \leq \text{ns}(G) + 1$.
- (2) $\text{ms}(G) \leq \text{es}(G) \leq \text{ms}(G) + 1$.
- (3) $\text{ms}(G) \leq \text{ns}(G) \leq \text{ms}(G) + 1$.

The following result shows that all three search problems are **NP**-complete.

Theorem 2 ([22, 89, 93, 100]) *Given a graph G and an integer k , the problem of determining whether $\text{es}(G) \leq k$ ($\text{ns}(G) \leq k$ or $\text{ms}(G) \leq k$) is **NP**-complete.*

Megiddo et al. [100] only showed that the edge search problem is **NP**-hard. This problem belongs to **NP** owing to the monotonicity result of [93], in which LaPaugh showed that recontamination of edges cannot reduce the number of searchers needed to clear a graph. A search strategy is *monotonic* if the set of cleared edges before any step is always a subset of the set of cleared edges after the step. Monotonicity is an important issue in graph search problems. Bienstock and Seymour [22] proposed a method that gives a succinct proof for the monotonicity of the mixed search problem, which implies the monotonicity of the edge search problem and the node search problem. Fomin and Thilikos [61] provided a general framework that can unify monotonicity results in a unique min-max theorem.

Theorem 3 ([22, 93]) *The edge search, node search, and mixed search problems are monotonic.*

Search numbers have close relationships with pathwidth and treewidth [117, 118]. Given a graph G , a *tree decomposition* of G is a pair (T, W) with a tree $T = (I, F)$, $I = \{1, 2, \dots, m\}$, and a family of nonempty subsets $W = \{W_i \subseteq V : i = 1, 2, \dots, m\}$, such that

- (1) $\bigcup_{i=1}^m W_i = V$.
 - (2) For each edge $uv \in E$, there is an $i \in I$ with $\{u, v\} \subseteq W_i$.
 - (3) For all $i, j, k \in I$, if j is on the path from i to k in T , then $W_i \cap W_k \subseteq W_j$.
- The *width* of a tree decomposition (T, W) is

$$\max\{|W_i| - 1 : 1 \leq i \leq m\}.$$

The *treewidth* of G , denoted by $\text{tw}(G)$, is the minimum width over all tree decompositions of G . A tree decomposition (T, W) is a *path decomposition* if T is a path; the *pathwidth* of a graph G , denoted by $\text{pw}(G)$, is the minimum width over all path decompositions of G . One can find more information on treewidth and related problems in the survey papers [23, 115].

Another related graph parameter is the vertex separation introduced by Ellis et al. [52]. A *layout* of a connected graph G is a one-to-one mapping $L: V \rightarrow \{1, 2, \dots, |V|\}$. Let $V_L(i) = \{x : x \in V, \text{ and there exists } y \in V \text{ such that } xy \in E, L(x) \leq i \text{ and } L(y) > i\}$. The *vertex separation of G with respect to L* , denoted by $\text{vs}_L(G)$, is defined as

$$\text{vs}_L(G) = \max\{|V_L(i)| : 1 \leq i \leq |V|\}.$$

The *vertex separation* of G is defined as

$$\text{vs}(G) = \min\{\text{vs}_L(G) : L \text{ is a layout of } G\}.$$

Yang [135] introduced the strong-mixed search game, which is a generalization of the mixed search game. The strong-mixed search game has the same setting as the mixed search game except that searchers have an extra power to clear a neighborhood subgraph; that is, the subgraph induced by $N[v]$ (a set contains v and its neighbors) is cleared if all vertices in $N(v)$ (a set contains only the neighbors of v) are occupied by searchers. The *strong-mixed search number* of G , denoted by $\text{sms}(G)$, is the smallest positive integer k such that k searchers can clear G . The following relationships were given in [87, 89, 135].

Theorem 4 ([87, 89, 135]) *If G is a connected graph, then*

$$\text{sms}(G) = \text{pw}(G) = \text{vs}(G) = \text{ns}(G) - 1.$$

For an edge search strategy, if the subgraph induced by cleared edges is connected in every step, then call it *connected search*. We denote by $\text{cs}(G)$ and $\text{mcs}(G)$, respectively, the connected and monotonic connected search numbers of a graph G defined in the natural way.

The following theorem from [141, 143] demonstrates that the connected search game is not monotonic.

Theorem 5 ([141, 143]) *For any positive integer k , there is a graph W_k such that $\text{cs}(W_k) = 280k + 1$ and $\text{mcs}(W_k) = 290k$.*

Fraigniaud and Nisse [57] investigated visible robber connected node search. They proved that recontamination does help as well to catch a visible robber in a connected way.

Barrière et al. [15] proved that $\text{mcs}(T) \leq 2\text{es}(T)$ for any tree T . Dereniowski [48] extended the result to graphs as follows.

Theorem 6 ([48]) *For any graph G , $\text{mcs}(G) \leq 2\text{es}(G) + 3$.*

Fomin et al. [63] introduced the domination search game, in which the robber hides only on vertices and searchers are placed or are removed from vertices of a graph. In the domination search game, searchers are more powerful than those in the node search game. A searcher “captures” the robber if she can “see” him; that is, a searcher on a vertex v can clear v and all its neighbors. The following theorem gives a relation between the *domination search number*, written $\text{dsn}(G)$, node search number, and the domination number, written $\gamma(G)$.

Theorem 7 ([63]) *For any graph G , let G' be a graph obtained from G by replacing every edge by a path of length three and H be a graph obtained from G by connecting every two nonadjacent vertices by a path of length three. Then,*

$$\text{ns}(G) = \text{dsn}(G') \text{ and } \gamma(G) \leq \text{dsn}(H) \leq \gamma(G) + 1.$$

Kreutzer and Ordyniak [92] extended the domination game to distance d -domination games, in which each searcher can see a certain radius d around her position. That is, a searcher on vertex v can see any other vertex within distance d of v , and if this vertex is occupied by the robber, then the searcher can see the robber and capture him. They showed that all questions concerning monotonicity and complexity about d -domination games can be reduced to the case of $d = 1$. They gave a class of graphs on which two searchers can win on any graph in this class, but the number of searchers required for monotone winning strategies is unbounded. Due to the following result, the domination search problem is much harder than standard graph search problems. For background on **PSPACE** and other complexity classes, see Spiser [125].

Theorem 8 ([92]) *The domination search problem is **PSPACE**-complete. Moreover, the problem of deciding whether two searchers have a winning strategy on a graph is **PSPACE**-complete.*

2.2 Robber Is Visible or Lazy

Dendris et al. [47] introduced the *lazy-robber game*, which has the same setting as the node searching game except that the robber stays only on vertices and moves just before a searcher is going to be placed on the vertex currently occupied by the robber (i.e., robber is *lazy*). When a searcher is going to be placed on a vertex u currently occupied by the robber, the robber can move from u to another vertex v if there is a searcher-free path from u to v ; otherwise, the robber is captured. The following result from Dendris et al. [47] shows that the lazy-robber game is monotonic and the search number of a graph in the lazy-robber game is equal to the treewidth of the graph plus one.

Theorem 9 ([47]) *Let G be a graph. For a lazy robber with unbounded speed, the monotonic search number of G is equal to the search number of G , and moreover, it is equal to $\text{tw}(G) + 1$.*

Dendris et al. [47] also considered the speed-limited lazy robber. They showed that if the speed is 1, then the search number minus 1 is equal to a graph parameter, called width, which is polynomial time computable for arbitrary graphs. Given a layout L of a connected graph G , the *width of a vertex* $v \in V$ with respect to the layout L is the number of vertices which are adjacent to v and precede v in the layout. The *width of the layout* L is the maximum width of a vertex of G . The *width of G* is the minimum width of a layout of G .

Theorem 10 ([47]) *Let G be a graph. For a lazy robber with speed 1, the monotonic search number of G is equal to the search number of G , and moreover, it is equal to the width of G plus 1.*

Seymour and Thomas [122] introduced another variant of graph searching, *visible-robber game*. Its setting differs from node search in that the robber stands only on vertices and is visible to searchers. They showed that the visible-robber game is monotonic and the search number of a graph in the visible-robber game is equal to the treewidth of the graph plus one. They also showed that if there is a non-losing strategy for the robber, then there is a “nice” non-losing strategy with a particularly simple form. In order to show these results, Seymour and Thomas introduced jump-search and haven.

For a graph G and $X \subseteq V$, let $G - X$ be the graph obtained from G by deleting X . The vertex set of a component of $G - X$ is called an X -flap. Let $[V]^{<k}$ be the set of all subsets of V of cardinality less than k . The following lemma describes a strategy for the robber.

Lemma 1 ([122]) *A graph G cannot be cleared by less than k searchers if and only if there is a function σ mapping each $X \in [V]^{<k}$ to a nonempty union $\sigma(X)$ of X -flaps, such that if $X \subseteq Y \in [V]^{<k}$, then $\sigma(X)$ is the union of all X -flaps which intersect $\sigma(Y)$.*

Two vertex sets $X, Y \subseteq V(G)$ touch if either $X \cap Y \neq \emptyset$ or some vertex in X has a neighbor in Y . In *jump-searching*, each searcher can jump from a vertex to another vertex. At the start of the i th step, searchers occupy $X_{i-1} \in [V]^{<k}$ and the robber is in the subgraph R_{i-1} , which is an X_{i-1} -flap. After some searchers jump, the searchers occupy $X_i \in [V]^{<k}$ and the robber chooses an X_i -flap R_i , which touches R_{i-1} . Havens correspond to particularly nice winning strategies for the robber. A *haven* of order k is a function β which assigns an X -flap $\beta(X)$ to each $X \in [V]^{<k}$, in such a way that $\beta(X)$ touches $\beta(Y)$ for all $X, Y \in [V]^{<k}$. A *screen* in G is a set of subsets of $V(G)$ such that each subset induces a connected subgraph of G and any two subsets touch each other. A screen S has thickness at least k if there is no $X \in [V]^{<k}$ such that $X \cap H \neq \emptyset$ for all $H \in S$. The following result from Seymour

and Thomas [122] demonstrates the relations among screen, haven, visible-robber search, monotonic visible-robber search, and treewidth.

Theorem 11 ([122]) *For a graph G and an integer $k \geq 1$, the following are equivalent:*

- (1) *G has a screen of thickness at least k .*
- (2) *G has a haven of order at least k .*
- (3) *Fewer than k searchers cannot jump-search clear G .*
- (4) *Fewer than k searchers cannot clear G ,*
- (5) *Fewer than k searchers cannot monotonically clear G ,*
- (6) *G has treewidth at least $k - 1$.*

The problem of determining whether k searchers can capture the robber in the visible-robber game (or lazy-robber game) is **NP**-complete since it is monotonic and computing treewidth (equivalently, partial k -tree) is **NP**-complete [11].

Theorem 12 ([11]) *Given a graph G and a positive integer k , the problem of determining whether the treewidth of G is at most k is **NP**-complete.*

There are many variants of graph searching games, for example, time constrained searching [7], network security searching [14], weighted graphs searching [144], robber-and-marshals games [73], geometric environment searching [128–130], and more searching games on undirected graphs which can be found in Sect. 4.

3 Searching Digraphs

3.1 Robber Is Visible or Lazy

Johnson et al. [82] generalized the concept of treewidth to digraphs. For a digraph D , let Z and S be two disjoint subsets of V . The set S is Z -normal if for every directed path in D with first and last vertices in S , all vertices of the path belong to $S \cup Z$. An arborecence is a directed tree T with edges oriented away from a unique vertex $r \in V(T)$ (called the root). We write $t > e$ for $t \in V(T)$ and $e \in E(T)$ if e occurs on the unique directed path from r to t and $e \sim t$ if e is incident with t . An arboreal decomposition of a digraph D is a triple (T, X, W) where T is an arborecence and $X = \{X_e \subseteq V(D) : e \in E(T)\}$ and $W = \{W_t \subseteq V(D) : t \in V(T)\}$ satisfy the following items:

- (1) W is a partition of $V(D)$ into nonempty sets.
- (2) If $e \in E(T)$, then $\bigcup\{W_t : t \in V(T) \text{ and } t > e\}$ is X_e -normal.

The width of an arboreal decomposition (T, X, W) is the minimum k such that for all $t \in V(T)$,

$$\left| W_t \cup \bigcup_{e \sim t} X_e \right| \leq k + 1.$$

The *directed treewidth* of D is the least integer k such that D has an arboreal decomposition of width k .

Johnson et al. [82] introduced a generalization of the visible-robber game called the *strong-directed visible-robber game*. In the strong-directed visible-robber game, the robber occupies only vertices and must obey the edge directions when he moves along edges. The searchers have two actions – placing on vertices and removing from vertices. The robber is visible, and the robber can move from vertex u to v if there is a searcher-free directed cycle containing u and v .

Theorem 13 ([82]) *Let D be a digraph and k be an integer. If D has directed treewidth less than k , then k searchers have a winning strategy in the strong-directed visible-robber game.*

Johnson et al. [82] gave an example to show that the winning strategy in **Theorem 13** need not be searcher-monotone in the sense that searchers may have to revisit certain vertices. Adler [1] showed that it may not be robber-monotone either. She constructed a digraph where four searchers have a winning strategy but they have no robber-monotone winning strategy. This is very different from the undirected case, where a graph that has treewidth less than k implies that k searchers have a winning strategy where no vertex is revisited once it has been vacated. The following theorem gives a relation between the treewidth and directed treewidth.

Theorem 14 ([82]) *Let G be a graph, and let D be the digraph obtained from G by replacing every edge with two directed edges directed in opposite directions. Then the directed treewidth of D is equal to the treewidth of G .*

Let $k \geq 1$ be an integer. A *haven* of order k in a digraph D is a function β assigning to every $Z \subseteq V(D)$ with $|Z| < k$ the vertex set of a strong component of $D - Z$ in such a way that if $Z' \subseteq Z \subseteq V(D)$ with $|Z'| < k$, then $\beta(Z) \subseteq \beta(Z')$. In the strong-directed visible-robber game, if $k - 1$ searchers have a winning strategy on the digraph D , then D has no haven of order k . If β is a haven of order k in D , then the robber wins against $k - 1$ searchers by staying in $\beta(Z)$, where Z is the set of vertices occupied by the searchers. Johnson et al. [82] proved the following relation between the directed treewidth and haven.

Theorem 15 ([82]) *Let D be a digraph and k be a positive integer. If D has a haven of order k , then its directed treewidth is at least $k - 1$.*

Adler [1] showed that the converse of **Theorem 15** is not true by constructing a digraph whose directed treewidth is at least 4, but it has no haven of order 5. Johnson et al. [82] showed the following upper bound.

Theorem 16 ([82]) *Let D be a digraph and k be a positive integer. Then either D has directed treewidth at most $3k - 2$ or it has a haven of order k .*

Corollary 1 ([82]) *Let D be a digraph and k be a positive integer. Then either D has directed treewidth at most $3k - 1$ or k searchers do not have a winning strategy in the strong-directed visible-robber game on D .*

Berwanger et al. [19] and Obdržálek [108] introduced another width measure for digraphs called *DAG-width*. Given a digraph D , a set $X \subseteq V$ guards a set $Y \subseteq V$ if $X \cap Y = \emptyset$ and whenever there is an edge $(u, v) \in E$ such that $u \in Y$ and $v \notin X$, then $v \in X$. A DAG-decomposition of D is a pair (T, W) with T an acyclic digraph and $W = \{W_t \subseteq V : t \in V(T)\}$ a family of nonempty subsets, such that the following hold:

- (1) $\bigcup_{t \in V(T)} W_t = V$.
- (2) For all $t, t', t'' \in V(T)$, if t' is on a directed path from t to t'' in T , then $W_t \cap W_{t''} \subseteq W_{t'}$.
- (3) For all edges $(t, t') \in E(T)$, $W_t \cap W_{t'}$ guards $W_{\geq t'} \setminus W_t$, where $W_{\geq t'} = \bigcup\{W_{t''} : t'' \in V(T) \text{ and there is a directed path from } t' \text{ to } t'' \text{ in } T\}$. For all roots $r \in V(T)$, $W_{\geq r}$ is guarded by \emptyset .

The *width* of a DAG-decomposition (T, W) is

$$\max\{|W_t| : t \in V(T)\}.$$

The *DAG-width* of D is the minimum width of any of its DAG-decompositions.

Berwanger et al. [19] and Obdržálek [108] introduced the *directed visible-robber game*, which has the same setting as the strong-directed visible-robber game except that the robber is more powerful. The robber is allowed to move from vertex u to v if there is a searcher-free directed path from u to v . They use this game to characterize the DAG-width.

The directed visible-robber game and the strong-directed visible-robber game have some different properties owing to the different behavior of robbers. One difference is that the search number in the strong-directed visible-robber game is invariant under edge reversal; that is, it does not change if the directions of all edges of the digraph are reversed. But this is not the case for the directed visible-robber game. Another difference is that for the strong-directed visible-robber game with k searchers, there are digraphs that have a robber-monotone winning strategy, but no searcher-monotone winning strategy. However, Berwanger et al. [19] proved the following.

Theorem 17 ([19]) *For the directed visible-robber game with k searchers, if the searchers have a searcher-monotone or robber-monotone winning strategy, then they also have a winning strategy that is both searcher- and robber-monotone.*

A *monotone strategy* is a strategy that is both searcher-monotone and robber-monotone. Berwanger et al. [19] and Obdržálek [108] established the following relation between the search number and the DAG-width.

Theorem 18 ([19, 108]) *Let D be a digraph and k be a positive integer. Then D has a DAG-decomposition of width k if and only if k searchers have a monotone winning strategy in the directed visible-robber game on D .*

Similar to Theorem 14, Berwanger et al. proved the following.

Theorem 19 ([19]) *Let G be a graph, and let D be the digraph obtained from G by replacing every edge with two directed edges directed in opposite directions. Then the DAG-width of D is equal to the treewidth of G minus one.*

Berwanger et al. [19] also proved that if a digraph has DAG-width k , its directed treewidth is at most $3k + 1$. However, there is a family of digraphs with directed treewidth one and arbitrarily large DAG-width. Kreutzer and Ordyniak [91] proved that the directed visible-robber game is non-monotone.

Theorem 20 ([91]) *For any positive integer $k \geq 2$, there is a digraph D_k such that in the directed visible-robber game on D_k , the monotonic search number of D_k is $4k - 2$ and the search number of D_k is $3k - 1$.*

Hunter and Kreutzer [80] extended the lazy-robber game to the *directed lazy-robber game* on digraphs, which has the same setting as the lazy-robber game on graphs except that the robber must obey the edge directions when he moves along edges. They use this game to characterize the digraph width measure called *Kelly-width*.

Given a digraph D , a *Kelly-decomposition* of D is a triple (T, B, W) with T an acyclic digraph, $B = \{B_t \subseteq V : t \in V(T)\}$, and $W = \{W_t \subseteq V : t \in V(T)\}$, such that the following properties hold:

- (1) The set B is a partition of V into nonempty sets.
- (2) For all $t \in V(T)$, W_t guards $B_{\geq t}$, where $B_{\geq t} = \bigcup\{B_{t'} : t' \in V(T)$ and there is a directed path from t to t' in $T\}$.
- (3) For all $s \in V(T)$, there is a linear order on its children t_1, \dots, t_p such that for all $1 \leq i \leq p$,

$$W_{t_i} \subseteq B_s \cup W_s \cup \bigcup_{j < i} B_{\geq t_j}.$$

Similarly, there is a linear order on roots r_1, \dots, r_q of T such that for all $1 \leq i \leq q$,

$$W_{r_i} \subseteq \bigcup_{j < i} B_{\geq r_j}.$$

The *width* of a Kelly-decomposition (T, B, W) is

$$\max\{|B_t \cup W_t| : t \in V(T)\}.$$

The *Kelly-width* of D is the minimum width of any of its Kelly-decompositions.

The following result from Hunter and Kreutzer [80] shows that k searchers have a monotone winning strategy to capture a lazy robber in a digraph D if and only if D has Kelly-width k .

Theorem 21 ([80]) *Let D be a digraph and k be a positive integer. Then D has a Kelly-decomposition of width k if and only if k searchers have a monotone winning strategy in the directed lazy-robbler game on D .*

Hunter and Kreutzer [80] gave the following relation between the directed lazy-robbler game and the directed visible-robbler game.

Theorem 22 ([80]) *Let D be a digraph and k be a positive integer. If k searchers have a robber-monotone winning strategy in the directed lazy-robbler game on D , then $2k - 1$ searchers have a winning strategy in the directed visible-robbler game on D .*

Theorem 23 ([80]) *Let D be a digraph and k be a positive integer. If k searchers have a monotone winning strategy in the directed visible-robbler game on D , then k searchers have a winning strategy in the directed lazy-robbler game on D .*

Kreutzer and Ordyniak [91] proved that the directed lazy-robbler game is non-monotone.

Theorem 24 ([91]) *For any positive integer $k \geq 2$, there is a digraph D_k such that in the directed lazy-robbler game on D_k , the monotonic search number of D_k is $7k$ and the search number of D_k is $6k$.*

The problem of determining whether k searchers can capture the robber in the directed visible-robbler game (resp. directed lazy-robbler game or strong-directed visible-robbler game) is NP-hard.

Besides the above width measures, several new digraph measures are introduced recently, which include K-width [70], DAG-depth [70], entanglement [18], D-width [119], directed pathwidth [13], directed vertex separation [140], and bi-rank-width [84]. See also the thesis of Hunter for more discussions [79].

3.2 Robber Is Invisible and Active

Reed, Seymour, and Thomas introduced the directed pathwidth. Given a digraph D , a directed path decomposition of D is a sequence of subsets of vertices W_1, W_2, \dots, W_m such that the following hold:

- (1) $\bigcup_{i=1}^m W_i = V$.
- (2) For $1 \leq i < j < k \leq m$, $W_i \cap W_k \subseteq W_j$.
- (3) For each edge in D , it either has both endpoints in the same W_i or has its tail in W_i and head in W_j , where $i < j$.

The width of a directed path decomposition of D is

$$\max_{1 \leq i \leq m} \{|W_i| - 1\}.$$

The *directed pathwidth* of D , denoted by $\text{dpw}(D)$, is the minimum width over all directed path decompositions of D .

Barát [13] introduced a generalization of the node searching, called the *directed node search game*, to characterize the directed pathwidth. The directed node search game has the same setting as the node search game on graphs except that the robber occupies only vertices and must obey the edge directions when he moves along edges. He proved that an optimal monotonic search strategy for a digraph needs at most one more searcher than the search number of the digraph, and he also proved that the directed pathwidth and the directed node search number differ by at most one. Specifically, he proved the following theorem.

Theorem 25 ([13]) *For a digraph D and an integer $k \geq 1$, the following downward implications apply:*

- (1) *There is a monotone capture of the robber in D with at most k searchers.*
- (2) *The directed pathwidth of D is at most $k - 1$.*
- (3) *There is a capture of the robber in D with at most k searchers.*
- (4) *There is a monotone capture of the robber in D with at most $k + 1$ searchers.*

Yang and Cao [137–139] generalized the edge searching problem to digraphs by introducing three digraph search games: directed search, strong search, and weak search. In all three games, the robber is invisible and searchers have three types of actions: placing, removing, and sliding. These games differ in the abilities of the searchers and robber depending on whether or not they must obey the edge directions. In the *directed search game*, both searchers and robber must move in the edge directions; in the *strong search game*, the robber must move in the edge directions but searchers need not; and in the *weak search game*, searchers must move in the edge directions but the robber need not. In particular, in the directed and strong search games, the robber can move from vertex u to vertex v along a searcher-free directed path from u to v at a great speed at any time, and in the weak search game, the robber can move from vertex u to vertex v along a searcher-free undirected path between u and v at a great speed at any time.

For a digraph D , let $\text{ds}(D)$, $\text{ss}(D)$, and $\text{ws}(D)$ denote the *directed*, *strong*, and *weak search number*, respectively. Let $\text{mds}(D)$, $\text{mss}(D)$, and $\text{mws}(D)$ denote the monotonic directed, monotonic strong, and monotonic weak search number, respectively. The following result shows that the directed, strong, and weak search problems are all monotonic and **NP**-complete.

Theorem 26 ([137–139]) Let D be a digraph and k be a positive integer. Then $\text{mds}(D) = \text{ds}(D)$, $\text{mss}(D) = \text{ss}(D)$, and $\text{mws}(D) = \text{ws}(D)$. Moreover, the problem of deciding whether $\text{ds}(D) \leq k$, $\text{ss}(D) \leq k$, or $\text{ws}(D) \leq k$ is NP-complete.

Let $\text{es}(D)$ be the edge search number of the underlying undirected graph of D . Yang and Cao gave the following relationships between search numbers.

Theorem 27 ([137–139]) Let D be a digraph. Then the following inequalities hold:

- (1) $\text{ss}(D) \leq \text{ds}(D) \leq \text{ws}(D)$.
- (2) $\text{ss}(D) \leq \text{es}(D) \leq \text{ws}(D)$.
- (3) $\text{ds}(D) - 1 \leq \text{ss}(D) \leq \text{ds}(D)$.
- (4) $\text{ds}(D) \leq \text{es}(D) + 1$.
- (5) $\text{ws}(D) \leq \text{es}(D) + 2$.

Nowakowski [106] and Alspach et al. [5] introduced another three digraph search games: internal directed search, internal strong search, and internal weak search, which have the same setting as the directed search, strong search, and weak search, respectively, except that in the internal games, searchers have only two types of actions: placing and sliding. Yang and Cao [138] proved that the internal directed search problem is monotonic; however, the internal strong and internal weak search problems are not monotonic. Although the internal strong search game is not monotonic, they still proved it is in NP. But the problem of whether the internal weak search game is in NP is still open.

Theorem 28 ([138]) Given a digraph D and an integer k , the problem of determining whether k searchers can capture the robber in the internal directed (or strong) search game is NP-complete, and the problem of determining whether k searchers can capture the robber in the internal weak search game is NP-hard.

Yang and Cao [140] extended the vertex separation to digraphs. Given a digraph D and a linear layout $L : V \rightarrow \{1, 2, \dots, |V|\}$, let $DV_L(i) = \{x \in V : \text{there exists } y \in V \text{ such that edge } (y, x) \in E \text{ and } L(x) \leq i \text{ and } L(y) > i\}$. The directed vertex separation of D with respect to L , denoted by $\text{dvs}_L(D)$, is defined as

$$\text{dvs}_L(D) = \max\{|DV_L(i)| : 1 \leq i \leq |V|\}.$$

The directed vertex separation of D is defined as $\text{dvs}(D) = \min\{\text{dvs}_L(D) : L \text{ is a linear layout of } D\}$. The following relationships were given in [140].

Theorem 29 ([140]) Let D be a digraph. Then the following inequalities hold:

- (1) $\text{dvs}(D) = \text{dpw}(D)$.
- (2) $\text{dvs}(D) + 1 \leq \text{ds}(D) \leq \text{dvs}(D) + 2$.
- (3) $\text{dvs}(D) \leq \text{ss}(D) \leq \text{dvs}(D) + 2$.

4 Other Searching Games

4.1 Mixed Searching with Multiple Robbers

As mentioned in Sect. 2.1, Bienstock and Seymour [22] introduced the mixed search game, in which searchers have three actions: placing, removing, and sliding. The mixed search game can be used to characterize the proper pathwidth introduced by Takahashi et al. [131].

A graph G is a *minor* of graph H if G can be obtained from a subgraph of H by edge contractions, where a contraction of edge uv is the deletion of uv , followed by identifying u and v such that all vertices adjacent to u or v are adjacent to the new vertex. The *proper pathwidth* of a graph G , denoted as $\text{ppw}(G)$, is the least integer k such that G is a minor of the graph $K_k \square P$ for some path P , where K_k is a clique of order k and \square is the Cartesian product operator. Similarly, the *proper treewidth* of a graph G , denoted as $\text{ptw}(G)$, is the least integer k such that G is a minor of the graph $K_k \square T$ for some tree T .

The following result from Takahashi et al. [132] shows the relation between the mixed search number and the proper pathwidth.

Theorem 30 ([132]) *For any graph G , $\text{ms}(G) = \text{ppw}(G)$.*

Richerby and Thilikos [116] studied the mixed search game with multiple robbers who stay only on vertices. A robber is captured if a searcher and the robber occupy the same vertex of G . A set of searchers win the game if they can capture all robbers; otherwise, robbers win.

For a graph G with $\text{ms}(G) = k$, if monotonicity is ignored, then k searchers can capture any number of visible active robbers one at a time by repeating the strategy to capture a single robber. So, monotonicity is crucial in the mixed search game with multiple robbers. The robbers' territory F_i at step i is the union of each robber's territory. A search strategy is *monotonic* if $F_{i+1} \subseteq F_i$ for all i . Let $\text{mvams}(G, r)$ denote the *monotonic mixed search number for r visible active robbers* in G , $\text{vams}(G, r)$ denote the *mixed search number for r visible active robbers* in G , $\text{milms}(G, r)$ denote the *monotonic mixed search number for r invisible lazy robbers* in G , and $\text{ilms}(G, r)$ denote the *mixed search number for r invisible lazy robbers* in G . Richerby and Thilikos showed the following relations.

Theorem 31 ([116]) *For any graph G with n vertices:*

- (1) $\text{mvams}(G, n) = \text{ms}(G) = \text{ppw}(G)$.
- (2) $\text{mvams}(G, 1) = \text{milms}(G, 1)$.
- (3) $\text{ilms}(G, r) = \text{ilms}(G, 1)$.
- (4) $\text{milms}(G, r) = \text{milms}(G, 1) = \text{ptw}(G)$.
- (5) $\text{mvams}(G, r) \leq \min\{\text{ppw}(G), \text{ptw}(G)(\lfloor \log r \rfloor + 1)\}$.

Theorem 32 ([116]) *For any tree T with r robbers,*

$$\text{mvams}(T, r) = \min\{\text{ppw}(T), \lfloor \log r \rfloor + 1\}.$$

4.2 Nondeterministic Graph Searching

Fomin et al. [65] introduced nondeterministic graph searching, which can be used in algorithm design and combinatorial analysis applying to both pathwidth and treewidth. In the nondeterministic graph searching, the searchers and robber stay only on vertices. There are three actions for searchers: placing a searcher on a vertex, removing a searcher from a vertex, and query the oracle that returns the connected component containing the robber. The robber has the same behavior as the node search game. For a nonnegative integer q , a q -limited nondeterministic searching is a nondeterministic searching that performs at most q query actions. Therefore, k searchers win a q -limited nondeterministic search game against a robber if they can capture the robber by querying the oracle at most q times. The q -limited nondeterministic search number of a graph G is the minimum number of searchers required to win the game.

Fomin et al. [65] extended the tree decomposition and treewidth to q -branched tree decomposition and q -branched treewidth. They used the nondeterministic graph search game to characterize the q -branched treewidth. A *rooted tree decomposition* of a graph G is a tree decomposition (T, W) of G in which T is a rooted tree and W is a family of nonempty subsets of $V(G)$ satisfying the three conditions of a tree decomposition. A *branching node* of a rooted tree decomposition is a node with at least two children. For a nonnegative integer q , a *q -branched tree decomposition* of a graph G is a rooted tree decomposition (T, W) of G such that every path in T from the root to a leaf contains at most q branching nodes. The *q -branched treewidth* of G , is the minimum width of any q -branched tree decomposition of G . The following theorem from Fomin et al. [65] shows the relation between the nondeterministic graph search game and the q -branched treewidth.

Theorem 33 ([65]) *For a graph G , a nonnegative integer q , and a positive integer k , at most k searchers win the q -limited nondeterministic search game in a monotonic way if and only if the q -branched treewidth is less than k .*

Fomin et al. used the q -limited nondeterministic graph searching to design an exact exponential-time algorithm for computing the q -branched treewidth of a graph.

Theorem 34 ([65]) *For a graph G with n vertices, there exists an algorithm that computes the q -branched treewidth and its corresponding optimal q -branched tree decomposition of G in time $O(2^n n \log n)$.*

Mazoit and Nisse [99] showed that the q -limited nondeterministic graph searching is monotone for any nonnegative integer q .

Theorem 35 ([99]) *For a graph G , a nonnegative integer q and an integer $k \geq 2$, k searchers win the q -limited nondeterministic search game if and only if they can do it in a monotonic way.*

4.3 Fast Searching

In the edge search game, the goal is to find the minimum number of searchers to clear a given graph. Yang [136] introduced a new game called the *fast edge searching*, which has the same setting as the edge search game except the goal. In the fast edge search game, the goal is to find the minimum number of steps (or equivalently, actions) to clear a given graph. The fast edge search game has a strong connection with the fast search game, which was first introduced by Dyer et al. [51]. The fast search game has the same setting as the edge search game except that every edge is traversed exactly once by a searcher and searchers cannot be removed.

The motivation to consider the fast edge searching and fast searching is that, in some real-life scenarios, the cost of a searcher may be relatively low in comparison to the cost of allowing a fugitive to be free for a long period of time. For example, if a dangerous fugitive hiding along streets in an area, policemen always want to capture the fugitive as soon as possible.

In the fast edge searching game, the minimum number of steps required to clear G is the *fast edge search time* of G , denoted by $\text{fet}(G)$, and the minimum number of searchers required so that G can be cleared in $\text{fet}(G)$ steps is the *fast edge search number* of G , denoted by $\text{fen}(G)$. Similarly, the minimum number of searchers required to clear G in the fast search game is the *fast search number* of G , denoted by $\text{fsn}(G)$. A difference between the two games is that the family of graphs $\{G : \text{fet}(G) \leq k\}$ is minor-closed for the fast edge searching, but the family of graphs $\{G : \text{fsn}(G) \leq k\}$ is not minor-closed for the fast searching. The difference between $\text{fen}(G)$ and $\text{fsn}(G)$ can be large. In fact, there exists a class of graphs H such that the ratio $\text{fsn}(H)/\text{fen}(H)$ is arbitrarily large. Yang gave the following relationship between the fast edge searching game and the fast searching game.

Theorem 36 ([136]) *For any graph G ,*

$$\text{fet}(G) = \text{fsn}(G) + |E|.$$

For a graph G , let G' be a graph obtained from G by adding a vertex a and connecting it to each vertex of G . Let A'_G be a multigraph obtained from G' by replacing each edge with four parallel edges. Let A_G be a graph obtained from A'_G by replacing each edge of A'_G with a path of length two. Then, the following relation between the node search number of G and the fast search number of A_G was given in [136].

Theorem 37 ([136]) *For a graph G and its corresponding graph A_G ,*

$$\text{ns}(G) = \text{fsn}(A_G) - 2 = \text{fen}(A_G) - 2.$$

Although the node search number and the fast search number have the above relation, Stanley and Yang [126] presented a linear time algorithm for computing the fast search number of cubic graphs, while it is **NP**-complete to find the node search number of cubic graphs [98].

Corollary 2 *The fast search problem and the fast edge search problem are **NP**-complete. They remain **NP**-complete for Eulerian graphs.*

Yang showed the following bounds for the fast search number.

Theorem 38 ([136]) *For a connected graph G with minimum degree at least 3,*

$$\max \left\{ \delta(G) + 1, \left\lceil \frac{\delta(G) + |V_{\text{odd}}(G)| - 1}{2} \right\rceil \right\} \leq \text{fsn}(G) \leq |E|,$$

where $\delta(G)$ is the minimum degree of G and $V_{\text{odd}}(G)$ is the set of all odd vertices in G .

The fast searching game has a close relation with the graph brushing problem [3, 71, 101] and the balanced vertex-ordering problem [20, 85]. For all graphs, the brush number is equal to the total imbalance of an optimal vertex-ordering. For some graphs, such as trees, the fast search number is equal to the brush number.

4.4 Graph Guarding

Fomin et al. [64] introduced the graph guarding games, in which a set of cops want to guard a region in a given graph against a robber. The robber and cops stay only on vertices of the graph, and both sides take alternative turns to play. All participants have complete information on the location of all other participants. Initially, the robber is placed on a vertex outside the protected region, and then all cops are placed on vertices inside the protected region. In robber's turn, he can move to any adjacent vertex. The robber wins if he can move to an unguarded vertex in the region. In the cops' turn, each cop can move to an adjacent vertex in the region or stay put. The cops win if they can forever prevent the robber to win. The guarding problem is to find the minimum number of cops needed to win the game.

Fomin et al. [64] showed that the guarding problem is polynomial time solvable if the robber's region is a path. Nagamochi [105] showed that the guarding problem is polynomial time solvable if the robber's region is a cycle.

Theorem 39 ([64]) *For a graph $G = (V, E)$, let C be the protected region (vertex set). If $V \setminus C$ induces a path in G , then the guarding problem can be solved in $O(n_1 n_2 m)$ time, where $n_1 = |C|$, $n_2 = |V \setminus C|$, and $m = |E|$.*

Fomin et al. showed that the guarding problem is **NP-hard** even if the robber's region is a star.

Theorem 40 ([64]) *Given a graph $G = (V, E)$, a protected region C (vertex set), and an integer k , it is **NP-hard** to decide whether k cops can win the guarding game. It remains **NP-hard** even if $V \setminus C$ induces a star. Moreover, the parameterized version of the problem with k (the number of cops) being a parameter is **W[2J-hard**.*

The guarding game can also play on digraphs, in which the robber and cops always follow edge directions when they move to neighbors.

Theorem 41 ([64]) *Given a digraph $D = (V, E)$, a protected region C (vertex set), and an integer k , if $V \setminus C$ induces an acyclic digraph, it is **PSPACE-complete** to decide whether k cops can win the guarding game.*

In the guarding game, the robber moves first. In [66], Fomin et al. also studied a variant of the guarding game, in which cops move first. They showed that the new guarding problem is **PSPACE-hard** on undirected graphs.

4.5 Minimum Cost Searching

Almost all graph search problems are to find search strategies such that the maximum number of searchers used at each step is minimized. Fomin and Golovach [59] proposed a different optimization criterion. They introduced a cost function in the node searching game, which is the sum of the number of searchers in every step of the node search process.

One can interpret the cost of a search as the total sum that searchers earn for doing their job. The *search cost* of a graph G , denoted by $\sigma(G)$, is the minimum cost of a search where the minimum is taken over all searches on G . For monotonic node searching, the *monotonic search cost* of G , denoted by $\sigma_m(G)$, can be defined similarly. Fomin and Golovach [59] proved that their minimum cost searching is monotonic.

Theorem 42 ([59]) *For any graph G , $\sigma(G) = \sigma_m(G)$.*

Kirousis and Papadimitriou [88] found a relation between node searching and interval graphs. An interval graph is one that has an interval model, which is a set of intervals of the real line, one for each vertex, such that two intervals intersect if and only if the corresponding vertices are adjacent. Every graph is a subgraph of an interval graph in a trivial way. The *interval thickness* of a graph G , denoted by $\theta(G)$, is the smallest max-clique over all interval supergraphs of G .

Theorem 43 ([88]) *For any graph G , $\text{ns}(G) = \theta(G)$.*

Fomin and Golovach proved the following relation between minimum cost searching and interval graphs.

Theorem 44 ([59]) *For any graph G and positive integer k , $\sigma(G) \leq k$ if and only if there is an interval supergraph I of G such that $E(I) \leq k$.*

Dyer et al. [51] introduced the following cost function in the edge searching game on a graph G :

$$C_G(s, t) = \alpha s + \beta st + \gamma t,$$

where $\text{es}(G) \leq s \leq \text{fsn}(G)$ and $t = t(s) \geq |E|$. The goal is to minimize the cost function C_G , instead of trying to minimize s , which corresponds to edge searching, or to minimize t , which corresponds to fast searching.

4.6 Pebbling

Pebble games on graphs and digraphs have been studied by both mathematicians and computer scientists [16, 41, 46, 77, 89, 102, 103]. Pebbling is a method of analyzing computational situations, especially situations in which notions such as time (number of operations) and space (number of memory locations) are of interest [45, 78, 83, 86, 90, 94]. We focus on the relationship between graph searching and pebbling [89].

Let D be an acyclic digraph. For an edge $(u, v) \in E(D)$, v is called a *successor* of u , and u is called a *predecessor* of v . We consider two versions of pebbling: one is black pebbling and the other is black and white pebbling. The rules for *black pebbling* are as follows:

- (1) All vertices of D start pebble-free.
- (2) All vertices of D end pebble-free.
- (3) Each vertex receives and loses a pebble at least once.
- (4) A pebble can be placed on a vertex v only if all the predecessors of v have a pebble. Vertices with zero in-degree can be pebbled at any time.
- (5) A pebble can be removed any time.

The rules for *black and white pebbling* are as follows:

- (1) All vertices of D start pebble-free.
- (2) All vertices of D end pebble-free.
- (3) Each vertex receives and loses a pebble at least once.
- (4) A white pebble can be placed on a vertex at any time.
- (5) A white pebble on vertex v turns black at the moment all the predecessors of v are pebbled.
- (6) A black pebble can be removed at any time, but no white pebble can be removed.

The principle application of pebbling is to model computations. We say that a vertex has been pebbled if it had a pebble placed on it at some point. A pebbling scheme is a sequence of placing and removing pebbles so that every vertex has

been pebbled. The size of a pebbling scheme is the maximum number of pebbles deployed at any step. The *black pebble number* of an acyclic digraph D , denoted $\text{bp}(D)$, is the smallest size of a black pebbling scheme. Similarly, the *black and white pebble number* of an acyclic digraph D , denoted $\text{bwp}(D)$, is the smallest size of a black and white pebbling scheme. A black pebbling scheme, or black and white pebbling scheme, is said to be *monotonic* if each vertex is pebbled only one time. The *monotonic black pebble number*, or *monotonic black and white pebble number*, for an acyclic digraph D is denoted by $\text{mbp}(D)$ or $\text{mbwp}(D)$, respectively. The following relation between the node search number and monotonic black pebble number or monotonic black and white pebble number was proved in [89].

Theorem 45 ([89]) *Given a graph G , let $\Omega(G)$ denote the set of acyclic digraphs that are orientations of G . Let $\text{mmbp}(G)$ be the minimum monotonic black pebble number over all digraphs in $\Omega(G)$ and $\text{mmbw}(G)$ be the minimum monotonic black and white pebble number over all digraphs in $\Omega(G)$. Then the following relationships hold:*

- (1) $\text{mmbp}(G) = \text{ns}(G) = \text{mmbw}(G)$.
- (2) *For all $D \in \Omega(G)$ with in-degree at most k ,*

$$\text{bwp}(D) \leq \text{mbwp}(D) \leq (k + 1)\text{ns}(G).$$

5 Open Problems on Searching Games

The area of graph searching games is relatively new; there are several open problems. We present a dozen of them, and note that more problems can be found in the references:

- (1) The problem of determining the edge search number is **NP**-complete [93, 100]. This problem remains **NP**-complete for graphs with a maximum vertex degree of 3 [98]. However, whether the problem remains **NP**-complete for planar graphs is still unknown. In fact, the complexity of determining the search number of planar graphs in all search games mentioned in Sect. 2 is unknown.
- (2) The problems of designing efficient polynomial time approximation algorithms for computing the search number of all search games mentioned in Part 1 are wide open. There are only few results for special classes of graphs. For example, Bodlaender and Fomin [24] gave an $O(n \log n)$ time 2-approximation algorithm for computing the pathwidth (or node search number) of outerplanar graphs.
- (3) Finding good lower bounds for search numbers is a challenge for all search games mentioned in this chapter. There are few results for lower bounds. For example, it is not clear how to improve the following lower bound given by Alspach et al. [6]:

$$\text{es}(G) \geq \max \{\delta(G) + g(G) - 2, \chi(G) - 1\},$$

where the minimum vertex degree $\delta(G) \geq 3$, the girth $g(G) \geq 3$, and the chromatic number $\chi(G) \geq 4$.

- (4) Megiddo et al. [100] showed that the edge search problem is **NP**-hard. This problem belongs to **NP** by the monotonicity result of [93], in which LaPaugh showed that recontamination does not help to clear a graph. Using a reduction from the edge search problem, it can be easily shown that the connected edge search problem is also **NP**-hard. However, Yang et al. [141, 143] showed that the connected search is not monotonic. An interesting problem left open is whether the connected edge search problem belongs to **NP**.
- (5) In [82], Johnson et al. gave a min-max theorem between directed treewidth and the number of searchers required to capture a robber in the strong-directed visible-robbert game. However, the directed treewidth and the search number in this theorem are given by a constant factor between them. Whether there is a modified version of the directed treewidth that has an exact min-max theorem with the search number in the associated game remains an open problem.
- (6) In [91], Kreutzer and Ordyniak gave examples showing that neither the directed visible-robbert game associated with DAG-width nor the directed lazy-robbert game associated with Kelly-width is monotone. It is not clear if the ratio of the monotonic search number to the search number is bounded by a constant in both games. That is, it is unknown if there is a constant c such that whenever k searchers have a winning strategy in one of the games, ck searchers have a monotone winning strategy.
- (7) In [80], Hunter and Kreutzer conjectured that for a digraph, the search number in the directed visible-robbert game and that in the directed lazy-robbert game lie within constant factors of one another.
- (8) Yang and Cao [138] proved that the internal strong and internal weak search problems are not monotonic. Although the internal strong search game is not monotonic, they still proved it is in **NP**. However, the problem of whether the internal weak search game is in **NP** is still open.
- (9) In [138], Yang and Cao showed by examples that the ratio of the monotonic internal strong search number to the internal strong search number may be as large as $\Omega(\log n)$, where n is the number of vertices in the digraph. They conjecture that $O(\log n)$ is an upper bound of the ratio for all digraphs. Another open problem is whether there is a constant c such that whenever k searchers have a winning strategy in the internal weak search game, then ck searchers have a monotone winning strategy. We conjecture that this constant is less than 2 for all digraphs.
- (10) In [139], Yang and Cao proved the monotonicity of the mixed weak searching problem using the crusade method proposed by Bienstock and Seymour [22]. But they applied the LaPaugh's method to prove the monotonicity of the weak searching problem. Since LaPaugh's method is more complicated than the crusade method, an interesting open problem is how to establish a relation between the mixed weak searching and the weak searching so that the monotonicity of the former implies the monotonicity of the latter.

- (11) Many relations between different search numbers are described by inequalities, such as [Theorems 1](#) and [27](#). The related open problems are to find the necessary and sufficient conditions for graphs or digraphs such that equalities hold. For example, for which graphs does the edge search number equal the node search number?
- (12) Bodlaender and Kloks [\[25\]](#) gave a polynomial time algorithm for computing the pathwidth of a graph with constant treewidth. Since the edge search number of a graph equals the pathwidth of its two-expansion, the edge search number of a graph with constant treewidth is polynomial time computable. However, the exponent in the running time of this algorithm is large. Even for a graph with treewidth two, it takes at least $\Omega(n^{11})$ time. The large exponent makes their algorithm impractical. Megiddo et al. [\[100\]](#) presented a linear time algorithm for computing the edge search number of trees, and Peng et al. [\[110\]](#) proposed a linear time algorithm to compute the optimal search strategy of trees. Yang et al. [\[142\]](#) presented a linear time algorithm for computing the edge search number of unicyclic graphs and some special cycle-disjoint graphs. The problem then, is how to design efficient algorithms for computing the search number of graphs with small treewidth, such as outerplanar graphs that have treewidth two.

Part 2. Cops and Robbers

Cops and Robbers is one aspect of graph searching that has received much recent attention. In this two-player game of perfect information, a set of cops tries to capture a robber by moving at unit speed from vertex to vertex. More precisely, *Cops and Robbers* is a game played on a reflexive graph (i.e., there is a loop at each vertex). There are two players consisting of a set of *cops* and a single *robber*. The game is played over a sequence of discrete time-steps or *rounds*, with the cops going first in round 0 and then playing alternate time-steps. The cops and robber occupy vertices; for simplicity, the player is identified with the vertex they occupy. The set of cops is referred to as C and the robber as R . When a player is ready to move in a round, they must move to a neighboring vertex. Because of the loops, players can *pass* or remain on their own vertex. Observe that any subset of C may move in a given round.

The cops win if after some finite number of rounds, one of them can occupy the same vertex as the robber (in a reflexive graph, this is equivalent to the cop landing on the robber). This is called a *capture*. The robber wins if he can evade capture indefinitely. A *winning strategy for the cops* is a set of rules that if followed, result in a win for the cops. A *winning strategy for the robber* is defined analogously.

If a cop is placed at each vertex, then the cops are guaranteed to win. Therefore, the minimum number of cops required to win in a graph G is a well-defined positive integer, named the *cop number* (or *copnumber*) of the graph G . The notation $c(G)$ is used for the cop number of a graph G . If $c(G) = k$, then G is *k-cop-win*. In the special case $k = 1$, G is *cop-win* (or *copwin*).

The game of Cops and Robbers was first considered by Quilliot [\[113\]](#) in his doctoral thesis and was independently considered by Nowakowski and Winkler [\[107\]](#).

Both [107, 113] refer only to one cop. The introduction of the cop number came in 1984 with Aigner and Fromme [2]. Many papers have now been written on cop number since these three early works; see the surveys [4, 74] and the recent book of Bonato and Nowakowski [29]. Cops and Robbers has even found recent application in artificial intelligence and so-called *moving target search*; see Isaza et al. [81] and Moldenhauer et al. [104].

Part 2 is a selective survey of recent results on Cops and Robbers, focusing on algorithmic and probabilistic results, as well as the game in infinite graphs. A much more exhaustive survey with proofs may be found in the book [29]. This part begins with the case of cop-win graphs in Sect. 6. Such graphs have a good characterization in terms of an elimination ordering of their vertices. In Sect. 7 a discussion of graphs with higher cop number is provided. The highlight here is Meyniel's conjecture, which gives an upper bound on the cop number of connected graphs. The cop number in graph classes is discussed in Sect. 8, algorithmic results in Sect. 9, random graphs in Sect. 10, and infinite graphs in Sect. 11. Part 2 closes with 12 of the most important open problems in the field.

6 Cop-Win Graphs

The game of Cops and Robbers historically first considered only the case of one cop, and that is the focus in the present section. The *closed neighbor set* of a vertex x , written $N[x]$, is the set of vertices joined to x (including x itself). A vertex u is a *corner* if there is some vertex v such that $N[u] \subseteq N[v]$.

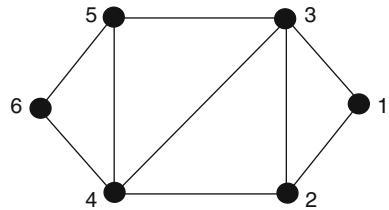
A graph is *dismantlable* if some sequence of deleting corners results in the graph K_1 . For example, each tree is dismantlable: delete leaves repeatedly until a single vertex remains. The same approach holds with chordal graphs, which always contains at least two simplicial vertices (i.e., vertices whose neighbor sets are cliques). The following result characterizes cop-win graphs.

Theorem 46 ([107]) *A graph is cop-win if and only if it is dismantlable.*

Cop-win (or dismantlable) graphs have a recursive structure, made explicit in the following sense. Observe that a graph is dismantlable if the vertices can be labeled by positive integers $[n] = \{1, 2, \dots, n\}$, in such a way that for each $i < n$, the vertex i is a corner in the subgraph induced by $\{i, i+1, \dots, n\}$. This ordering of $V(G)$ is called a *cop-win ordering*. See Fig. 1 for a graph with vertices labeled by a cop-win ordering. Cop-win orderings are sometimes called *elimination orderings*; vertices from lower to higher index are deleted until only vertex n remains.

Let H be an induced subgraph of G formed by deleting one vertex. The graph H is a *retract* of G if there is a homomorphism f from G onto H so that $f(x) = x$ for $x \in V(H)$; that is, f is the identity on H . The map f is called a *retraction* (or *one-point retraction* or *fold*). For example, the subgraph formed by deleting a vertex of degree 1 is a retract. If u is a corner, then the mapping

Fig. 1 A cop-win ordering of a cop-win graph



$$f(x) = \begin{cases} v & \text{if } x = u \\ x & \text{else} \end{cases}$$

is a retraction (recall that graphs are reflexive, so edges may map to a single vertex).

Retracts play an important role in characterizing cop-win graphs. The next theorem, due to Berarducci and Intrigila, shows that the cop number of a retract never increases.

Theorem 47 ([17]) *If H is a retract of G , then $c(H) \leq c(G)$.*

Cop-win orderings suggest a kind of linear structure to cop-win graphs; it roughly suggests that by “sweeping” from largest index vertex to smallest in the ordering, the robber can be captured. This intuition is made precise by the following winning strategy for the cops – first made explicit by Clarke and Nowakowski [44] – in a cop-win graph exploiting the cop-win ordering.

Cop-Win (or No-Backtrack) Strategy [44]. Assume that $[n]$ is a cop-win ordering of G , and for $1 \leq i \leq n$, define

$$G_i = G \upharpoonright \{n, n-1, \dots, i\}.$$

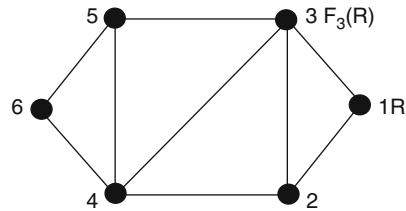
Note that $G_1 = G$ and G_n is just the vertex n . For each $1 \leq i \leq n-1$, let $f_i : G_i \rightarrow G_{i+1}$ be the retraction map from G_i to G_{i+1} mapping i onto a vertex that covers i in G_i . Define F_1 to be the identity mapping on $G : F_1(x) = x$ for all $x \in V(G)$. For $2 \leq i \leq n$, define

$$F_i = f_{i-1} \circ \dots \circ f_2 \circ f_1.$$

In other words, the F_i is the mapping formed by iteratively retracting corners $1, 2, \dots, i-1$. As the f_i are homomorphisms, so are the F_i (recall that graphs are reflexive). Further, for all i , as the f_i are retractions, $F_i(x)$ and $F_{i+1}(x)$ are either equal or joined. If the robber is on vertex x in G , then think of $F_i(x)$ as the robber’s shadow on G_i . See Fig. 2.

The cop-win strategy is described as follows. The cop begins on G_n (the vertex n), which is the shadow of the robber’s position under F_n (note that everything in G

Fig. 2 The robber and his shadow $F_3(R) = f_2 \circ f_1(R)$



maps to n under F_n). Suppose that the robber is on u and the cop is occupying the shadow of the robber in G_i equaling $F_i(u)$. If the robber moves to v , then the cop moves onto the image $F_{i-1}(v)$ of R in the larger graph G_{i-1} .

Theorem 48 ([44]) *The cop-win strategy results in a capture for the cop in at most n moves.*

If both players are playing *optimally* (i.e., the cop is trying to minimize the length of the game, while the robber is trying to maximize it), then in cop-win graph, a cop can win in no more than $n - 3$ moves if $n \geq 5$. See Bonato et al. [31].

Note that the dismantling characterization of [Theorem 46](#) fails badly for infinite graphs. For example, an infinite one-way path (or *ray*) is dismantlable (if infinitely many vertex deletions are allowed) but fails to be cop-win. There is a characterization of cop-win graphs of any order (finite or infinite) included here, owing to Nowakowski and Winkler [107].

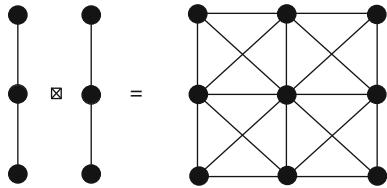
Define a relation \leq on vertices. The relation is defined recursively on ordinals, with $x \leq_0 x$ for all vertices u (in other words, \leq_0 is just the *diagonal* or *equality relation* on $V(G)$). Observe that $u \leq_\alpha v$ will mean that when a robber is on vertex u , a cop is on vertex v , and when it is the robber's turn to move, the robber will lose in at most α rounds. For an ordinal α , define $u \leq_\alpha v$ if and only if for each $a \in N[u]$, there exists a $b \in N[v]$ such that $a \leq_\beta b$ for some $\beta < \alpha$. Let ρ be the least ordinal such that $\leq_\rho = \leq_{\rho+1}$ and define $\leq = \leq_\rho$.

Note that if $\rho < \alpha$, then the relation \leq_ρ is a subset of \leq_α . As such relations are bounded above in cardinality, the ordinal ρ exists. A binary relation on a set X is *trivial* if it equals the Cartesian product of X with itself, $X \times X$.

Theorem 49 ([107]) *A graph G is cop-win if and only if the relation \leq on $V(G)$ is trivial.*

A generalization of [Theorem 49](#) has been found for graphs with higher cop number. Given a graph G , define the k th *strong power* of G , written G^k_{\boxtimes} , to have vertices the ordered k -tuples from G , with two tuples joined if they are equal or joined in each coordinate. See [Fig. 3](#) for an example. The positions of k -many cops in G are identified with a single vertex in G^k_{\boxtimes} . The definition of the strong power allows us to simulate movements of the cops in G by movements of a single cop in G^k_{\boxtimes} .

Fig. 3 The strong power $(P_3)^2$



Let $P = G_{\boxtimes}^k$. For $i \in \mathbb{N}$, the relation \leq_i on $V(G) \times V(P)$ is defined as follows by induction on i . For $x \in V(G)$ and $p \in V(P)$, $x \leq_0 p$ if in position p , at least one of the k cops is occupying x . For $i > 0$, $x \leq_i p$ if and only if for each $u \in N[x]$, there exists a $v \in N[p]$ such that $u \leq_j v$ for some $j < i$. Just as in the cop-win case, the relations \leq_i are nondecreasing sets in i , and hence, there is an ordinal M such that $\leq_M = \leq_{M+1}$ and set $\leq = \leq_M$. Although the notation \leq in this case clashes with the one for cop-win graphs, it is used again here for simplicity.

Theorem 50 ([43]) *A graph G is k -cop-win if and only if there exists $p \in V(P)$ such that $x \leq p$ for every $x \in V(G)$.*

7 Higher Cop Number

Graphs with higher cop number are much less understood than cop-win graphs. Hence, the focus is on finding bounds on the cop number.

7.1 Upper Bounds

An elementary upper bound for the cop number is

$$c(G) \leq \gamma(G), \quad (1)$$

where $\gamma(G)$ is the domination number of G . In general graphs, the inequality (1) is far from tight (e.g., consider a path).

We do not know how large the cop number of a connected graph can be as a function of its order. For a positive integer n , let $c(n)$ be the maximum value of $c(G)$, where G is of order n . *Meyniel's conjecture* states that

$$c(n) = O(\sqrt{n}).$$

The conjecture was mentioned in Frankl's paper [68] as a personal communication to him by Henri Meyniel in 1985 (see page 301 of [68] and reference [7] in that paper). Meyniel's conjecture stands out as one of the deepest (if not *the* deepest) problems on the cop number.

For many years, the best known upper bound for general graphs was the one proved by Frankl [68].

Theorem 51 ([68]) *If n is a positive integer, then*

$$c(n) \leq O\left(n \frac{\log \log n}{\log n}\right).$$

The key to proving **Theorem 51** is the notion of cops guarding an isometric path. For a fixed integer $k \geq 1$, an induced subgraph H of G is k -guardable if, after finitely many moves, k cops can move only in the vertices of H in such a way that if the robber moves into H at round t , then he will be captured at round $t + 1$. For example, a clique or a closed neighbor set (i.e., a vertex along with its neighbors) in a graph are 1-guardable. Given a connected graph G , the *distance* between vertices u and v in G is denoted $d_G(u, v)$. A path P in G is *isometric* if for all vertices u and v of P ,

$$d_P(u, v) = d_G(u, v).$$

The following theorem of Aigner and Fromme [2] on guarding isometric paths has found a number of applications.

Theorem 52 [2] *An isometric path is 1-guardable.*

In 2008 Chiniforooshan [39] gave an improved upper bound once again using a guarding argument.

Theorem 53 ([39]) *For a positive integer n ,*

$$c(n) = O\left(\frac{n}{\log n}\right). \quad (2)$$

The bound (2), therefore, represents the first important step forward in proving Meyniel's conjecture in over 25 years. The key to proving (2) comes again from the notion of guarding an induced subgraph.

An improvement exists to the bound (2) in **Theorem 53**. The following theorem was proved independently by three sets of authors.

Theorem 54 ([69, 95, 121]) *For a positive integer n ,*

$$c(n) \leq O\left(\frac{n}{2^{(1-o(1))\sqrt{\log_2 n}}}\right). \quad (3)$$

The bound in (3) is currently the best upper bound for general graphs that is known, but it is still far from proving Meyniel's conjecture or even the soft version of the conjecture.

7.2 Lower Bounds

A useful theorem of Aigner and Fromme [2] is the following. The girth of a graph is the length of minimum order cycle. The *minimum degree* of G is written $\delta(G)$.

Theorem 55 ([2]) *If G has girth at least 5, then $c(G) \geq \delta(G)$.*

Frankl [68] proved the following theorem generalizing [Theorem 55](#) (which is the case $t = 1$).

Theorem 56 ([2]) *For a fixed integer $t \geq 1$, if G has girth at least $8t - 3$ and $\delta(G) > d$, then $c(G) > d^t$.*

Meyniel's conjecture states that the cop number is at most approximately \sqrt{n} . For graphs with large cop number near the conjectured bound, consider projective planes. A *projective plane* consists of a set of points and lines satisfying the following axioms:

- (1) There is exactly one line incident with every pair of distinct points.
- (2) There is exactly one point incident with every pair of distinct lines.
- (3) There are four points such that no line is incident with more than two of them.

Finite projective planes have $q^2 + q + 1$ points for some integer $q > 0$ (called the *order* of the plane). For a given projective plane P , define $G(P)$ to be the bipartite graph with red vertices representing the points of P and the blue vertices representing the lines. Vertices of different colors are joined if they are incident. This is the *incidence graph* of P . See [Fig. 4](#) for $G(P)$, where P is the Fano plane (i.e., the projective plane of order 2). The incidence graph of the Fano plane is isomorphic to the famous *Heawood graph*.

Hence, [Theorem 55](#) proves that $c(G(P)) \geq q + 1$. As proven in Prałat [111], $c(G(P)) = q + 1$. However, the orders of $G(P)$ depend on the orders of projective planes. The only orders where projective planes are known to exist are prime powers; indeed, this is a deep conjecture in finite geometry. What about integers which are not prime powers? An infinite family of graphs $(G_n : n \geq 1)$ is *Meyniel extremal* if there is a constant d such that for sufficiently large n , $c(G_n) \geq d \sqrt{|V(G_n)|}$.

Recall the famous *Bertrand's postulate*.

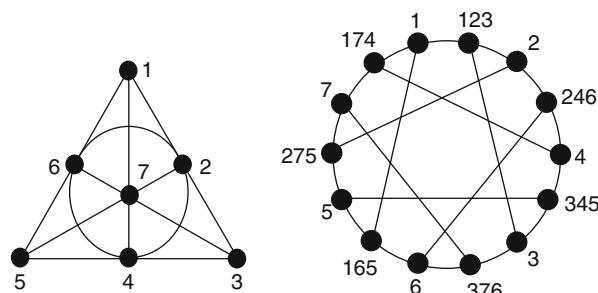


Fig. 4 The Fano plane and its incidence graph. Lines are represented by triples

Theorem 57 ([38]) *For any integer $x > 1$, there is a prime in the interval $(x, 2x)$.*

In Prałat [111] a Meyniel extremal family was given using incidence graphs of projective planes and [Theorem 57](#). Using Bertrand's postulate, it was shown that

$$c(n) \geq \sqrt{\frac{n}{8}}$$

for $n \geq 72$. Using this theorem and a result from number theory, it was shown in Prałat [111] that for sufficiently large n ,

$$c(n) \geq \sqrt{\frac{n}{2}} - n^{0.2625}. \quad (4)$$

Define m_k to be the minimum order of a connected graph G satisfying $c(G) \geq k$. Define M_k to be the minimum order of a connected k -cop-win graph. It is evident that the m_k are monotonically increasing and $m_k \leq M_k$. A recent work [12] establishes that

$$m_3 = 10.$$

The fact that $m_3 \geq 10$ follows by a computer search. The upper bound follows by considering the Petersen graph, which is 3-cop-win. In fact, a computer search in Baird and Bonato [12] shows the surprising fact that the Petersen graph is the *unique* smallest order isomorphism type of connected graph with cop number 3.

Define $f_k(n)$ to be the number of non-isomorphic connected k -cop-win graphs of order n (i.e., the *unlabeled* graphs G of order n with $c(G) = k$). Define $g(n)$ to be the number of non-isomorphic (not necessarily connected) graphs of order n and $g_c(n)$ the number of non-isomorphic connected graphs of order n . Trivially, for all k , $f_k(n) \leq g(n)$. The table below presents the values of g , g_c , f_1 , and f_2 for small orders. The values of g and g_c come from [124]; f_1 was computed by checking for cop-win orderings [107], while f_2 and f_3 were computed using [Algorithm 1](#) given in [Sect. 9](#).

Order n	$g(n)$	$g_c(n)$	$f_1(n)$	$f_2(n)$	$f_3(n)$
1	1	1	1	0	0
2	2	1	1	0	0
3	4	2	2	0	0
4	11	6	5	1	0
5	34	21	16	5	0
6	156	112	68	44	0
7	1,044	853	403	450	0
8	12,346	11,117	3,791	7,326	0
9	274,668	261,080	65,561	195,519	0
10	12,005,168	11,716,571	2,258,313	9,458,257	1

Algorithm 1 CHECK K COP NUMBER

Require: $G = (V, E)$, $k \geq 1$

- 1: initialize $f(u)$ to $V(G) \setminus N_G[u]$, for all $u \in V(G_{\boxtimes}^k)$
- 2: **repeat**
- 3: **for all** $uv \in E(G_{\boxtimes}^k)$ **do**
- 4: $f(u) \leftarrow f(u) \cap N_G[f(v)]$
- 5: $f(v) \leftarrow f(v) \cap N_G[f(u)]$
- 6: **end for**
- 7: **until** the value of f is unchanged
- 8: **if** there exists $u \in V(G_{\boxtimes}^k)$ such that $f(u) = \emptyset$, **then**
- 9: $c(G) \leq k$
- 10: **else**
- 11: $c(G) > k$
- 12: **end if**

The next theorem sets up an unexpected connection between Meyniel's conjecture and the order of m_k .

Theorem 58 ([12])

- (1) $m_k = O(k^2)$.
- (2) *Meyniel's conjecture is equivalent to the property that*

$$m_k = \Omega(k^2).$$

Hence, if Meyniel's conjecture holds, then [Theorem 58](#) implies that $m_k = \Theta(k^2)$.

8 Graph Classes

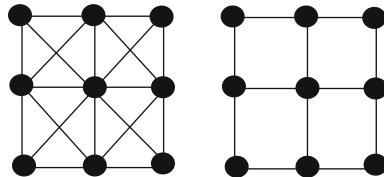
Planar graphs have inspired some of the deepest results in graph theory, most notably the four color theorem (which states that every planar graph has chromatic number at most four; see Appel, Haken, and Koch [10]). A graph is *planar* if it can be drawn in \mathbb{R}^2 without any two of its edges crossing. Aigner and Fromme [2] showed in fact that planar graphs require no more than three cops.

Theorem 59 ([2]) *If G is a planar graph, then $c(G) \leq 3$.*

The idea of the proof of [Theorem 59](#) is to increase the *cop territory*; that is, a set of vertices so that if the robber moved to one of them he would be caught. Hence, the number of vertices the robber can move to without being caught is eventually reduced to the empty set, and so the robber is captured.

For a fixed graph H , Andreae [8] generalized this result by proving that the cop number of a K_5 -minor-free graph (or $K_{3,3}$ -minor-free graph) is at most 3 (recall that planar graphs are exactly those which are K_5 -minor-free and $K_{3,3}$ -minor-free).

Fig. 5 The graph on the *left* is a cop-win non-outerplanar graph, while the graph on the *right* is non-outerplanar with cop number 2



Andreae [9] also proved that for any graph H , the cop number of the class of H -minor-free graphs is bounded above by a constant.

Less is known about the cop number of graphs with positive genus. As such, the survey of such graphs is brief. The main conjecture in this area is due to Schroeder. In [120], Schroeder conjectured that if G is a graph of genus g , then $c(G) \leq g + 3$. Quilliot [114] had shown the following.

Theorem 60 ([114]) *If G is a graph of genus g , then $c(G) \leq 2g + 3$.*

In the same paper as the conjecture, Schroeder showed the following.

Theorem 61 ([120]) *If G is a graph of genus g , then*

$$c(G) \leq \left\lfloor \frac{3g}{2} \right\rfloor + 3.$$

Schroeder also proved the following theorem.

Theorem 62 ([120]) *If G is a graph that can be embedded on a torus, then $c(G) \leq 4$.*

A graph G is *outerplanar* if it has an embedding in the plane with the following properties:

- (1) Every vertex lies on a circle.
 - (2) Every edge of G either joins two consecutive vertices around the circle or is a chord across the circle.
 - (3) If two chords intersect, then they do so at a vertex
- (See Fig. 5). Clarke proved the next result in her doctoral thesis.

Theorem 63 ([42]) *If G be an outerplanar graph, then $c(G) \leq 2$.*

Theorem 63 was generalized by Theis [133] to series-parallel graphs.

Lu and Peng [95] showed that Meyniel's conjecture holds in the class of graphs with diameter two. The proof uses the notion of guarding subgraph but also uses a randomized argument.

Theorem 64 ([95]) *If G is a graph on n vertices with diameter two, then*

$$c(G) \leq 2\sqrt{n} - 1. \quad (5)$$

The same bound (5) was also shown in [95] in the case when G is bipartite and of diameter at most three. The incidence graphs of projective planes are bipartite of diameter three and so show that the bound (5) is asymptotically tight in that class.

9 Algorithmic Results

Another approach to the Cops and Robbers game is an algorithmic one. Consider the following two graph decision problems:

k -COP NUMBER: Given a graph G and a positive integer k , is $c(G) \leq k$?

k -FIXED COP NUMBER: Given a graph G and a fixed positive integer k , is $c(G) \leq k$?

The main difference between the two problems is that in k -COP NUMBER the integer k may be a function of n and so grows with n . In k -FIXED COP NUMBER, k is fixed and not part of the input and so is independent of n .

There is the following result.

Theorem 65 ([17, 75]) *The problem FIXED COP NUMBER is in P .*

An algorithm from Bonato et al. [33] is described which may be used to prove **Theorem 65**. Given a graph G , recall that the k th *strong power* of G , written G_{\boxtimes}^k , is the strong product of G with itself k times. For a set X , define 2^X to be the set of subsets of X . For $S \subseteq V(G)$, define $N_G[S]$ to be the union of the closed neighbor sets of vertices in S .

Theorem 66 ([33]) *Suppose $k \geq 1$ is an integer. Then $c(G) > k$ if and only if there is a mapping $f : V(G_{\boxtimes}^k) \rightarrow 2^{V(G)}$ with the following properties:*

(1) *For every $u \in V(G_{\boxtimes}^k)$,*

$$\emptyset \neq f(u) \subseteq V(G) \setminus N_G[u].$$

(2) *For every $uv \in E(G_{\boxtimes}^k)$,*

$$f(u) \subseteq N_G[f(v)].$$

Consider [Algorithm 1](#) for determining whether $c(G) \leq k$ based on [Theorem 66](#). The following theorem gives [Theorem 65](#) as a corollary.

Theorem 67 ([33]) *[Algorithm 1](#) runs in time $O(n^{3k+3})$.*

If k is not fixed (and hence can be a function of n), then the problem becomes less tractable.

Theorem 68 ([67]) *The problem k -COP NUMBER is NP-hard.*

Theorem 68 does not say that k -COP NUMBER is in NP; that is an open problem! See Sect. 12 below. Theorem 68 is proved in Fomin et al. [67] by using a reduction from the following NP-complete problem:

Domination: Given a graph G and an integer $k \geq 2$, is $\gamma(G) \leq k$?

10 Random Graphs

Random graphs are a central topic in graph theory; however, only recently have researchers considered the cop number of random graphs. Define a probability space on graphs of a given order $n \geq 1$ as follows. Fix a vertex set V consisting of n distinct elements, usually taken as $[n]$, and fix $p \in [0, 1]$. Define the space of *random graphs of order n with edge probability p* , written $G(n, p)$ with sample space equaling the set of all $2^{\binom{n}{2}}$ (labeled) graphs with vertex set V , and

$$\mathbb{P}(G) = p^{|E(G)|} (1-p)^{\binom{n}{2}-|E(G)|}.$$

Informally, $G(n, p)$ may be viewed as the space of graphs with vertex set V , so that two distinct vertices are joined independently with probability p . Even more informally, toss a (biased) coin to determine the edges of your graph. Hence, V does not change, but the number of edges is not fixed: it varies according to a binomial distribution with expectation $\binom{n}{2}p$. Despite the fact that $G(n, p)$ is a space of graphs, it is often called *the random graph of order n with edge probability p* . Random graphs were introduced in a series of papers by Erdős and Rényi [53–55]. See the book [26] for a modern reference.

The cases when p is fixed and when it is a function of n are considered. Graph parameters, such as the cop number, become random variables in $G(n, p)$. For notational ease, the cop number of $G(n, p)$ is referred to simply by $c(G(n, p))$.

An event holds *asymptotically almost surely* (or *a.a.s.* for short) if it holds with probability tending to 1 as $n \rightarrow \infty$. For example, if p is constant, then a.a.s. $G(n, p)$ is diameter two and not planar.

10.1 Constant p

The cop number of $G(n, p)$ was studied in Bonato et al. [30] for constant p , where the following result was proved. For $p \in (0, 1)$ or $p = p(n) = o(1)$, define

$$\mathbb{L}n = \log_{\frac{1}{1-p}} n.$$

Theorem 69 ([30]) Let $0 < p < 1$ be fixed. For every real $\varepsilon > 0$, a.a.s.

$$(1 - \varepsilon)\mathbb{L}n \leq c(G(n, p)) \leq (1 + \varepsilon)\mathbb{L}n. \quad (6)$$

In particular,

$$c(G(n, p)) = \Theta(\log n).$$

The upper bound in [Theorem 69](#) follows by considering the domination number of $G(n, p)$ [50], while the lower bounds follow by considering an adjacency property. If the case of $p = 1/2$ is considered, then $G(n, p)$ corresponds to the case of uniformly choosing a labeled graph of order n from the space of all such graphs. Hence, [Theorem 69](#) may be interpreted as saying “most” finite graphs of order n have cop number approximately $\log n$.

Properties of randomly chosen k -cop-win graphs (with the uniform distribution) are described next. For this, it is equivalent to work in the probability space $G(n, 1/2)$. Let **cop-win** be the event in $G(n, 1/2)$ that the graph is cop-win, and let **universal** be the event that there is a universal vertex. If a graph has a universal vertex w , then it is cop-win; in a certain sense, graphs with universal vertices are the simplest cop-win graphs. The probability that a random graph is cop-win can be estimated as follows:

$$\begin{aligned} \mathbb{P}(\text{cop-win}) &\geq \mathbb{P}(\text{universal}) = n2^{-n+1} - O(n^22^{-2n+3}) \\ &= (1 + o(1))n2^{-n+1}. \end{aligned}$$

A recent surprising result of Bonato et al. [35] showed that this lower bound is in fact the correct asymptotic value for $\mathbb{P}(\text{cop-win})$.

Theorem 70 ([35]) In $G(n, 1/2)$,

$$\mathbb{P}(\text{cop-win}) = (1 + o(1))n2^{-n+1}.$$

Hence, almost all cop-win graphs contain a universal vertex.

10.2 The Dense Case

Now consider the cop number of *dense* random graphs, with average degree pn at least \sqrt{n} . The main results in this case were given in Bonato et al. [32].

Theorem 71 ([32])

(1) Suppose that $p \geq p_0$ where p_0 is the smallest p for which

$$p^2/40 \geq \frac{\log((\log^2 n)/p)}{\log n}$$

holds. Then, a.a.s.

$$\mathbb{L}n - \mathbb{L}((p^{-1}\mathbb{L}n)(\log n)) \leq c(G(n, p)) \leq \mathbb{L}n - \mathbb{L}((\mathbb{L}n)(\log n)) + 2.$$

(2) If $(2\log n)/\sqrt{n} \leq p = o(1)$ and $\omega(n)$ is any function tending to infinity, then a.a.s.

$$\mathbb{L}n - \mathbb{L}((p^{-1}\mathbb{L}n)(\log n)) \leq c(G(n, p)) \leq \mathbb{L}n + \mathbb{L}(\omega(n)).$$

By [Theorem 71](#), the following corollary gives a concentration result for the cop number. In particular, for a wide range of p , the cop number of $G(n, p)$ concentrates on just the one value $\mathbb{L}n$.

Corollary 3 ([32]) If $p = n^{-o(1)}$ and $p < 1$, then a.a.s.

$$c(G(n, p)) = (1 + o(1))\mathbb{L}n.$$

From [Theorem 71](#) part (1), it follows that if p is a constant, then there is the concentration result that

$$c(G(n, p)) = \mathbb{L}n - 2\mathbb{L}\log n + \Theta(1) = (1 + o(1))\mathbb{L}n.$$

10.3 The Sparse Case and Zigzag Theorem

Bollobás, Kun, and Leader [\[27\]](#) established the following bounds on the cop number in the sparse case, when the expected degree is $np = O(n^{1/2})$.

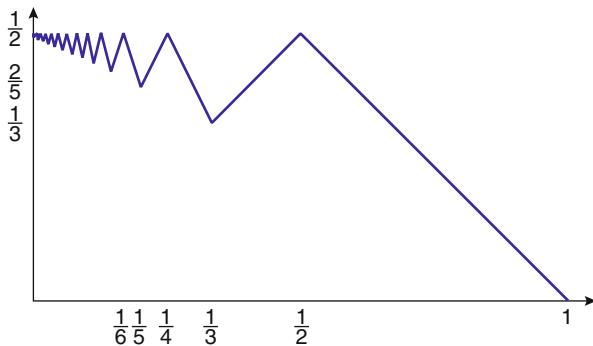
Theorem 72 ([27]) If $p(n) \geq 2.1 \log n/n$, then a.a.s.

$$\frac{1}{(np)^2} n^{\frac{1}{2} \frac{\log \log(np)-9}{\log \log(np)}} \leq c(G(n, p)) \leq 160,000\sqrt{n} \log n. \quad (7)$$

In particular, [Theorem 72](#) proves Meyniel's conjecture for random graphs up to a logarithmic factor of n from the upper bound in (7). Recent work by Prałat and Wormald [\[112\]](#) removes the $\log n$ factor in the upper bound of (7) and hence proves the Meyniel bound for random graphs.

It would be natural to assume that the cop number of $G(n, p)$ is close to \sqrt{n} also for $np = n^{\alpha+o(1)}$, where $0 < \alpha < 1/2$. The so-called zigzag theorem of Łuczak and Prałat [\[96\]](#) demonstrated that the actual behavior of $c(G(n, p))$ is much more complicated.

Fig. 6 The zigzag-shaped graph of the cop number of $G(n, p)$



Theorem 73 (Zig-Zag Theorem, [96]) Let $0 < \alpha < 1$, and $d = d(n) = np = n^{\alpha+o(1)}$.

(1) If $\frac{1}{2j+1} < \alpha < \frac{1}{2j}$ for some $j \geq 1$, then a.a.s.

$$c(G(n, p)) = \Theta(d^j).$$

(2) If $\frac{1}{2j} < \alpha < \frac{1}{2j-1}$ for some $j \geq 1$, then a.a.s.

$$\mathcal{O}\left(\frac{n}{d^j}\right) = c(G(n, p)) = O\left(\frac{n \log n}{d^j}\right).$$

Consider the function $f : (0, 1) \rightarrow \mathbb{R}$ defined by

$$f(x) = \frac{\log \bar{c}(G(n, n^{x-1}))}{\log n},$$

where $\bar{c}(G(n, n^{x-1}))$ is the median of the cop number for $G(n, p)$. See Fig. 6, which justifies Theorem 73's moniker. In particular,

$$f(x) = \begin{cases} \alpha j & \text{if } \frac{1}{2j+1} \leq \alpha < \frac{1}{2j} \text{ for some } j \geq 1, \\ 1 - \alpha j & \text{if } \frac{1}{2j} \leq \alpha < \frac{1}{2j-1} \text{ for some } j \geq 1. \end{cases}$$

11 Infinite Graphs

Infinite graphs exhibit properties often quite different than finite ones. In this regard, the cop number is no exception. For example, the *ray* (i.e., the one-way infinite path) is an infinite tree with infinite cop number: one robber can always stay ahead of finitely many cops.

11.1 Cop Density

When dealing with countable graphs, note that they are limits of chains of finite graphs. To analyze the cop number of infinite graphs, consider the *cop density* of a finite graph first introduced in [30]. Define

$$D_c(G) = \frac{c(G)}{|V(G)|}.$$

Note that $D_c(G)$ is a rational number in $[0, 1]$. Every countable graph G is the limit of a chain of finite graphs, and there are infinitely many distinct chains with limit G . Suppose that $G = \lim_{n \rightarrow \infty} G_n$, where $\mathcal{C} = (G_n : n \in \mathbb{N})$ is a fixed chain of induced subgraphs of G . The chain \mathcal{C} is a *full chain for* G . Define

$$D(G, \mathcal{C}) = \lim_{n \rightarrow \infty} D_c(G_n),$$

if the limit exists (and then it is a real number in $[0, 1]$). This is the *cop density of G relative to \mathcal{C}* ; if \mathcal{C} is clear from context, this is referred to as the *cop density of G* .

The *upper cop density of G* , written $UD(G)$, is defined as

$$\sup\{D(G, \mathcal{C}) : \mathcal{C} \text{ is a full chain for } G\}.$$

Note that $UD(G)$ does not depend on the chain and is a parameter of G .

For a positive integer n , a graph G is *strongly n -e.c.* if for all disjoint finite sets of vertices A and B from G with $|A| \leq n$, there is a vertex z correctly joined to A and B . Note that the infinite random graph is the unique isomorphism type of countable graph which is strongly n -e.c. for all n ; see [37]. The following result, proved in Bonato et al. [30], finds connections between infinite-cop-win graphs and the strongly 0- and 1-e.c. properties.

Theorem 74 ([30])

- (1) If G is strongly 1-e.c., then $c(G)$ is infinite.
- (2) If $c(G)$ is infinite, then G satisfies the strongly 0-e.c. property.

If G is strongly 1-e.c., then the cop density of G may be *any* real number in $[0, 1]$. This property applies, therefore, to a large number of graphs: for each $n \geq 0$, there are 2^{\aleph_0} many non-isomorphic countable graphs that are strongly n -e.c. but not strongly $(n + 1)$ -e.c.; see Theorem 4.1 of [28].

Theorem 75 ([30]) If G is strongly 1-e.c., then for all $r \in [0, 1]$, there is a chain \mathcal{C} in G such that $D(G, \mathcal{C}) = r$.

The next result completely characterizes the upper cop density of a graph G : $UD(G)$ takes on one of the two values 0 or 1 and equals 1 exactly when G is strongly 0-e.c. This fact, proven in [30], is somewhat unexpected.

Theorem 76 ([30]) *The following are equivalent:*

- (1) $UD(G) = 1$.
- (2) $UD(G) > 0$.
- (3) G is strongly 0-e.c.

We note that the strongly 0-e.c. graphs are precisely the spanning subgraphs of the infinite random graph; see Cameron [37].

11.2 Chordal Graphs

As another instance where results from finite graphs do not translate to infinite ones, consider chordal graphs. The graph G is *chordal* if each cycle of length at least four has a chord. A vertex of G is *simplicial* if its neighborhood induces a complete graph. Every finite chordal graph contains at least two simplicial vertices, and the deletion of a simplicial vertex leaves a chordal graph. As a simplicial vertex is a corner, a finite chordal graph is dismantlable and so cop-win.

However, an infinite tree containing a ray is chordal but not cop-win. Such trees have infinite diameter. Inspired by a question of Martin Farber which asked if infinite chordal graphs (more generally, bridged graphs) of finite diameter are cop-win, it was shown in Hahn et al. [76] that there exist infinite chordal graphs of diameter two that are not cop-win. The difficulty lies in finding examples with finite diameter.

Theorem 77 ([76]) *For each infinite cardinal κ , there exist chordal, robber-win graphs of order κ with diameter two.*

11.3 Large Families of Cop-Win Graphs

From Theorem 77, the cop number of infinite graphs behaves rather differently than in the finite case. Vertex-transitive cop-win finite graphs are cliques; see Nowakowski and Winkler [107]. However, this fails badly in the infinite case, which is the focus of this section.

A class of graphs is *large* if for each infinite cardinal κ , there are 2^κ many non-isomorphic graphs of order κ in the class. In other words, a large class contains as many as possible non-isomorphic graphs of each infinite cardinality. For example, the classes of all graphs, all trees, and all k -chromatic graphs for k finite are large. Recall that a graph G is *vertex-transitive* if for each pair of vertices x and y , there is an automorphism of G mapping x to y . The following result of Bonato et al. [34] showed that for any integer $k > 0$, there are large families of k -cop-win graphs that are vertex-transitive.

Theorem 78 ([34]) *The class of cop-win, vertex-transitive graphs with the property that the cop can win in two moves is large.*

To describe the large family in [Theorem 78](#), recall some properties of the strong product of a set of graphs over a possibly infinite index set. Let I be an index set. The *strong product* of a set $\{G_i : i \in I\}$ of graphs is the graph $\boxtimes_{i \in I} G_i$ defined by

$$\begin{aligned} V(\boxtimes_{i \in I} G_i) &= \{f : I \rightarrow \bigcup_{i \in I} V(G_i) : f(i) \in V(G_i) \text{ for all } i \in I\}, \\ E(\boxtimes_{i \in I} G_i) &= \{fg : f \neq g \text{ and for all } i \in I, \\ &\quad f(i) = g(i) \text{ or } f(i)g(i) \in E(G_i)\}. \end{aligned}$$

Products exhibit unusual properties if there are infinitely many factors.

Fix a vertex $f \in V(\boxtimes_{i \in I} G_i)$ and define the *weak strong product* of $\{G_i : i \in I\}$ with base f as the subgraph $\boxtimes_f^I G_i$ of $\boxtimes_{i \in I} G_i$ induced by the set of all $g \in V(\boxtimes_{i \in I} G_i)$ such that $\{i \in I : g(i) \neq f(i)\}$ is finite. The graph $\boxtimes_f^I G_i$ is connected if each factor is, and if $|I| \leq \kappa$ and $|V(G_i)| \leq \kappa$ for each $i \in I$, then $|V(\boxtimes_f^I G_i)| \leq \kappa$. For $i \in I$, the *projection mapping* $\pi_i : \boxtimes_f^I G_i \rightarrow G_i$ is defined by $\pi_i(g) = g(i)$.

Let $\{G_i : i \in I\}$ be a set of isomorphic copies of G . Denote by $\boxtimes^I G$ the strong product $\boxtimes_{i \in I} G_i$. If $f \in V(\boxtimes^I G)$ is fixed, denote by G_f^I the weak strong product $\boxtimes_f^I G$ with base f . One particular power of a graph is of special interest as it allows us to construct vertex-transitive graphs out of non-transitive ones. Let κ be a cardinal, and let H be a graph of order κ . Let $I = \kappa \times V(H)$, and define $f : I \rightarrow V(H)$ by $f(\beta, v) = v$. The power H_f^I of H with base f will be called the *canonical power* of H and will be denoted by H^H .

As automorphisms of the factors applied coordinate-wise yield an automorphism of the product, it follows that the weak strong product of vertex-transitive graphs is vertex-transitive. However, the following technical lemma from [34] demonstrates the paradoxical property that if there are infinitely many factors, the weak strong product may be vertex-transitive even if none of the factors are!

Lemma 2 ([34]) *If H is an infinite graph, then the canonical power of H^H is vertex-transitive.*

12 A Dozen Problems on Cops and Robbers

To serve as a record and a challenge, 12 open problems on Cops and Robbers are stated. These are arguably the most central (and relevant) problems in the field at the moment. Solutions to problems (1) and (11), in particular, would be exceptional breakthroughs:

- (1) Most likely the deepest open problem in the area is to settle *Meyniel's conjecture*: If G is a graph of order n , then

$$c(G) = O(\sqrt{n}). \quad (8)$$

- (2) An easier problem than (1) would be to settle the so-called *soft Meyniel's conjecture*: For a fixed constant $\epsilon > 0$,

$$c(n) = O(n^{1-\epsilon}),$$

- (3) Meyniel's conjecture remains open for familiar graph classes. For example, does it hold in graphs whose chromatic number is bounded by some constant k ?
(4) While the parameters m_k are nonincreasing, an open problem is to determine whether the M_k are in fact nonincreasing. A possibly more difficult problem is to settle whether $m_k = M_k$ for all $k \geq 1$.
(5) Can the lower bound (Eq. 4)

$$c(n) \geq \sqrt{\frac{n}{2}} - n^{0.2625}$$

- be improved for sufficiently large n ?
(6) For $k > 1$, it is conjectured in [35] that almost all k -cop-win graphs in fact have a dominating set of cardinality k . If this is the case, then the number labeled k -cop-win graphs of order n satisfies

$$F_k(n) = 2^{o(n)} (2^k - 1)^{n-k} 2^{\binom{n-k}{2}}.$$

- (7) Characterize the planar graphs with cop numbers 1, 2, and 3.
(8) Determine the cop number of the giant component of $G(n, p)$, where $p = \Theta(1/n)$. In particular, show that the cop number is a.a.s. $O(\sqrt{n})$.
(9) *Schroeder's conjecture*: If G is a graph of genus g , then $c(G) \leq g + 3$.
(10) Is the decision problem **k -COP NUMBER** in **NP**?
(11) Is **COP NUMBER** **EXPTIME**-complete? Goldstein and Reingold [72] proved that the version of the Cops and Robbers game on directed graphs is **EXPTIME**-complete. They also proved that the version of the game on undirected graphs when the cops and the robber are given their initial positions is also **EXPTIME**-complete.
(12) Are there large classes of infinite cop-win graphs whose members are k -chromatic, where $k \geq 2$ is an integer?

13 Conclusion

The chapter focused on two major aspects of graph searching: *searching* games and *Cops and Robbers* games. A broad overview of searching was given for both graphs and digraphs, focusing on the cases when the robber is invisible and active or is visible or lazy. Related games such as minimum cost searching and pebbling were also considered. Structural characterizations were given in these cases, along with

complexity results and bounds on various searching parameters. In the game of Cops and Robbers, a summary of known bounds and techniques were presented, focusing on Meyniel's conjecture as one of the most important problems on the cop number. Results were presented on algorithmic and probabilistic aspects of the cop number, and results were considered on infinite graphs. For both searching and Cops and Robbers, several problems and conjectures were given.

Given the broad diversity of techniques, applications, and problems in graph searching, it is evident that the field is now occupying an increasingly central position in graph theory and theoretical computer science. The subject has seen an explosive growth of interest in recent years. The wealth of new research on graph searching points to many more years of fruitful research in the field.

Recommended Reading

1. I. Adler, Directed tree-width examples. *J. Comb. Theory Ser. B* **97**, 718–725 (2007)
2. M. Aigner, M. Fromme, A game of cops and robbers. *Discret. Appl. Math.* **8**, 1–12 (1984)
3. N. Alon, P. Pralat, R. Wormald, Cleaning regular graphs with brushes. *SIAM J. Discret. Math.* **23**, 233–250 (2008)
4. B. Alspach, Sweeping and searching in graphs: a brief survey. *Matematiche* **5**, 95–37 (2006)
5. B. Alspach, D. Dyer, D. Hanson, B. Yang, Arc searching digraphs without jumping, in *Proceedings of the 1st International Conference on Combinatorial Optimization and Applications (COCOA'07)*, Xi'an, China, 2007
6. B. Alspach, D. Dyer, D. Hanson, B. Yang, Lower bounds on edge searching, in *Proceedings of the 1st International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies (ESCAPE'07)*, Hangzhou, China. Lecture Notes in Computer Science, vol. 4614, 2007, pp. 516–527
7. B. Alspach, D. Dyer, D. Hanson, B. Yang, Time constrained Searching. *Theor. Comput. Sci.* **399**, 158–168 (2008)
8. T. Andreae, Note on a pursuit game played on graphs. *Discret. Appl. Math.* **9**, 111–115 (1984)
9. T. Andreae, On a pursuit game played on graphs for which a minor is excluded. *J. Comb. Theory Ser. B* **41**, 37–47 (1986)
10. K. Appel, W. Haken, J. Koch, Every planar map is four colorable. *Ill. J. Math.* **21**, 439–567 (1977)
11. S. Arnborg, D. Corneil, A. Proskurowski, Complexity of finding embeddings in a k -tree. *SIAM J. Algebr. Discret. Methods* **8**, 277–284 (1987)
12. W. Baird, A. Bonato, Meyniel's conjecture on the cop number: a survey. *J. Comb.* **3**, 225–238 (2012)
13. J. Barat, Directed path-width and monotonicity in digraph searching. *Graphs Comb.* **22**, 161–172 (2005)
14. L. Barrière, P. Flocchini, P. Fraigniaud, N. Santoro, Capture of an intruder by mobile agents, in *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA'02)*, Winnipeg, MB, Canada, 2002
15. L. Barrière, P. Fraigniaud, N. Santoro, D. Thilikos, Searching is not jumping, in *Proceedings of the 29th Workshop on Graph Theoretic Concepts in Computer Science (WG'03)*, Elspeet, The Netherlands, 2003
16. M. Bender, A. Fernández, D. Ron, A. Sahai, S. Vadhan, The power of a pebble: exploring and mapping directed graphs, in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, Dallas, TX, 1998
17. A. Berarducci, B. Intrigila, On the cop number of a graph. *Adv. Appl. Math.* **14**, 389–403 (1993)

18. D. Berwanger, E. Grädel, Entanglement – a measure for the complexity of directed graphs with applications to logic and games, in *Proceedings of the 11th International Conference on Logic for Programming, Artificial Intelligence*, Montevideo, Uruguay, 2004
19. D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, DAG-width and parity games, in *Proceedings of the 23rd Symposium of Theoretical Aspects of Computer Science (STACS'06)*, Marseille, France, 2006
20. T. Biedl, T. Chan, Y. Ganjali, M.T. Hajiaghayi, D. Wood, Balanced vertex-ordering of graphs. *Discret. Appl. Math.* **148**, 27–48 (2005)
21. D. Bienstock, Graph searching, path-width, tree-width and related problems (a survey), *DIMACS Ser. Discret. Math. Theor. Comput. Sci.* **5**, 33–49 (1991)
22. D. Bienstock, P. Seymour, Monotonicity in graph searching. *J. Algorithms* **12**, 239–245 (1991)
23. H.L. Bodlaender, A partial k -arboretum of graphs with bounded treewidth. *Theor. Comput. Sci.* **209**, 1–45 (1998)
24. H. Bodlaender, F. Fomin, Approximation of pathwidth of outerplanar graphs. *J. Algorithms* **43**, 190–200 (2002)
25. H. Bodlaender, T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms* **21**, 358–402 (1996)
26. B. Bollobás, *Random Graphs*. Cambridge Studies in Advanced Mathematics, vol. 73, 2nd edn. (Cambridge University Press, Cambridge, 2001)
27. B. Bollobás, G. Kun, I. Leader, Cops and robbers in a random graph, Preprint 2013
28. A. Bonato, D. Delić, On a problem of Cameron's on inexhaustible graphs. *Combinatorica* **24**, 35–51 (2004)
29. A. Bonato, R.J. Nowakowski, *The Game of Cops and Robbers on Graphs* (American Mathematical Society, Providence, 2011)
30. A. Bonato, G. Hahn, C. Wang, The cop density of a graph. *Contrib. Discret. Math.* **2**, 133–144 (2007)
31. A. Bonato, G. Hahn, P.A. Golovach, J. Kratochvíl, The capture time of a graph. *Discret. Math.* **309**, 5588–5595 (2009)
32. A. Bonato, P. Prałat, C. Wang, Pursuit-evasion in models of complex networks. *Internet Math.* **4**, 419–436 (2009)
33. A. Bonato, E. Chiniforooshan, P. Prałat, Cops and Robbers from a distance. *Theor. Comput. Sci.* **411**, 3834–3844 (2010)
34. A. Bonato, G. Hahn, C. Tardif, Large classes of infinite k -cop-win graphs. *J. Graph Theory* **65**, 334–242 (2010)
35. A. Bonato, G. Kemkes, P. Prałat, Almost all cop-win graphs contain a universal vertex. *Discret. Math.* **312**, 1652–1657 (2012)
36. R. Breisch, An intuitive approach to speleotopology. *Southwest. Cavers* **6**, 72–78 (1967)
37. P.J. Cameron, The random graph, in *The Mathematics of Paul Erdős, II*. Algorithms and Combinatorics, vol. 14 (Springer, Berlin, 1997), pp. 333–351
38. P. Chebyshev, Mémoire sur les nombres premiers. *Mém. Acad. Sci. St. Pétersbourg* **7**, 17–33 (1850)
39. E. Chiniforooshan, A better bound for the cop number of general graphs. *J. Graph Theory* **58**, 45–48 (2008)
40. F.R.K. Chung, On the cutwidth and the topological bandwidth of a tree. *SIAM J. Algebr. Discret. Methods* **6**, 268–277 (1985)
41. F.R.K. Chung, Pebbling in Hypercubes. *SIAM J. Discret. Math.* **2**, 467–472 (1989)
42. N.E. Clarke, *Constrained Cops and Robber*, Ph.D. Thesis, Dalhousie University, 2002
43. N.E. Clarke, G. MacGillivray, Characterizations of k -copwin graphs. *Discret. Math.* **312**, 1421–1425 (2012)
44. N.E. Clarke, R.J. Nowakowski, Cops, robber, and traps. *Util. Math.* **60**, 91–98 (2001)
45. S. Cook, R. Sethi, Storage requirements for deterministic polynomial time recognizable languages. *J. Comput. Syst. Sci.* **13**, 25–37 (1976)
46. B. Crull, T. Cundiff, P. Feltman, G. Hurlbert, L. Pudwell, Z. Szaniszlo, Z. Tuza, The cover pebbling number of graphs. *Discret. Math.* **296**, 15–23 (2005)

47. N. Dendris, L. Kirousis, D. Thilikos, Fugitive-search games on graphs and related parameters. *Theor. Comput. Sci.* **172**, 233–254 (1997)
48. D. Dereniowski, From Pathwidth to Connected Pathwidth, in *Proceedings of the 28th Symposium on Theoretical Aspects of Computer Science (STACS'11)*, Dortmund, Germany, 2011
49. R. Diestel, *Graph Theory* (Springer, New York, 2000)
50. P.A. Dreyer, *Applications and Variations of Domination in Graphs*, Ph.D. Dissertation, Department of Mathematics, Rutgers University, 2000
51. D. Dyer, B. Yang, Ö. Yaşar, On the fast searching problem, in *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM'08)*, Shanghai, China, 2008
52. J. Ellis, I. Sudborough, J. Turner, The vertex separation and search number of a graph. *Inf. Comput.* **113**, 50–79 (1994)
53. P. Erdős, A. Rényi, On random graphs I. *Publ. Math. Debr.* **6**, 290–297 (1959)
54. P. Erdős, A. Rényi, On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.* **5**, 17–61 (1960)
55. P. Erdős, A. Rényi, Asymmetric graphs. *Acta Math. Acad. Sci. Hungar.* **14**, 295–315 (1963)
56. M. Fellows, M. Langston, On search, decision and the efficiency of polynomial time algorithm, in *Proceedings of the 21st ACM Symposium on Theory of Computing (STOC'89)*, Seattle, WA, 1989
57. P. Fraigniaud, N. Nisse, Monotony properties of connected visible graph searching. *Inf. Comput.* **206**, 1383–1393 (2008)
58. M. Frankling, Z. Galil, M. Yung, Eavesdropping games: a graph-theoretic approach to privacy in distributed systems. *J. ACM* **47**, 225–243 (2000)
59. F.V. Fomin, P. Golovach, Graph searching and interval completion. *SIAM J. Discret. Math.* **13**, 454–464 (2000)
60. F.V. Fomin, N. Petrov, Pursuit-evasion and search problems on graphs, in *Proceedings of the Twenty-seventh Southeastern International Conference on Combinatorics, Graph Theory and Computing*, Baton Rouge, LA, 1996
61. F.V. Fomin, D. Thilikos, On the monotonicity of games generated by symmetric submodular functions. *Discret. Appl. Math.* **131**, 323–335 (2003)
62. F.V. Fomin, D. Thilikos, An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.* **399**, 236–245 (2008)
63. F.V. Fomin, D. Kratsch, H. Müller, On the domination search number. *Discret. Appl. Math.* **127**, 565–580 (2003)
64. F.V. Fomin, P. Golovach, A. Hall, M. Mihalak, E. Vicari, P. Widmayer, How to guard a graph? in *Proceedings of 15th International Symposium on Algorithms and Computation (ISAAC'08)*, Gold Coast, Australia, 2008
65. F.V. Fomin, P. Fraigniaud, N. Nisse, Nondeterministic graph searching: from pathwidth to treewidth. *Algorithmica* **53**, 358–373 (2009)
66. F.V. Fomin, P. Golovach, D. Lokshtanov, Guard games on graphs: keep the intruder Out, in *Proceedings of the 7th International Workshop on Approximation and Online Algorithms (WAOA'09)*, Copenhagen, Denmark, 2010
67. F.V. Fomin, P.A. Golovach, J. Kratochvíl, N. Nisse, Pursuing fast robber in graphs. *Theor. Comput. Sci.* **411**, 1167–1181 (2010)
68. P. Frankl, Cops and robbers in graphs with large girth and Cayley graphs. *Discret. Appl. Math.* **17**, 301–305 (1987)
69. A. Frieze, M. Krivelevich, P. Loh, Variations on Cops and Robbers. *J. Graph. Theor.* **69**, 383–402 (2012)
70. R. Ganian, P. Hliněný, J. Kneis, A. Langer, J. Obdržálek, P. Rossmanith, On digraph width measures in parameterized algorithmics, in *Proceedings of the 4th International Workshop on Parameterized and Exact Computation (IWPEC 2009)*, Copenhagen, Denmark, 2009
71. S. Gaspers, M.E. Messinger, R. Nowakowski, P. Pralat, Clean the graph before you draw it. *Inf. Process. Lett.* **109**, 463–467 (2009)
72. A.S. Goldstein, E.M. Reingold, The complexity of pursuit on a graph. *Theor. Comput. Sci.* **143**, 93–112 (1995)

73. G. Gottlob, N. Leone, F. Scarcello, Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. *J. Comput. Syst. Sci.* **66**, 775–808 (2003)
74. G. Hahn, Cops, robbers and graphs. *Tatra Mt. Math. Publ.* **36**, 163–176 (2007)
75. G. Hahn, G. MacGillivray, A characterization of k -cop-win graphs and digraphs. *Discret. Math.* **306**, 2492–2497 (2006)
76. G. Hahn, F. Laviolette, N. Sauer, R.E. Woodrow, On cop-win graphs. *Discret. Math.* **258**, 27–41 (2002)
77. D. Herscovici, Graham's pebbling conjecture on products of cycles. *J. Graph Theory* **42**, 141–154 (2003)
78. J. Hopcroft, W. Paul, L. Valiant, On time versus space. *J. ACM* **24**, 332–337 (1977)
79. P. Hunter, *Complexity and Infinite Games on Finite Graphs*, Ph.D. thesis, University of Cambridge, Computer Laboratory, 2007
80. P. Hunter, S. Kreutzer, Digraph measures: Kelly decompositions, games, and orderings. *Theor. Comput. Sci.* **399**, 206–219 (2008)
81. A. Isaza, J. Lu, V. Buleitko, R. Greiner, A cover-based approach to multi-agent moving target pursuit, in *Proceedings of the 4th Conference on Artificial Intelligence and Interactive Digital Entertainment*, Stanford, CA, 2008
82. T. Johnson, N. Robertson, P. Seymour, R. Thomas, Directed tree-width. *J. Comb. Theory Ser. B* **82**, 138–154 (2001)
83. B. Kalyanasundaram, G. Schnitger, On the power of white pebbles, in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, Chicago, IL, 1988
84. M. Kanté, The rank-width of directed graphs, Preprint 2013
85. K. Kara, J. Kratochvil, D. Wood, On the complexity of the balanced vertex ordering problem. *Discret. Math. Theor. Comput. Sci.* **9**, 193–202 (2007)
86. T. Kasai, A. Adachi, S. Iwata, Classes of pebble games and complete problems. *SIAM J. Comput.* **8**, 574–586 (1979)
87. N.G. Kinnersley, The vertex separation number of a graph equals its path-width. *Inf. Process. Lett.* **42**, 345–350 (1992)
88. L.M. Kirousis, C.H. Papadimitriou, Interval graphs and searching. *Discret. Math.* **55**, 181–184 (1985)
89. L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling. *Theor. Comput. Sci.* **47**, 205–216 (1986)
90. M. Klawe, A tight bound for black and white pebbles on the pyramids. *J. ACM* **32**, 218–228 (1985)
91. S. Kreutzer, S. Ordyniak, Digraph decompositions and monotonicity in digraph searching, in *Proceedings of the 34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'08)*, Durham, UK, 2008
92. S. Kreutzer, S. Ordyniak, Distance d-Domination Games, in *Proceedings of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'09)*, Montpellier, France, 2009
93. A.S. LaPaugh, Recontamination does not help to search a graph. *J. ACM* **40**, 224–245 (1993)
94. T. Lengauer, R. Tarjan, Asymptotically tight bounds on time-space trade-offs in a pebble game. *J. ACM* **29**, 1087–1130 (1982)
95. L. Lu, X. Peng, On Meyniel's conjecture of the cop number. *J. Graph. Theor.* **71**, 192–205 (2012)
96. T. Łuczak, P. Pralat, Chasing robbers on random graphs: zigzag theorem. *Random Struct. Algorithms* **37**, 516–524 (2010)
97. F. Makedon, I. Sudborough, On minimizing width in linear layouts. *Discret. Appl. Math.* **23**, 243–265 (1989)
98. F. Makedon, C. Papadimitriou, I. Sudborough, Topological Bandwidth. *SIAM J. Algebr. Discret. Methods* **6**, 418–444 (1985)
99. F. Mazoit, N. Nisse, Monotonicity property of non-deterministic graph searching. *Theor. Comput. Sci.* **399**, 169–178 (2008)

100. N. Megiddo, S. L. Hakimi, M. Garey, D. Johnson, C.H. Papadimitriou, The complexity of searching a graph. *J. ACM* **35**, 18–44 (1998)
101. M.E. Messinger, R.J. Nowakowski, P. Prałat, Cleaning a network with brushes. *Theor. Comput. Sci.* **399**, 191–205 (2008)
102. K. Milans, B. Clark, The complexity of graph pebbling. *SIAM J. Discret. Math.* **20**, 769–798 (2006)
103. D. Moews, Pebbling graphs. *J. Comb. Theory Ser. B* **55**, 244–252 (1992)
104. C. Moldenhauer, N. Sturtevant, Evaluating strategies for running from the cops, in *Proceedings of IJCAI*, Pasadena, CA, 2009
105. H. Nagamochi, Cop-robber guarding game with cycle robber region, in *Proceedings of the 3rd International Frontiers of Algorithmics Workshop (FAW'09)*, Hefei, China, 2009
106. R. Nowakowski, Search and sweep numbers of finite directed acyclic graphs. *Discret. Appl. Math.* **41**, 1–11 (1993)
107. R.J. Nowakowski, P. Winkler, Vertex-to-vertex pursuit in a graph. *Discret. Math.* **43**, 235–239 (1983)
108. J. Obdržálek, DAG-width: connectivity measure for directed graphs, in *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, Miami, FL, 2006
109. T.D. Parsons, Pursuit-evasion in graphs, in *Theory and Applications of Graphs*, ed. by Y. Alavi, D.R. Lick (Springer, Berlin/New York, 1976), pp. 426–441
110. S. Peng, C. Ho, T. Hsu, M. Ko, C. Tang, Edge and node searching problems on trees. *Theor. Comput. Sci.* **240**, 429–446 (2000)
111. P. Prałat, When does a random graph have constant cop number? *Australas. J. Comb.* **46**, 285–296 (2010)
112. P. Prałat, N. Wormald, Meyniel's conjecture holds in random graphs, Preprint 2013
113. A. Quilliot, Jeux et pointes fixes sur les graphes. *Thèse de 3ème cycle*, Université de Paris VI, 1978, pp. 131–145
114. A. Quilliot, A short note about pursuit games played on a graph with a given genus. *J. Comb. Theory Ser. B* **38**, 89–92 (1985)
115. B. Reed, Treewidth and tangles: a new connectivity measure and some applications, in *Surveys in Combinatorics*, ed. by R.A. Bailey (Cambridge University Press, Cambridge, 1997), pp. 87–162
116. D. Richerby, D. Thilikos, Graph searching in a crime wave. *SIAM J. Discret. Math.* **23**, 349–368 (2009)
117. N. Robertson, P. Seymour, Graph minors I: excluding a forest. *J. Comb. Theory Ser. B* **35**, 39–61 (1983)
118. N. Robertson, P. Seymour, Graph minors II: algorithmic aspects of tree-width. *J. Algorithms* **7**, 309–322 (1986)
119. M. Safari, D-width: a more natural measure for directed tree-width, in *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science (MFCS'05)*, Gdańsk, Poland, 2005
120. B.S.W. Schroeder, The copnumber of a graph is bounded by $\lfloor \frac{3}{2}\text{genus}(G) \rfloor + 3$, in *Categorical Perspectives* (Kent, OH, 1998). Trends Mathematics (Birkhäuser, Boston, 2001), pp. 243–263
121. A. Scott, B. Sudakov, A new bound for the cops and robbers problem. *SIAM J. Discret. Math.* **25**, 1438–1442 (2011)
122. P. Seymour, R. Thomas, Graph searching and a min-max theorem for tree-width. *J. Comb. Theory Ser. B* **58**, 22–33 (1993)
123. P. Seymour, R. Thomas, Call routing and the ratcatcher. *Combinatorica* **14**, 217–241 (1994)
124. N.J.A. Sloane, Sequences A000088 and A001349, *The On-Line Encyclopedia of Integer Sequences* published electronically at <http://oeis.org>, 2011
125. M. Sipser, *Introduction to the Theory of Computation*, 2nd edn. (Thomson Course Technology, Boston, 2006)
126. D. Stanley, B. Yang, Fast searching games on graphs. *J. Comb. Optim.* **22**, 763–777 (2011)

127. K. Sugihara, I. Suzuki, Optimal algorithms for a pursuit-evasion problem in grids. *SIAM J. Discret. Math.* **2**, 126–143 (1989)
128. K. Sugihara, I. Suzuki, M. Yamashita, The searchlight scheduling problem. *SIAM J. Comput.* **19**, 1024–1040 (1990)
129. I. Suzuki, M. Yamashita, Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.* **21**, 863–888 (1992)
130. I. Suzuki, Y. Tazoe, M. Yamashita, T. Kameda, Searching a polygonal region from the boundary. *Int. J. Comput. Geom. Appl.* **11**, 529–553 (2001)
131. A. Takahashi, S. Ueno, Y. Kajitani, Minimal acyclic forbidden minors for the family of graphs with bounded path-width. *Discret. Appl. Math.* **127**, 293–304 (1994)
132. A. Takahashi, S. Ueno, Y. Kajitani, Mixed searching and proper-path-width. *Theor. Comput. Sci.* **137**, 253–268 (1995)
133. D.O. Theis, The cops and robber game on series-parallel graphs. Preprint 2013
134. D.B. West, *Introduction to Graph Theory*, 2nd ed. (Prentice Hall, Upper Saddle River, 2001)
135. B. Yang, Strong-mixed searching and pathwidth. *J. Comb. Optim.* **13**, 47–59 (2007)
136. B. Yang, Fast edge-searching and fast searching on graphs. *Theor. Comput. Sci.* **412**, 1208–1219 (2011)
137. B. Yang, Y. Cao, Monotonicity of strong searching on digraphs. *J. Comb. Optim.* **14**, 411–425 (2007)
138. B. Yang, Y. Cao, Monotonicity in digraph search problems. *Theor. Comput. Sci.* **407**, 532–544 (2008)
139. B. Yang, Y. Cao, On the monotonicity of weak searching, in *Proceedings of the 14th International Computing and Combinatorics Conference (COCOON'08)*, Dalian, China, 2008
140. B. Yang, Y. Cao, Digraph searching, directed vertex separation and directed pathwidth. *Discret. Appl. Math.* **156**, 1822–1837 (2008)
141. B. Yang, D. Dyer, B. Alspach, Sweeping graphs with large clique number (extended abstract), in *Proceedings of 15th International Symposium on Algorithms and Computation (ISAAC'04)*, HongKong, China, 2004
142. B. Yang, R. Zhang, Y. Cao, Searching cycle-disjoint graphs, in *Proceedings of the 1st International Conference on Combinatorial Optimization and Applications (COCOA'07)*, Xi'an, China. Lecture Notes in Computer Science, vol. 4616, 2007, pp. 32–43
143. B. Yang, D. Dyer, B. Alspach, Sweeping graphs with large clique number. *Discret. Math.* **309**, 5770–5780 (2009)
144. Ö. Yaşar, D. Dyer, D. Pike, M. Kondratieva, Edge searching weighted graphs. *Discret. Appl. Math.* **157**, 1913–1923 (2009)

Graph Theoretic Clique Relaxations and Applications

Balabhaskar Balasundaram and Foad Mahdavi Pajouh

Contents

1	Introduction	1560
1.1	Notations.....	1561
1.2	Cliques and Related Models.....	1562
2	Degree-Based Clique Relaxation: k -Plex	1563
2.1	Complexity and Approximation.....	1566
2.2	Polyhedral Results.....	1568
2.3	Algorithms.....	1573
3	Distance-Based Clique Relaxations: k -Cliques and k -Clubs	1574
3.1	Complexity and Approximation.....	1578
3.2	Polyhedral Results.....	1582
3.3	Algorithms.....	1584
4	Density-Based Models: Densest t -Subgraph and γ -Quasi-clique	1584
4.1	The Densest t -Subgraph Problem.....	1585
4.2	The Maximum γ -Quasi-clique Problem.....	1587
5	Selected Applications	1590
6	Conclusion	1591
	Cross-References	1592
	Recommended Reading	1592

Abstract

Cliques and graph theoretic *clique relaxations* are used to model clusters in graph-based data mining, where data is modeled by a graph in which an edge implies some relationship between the entities represented by its endpoints. The need for relaxations of the clique model arises in practice when dealing with massive data sets which are error prone, resulting in false or missing edges. The clique definition which requires complete pairwise

B. Balasundaram (✉) • F. Mahdavi Pajouh

School of Industrial Engineering and Management, Oklahoma State University, Stillwater, OK,
USA

e-mail: baski@okstate.edu; mahdavi@okstate.edu

adjacency in the cluster becomes overly restrictive in such situations. Graph theoretic clique relaxations address this need by relaxing structural properties of a clique in a controlled manner via user-specified parameters. This chapter surveys such clique relaxations available in the literature primarily focusing on polyhedral results, complexity studies, approximability, and exact algorithmic approaches.

1 Introduction

A clique in a graph is a set of pairwise adjacent vertices. Given a simple, undirected, and finite graph, finding a clique of maximum cardinality, clustering the graph into minimum number of cliques that cover or partition the vertex set, and enumerating inclusionwise maximal cliques are fundamental combinatorial optimization problems associated with cliques. This basic model and the associated problems have been well studied in graph theory, polyhedral combinatorics, and complexity theory, leading to many deep results in these areas.

Cliques and many graph theoretic *clique relaxations* were proposed in *social network analysis* (SNA) to model *cohesive subgroups* in social networks [149]. Such applications of the clique model in SNA can be viewed within the broader scope of graph-based clustering and data mining [18, 41, 55] where data is modeled by a graph in which an edge implies similarity, dissimilarity, or some other relationship between the entities represented by the endpoints of the edge.

One could formalize the notion of a graph-theoretic cluster by requiring one or more of the following structural properties:

1. Each vertex has a large number of vertices adjacent to it inside the cluster.
2. The pairwise distances between vertices in the cluster are small.
3. The cluster has a large number of edges with both endpoints inside the cluster.
4. The cluster has a large fraction of the maximum possible number of edges, with both endpoints inside the cluster.
5. The cluster requires a large number of vertices or edges to be deleted in order to be disconnected.

Clique is an *ideal* graph model for this purpose as it represents a cluster in which every vertex has maximum possible number of adjacent vertices, maximum possible edges within a cluster, and minimum possible pairwise distances within a cluster. Furthermore, for a clique with n vertices, at least $n - 1$ edges must be deleted to disconnect the clique or $n - 1$ vertices must be deleted resulting in a trivial graph.

The need for relaxations of the clique model arises in practice when dealing with massive data sets which are inevitably prone to errors that manifest in the graph model as false or missing edges. The clique definition which requires complete pairwise adjacency in the cluster becomes overly restrictive in such situations. Graph theoretic clique relaxations address this need by relaxing the aforementioned structural properties in a controlled manner via user-specified parameters.

This chapter surveys graph theoretic clique relaxations that have witnessed a resurgent interest in recent literature in operations research, computer science, and graph-based data mining. Several comprehensive surveys on the maximum clique problem and its applications are available that also cover application areas other than data mining such as coding theory and fault diagnosis [31, 122]. This chapter adds to recent focused surveys on clique relaxations [27, 103, 125] by providing a more comprehensive overview and surveying some more recent developments. In particular, this chapter focuses on polyhedral results, complexity studies, approximability, and exact algorithmic approaches addressing clique relaxation models. When appropriate, proofs for some of the relevant results are also included from the original works.

1.1 Notations

In this chapter, an arbitrary, simple, finite, undirected graph of order n and size m is denoted by $G = (V, E)$ where $V = \{1, \dots, n\}$ and $(i, j) \in E$ when vertices i and j are adjacent, with $|E| = m$. For an introduction to graph theory, readers are referred to texts by West [150] or Diestel [62]. The edge density of a nontrivial graph G is the ratio of number of edges in G to the number of maximum possible edges in this graph, that is, $m / \binom{n}{2}$. The edge density of a single vertex graph is taken to be unity. The *complement graph* of G is denoted by $\overline{G} = (V, \overline{E})$ where \overline{E} is the complement of the edge set E .

Given $S \subseteq V$, *induced subgraph* $G[S]$ is obtained by deleting from G all vertices (and incident edges) in $V \setminus S$. The graph obtained by the deletion of a vertex i , an edge e , a set of vertices I , or a set of edges J from G is denoted by $G - i$, $G - e$, $G - I$, and $G - J$, respectively.

The *complete graph* on n vertices is denoted by K_n , and the *complete bipartite graph* with partitions of sizes p and q is denoted by $K_{p,q}$. In particular, the graph $K_{1,n}$ is called a *star*, and $K_{1,3}$ is called a *claw*. *Cycle* and *path* on n vertices are denoted by C_n and P_n , respectively. The *adjacency matrix* A_G is a symmetric 0,1-matrix of order $n \times n$ with $a_{i,j} = 1 \Leftrightarrow (i, j) \in E$.

For a vertex $i \in V$, $N_G(i)$ denotes the set of vertices adjacent to i in G , called the *neighborhood* of i , and $N_G[i] = \{i\} \cup N_G(i)$ denotes the *closed neighborhood* of i . The set of *non-neighbors* of i is given by $V \setminus N_G[i]$. Denote by $\deg_G(i)$, the degree of vertex i in G given by $|N_G(i)|$. The maximum and minimum degrees in a graph are denoted, respectively, by $\Delta(G)$ and $\delta(G)$.

For two vertices $i, j \in V$, $d_G(i, j)$ denotes the length of a shortest path between i and j in G . By convention, when no path exists between two vertices, the shortest distance between them is infinity. The *diameter* of a graph is defined as $\text{diam}(G) = \max_{i,j \in V} d_G(i, j)$. For a graph $G = (V, E)$ and positive integer k , the k th power of G is defined as $G^k = (V, E^k)$, where $E^k = \{(i, j) : i, j \in V, d_G(i, j) \leq k\}$. The simple graph G^k is constructed from the original graph by adding edges corresponding to all pairs of vertices with distance no more than k between them in G .

The *vertex connectivity* $\kappa(G)$ and *edge connectivity* $\lambda(G)$ of a graph are the minimum number of vertices and edges, respectively, whose removal results in a disconnected or trivial graph. A graph is said to be t -connected if $\kappa(G) \geq t$ and t -edge connected if $\lambda(G) \geq t$. It is known that when G is nontrivial, $\kappa(G) \leq \lambda(G) \leq \delta(G)$ [150]. If the graph G under consideration is obvious, the subscript G in the neighborhood, degree, and distance notations is sometimes dropped for simplicity.

1.2 Cliques and Related Models

Definition 1 A clique is a subset of vertices that are pairwise adjacent in the graph.

Definition 2 An independent set (stable set, vertex packing) is a subset of vertices that are pairwise nonadjacent in the graph.

The *maximum clique problem* is to find a clique of maximum cardinality. The *clique number* $\omega(G)$ is the cardinality of a maximum clique in G . The maximum cardinality of an independent set of G is called the *independence number* of the graph G and is denoted by $\alpha(G)$. The associated problem of finding a largest independent set is the *maximum independent set problem*. Clearly, I is an independent set in G if and only if I is a clique in \overline{G} and consequently $\alpha(G) = \omega(\overline{G})$.

In this chapter, *maximality* and *minimality* of sets are always defined based on inclusion and exclusion, respectively. For instance, a *maximal clique* is one that is not a proper subset of another clique. The maximum clique and independent set problems on arbitrary graphs are NP-hard [74] and are hard to approximate within $n^{1-\epsilon}$ for any $\epsilon > 0$ [79]. Furthermore, finding cliques and independent sets parameterized by solution size are not known to be fixed-parameter tractable problems and, in fact, are basic $W[1]$ -hard problems [65].

Definition 3 A proper coloring of a graph is one in which every vertex is colored such that no two vertices of the same color are adjacent.

A graph is said to be t -colorable if it admits a proper coloring with t colors. Vertices of the same color are referred to as a *color class*, and they induce an independent set. The *chromatic number* of the graph, denoted by $\chi(G)$, is the minimum number of colors required to properly color G . For any graph G , $\omega(G) \leq \chi(G)$, as different colors are required to color the vertices of a clique.

Definition 4 A vertex cover is a subset of vertices such that every edge in the graph is incident at some vertex in the subset.

Clearly, C is a vertex cover of G if and only if $V \setminus C$ is an independent set in G . The *minimum vertex cover problem* seeks to find a vertex cover of minimum cardinality.

Definition 5 A dominating set is a subset of vertices such that every vertex in the graph is either in this set or has a neighbor in this set.

The minimum cardinality of a dominating set is called the *domination number*, denoted by $\gamma(G)$. It should be noted that every maximal independent set is also a minimal dominating set.

2 Degree-Based Clique Relaxation: k -Plex

The degree-based clique relaxation, called a k -plex, was introduced by Seidman and Foster for modeling cohesive subgroups in social network analysis [133]. The parameter $k \geq 1$ is a positive integer that controls the “degree” of relaxation and includes clique as a special case. A complementary model to k -plex can also be defined along similar lines that relaxes independent sets.

Definition 6 A subset $S \subseteq V$ is a k -plex if $\delta(G[S]) \geq |S| - k$.

Definition 7 A subset $J \subseteq V$ is a co- k -plex if $\Delta(G[J]) \leq k - 1$.

Clearly, 1-plexes and co-1-plexes are simply cliques and independent sets, respectively. For $k > 1$, a k -plex is a degree-based relaxation of the clique definition which allows for at most $k - 1$ non-neighbors in S , while the co- k -plex is a degree-based relaxation of the independent set definition which allows for at most $k - 1$ neighbors in J . Clearly, S is a k -plex in G if and only if S is a co- k -plex in \bar{G} . Figure 1 illustrates this concept. The set $\{1, 2, 3, 4\}$ is a 1-plex (clique), sets $\{1, 2, 3, 4, 5\}$ and $\{1, 2, 3, 4, 6\}$ are 2-plexes (maximal and maximum), and the entire vertex set forms a 3-plex. In the same graph $\{5, 6\}$ forms a co-1-plex (independent set), while $\{1, 5, 6\}$ forms a co-2-plex.

The *maximum k -plex problem* is to find a largest cardinality k -plex in G , the cardinality of which is the *k -plex number* of G denoted by $\omega_k(G)$. The *maximum co- k -plex problem* is similarly defined with the *co- k -plex number* denoted by $\alpha_k(G)$. Some early results on k -plexes established by Seidman and Foster [133] are summarized in Theorem 1. An alternate characterization of k -plexes, Theorem 2, was also established therein.

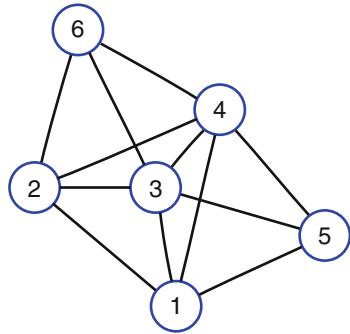
Theorem 1 ([133]) *Let graph G be a k -plex on n vertices. Then,*

1. *Any vertex-induced subgraph of G is a k -plex.*
2. *If $k < \frac{n+2}{2}$, then $\text{diam}(G) \leq 2$.*
3. $\kappa(G) \geq n - 2k + 2$.

Proof

1. Every induced subgraph of G can be obtained by deleting vertices, one at a time. If G is a k -plex, $\delta(G) \geq n - k$. In G' obtained by deleting some vertex from G ,

Fig. 1 Illustration of k -plexes and co- k -plexes



- the degree of all the vertices and hence the minimum degree can drop by at most 1. Hence, the new graph continues to be a k -plex as $\delta(G') \geq (n - 1) - k$.
2. For any vertex v in the k -plex G , the number of vertices in the closed neighborhood $|N[v]| \geq n - k + 1$. Thus, for two arbitrary vertices u, v we have $|N[u]| + |N[v]| \geq 2n - 2k + 2 > n$ by the given condition. Hence, $N[u]$ and $N[v]$ are not disjoint. Thus, for all u, v , $d_G(u, v) \leq 2 \Rightarrow \text{diam}(G) \leq 2$.
 3. Suppose we delete t vertices from G . The resulting graph G' is still a k -plex and the above argument still holds. If $k < \frac{n-t+2}{2}$, then $\text{diam}(G') \leq 2$. Thus, G' is still connected if $t < n - 2k + 2 \Rightarrow \kappa(G) \geq n - 2k + 2$.

Remark 1 Note that the condition for a diameter-two bound is sharp. Consider the graph $K_{k-1} \cup K_{k-1}$ which is a k -plex for every k . Thus, a k -plex could be disconnected even for $k = \frac{n+2}{2}$. However, no isolated vertex could be present in a k -plex of size $k + 1$ or more.

Remark 2 It is also useful to note that in an arbitrary graph G , any k -element subset of vertices is a k -plex and any k -plex is also a $k + 1$ -plex.

Theorem 2 ([133]) $G = (V, E)$ is a k -plex if and only if every k -element subset of vertices forms a dominating set in G .

Proof Suppose there exist k vertices that do not form a dominating set, then there must be some vertex v distinct from these k vertices that is not adjacent to any of these k vertices. This contradicts the fact that G is a k -plex.

Now suppose that G is not a k -plex. Hence, $|V| \geq k + 1$, and there exists a vertex $v \in V$ such that $|N(v)| \leq |V| - k - 1$. Hence, $|V \setminus N[v]| \geq k$, and it does not dominate G , specifically the vertex v . \square

A characterization of 2-plexes via the co-2-plex number is provided in [Proposition 1 \[105\]](#). This characterization does not extend to general k in a straightforward manner.

Proposition 1 ([105]) $G = (V, E)$ is a 2-plex if and only if $\alpha_2(G) = \min\{2, |V|\}$.

Proof Suppose G is a 2-plex and $\alpha_2(G) \geq 3$. Then, V contains a subset $S = \{u, v, w\}$ that forms a co-2-plex. At least one vertex in S must have zero degree in $G[S]$, which contradicts the fact that $G[S]$ is a 2-plex.

Conversely, suppose G is graph with $|V| \geq 3$ and $\alpha_2(G) = 2$. If G is not a 2-plex, there exists $u \in V$ such that $|N(u) \cap V| \leq |V| - 3$. Hence, there exist vertices $v, w \in V \setminus N[u]$, and the set $\{u, v, w\}$ forms a co-2-plex of size 3. \square

There is a close relationship between the maximum k -plex problem and the *minimum bounded-degree- d vertex deletion problem* (d -BDD) studied in [114]. The d -BDD problem defined next is a generalization of the minimum vertex cover problem.

Definition 8 Given a graph $G = (V, E)$ and a fixed integer $d \geq 0$, a bdd- d -set is defined as a subset $S \subseteq V$ for which $G[V \setminus S]$ is a graph with maximum vertex degree at most d [113].

The minimum bounded-degree- d vertex deletion problem is to find a minimum cardinality bdd- d -set in G . Note that for $d = 0$, a bdd-0-set is precisely a vertex cover of G . Further note that if S is a bdd- d -set, then $V \setminus S$ by definition is a co- k -plex with $k = d + 1$. Lemma 2 shows the relationship between the maximum k -plex problem and the minimum d -BDD problem.

Proposition 2 ([113]) A graph $G = (V, E)$ contains a k -plex of size c if and only if \overline{G} contains a bdd- d -set of size $|V| - c$ with $d = k - 1$.

Proof It follows from the observation that $S \subseteq V$ is a k -plex of size c in G if and only if S is a co- k -plex in \overline{G} . \square

A natural degree-based generalization of graph coloring is the following.

Definition 9 A proper co- k -plex coloring of a graph $G = (V, E)$ is an assignment of colors to V such that each vertex has at most $k - 1$ neighbors in the same color class.

Co- k -plexes have also been called $(k - 1)$ -dependent sets in the literature [64] and co- k -plex coloring has also been studied under other names such as defective coloring [59] and k -improper coloring [80]. A proper co- k -plex coloring of G partitions G into co- k -plexes and is equivalent to a k -plex partitioning of \overline{G} . Obviously, an independent set cannot contain a clique of size larger than one, while k -plexes of size two or more can be found inside a co- k -plex for $k \geq 2$. A sharp upper bound on the maximum size of the intersection was established in [20] showing that if $\Delta(G) \leq k - 1$, then $\omega_k(G) \leq 2k - 2 + (k \bmod 2)$. In light of this observation, two graph invariants can be defined to relax the graph coloring problem as discussed next.

The minimum number of colors required in a proper co- k -plex coloring of G is called the *co- k -plex chromatic number* denoted by $\chi_k(G)$ [21]. The *weighted co- k -plex chromatic number* denoted by $\chi_k^w(G)$ is defined as the minimum weighted sum of the color classes, where the weight of a color class is the size of a maximum k -plex contained in the color class [104]. These definitions are equivalent when $k = 1$, but not for $k > 1$. The latter provides a tighter upper bound on the k -plex number as the following inequality is true for any graph G [20, 104]:

$$\omega_k(G) \leq \chi_k^w(G) \leq [2k - 2 + (k \bmod 2)]\chi_k(G).$$

2.1 Complexity and Approximation

For arbitrary k , where the parameter is considered as part of the input for each instance of the problem, the maximum k -plex problem is trivially NP-hard as it includes the maximum clique problem as a special case. However, the complexity of the problems for fixed k (prescribed as part of the problem definition) is a nontrivial question.

A graph property Π is said to be *hereditary on induced subgraphs*, if G is a graph with property Π and the deletion of any subset of vertices does not produce a graph violating Π . Property Π is said to be *nontrivial* if it is true for a single vertex graph and is not satisfied by every graph. A property Π is said to be *interesting* if there are arbitrarily large graphs satisfying Π . The *maximum Π problem* is to find the largest order induced subgraph that does not violate property Π . Naturally, this optimization problem is meaningful only if Π is nontrivial and interesting.

Yannakakis [153] obtained a general complexity result for such properties Π showing that the maximum Π problem for nontrivial, interesting graph properties that are hereditary on induced subgraphs is NP-hard. Inapproximability of such problems was also established by Lund and Yannakakis in [96]. Since k -plexes and co- k -plexes are nontrivial, interesting, and hereditary, the maximum k -plex and co- k -plex problems are NP-hard based on this result and hard to approximate within $O(n^\epsilon)$ for some $\epsilon > 0$ unless $P = NP$.

The NP-hardness of the maximum k -plex problem and the maximum co- k -plex problem for every fixed k was also independently established in [20, 64], respectively. In [61], the maximum co- k -plex problem was shown to be NP-hard even on planar and bipartite graphs for fixed $k \geq 2$. Recently in [81, 88], this problem was shown to be NP-hard for every fixed k on unit-disk graphs (UDGs, intersection graphs of unit-disks in a Euclidean plane [54]). Note that the maximum independent set problem is NP-hard on planar graphs [74] and unit-disk graphs [54], but it is polynomial time solvable on bipartite graphs through graph-matching techniques [56]. Analogous to what is known for cliques and vertex covers, the maximum k -plex problem parameterized by solution size is $W[1]$ -hard [91], while the d -BDD problem is fixed-parameter tractable when parameterized by solution size [91, 118].

The maximum co- k -plex problem admits a polynomial time approximation scheme (PTAS) on UDGs [81, 88] and a $3k$ -approximation algorithm on

UDGs [21]. The PTAS employs a “divide-and-conquer” approach similar to that used for independent sets in UDGs [84], and the constant factor approximation employs approaches similar to those developed in [100] for independent sets. The minimum co- k -plex coloring problem is also NP-hard for every fixed k on UDGs [81, 88]. A polynomial time 6-approximation algorithm for this problem was developed in [81, 88], and a 3-approximate algorithm for this problem was presented in [21]. Some bounds on the graph invariants $\omega_k(G)$, $\alpha_k(G)$, and $\chi_k(G)$ were also developed in [21], used to establish the aforementioned approximation guarantees for UDGs and generalized claw-free graphs. These identities summarized in **Theorem 3** generalize some well-known results for cliques, independent sets, and coloring (see for instance [83, 100]).

Theorem 3 ([21]) Consider a graph $G = (V, E)$,

1. $\alpha_k(G) \leq k\alpha(G)$.
2. $\omega_k(G) \leq k\omega(G)$.
3. If G is a $(p + 1, k)$ -claw-free, that is, $\alpha_k(G[N(v)]) \leq p \forall v \in V$, then

$$|J| \geq \frac{\alpha_k(G)}{p}$$

for any maximal co- k -plex J .

4. If G is a unit-disk graph with an embedding specified in the Euclidean X - Y plane, then $\alpha_k(G[N(v)]) \leq 5k \forall v \in V$, and if v is the vertex corresponding to the minimum X -coordinate, then $\alpha_k(G[N(v)]) \leq 3k$.
5. $\chi_k(G) \leq \lfloor \frac{d(G)}{k} \rfloor + 1$ where the polynomial time computable invariant $d(G)$ denotes the largest d such that G has an induced subgraph of minimum degree at least d .
6. If G is a unit-disk graph, then $\chi_k(G) \geq \frac{d(G)}{3k}$.

Proof

1. Suppose $\alpha_k(G) \geq k\alpha(G) + 1$. Let $G^{(0)}$ be a co- k -plex induced in G of size $k\alpha(G) + 1$. Apply the following greedy algorithm for a maximal independent set in graph $G^{(0)}$, with set I initially empty and $i = 0$. Pick any vertex v from $V(G^{(i)})$, let $I \leftarrow I \cup \{v\}$ and $G^{(i+1)} \leftarrow G^{(i)} - N[v]$. Increment i and repeat until $G^{(i+1)}$ is null. Since $G^{(0)}$ is a co- k -plex, so are all $G^{(i)}$, and thus $\Delta(G^{(i)}) \leq k - 1$. So $|V(G^{(i+1)})| = |V(G^{(i)})| - 1 - |N(v) \cap V(G^{(i)})| \geq |V(G^{(i)})| - k$. After $\alpha(G)$ iterations $|I| = \alpha(G)$ and $|V(G^{(\alpha(G))})| \geq k\alpha(G) + 1 - k\alpha(G) = 1$. So the algorithm can add at least one more vertex to independent set I , a contradiction.
2. Implied by 1.
3. Suppose J^* is a maximum co- k -plex in G . Further assume that $p \geq k$, as the only $(p + 1, k)$ -claw-free graphs with $p < k$ are themselves co- k -plexes. Then

$$J^* \setminus \bigcup_{v \in J} N[v] = \emptyset;$$

otherwise, there exists a $v \in J^*$ such that $N[v] \cap J = \emptyset$ contradicting the maximality of J . Hence, we have

$$|J^*| = \alpha_k(G) \leq \sum_{v \in J} |N[v] \cap J^*| \leq p|J|.$$

4. By Part 1 we have $\alpha_k(N(v)) \leq k\alpha(N(v)) \leq 5k$ where the last inequality follows from the result of Marathe et al. [100] that for a UDG, $\alpha(N(v)) \leq 5 \forall v \in V$. The bound for the minimum X-coordinate can be obtained similarly by extending an analogous result from [100].
5. Denote by $d(G)$ the largest d such that G has an induced subgraph of minimum degree at least d . The invariant $d(G)$ can be found in polynomial time using a simple algorithm [83] along with an associated ordering v_1, \dots, v_n of vertices of G such that any v_i has at most $d(G)$ neighbors in v_1, \dots, v_{i-1} . In that order, for each vertex, assign the smallest *available* color. A color is said to be *available* at a vertex if it has been assigned to at most $k - 1$ neighbors of the current vertex, already processed in the list ordering.

Consider any vertex v_i , $i = 2, \dots, n$. It can have at most $d(G)$ neighbors already colored in the list so far. There must exist a color from the set $\{1, \dots, \lfloor \frac{d(G)}{k} \rfloor + 1\}$ that has been assigned to at most $k - 1$ neighbors of v_i . If not, the number of colored neighbors of v_i exceeds $d(G)$. Since this is true for any v_i , the above algorithm uses no more than $\lfloor \frac{d(G)}{k} \rfloor + 1$ colors.

6. Let H be the induced subgraph of G such that $\delta(H) = d(G)$. Consider a vertex $v \in V(H)$ of minimum X-coordinate, and let H' denote the graph induced by $N(v)$. Any proper co- k -plex coloring of G is also proper for H' , and it partitions H' into co- k -plexes. Hence, we have

$$\chi_k(G) \geq \chi_k(H') \geq \frac{|V(H')|}{\alpha_k(H')} \geq \frac{d(G)}{3k}. \quad \square$$

2.2 Polyhedral Results

The maximum k -plex problem can be formulated as the following integer program:

$$\omega_k(G) = \max \sum_{i \in V} x_i$$

Subject to

$$\begin{aligned} \sum_{j \in V \setminus N[i]} x_j &\leq (k-1)x_i + |V \setminus N[i]|(1-x_i) \quad \forall i \in V \\ x_i &\in \{0, 1\} \quad \forall i \in V. \end{aligned}$$

Given a graph $G = (V, E)$, the k -plex polytope of G denoted by $P_k(G)$ is the convex hull of the incidence vectors of all k -plexes in G . The k -plex polytope is full dimensional and the variable 0,1-bound constraints are facet inducing as stated in [Theorem 4](#). The case $k = 1$ which corresponds to the well-known clique polytope is excluded from consideration.

Theorem 4 ([20]) *Given a graph $G = (V, E)$ and an integer $k > 1$,*

1. $\dim(P_k(G)) = n$,
2. *For each $i \in V$, inequalities $x_i \geq 0$ and $x_i \leq 1$ are facet inducing for $P_k(G)$.*

Several facets and valid inequalities for $P_k(G)$ for arbitrary and restricted graph classes are available from [20, 105]. The results for the co- k -plex polytope developed in [105] are adapted via complementation for $P_k(G)$ in this chapter. It should be noted that these results extend several analogous results known for the clique and stable set polytopes [49, 50, 116, 117, 121, 142].

2.2.1 Co- k -plex and Independent Set Inequalities

Proposition 3 ([20]) *(Co- k -plex inequalities) Given a graph $G = (V, E)$ and an integer $k \geq 1$, let $r_k = 2k - 2 + (k \bmod 2)$. If $I \subseteq V$ is a maximal co- k -plex of size at least $r_k + 1$ in G , then the following inequality is valid for $P_k(G)$:*

$$\sum_{i \in I} x_i \leq r_k.$$

[Proposition 3](#) can be further strengthened for the case, $k = 2$ as outlined in [Theorem 5](#).

Theorem 5 ([20]) *(Co-2-plex facets) Given a graph $G = (V, E)$ and $J \subseteq V$ such that $|J| \geq 3$, the inequality $\sum_{i \in J} x_i \leq 2$ induces a facet of $P_2(G)$ if and only if J is a maximal co-2-plex.*

Proof The validity of this inequality for $P_2(G)$ follows from [Proposition 3](#). So, it suffices to show that the face of $P_2(G)$ induced by this inequality,

$$\left\{ x \in P_2(G) : \sum_{i \in J} x_i = 2 \right\},$$

is of dimension $n - 1$.

Note that if J is a maximal co- k -plex, for every $v \in V \setminus J$, at least one of the following conditions must hold:

1. $\exists j \in J \cap N(v)$ such that $|N(j) \cap J| = k - 1$ and including v would cause degree of j in the induced subgraph to be k .
2. $|N(v) \cap J| \geq k$ and upon inclusion, v would have degree k or more in the induced subgraph.

For every $v \in V \setminus J$, the above two conditions for a maximal co-2-plex imply the existence of two vertices $u, w \in J$ such that $\{v, u, w\}$ is a 2-plex. Indeed, if the first

case holds, let $u \in J \cap N(v)$, then $N(u) \cap J = \{w\}$ and $\{v, u, w\}$ is a 2-plex. If the second case holds, $\{u, w\} \subseteq J \cap N(v)$ and again $\{v, u, w\}$ is a 2-plex.

W.l.o.g. assume that $J = \{1, \dots, r\}$ and $V \setminus J = \{r + 1, \dots, n\}$, where $r \geq 3$. Let $e_i \in \mathbb{R}^n$ denote the unit vector with i th component one and all others zero. The affinely independent vectors are constructed as follows:

$$\begin{aligned} x^v &= e_v + e_r, \quad \forall v = 1, \dots, r - 1, \\ x^r &= e_1 + e_2 \text{ (note that } x^r \text{ is distinct from } x^1, \dots, x^{r-1} \text{ as } r \geq 3), \\ x^v &= e_v + e_u + e_w, \quad \forall v = r + 1, \dots, n, \end{aligned}$$

where for each $v \in V \setminus J$, $u, w \in J$ are particular vertices described before. Clearly, $x^v \in \{x \in P_2(G) : \sum_{i \in J} x_i = 2\}$, $\forall v \in V$, and it is easy to verify that they are affinely independent.

Conversely, suppose $\sum_{i \in J} x_i \leq 2$ induces a facet of $P_2(G)$ and J is not a co-2-plex. Then, there exists some $v \in J$ with 2 neighbors in J which form a 2-plex that violates the facet-inducing inequality, leading to a contradiction. Hence, J must be a co-2-plex. If J is not maximal, then there exists a valid maximal co-2-plex inequality that dominates the given facet-inducing inequality. Hence, J must be a maximal co-2-plex. \square

Padberg [121] (see also [71]) showed that the maximal independent set inequalities are facet inducing for the clique polytope, and [Theorem 5](#) is a direct analogue for the 2-plex polytope. However, for $k \geq 3$, co- k -plex inequalities are not necessarily facets of $P_k(G)$ for arbitrary graphs. It should be noted that the bound r_k while sharp is not necessarily achieved in all co- k -plexes and the k -plex number could be strictly smaller than r_k . Even in co- k -plexes where the bound is achieved, there may not be enough affinely independent incidence vectors of k -plexes achieving that bound. This motivated the study of k -plex rank inequalities of $P_k(G)$ [105]. For any $I \subseteq V$, the following k -plex rank inequality is valid for $P_k(G)$:

$$\sum_{i \in I} x_i \leq \omega_k(G[I]).$$

A sufficient condition under which the k -plex rank inequality derived based on the k -plex number of G induces a facet of $P_k(G)$ was developed in [105], extending a result of Chvátal [53].

Definition 10 Given a graph $G = (V, E)$ and an edge $e \in \overline{E}$ in the complement graph \overline{G} , e is called k -plex critical if $\omega_k(G + e) = \omega_k(G) + 1$ where $G + e = (V, E \cup \{e\})$.

Theorem 6 ([105]) Let $E^* \subseteq \overline{E}$ be the set of all k -plex critical edges of G . If the graph $G^* = (V, E^*)$ is connected, then the k -plex rank inequality $\sum_{i \in V} x_i \leq \omega_k(G)$ induces a facet of $P_k(G)$.

Maximal independent set inequalities play an important role in describing the clique polytope of graphs. Specifically, along with nonnegativity constraints, they are sufficient to completely describe the clique polytope of perfect graphs [58]. Preliminary results in this direction established in [105] show that the co-2-plex facets and the variable bounds are sufficient to describe the 2-plex polytope of paths, mod-3 cycles (see [Definition 11](#)), and co-2-plexes.

The co- k -plex inequalities for the k -plex polytope generalize the independent set inequalities for the clique polytope. Note that $r_1 = 1$, and hence, the co-1-plex inequalities are precisely the independent set inequalities for the 1-plex (clique) polytope. An alternate approach based on the observation that an independent set can contain at most k vertices from a k -plex for any k leads to the independent set inequalities for the k -plex polytope.

Proposition 4 ([20]) (*Independent set inequalities*) Given a graph $G = (V, E)$ and an integer $k \geq 1$, if $I \subseteq V$ is a maximal independent set of size at least $k + 1$ in G , then $\sum_{i \in I} x_i \leq k$ is a valid inequality for $P_k(G)$. Furthermore, if $E = \emptyset$ and $n \geq k + 1$, the inequality induces a facet of $P_k(G)$.

When $k = 1$, co-1-plex and independent set inequalities described here coincide with the well-known inequalities for the clique polytope. For $k = 2$, independent set inequalities are dominated by the facet-inducing co-2-plex inequalities since $r_k = k = 2$. However, that is not the case for $k \geq 3$ as maximal independent set inequalities upon lifting can produce facets of $P_k(G)$ that are not identified as co- k -plex inequalities or k -plex rank inequalities [20].

In addition to independent set inequalities, some other well-known class of inequalities for the clique polytope are hole and antihole inequalities [116, 121], wheel inequalities and their generalizations [23, 49], web and antiweb inequalities [142], and many others [24, 33, 75]. Extensions of some of these results to the k -plex polytope is presented next.

2.2.2 Hole Inequalities and Special Cases

Definition 11 A hole is an induced chordless cycle of size 4 or more. For a positive integer k , a hole on n vertices with $n \equiv 0 \pmod{k}$ is called a mod- k hole, otherwise it is an odd mod- k hole. The complement of a hole is called an antihole.

Valid inequalities based on holes were developed in [20] for the k -plex polytope. These are also known to be facet inducing for $P_k(C_n)$ depending on n and k .

Proposition 5 ([20]) (*Hole inequalities*) Given a graph $G = (V, E)$ and a positive integer k , if $I \subseteq V$ is a hole of size $\geq k + 3$, then $\sum_{i \in I} x_i \leq k + 1$ is a valid inequality for $P_k(G)$.

Theorem 7 ([20]) Consider a cycle C_n on n vertices, positive integer k with $n \geq k + 3$ such that n and $k + 1$ are relatively prime. Then, the inequality $\sum_{i \in V} x_i \leq k + 1$ induces a facet of $P_k(C_n)$.

Theorem 7 for odd mod- $(k+1)$ holes is a direct analogue of the result developed by Nemhauser and Trotter [116]. Interestingly, however, mod- $(k+1)$ holes can also induce facets of $P_k(C_n)$ [20].

Theorem 8 ([20]) Consider a cycle C_n on n vertices and a positive integer $k \geq 4$ such that $n \geq k+3$ and n is a multiple of $k+1$. Then, the inequality $\sum_{i \in V} x_i \leq k+1$ induces a facet of $P_k(C_n)$.

Note that antiholes are k -plexes for $k \geq 3$. However, when $k=2$, antiholes do lead to valid inequalities for the 2-plex polytope [105]. Lifting this inequality leads to the *antiwheel* inequality, details of which can be found in [105].

Theorem 9 ([105]) Consider the complement of a cycle on n vertices \bar{C}_n with $n \geq 4$ such that $n \not\equiv 0 \pmod{3}$. Then, the inequality $\sum_{i \in V} x_i \leq \lfloor \frac{2n}{3} \rfloor$ induces a facet of $P_2(\bar{C}_n)$.

Definition 12 Given a graph $G = (V, E)$ and an integer p , suppose $n \geq 2$, $1 \leq p \leq \frac{n}{2}$. If $E = \{(i, j) : j = i + p, \dots, i + n - p, \forall i \in V\}$ where sums are written modulo n , then G is called a web, denoted by $W(n, p)$. The complement of a web is an antiweb.

Webs were introduced by Trotter [142] to generalize odd hole and antihole inequalities for the stable set polytope. This result was extended in [105] to show that antiwebs induce facets of the 2-plex polytope as outlined next. It should be noted that $W(n, 1)$ is a complete graph, for $p \geq 2$, $W(2p+1, p)$ is an odd hole and $W(2p+1, 2)$ is an odd antihole.

Theorem 10 ([105]) (Web inequalities) Consider the antiweb $\bar{W}(n, p)$ where n and $p+1$ are relatively prime and $p < \lfloor \frac{n}{2} \rfloor$, then the inequality $\sum_{i \in V} x_i \leq p+1$ induces a facet of $P_2(\bar{W}(n, p))$.

This section is concluded with valid inequalities produced by generalizing claws. These were originally developed for the co- k -plex polytope in [105].

Definition 13 Given a graph $G = (V, E)$ and an integer $k \geq 1$, if there exists a vertex $u \in V$ such that $N(u) = V \setminus \{u\}$, $N(u)$ is a co- k -plex, and $|N(u)| \geq \max\{3, k\}$, then G is called a k -claw. Vertex u is called the center of the k -claw.

Theorem 11 ([105]) Consider a graph $G = (V, E)$ such that \bar{G} is a k -claw with center $u \in V$ and integer $k \geq 2$, the inequality $(n-k)x_u + \sum_{i \notin N_G[u]} x_i \leq n-1$ is a facet of $P_k(G)$.

Of particular note is that a k -claw can properly contain another k -claw, and both would give rise to distinct facets.

2.3 Algorithms

Exact algorithms for the maximum k -plex problem can be broadly categorized as combinatorial approaches and polyhedral approaches. A branch-and-cut framework is developed in [20] that incorporates maximal independent set and co- k -plex inequalities identified via heuristic separation procedures. An *iterative peel-branch-and-cut* algorithm to solve the maximum k -plex problem to optimality on “scale-free” graphs is also developed in [20] that uses the branch-and-cut algorithm in combination with preprocessing and graph decomposition approaches.

Scale-free graphs [22, 52] are very large and sparse graphs which exhibit a power-law degree distribution. The large-scale and sparse nature of such graphs present computational challenges that are addressed by exploiting the power-law nature of such graphs. The need for such specialized techniques are motivated by graph-based data mining applications where the graph models exhibit a power-law degree distribution [20].

Numerous exact combinatorial algorithms are known for the maximum clique problem such as those developed by Bron and Kerbosch [38], Balas and Yu [16], Applegate and Johnson [7], Carraghan and Pardalos [43], Babel [13], Balas and Xue [15], Wood [151], Sewell [134], Östergård [120], and Tomita and Kameda [141].

Östergård’s algorithm [120], which extends the approaches proposed by Applegate and Johnson [7] and Carraghan and Pardalos [43], has been extended to the maximum k -plex problem by McClosky and Hicks [102, 106] and by Trukhanov et al. [143, 144]. Design of such algorithms and variations, implementation details, computational results on DIMACS benchmarks [63], as well as comparisons between algorithms can be found in [106, 114, 144].

Algorithm 1 presents a basic extension of Östergård’s clique algorithm [120] to the maximum k -plex problem, which works by first ordering vertices in V based on degree or other criteria as v_1, v_2, \dots, v_n . Let set $S_i \subseteq V$ be defined as $S_i = \{v_i, v_{i+1}, \dots, v_n\}$ and let $c(i)$ denote the size of the maximum k -plex in $G[S_i]$. Clearly $c(n) = 1$ and $c_1 = \omega_k(G)$. **Algorithm 1** recursively computes $c(i)$ starting from $c(n)$ and down to $c(1)$ for which the following holds:

$$c(i) = \begin{cases} c(i + 1) + 1, & \text{if the corresponding solution contains } v_i, \\ c(i + 1), & \text{otherwise.} \end{cases}$$

A different approach is undertaken in [114] where the close relationship to the minimum bounded-degree- d vertex deletion problem and its fixed-parameter tractability with respect to the solution size is exploited to develop an exact algorithm for the maximum k -plex problem. Another important result in advancing this line of research is the following. A basic result of Nemhauser and Trotter [117] is fundamental to many fixed-parameter tractable and approximation algorithms for the minimum vertex cover problem. In [70], a generalization of this result is

Algorithm 1 Maximum k -plex algorithm

```

1: procedure MAXKPLEX(G)
2:   Order vertices in  $V$  as  $v_1, \dots, v_n$ 
3:   max := 0
4:   for  $i := n$  downto 1 do
5:      $C := \{v \in S_i \setminus \{v_i\} : \{v, v_i\}$  is a  $k$ -plex}
6:     found := false
7:     FINDKPLEX( $C, \{v_i\}$ )
8:      $c(i) := \max$ 
9:   end for
10:  return max
11: end procedure
12: procedure FINDKPLEX(C,P)
13:   if  $C = \emptyset$  then
14:     if  $|P| > \max$  then
15:       max :=  $|P|$ 
16:       found := true
17:     end if
18:     return
19:   end if
20:   while  $C \neq \emptyset$  do
21:     if  $|C| + |P| \leq \max$  then
22:       return
23:     end if
24:      $i := \min\{j : v_j \in C\}$ 
25:     if  $c(i) + |P| \leq \max$  then
26:       return
27:     end if
28:      $C := C \setminus \{v_i\}; P' := P \cup \{v_i\}; C' := \{v \in C : P' \cup \{v\}$  is a  $k$ -plex}
29:     FINDKPLEX( $C', P'$ )
30:     if found = true then
31:       return
32:     end if
33:   end while
34: end procedure

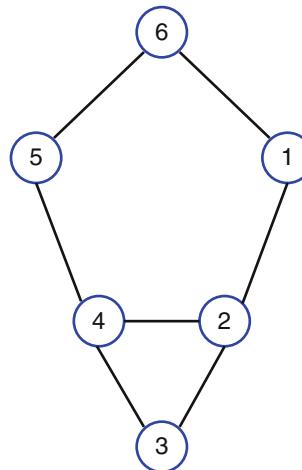
```

presented which can further improve the parameterized algorithmics of the d -BDD problem.

3 Distance-Based Clique Relaxations: k -Cliques and k -Clubs

The distance-based clique relaxations were introduced in social network analysis by Luce [94] soon after cliques were used to model cohesive subgroups [95]. These models relaxed the adjacency requirement of cliques to a short path requirement, with the models being refined and redefined by others [3, 112]. A discussion of the different definitions can be found in [19]. Two basic models based on relaxing pairwise distances can be defined as follows.

Fig. 2 A graph in which set $S = \{1, 2, 3, 4, 5\}$ is a 2-clique but not a 2-club



Definition 14 A subset of vertices $S \subseteq V$ is a k -clique if $d_G(u, v) \leq k$ for all $u, v \in S$.

Definition 15 A subset of vertices $S \subseteq V$ is a k -club if $d_{G[S]}(u, v) \leq k$ for all $u, v \in S$, that is, $\text{diam}(G[S]) \leq k$.

By definition, 1-clique and 1-club correspond to the classical clique. Clearly, every k -club is also a k -clique since $\text{diam}(G[S]) \leq k \Rightarrow d_{G[S]}(u, v) \leq k \forall u, v \in S \Rightarrow d_G(u, v) \leq k \forall u, v \in S$. The converse, however, is not always true. In fact for $k \geq 2$, a k -clique S can contain vertices $u, v \in S$ such that $d_G(u, v) \leq k$ but $d_{G[S]}(u, v) > k$. In Fig. 2 [3], set $S = \{1, 2, 3, 4, 5\}$ forms a 2-clique which is not a 2-club. It should be noted that $d_G(1, 5) = 2$ and $d_{G[S]}(1, 5) = 3$.

The k -clique model was originally introduced in social network analysis by Luce [94] to model cohesive subgroups. The k -club model [3, 112] was developed to address the drawback of k -cliques where members outside the cohesive subgroup can be used on the short paths required between members inside the group. The k -club number of G denoted by $\overline{\omega}_k(G)$ is the cardinality of a maximum k -club in G , and the maximum k -club problem is to find a k -club of the maximum cardinality in G . The k -clique number of G denoted by $\widetilde{\omega}_k(G)$ and the maximum k -clique problem are similarly defined. For a given graph G and a positive integer k , the following inequality is true:

$$\omega(G) \leq \overline{\omega}_k(G) \leq \widetilde{\omega}_k(G).$$

It should be noted that $S \subseteq V$ is a k -clique in G , if and only if S is a clique in the power graph G^k . Accordingly, given a graph G and a positive integer k , $\widetilde{\omega}_k(G) = \omega(G^k)$. Hence, the maximum k -clique problem for any k is equivalent to the maximum clique problem on the corresponding power graph. Because of this

close correspondence between k -cliques in original graph and cliques in the power graph, the k -clique model has not been extensively studied.

Two concepts of particular relevance to these problems are distance- k independence [45] and distance- k coloring [107].

Definition 16 Given a graph $G = (V, E)$, $I \subseteq V$ is a distance- k -independent set if for every distinct pair $i, j \in I$, $d_G(i, j) \geq k + 1$.

Clearly, distance-1 independence is equivalent to pairwise nonadjacency, which corresponds to a classical independent set. It should be noted that the definition becomes more restrictive as k increases. A distance- $(k + 1)$ independent set is also distance- k independent, but the converse is not always true. Further, a k -clique or a k -club can intersect a distance- k -independent set in at most one vertex.

Definition 17 Given a graph $G = (V, E)$ and a positive integer k , a proper distance- k coloring is one in which the pairwise distance in G between any two vertices that receive the same color is at least $k + 1$.

A graph is distance- k t -colorable if it has a proper distance- k coloring that uses at most t colors. The minimum number of colors t that admits a proper distance- k t -coloring is the distance- k chromatic number of G denoted by $\chi_k^d(G)$. It should be noted that each color class forms a distance- k -independent set. [Figure 3](#) shows a proper distance-2 5-coloring of a graph.

By definition, in any proper distance- k coloring of a graph G , a k -clique in G can intersect any color class in at most one vertex. Then the following relationship [97] holds:

$$\overline{\omega}_k(G) \leq \widetilde{\omega}_k(G) \leq \chi_k^d(G).$$

Some remarks on the nonhereditary nature of k -clubs. For any integer $k \geq 1$, the k -clique model like the k -plex model is hereditary on induced subgraphs in the sense described in [Sect. 2.1](#). However, this property does not hold for the k -club model for $k \geq 2$. For example, the graph shown in [Fig. 4b](#) is a 2-club which means it is also a 2-clique. Every subset of this set is a 2-clique, but $\{1, 2, 3, 4\}$ for instance is not a 2-club.

Inclusionwise maximality of any polynomially verifiable hereditary property such as k -clique can be tested in polynomial time. For example, to verify maximality of a k -clique S , it suffices to show that there is no single vertex in $V \setminus S$ that could be added to S to form a larger k -clique. However, for the k -club model, nonexistence of a vertex that could increase the size of the k -club by one is a necessary but not sufficient condition for its maximality. For example, in the graph shown in [Fig. 4a](#), set $S_1 = \{1, 2, 3\}$ is a 2-clique and since sets $S_2 = S_1 \cup \{4\}$ and $S_3 = S_1 \cup \{5\}$ are not 2-cliques, it can be concluded that S_1 is a maximal 2-clique. In [Fig. 4b](#), S_1 is not a maximal 2-clique which can be deduced by observing that both S_2 and S_3 are also

Fig. 3 A proper distance-2 5-coloring using colors $\{a, b, c, d, e\}$

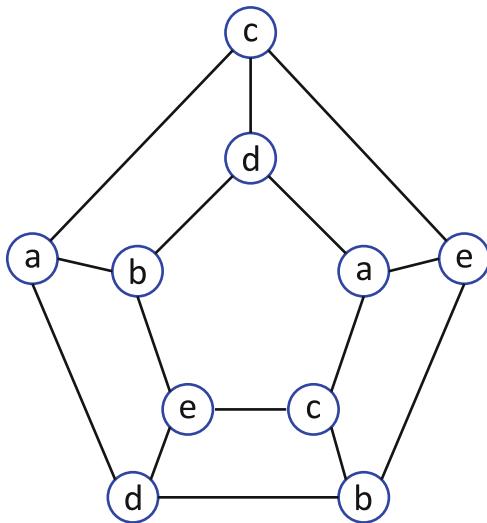
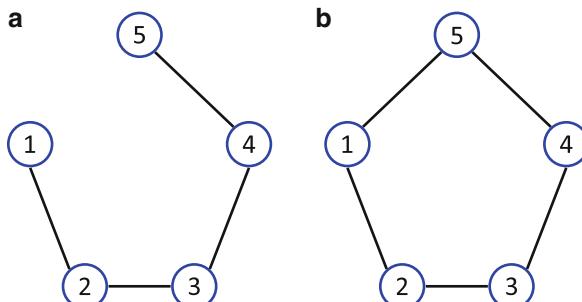


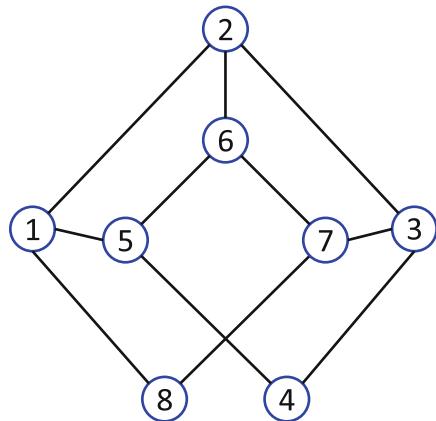
Fig. 4 Inclusionwise maximality testing of 2-cliques and 2-clubs



2-cliques. Set S_1 in the graph shown in Fig. 4b forms a 2-club, while sets S_2 and S_3 are not 2-clubs. But S_1 is not a maximal 2-club since $V = S_1 \cup \{4, 5\}$ is a larger 2-club containing S_1 .

Since the k -club definition is more restrictive than the k -clique, it is possible that a maximal k -club is not a maximal k -clique. For instance, $\{1, 2, 3, 4\}$ is a maximal 2-club in Fig. 2, but not a maximal 2-clique as it is contained in 2-clique $\{1, 2, 3, 4, 5\}$. By contrast, if a maximal k -clique satisfies the diameter requirement, it is also a maximal k -club. However, it is possible in a graph for $k \geq 2$ that no maximal k -clique is a k -club. Figure 5 shows a graph with exactly two maximal 2-cliques $\{1, 2, 3, 4, 5, 6, 7\}$ and $\{1, 2, 3, 5, 6, 7, 8\}$, neither of which is a 2-club. Hence, even enumerating all maximal k -cliques in the graph to select the ones satisfying the diameter- k condition may not identify a maximum k -club. Furthermore, in the extreme case, it fails to identify a single k -club.

Fig. 5 A graph in which every maximal 2-clique is not a 2-club



3.1 Complexity and Approximation

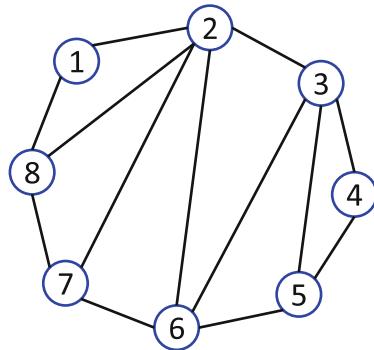
The maximum k -clique and k -club problems are NP-hard for any *fixed* positive integer k [19, 35], and they remain NP-hard even in graphs of fixed diameter with $\text{diam}(G) > k$ [19]. Hence, the maximum k -club problem is known to be NP-hard for every fixed k even on graphs of diameter $k + 1$.

For given positive integers k and l ($k \neq l$), the problem of recognizing whether $\overline{\omega}_k(G) = \overline{\omega}_l(G)$ is also NP-hard. This gap-recognition complexity result was further used to show that for an integer $k \geq 2$ and in a graph G with $\overline{\omega}_k(G) > \Delta(G) + 1$, unless $P=NP$, there does not exist a polynomial time algorithm for finding a k -club of size strictly larger than $\Delta(G) + 1$ [40]. Recalling that $\Delta(G)$ denotes the maximum vertex degree in G , it is clear that a vertex of maximum degree along with all its neighbors forms a k -club in G for any $k \geq 2$.

The maximum k -club problem for fixed $k \geq 2$ was shown to be inapproximable within a factor of $n^{\frac{1}{3}-\epsilon}$ for any $\epsilon > 0$, if $NP \neq ZPP$ [101], which has been strengthened to $n^{\frac{1}{2}-\epsilon}$ recently under the assumption $P \neq NP$ [12]. Approximation algorithms of factor $n^{\frac{1}{2}}$ and $n^{\frac{2}{3}}$ for even and odd k , respectively, have also been proposed recently [12].

In contrast to what is observed for cliques, an encouraging result for k -clubs is that it is fixed-parameter tractable when parameterized by solution size [131, 132]. The k -club problem is one of the few parameterized problems for which nonexistence of a polynomial-size many-to-one kernel and existence of a polynomial-size Turing kernel are known [131, 132]. The k -club problem can be solved on trees and interval graphs in $O(nk^2)$ and $O(n^2)$, respectively, and it is polynomial time solvable on graphs with bounded tree- or cliquewidth [131]. The 2-club problem can be solved on bipartite graphs in $O(n^5)$ [131].

Fig. 6 An asymmetric partitionable cycle with respect to nodes 1 and 4



Recently, it has been shown that testing inclusionwise maximality of k -clubs is NP-hard, which is a direct consequence of their nonhereditary property [97]. However, maximality of a k -club can be tested in polynomial time in graphs that do not contain an *asymmetric partitionable cycle* of a certain size [97]. We provide some details on this result next.

A k -clique which induces a connected subgraph is called a *connected k -clique*. Given a graph $G = (V, E)$, maximality of a connected k -clique D can be checked by testing nodes in $V \setminus D$ that have at least one edge to some node in D , one at a time, to see if they can be added to D to form a larger k -clique. For graphs in which every connected k -clique is also a k -club, maximality of k -clubs can then be checked in polynomial time. By definition, every k -club is a connected k -clique, and in such graphs every connected k -clique is also a k -club, and hence, a maximal connected k -clique is also a maximal k -club.

Definition 18 A graph $G = (V, E)$ is called a partitionable cycle, if it contains a spanning cycle W and a pair of nonadjacent nodes i and j such that every edge in $E \setminus E(W)$ has one endpoint in $A_W(i, j)$ and the other endpoint in $A_W(j, i)$, where $A_W(i, j)$ and $A_W(j, i)$ are the internal nodes on the two paths between i and j in W . A partitionable cycle is said to be asymmetric if $|A_W(i, j)| \neq |A_W(j, i)|$. (See Fig. 6.)

Theorem 12 ([97]) *Given a graph $G = (V, E)$ and fixed integer $k \geq 2$, every connected k -clique is a k -club if G does not contain an asymmetric partitionable cycle on c vertices as an induced subgraph, where $5 \leq c \leq 2k + 1$.*

Corollary 1 ([97]) *In a bipartite graph, every connected 2-clique is a 2-club.*

3.1.1 Formulations

The maximum k -club problem admits the following integer programming formulation [19, 35]:

$$\bar{\omega}_k(G) = \max \sum_{i \in V} x_i$$

Subject to

$$\begin{aligned}
 x_i + x_j &\leq 1 + \sum_{l:P_{ij}^l \in \mathcal{P}_{ij}} y_{ij}^l \quad \forall (i, j) \notin E \\
 x_p &\geq y_{ij}^l \quad \forall p \in V(P_{ij}^l), P_{ij}^l \in \mathcal{P}_{ij}, (i, j) \notin E \\
 x_i &\in \{0, 1\} \quad \forall i \in V \\
 y_{ij}^l &\in \{0, 1\} \quad \forall P_{ij}^l \in \mathcal{P}_{ij}, (i, j) \notin E,
 \end{aligned}$$

where \mathcal{P}_{ij} is an indexed collection of all paths of length at most k between vertices i, j in G and P_{ij}^l is the path with index l between vertices i, j . The formulation ensures that if two vertices are in a k -club, then all the vertices in at least one path between them with length less than or equal to k are also included in the k -club. It should be noted that the size of \mathcal{P}_{ij} could be very large, making this formulation difficult to handle. But the formulation for the maximum 2-club problem stated next is much more compact.

$$\bar{\omega}_2(G) = \max \sum_{i \in V} x_i$$

Subject to

$$\begin{aligned}
 x_i + x_j - \sum_{k \in N(i) \cap N(j)} x_k &\leq 1 \quad \forall (i, j) \notin E \\
 x_i &\in \{0, 1\} \quad \forall i \in V.
 \end{aligned}$$

It should also be noted that while the exponential formulation for the maximum k -club problem discussed here is unsuitable for explicit use with integer programming solvers for large k , a more compact formulation has been recently developed in [145] that permits such use. We present this formulation next.

Recall that $A_G = [a_{i,j}]_{n \times n}$ is the symmetric adjacency matrix of G . We use subscripts of the form $\sigma(u, v)$ where u refers to the position of the term and v refers to the position of the summation. For example, $\sigma(3, 2)$ is the index over which the second summation of the third term runs. The formulation is such that $1 \leq v \leq u \leq k - 1$. Specifically, the term using indices $\sigma(u, .)$ models paths of length $u + 1$. The formulation requires that whenever vertices i and j are included in a k -club, at least one of these terms is 1, thereby requiring a path between i and j of length at most k as well as the requirement that the vertices on that path are included. The compact 2-club formulation presented above can be viewed as a special case of the formulation presented next.

$$\bar{\omega}_k(G) = \max \sum_{i \in V} x_i$$

subject to

$$\begin{aligned}
& a_{i,j} + \sum_{\sigma(1,1)=1}^n a_{i,\sigma(1,1)} a_{\sigma(1,1),j} x_{\sigma(1,1)} \\
& + \sum_{\sigma(2,1)=1}^n \sum_{\sigma(2,2)=1}^n a_{i,\sigma(2,1)} a_{\sigma(2,1),\sigma(2,2)} a_{\sigma(2,2),j} x_{\sigma(2,1)} x_{\sigma(2,2)} \\
& + \sum_{\sigma(3,1)=1}^n \sum_{\sigma(3,2)=1}^n \sum_{\sigma(3,3)=1}^n a_{i,\sigma(3,1)} a_{\sigma(3,1),\sigma(3,2)} a_{\sigma(3,2),\sigma(3,3)} a_{\sigma(3,3),j} x_{\sigma(3,1)} x_{\sigma(3,2)} x_{\sigma(3,3)} \\
& \quad + \cdots + \\
& \sum_{\sigma(k-1,1)=1}^n \cdots \sum_{\sigma(k-1,k-1)=1}^n a_{i,\sigma(k-1,1)} a_{\sigma(k-1,1),\sigma(k-1,2)} \cdots a_{\sigma(k-1,k-1),j} x_{\sigma(k-1,1)} x_{\sigma(k-1,2)} \\
& \quad \cdots x_{\sigma(k-1,k-1)} \\
& \geq x_i + x_j - 1, \quad \forall i, j \in V : i < j \\
& x_i \in \{0, 1\} \quad \forall i \in V.
\end{aligned}$$

To see the correctness of this formulation, consider $x \in \{0, 1\}^n$ which is an incidence vector of a k -club in G . Then for a pair i, j such that $x_i = x_j = 1$ and $i < j$, the corresponding constraint requires the left-hand side of the constraint to be at least 1. Since x corresponds to a k -club, if $(i, j) \in E$ then $a_{i,j} = 1$ and the constraint is satisfied. Otherwise, there exists a path $i = i_0 - i_1 - \cdots - i_u = j$ where $2 \leq u \leq k$ such that $x_{i_s} = 1, s = 0, \dots, u$ and the term corresponding to u has the product corresponding to $\sigma(u, s) = i_s, s = 1, \dots, u-1$ equal to 1. Hence, every constraint is satisfied by incidence vectors of k -clubs.

Conversely, let $x \in \{0, 1\}^n$ be a feasible solution to the formulation. Then, it suffices to show that $S = \{i \in V : x_i = 1\}$ is a k -club. Consider $i, j \in S, i < j$. Since x satisfies the corresponding constraint, at least one term in the left-hand side is 1. Suppose $a_{i,j} = 0$, then for some u between 1 and $k-1$, a product in term u equals 1. Suppose

$$a_{i,\sigma(u,1)} a_{\sigma(u,1),\sigma(u,2)} \times \cdots \times a_{\sigma(u,u-1),\sigma(u,u)} a_{\sigma(u,u),j} x_{\sigma(u,1)} x_{\sigma(u,2)} \cdots x_{\sigma(u,u)} = 1.$$

Hence, $\sigma(u, s) \in S$ for $s = 1, \dots, u$, and $i - \sigma(u, 1) - \sigma(u, 2) - \cdots - \sigma(u, u-1) - \sigma(u, u) - j$ is a path of length $u+1 \leq k$ in G . This shows S is a k -club.

The approach taken here is to formulate the problem as a nonlinear (binary) integer program. Note that the polynomial products of binary variables can be linearized using standard techniques. The authors [145] show that this formulation can be linearized into a linear integer program which uses $O(kn^2)$ variables and constraints by exploiting the properties of a k -club.

The authors of [145] also introduce the notion of an *r-robust k-club* which is a subset of vertices S such that $G[S]$ has at least r internally vertex disjoint, edge disjoint, or simply distinct paths (differing in at least one edge) of length at most k (in $G[S]$) between every pair of vertices. Clearly when $k = 2$, distinct paths are also edge/vertex disjoint, and the three ideas coincide. Note that a 2-club which is 2-connected is not necessarily a 2-robust 2-club although the converse is true. A cycle on five vertices is a 2-connected 2-club, but it is not a 2-robust 2-club since the pair of paths between any two vertices are not both of length at most two.

The formulation for the maximum k -club problem presented above can be extended for the case where r distinct paths of length at most k are required between pairs of vertices. This can be accomplished by replacing the right-hand side with $r(x_i + x_j - 1)$ instead. Specifically, the r -robust 2-club model allows the detection of diameter-two clusters that are robust, in the sense that they preserve the diameter-two bound under vertex or edge deletions (up to $r - 1$).

3.2 Polyhedral Results

Given a graph $G = (V, E)$, the convex hull of the incidence vectors of k -clubs in G is called the k -club polytope of G denoted by $C_k(G)$. Some preliminary results on these polytopes, especially the 2-club polytope, are available from [19, 44]. **Theorem 13** [19] presents the basic results on $C_2(G)$ and the first family of facets developed. It should also be noted that the maximal 2-independent set inequalities are analogous to Padberg's maximal independent inequalities for the clique polytope [121].

Proposition 6 ([17]) (*Distance-k-independent set inequalities*) *Given a graph $G = (V, E)$, let $I \subseteq V$ be a maximal k -independent set. Then the following inequality is valid for $C_k(G)$:*

$$\sum_{i \in I} x_i \leq 1.$$

Theorem 13 ([19]) *Consider the 2-club polytope $C_2(G)$ of a graph $G = (V, E)$.*

1. $\dim(C_2(G)) = n$.
2. $x_i \geq 0$ induces a facet of $C_2(G)$ for every $i \in V$.
3. For $i \in V$, $x_i \leq 1$ induces a facet of $C_2(G)$ if and only if $d_G(i, j) \leq 2 \forall j \in V$.
4. The inequality $\sum_{i \in I} x_i \leq 1$ induces a facet of $C_2(G)$ if and only if I is a maximal distance-2-independent set in G .

Proof The full dimension of the 2-club polytope follows from the fact that $\emptyset, \{i\}$ are 2-clubs for each $i \in V$. The nonnegativity facet also follows from the same observation. The unit upper-bound facet is a special case of a 2-independent set facet that is proved next.

Suppose I is a maximal distance-2-independent set. We establish the maximality of the face $F_I = \left\{ x \in C_2(G) : \sum_{i \in I} x_i = 1 \right\}$, thereby showing it is a facet. Suppose

there exists a valid inequality $\alpha^T x \leq \beta$ such that, $F = \{x \in C_2(G) : \alpha^T x = \beta\} \supseteq F_I$. Since $e_i \in F_I \subseteq F \forall i \in I$, we have $\alpha_i = \beta \forall i \in I$. Now for every $j \in V \setminus I$, there exists a vertex $i \in I$ such that at least one of the following two conditions are satisfied:

1. $(i, j) \in E$ and so $e_i + e_j \in F_I \subseteq F$.
2. $N(i) \cap N(j) \neq \emptyset$, that is, they have a common neighbor $k \in V \setminus I$ in which case $e_i + e_j + e_k, e_i + e_k \in F_I \subseteq F$.

Now for every $j \in V \setminus I$, in the first case we obtain, $\alpha_i + \alpha_j = \beta \Rightarrow \alpha_j = 0$ and in the second case we obtain $\alpha_i + \alpha_k + \alpha_j = \beta$ and $\alpha_i + \alpha_k = \beta \Rightarrow \alpha_j = \alpha_k = 0$. Thus, $F_I = F$ is a facet.

Suppose the inequality induces a facet of $C_2(G)$. If I is not a 2-independent set, then there exist $i, j \in I$ such that either $(i, j) \in E$ or if I is independent, there exists $k \in N(i) \cap N(j)$. In the first case, $e_i + e_j$ which is a feasible point violates the inequality. In the second case, $e_i + e_j + e_k$ which is feasible violates the inequality. Either situation contradicts the validity of this inequality. Hence, I must be a 2-independent set. If I is not maximal, there exists $I \cup \{u\}$ which is 2-independent and $x_u + \sum_{i \in I} x_i \leq 1$ is valid and dominates a facet-defining inequality, which is a contradiction. Hence, I is maximal. \square

This line of research has been recently furthered in [44]. **Theorem 14** provides necessary and sufficient conditions under which the common neighborhood constraint in the formulation induces a facet. **Theorem 15** introduces another facet-defining inequality for $C_2(G)$.

Theorem 14 ([44]) *Given a graph $G = (V, E)$, let $a, b \in V$, and $I \subseteq V \setminus \{a, b\}$ be such that $I \cup \{a\}$ and $I \cup \{b\}$ are distance-2-independent sets in G and $\text{dist}_G(a, b) = 2$. The following inequality is valid for $C_2(G)$ and induces a facet if and only if I is a maximal set:*

$$\sum_{i \in I \cup \{a, b\}} x_i - \sum_{j \in N(a) \cap N(b)} x_j \leq 1.$$

Theorem 15 ([44]) *Given a graph $G = (V, E)$, let $a, b, c \in V$, and $I \subseteq V \setminus \{a, b, c\}$ be such that set $\{a, b, c\}$ is independent and sets $I \cup \{a\}$, $I \cup \{b\}$ and $I \cup \{c\}$ are distance-2-independent in G . The following inequality is valid for $C_2(G)$ and induces a facet if and only if I is a maximal set:*

$$\sum_{i \in I \cup \{a, b, c\}} x_i - \sum_{j \in V} \alpha_j x_j \leq 1,$$

where

$$\alpha_j = \begin{cases} 2, & \text{if } j \in N(a) \cap N(b) \cap N(c), \\ 1, & \text{if } j \in [N(a) \cap N(b)] \cup [N(a) \cap N(c)] \cup [N(b) \cap N(c)] \setminus [N(a) \cap N(b) \cap N(c)], \\ 0, & \text{Otherwise.} \end{cases}$$

Recently in [98], a new family of facets called the *independent distance-2-dominating set inequalities* for the 2-club polytope have been introduced that include all known facets other than nonnegativity constraints as special cases. The k -club polytope for $k \geq 3$ has fewer associated results, and there is ongoing research on polyhedral approaches to these cases. Alternate formulations and valid inequalities for the maximum 3-club problem were introduced along with computational experiments in [4, 5].

3.3 Algorithms

Combinatorial exact algorithms appear to be more prevalent presently for solving the maximum k -club problem for arbitrary k . The results on the 2-club polytope have also led to preliminary branch-and-cut approaches incorporating maximal distance-2-independent set facets discussed in [17, 19]. A sequential cutting-plane method is also proposed in [44]. Heuristic algorithms for finding large k -cliques or k -clubs in a given graph have been proposed in [34, 44, 66] that can be used to produce lower bounds in some exact approaches like the combinatorial branch-and-bound algorithms.

Due to the intractability of testing inclusionwise maximality of k -clubs and its nonhereditary nature, well-known exact combinatorial algorithms for cliques like Carraghan-Pardalos and Östergård's algorithms [43, 120] are unlikely to be applicable to k -clubs although they can naturally be applied to the power graph to find k -cliques. The existing branch-and-bound algorithms from [35, 97] are classical partitioning approaches based on variable dichotomy but differ in the approach taken to obtain upper bounds.

The approach taken in [35] is to obtain upper bounds at each node of the search tree via the k -clique number. In [97], the authors employ the distance- k coloring problem to obtain upper bounds in the search tree. From the inequality presented in Sect. 3, the former has the potential to produce tighter bounds than the latter approach. However, in the first approach, the maximum k -clique problem (which is also NP-hard) must be solved to optimality to produce a valid bound. Whereas, a heuristic coloring algorithm is sufficient with the second approach to produce potentially weaker upper bounds. Details on these algorithms and available benchmark instances can be found in [35, 97]. A fixed-parameter tractable algorithm is presented in [131] to find a k -club of size c if one exists in G in $O((c - 2)^c \cdot c! \cdot c^3 n + nm)$ time.

4 Density-Based Models: Densest t -Subgraph and γ -Quasi-clique

Two types of optimization problems can be developed to identify dense clusters in graphs. In the first type, given a graph $G = (V, E)$ and a fixed positive integer $t \leq n$, the problem is to find a subgraph on t vertices of maximum edge density.

Since the order of the subgraph is fixed (t), this is equivalent to finding a t -vertex subgraph with maximum number of edges or average degree. In the second type, given a graph $G = (V, E)$ and a fixed real number $\gamma \in [0, 1]$, the problem is to find a subgraph of maximum order with edge density at least γ . The first type is analogous to finding a k -plex, k -clique, or a k -club of fixed order t with minimum k . In other words, the most cohesive or tightly knit cluster of a given order t where cluster cohesion is quantified by the parameter k . The second type generalizes the maximum clique problem which is a special case when $\gamma = 1$ and hence is a clique relaxation in a manner similar to the foregoing models in this chapter.

4.1 The Densest t -Subgraph Problem

The problem of the first type is referred to in the literature by various names such as *the t -cluster problem* [57], *the heaviest (unweighted) t -subgraph problem* [92], *the densest t -subgraph problem* [68], and *the maximum edge subgraph problem* [10]. The edge-weighted version of this problem has also been studied in [10, 11, 78, 129]. A different but closely related *densest subgraph problem* is to choose a nonempty subgraph $G' = (V', E')$ of an edge-weighted graph G such that the ratio $w(E')/|V'|$ is maximized. The densest subgraph problem can be solved in polynomial time using maximum-flow algorithms [73, 92] (see also [46]).

4.1.1 Complexity and Approximation

The densest t -subgraph problem is NP-hard even on bipartite graphs, perfect graphs, chordal graphs [57], and planar graphs [89]. No PTAS exists under the assumption that NP does not have subexponential time algorithms [90]. It is shown to have a PTAS on dense instances ($|E| = \Omega(n^2)$) if $t = \Omega(n)$ in [8]. Other types of approximation algorithms for the problem are also available in the literature. A $O(n^{0.3885})$ approximation algorithm is available in [92], and an improved approximation guarantee of $O(n^{\frac{1}{3}-\epsilon})$ for some $\epsilon > 0$ is presented in [68, 69]. A more recent algorithm [25] provides an approximation guarantee of $O(n^{\frac{1}{4}+\epsilon})$ for any $\epsilon > 0$.

The approximation guarantee of $O(n/t)$ from the greedy algorithm where the vertex of minimum degree is recursively deleted until a set of size t remains is also established in [10]. A 3-approximate algorithm is available for the problem on chordal graphs [93] and a 1.5-approximate algorithm is known for proper interval graphs and bipartite permutation graphs (the complexity on these graph classes is unknown) [69, 138].

The complexity threshold of this problem was studied in [11], designated as *the t - $f(t)$ -dense subgraph problem* which asks if G contains a t -vertex subgraph with at least $f(t)$ edges. This is the decision version of the maximum clique problem when $f(t) = \binom{t}{2}$ which is known to be NP-complete. It is established

that the problem remains NP-complete when $f(t) = \Theta(t^{1+\epsilon})$ and when $f(t) = mt^2/n^2(1 + O(n^{\epsilon-1}))$ for any constant ϵ such that $0 < \epsilon < 1$.

Semidefinite programming-based approximation algorithms are also available for this problem in [69, 138]. A deterministic approximation algorithm for the problem is also presented in [130] that applies rounding techniques to the optimal solution of a linear programming relaxation of the following binary quadratic program:

$$\max \sum_{(ij) \in E} x_i x_j$$

Subject to

$$\begin{aligned} \sum_{i \in V} x_i &= t \\ x_i &\in \{0, 1\} \quad \forall i \in V. \end{aligned}$$

4.1.2 Polyhedral Results and Algorithms

Different integer programming formulations are presented for this problem in [26], obtained by different linearizations and tightening of the aforementioned binary quadratic program. Polyhedral study of a particular linearization of this formulation undertaken in [32] is presented next. Summarized are some families of valid inequalities, sufficient conditions under which they are facet inducing, and the associated separation problems investigated in [32].

$$\max \sum_{(ij) \in E} z_{ij}$$

Subject to

$$\begin{aligned} \sum_{i \in V} x_i &= t \\ z_{ij} &\leq x_i \quad \forall (ij) \in E \\ z_{ij} &\leq x_j \quad \forall (ij) \in E \\ x_i &\in \{0, 1\} \quad \forall i \in V \\ z_{ij} &\in \{0, 1\} \quad \forall (ij) \in E. \end{aligned}$$

Theorem 16 ([32]) Given $G = (V, E)$ and a positive integer $t \leq n$, let $D_t(G)$ denote the convex hull of feasible solutions to the above integer program. Then, $\dim(D_t(G)) = n + m - 1$.

Theorem 17 ([32]) (Generalized neighborhood inequalities) Given $G = (V, E)$, $i \in V$, and a set $A^i \subseteq V \setminus \{i\}$ with $|A^i| \leq t - 2$, the following inequality is valid for $D_t(G)$:

$$\sum_{j \in A^i} x_j + \sum_{j \in N(i) \setminus A^i} z_{ij} \leq |A^i| + (t - |A^i| - 1)x_i.$$

Furthermore, if $|N(i) \cap A^i| \geq t$ and $t \leq n - 2$, the generalized neighborhood inequality induces a facet of $D_t(G)$.

In [32], the authors also show that the generalized neighborhood inequalities can be separated in polynomial time. Another family of inequalities, similar to the odd-set inequalities for the matching polyhedron [67], were also developed and the associated separation problem was shown to be polynomial time solvable in [32].

Theorem 18 ([32]) (Matching inequalities) Given $G = (V, E)$, $T \subseteq V$ and a maximal matching M of $G[V \setminus T]$, the following inequality is valid for $D_t(G)$, if $|T| + t$ is odd:

$$\sum_{j \in T} x_j + \sum_{(ij) \in M} z_{ij} \leq \frac{|T| + t - 1}{2}.$$

In addition, if $t - 2|M| + 1 \leq |T| \leq t - 1$ and $t \leq n - 3$, then the matching inequality induces a facet of $D_t(G)$.

Similar results on validity and facet-inducing conditions are developed in [32] for more families such as forest inequalities, tree inequalities, and disjoint matching inequalities. Separating these inequalities is shown to be NP-hard. Computational experiments to study the effectiveness of these inequalities in a branch-and-cut framework are also conducted in [32]. This work appears to be the only polyhedral approach to this problem in the literature presently. In addition to the approximation algorithms discussed in Sect. 4.1.1, a variable neighborhood search [111] metaheuristic is also available for the problem in [36].

4.2 The Maximum γ -Quasi-clique Problem

The second type of problem designed to identify dense clusters in a graph is called the maximum γ -quasi-clique problem studied in [1, 2, 124].

Definition 19 Given a graph $G = (V, E)$ and a real number $0 \leq \gamma \leq 1$, $S \subseteq V$ is a γ -quasi-clique¹ if the edge density of $G[S]$ is at least γ .

The maximum γ -quasi-clique problem is to find a maximum cardinality γ -quasi-clique in G , and this cardinality is the γ -quasi-clique number of G , denoted by $\omega_\gamma^q(G)$. This problem includes the maximum clique problem as a special case when $\gamma = 1$ and is shown to be NP-hard for fixed $\gamma \in (0, 1)$ in [124]. This result is also extended to show a variant of the model studied in [39] is also NP-hard.

¹It should be noted that a subset $S \subseteq V$ such that $\delta(G[S]) \geq \gamma(|S| - 1)$ has also been called a γ -quasi-clique in some literature (see for instance [39, 87, 154]).

Similar to k -clubs, the γ -quasi-cliques are nonhereditary. But they are said to be *quasi-hereditary*, a notion introduced in [124]. A graph property Π is said to be *quasi-hereditary*; if G has property Π , then there exists a vertex $v \in V$ such that $G - v$ also satisfies property Π . If G has edge density at least γ , deleting a vertex with degree no larger than the average degree (specifically a vertex of minimum degree) will result in a graph with edge density at least γ [124].

Summarized next are bounds on $\omega_\gamma^q(G)$ extending results known for $\omega(G)$ [6, 115] and its relationship to the clique number developed in [124].

Proposition 7 ([124]) *Given a graph $G = (V, E)$, and $0 < \gamma \leq 1$*

$$\omega_\gamma^q(G) \leq \frac{\gamma + \sqrt{\gamma^2 + 8m\gamma}}{2\gamma}.$$

Moreover if G is connected,

$$\omega_\gamma^q(G) \leq \frac{\gamma + 2 + \sqrt{(\gamma + 2)^2 + 8(m - n)\gamma}}{2\gamma}.$$

If $\gamma > 1 - \frac{1}{\omega(G)}$,

$$\omega_\gamma^q(G) \leq \frac{\gamma\omega(G)}{1 - \omega(G)(1 - \gamma)}.$$

The maximum γ -quasi-clique problem can be formulated as the following quadratically constrained binary integer program [124]. Recall that $A_G = [a_{i,j}]_{n \times n}$ is the symmetric adjacency matrix of G . Denote by M , a zero-diagonal $n \times n$ matrix in which every entry except the diagonal entries are ones.

$$\omega_\gamma^q(G) = \max \sum_{i \in V} x_i$$

Subject to

$$\begin{aligned} x^T A_G x &\geq \gamma x^T M x \\ x_i &\in \{0, 1\} \forall i \in V. \end{aligned}$$

Some preliminary computational experiments are performed in [124] by linearizing this formulation into two different linear mixed-integer programs, solved using a commercial solver. These linearizations are presented next.

The following formulation is obtained by a simple linearization that introduces a new variable (and associated constraints) for each binary quadratic term. Note that this formulation has $O(n^2)$ variables and constraints.

$$\omega_{\gamma}^q(G) = \max \sum_{i=1}^n x_i$$

Subject to

$$\begin{aligned} \sum_{i=1}^n \sum_{j=i+1}^n (\gamma - a_{ij}) z_{ij} &\leq 0 \\ z_{ij} &\leq x_i, \forall j > i = 1, \dots, n \\ z_{ij} &\leq x_j, \forall j > i = 1, \dots, n \\ z_{ij} &\geq x_i + x_j - 1, \forall j > i = 1, \dots, n \\ z_{ij} &\geq 0, \forall j > i = 1, \dots, n \\ x_i &\in \{0, 1\}, \forall j > i = 1, \dots, n. \end{aligned}$$

The following formulation is also a binary mixed-integer program based on an alternate linearization scheme which results in a more compact formulation with $O(n)$ variables and constraints.

$$\omega_{\gamma}^q(G) = \max \sum_{i=1}^n x_i$$

Subject to

$$\begin{aligned} \sum_{i=1}^n y_i &\geq 0 \\ l_i x_i &\leq y_i \leq u_i x_i, \forall i = 1, \dots, n \\ \gamma x_i + \sum_{j=1}^n (a_{ij} - \gamma) x_j - u_i(1 - x_i) &\leq y_i, \forall i = 1, \dots, n \\ \gamma x_i + \sum_{j=1}^n (a_{ij} - \gamma) x_j - l_i(1 - x_i) &\geq y_i, \forall i = 1, \dots, n \\ x_i &\in \{0, 1\}, \forall i = 1, \dots, n \\ y_i &\in \mathbb{R}, \forall i = 1, \dots, n \end{aligned}$$

$$\text{where } u_i = (1 - \gamma) \sum_{j=1}^n a_{ij} \text{ and } l_i = - \left(n - 1 - \sum_{j=1}^n a_{ij} \right) \gamma.$$

The asymptotic behavior of the maximum γ -quasi-clique size in uniform random graphs has been studied in [146]. A greedy randomized adaptive search procedure (GRASP) was introduced in [1] to find large γ -quasi-cliques in a graph. In [2], the

authors defined a *potential function* by introducing the notion of the potential of a vertex or an edge set with respect to a given γ -quasi-clique. This potential function was used to develop another GRASP to find maximal γ -quasi-cliques in a given graph. A combinatorial branch-and-bound algorithm for the maximum γ -quasi-clique problem was recently developed in [99]. Clearly, theoretical developments on this model are presently limited even though numerous heuristic approaches have been employed to find quasi-cliques in various applications.

5 Selected Applications

Recent advances in high-throughput data collection in different fields such as Internet research, social network analysis, text analytics, bioinformatics, computational finance, and telecommunication among others have led to an increased demand for effective models and tools for data mining. Data mining is the process of summarizing, visualizing, and processing large data sets in order to extract useful knowledge from the data by using advanced mathematical techniques [139].

In a variety of data mining applications, the elements of a system and the relationships among them are modeled as a graph, wherein graph algorithms and optimization techniques are used to uncover the specific patterns in such data sets [55, 148]. Graph-based data mining is particularly advantageous for studying unstructured or partially structured data compared to methods that almost exclusively deal with numerical data. Cliques and their relaxations have been used to identify “tightly knit” clusters which often provide valuable insights in a variety of different application settings.

An *Internet graph* has vertices representing IP addresses, and edges in such graphs are determined based on information from routing protocols or using traceroute probes [37]. In web mining and Internet research, the web can be modeled by a graph in which each webpage is represented by a vertex and two vertices are adjacent if the corresponding webpages are linked together [140]. The 2-club model has been used for topical clustering of hyperlinked documents [110] and web sites [140] to expedite search procedures. A k -plex-based approach to mine databases is proposed in [154].

In *call graphs*, vertices represent telephone numbers and an edge represents a call placed from one vertex to another in a specified time interval [1]. A call graph representing telecommunications traffic data over one 20-day period with 290 million vertices and 4 billion edges was mined using quasi-cliques in [1].

Stock-market graphs have vertices representing stocks, and two stocks are connected by an edge if they are positively correlated over some threshold value based on calculations over a period of time in history [28–30]. In [29, 30], the authors studied the US stock market and used clique relaxation models in order to classify this market and find robust diversified portfolios.

Social networks are graph models representing sociological information such as friendship or acquaintance among people. Vertices usually represent people, and an

edge indicates a “tie” between two people. A tie could mean that they know each other, they visited the same place, or any other sociological connection. *Scientific collaboration networks* with vertices representing authors (in a particular field or with publications in a particular journal) and edges indicating coauthorship fall under this category [42, 77, 82, 85, 135].

In social network analysis, finding cohesive subgroups in contact networks, friendship networks, and collaboration networks are traditional applications that employ cliques and clique relaxations. A special case is criminal network analysis [9, 47, 48, 60, 136], used to detect organized criminal activities such as money laundering [119] and terrorism [108, 109]. Detecting cohesive subgroups in contact networks has also been used for disease contagiousness studies and what-if analysis by healthcare providers [76, 135]. Mining friendship networks and finding dense clusters can provide useful information for advertising and marketing purposes [82, 85, 152].

Biological networks such as *protein interaction networks* and *gene co-expression networks* are used to model biological information. A protein interaction network is represented by a graph with the proteins of an organism as vertices, and an edge exists between two vertices if the proteins are known to interact based on two-hybrid assays and other biological experiments [72, 86, 137]. In gene co-expression networks, vertices represent genes, and an edge exists between two vertices if the corresponding genes are co-expressed with correlation higher than a specified threshold in microarray experiments [127].

Detecting clusters in protein interaction networks helps identify *protein complexes* and *functional modules* that influence cellular functions [137]. Clustering of gene co-expression networks has been used to detect network motifs using cliques, quasi-cliques, and k -plexes in [51, 87, 127, 147]. Protein interaction networks of organisms like *Helicobacter pylori* and *Saccharomyces cerevisiae* have been mined using k -clubs [19, 123], quasi-cliques [14, 137], and k -plexes [126]. Additionally, finding cliques in graph models of electroencephalogram time series data has been used to help identify epileptic seizures [128].

6 Conclusion

Graph theoretic clique relaxations provide practical alternatives to cliques for modeling clusters in graph-based data mining applications. The variety of models available make this an attractive approach to consider in many such applications where the best clustering model can only be identified after it has proved itself to be useful in the application context. In massive data sets, the clique relaxations discussed in this chapter allow a level of tolerance to edges missed due to data errors. At the same time, these models provide specific structural guarantees controlled by the user, so that the significance of the cluster under edges included in error is also satisfactory. Hence, clique relaxations, for appropriate values of their respective parameters, provide a hierarchy of models that enable the user to derive valuable insights by detecting large clusters in massive data sets.

The results presented in this chapter are focused on polyhedral and combinatorial approaches, as well as complexity and approximation results that indicate the limits of solvability. More is needed with regard to results of this nature, as well as practical metaheuristic approaches designed for massive data sets. Studying these models under uncertainty with respect to the edges in the graph is also important. With ongoing active research in this area driven by practical needs and theoretical challenges, interesting developments can be expected in the future.

Acknowledgements This work was partially supported by the US Department of Energy Grant DE-SC0002051.

Cross-References

- ▶ [Combinatorial Optimization in Data Mining](#)
- ▶ [Combinatorial Optimization Techniques for Network-Based Data Mining](#)
- ▶ [Optimization Problems in Online Social Networks](#)

Recommended Reading

1. J. Abello, P.M. Pardalos, M.G.C. Resende, On maximum clique problems in very large graphs, in *External Memory Algorithms and Visualization*, ed. by J. Abello, J. Vitter, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 50 (American Mathematical Society, Providence, 1999), pp. 119–130
2. J. Abello, M.G.C. Resende, S. Sudarsky, Massive quasi-clique detection, in *LATIN 2002: Proceedings of the 5th Latin American Symposium on Theoretical Informatics*, ed. by S. Rajsbaum (Springer, London, 2002), pp. 598–612
3. R.D. Alba, A graph-theoretic definition of a sociometric clique. *J. Math. Sociol.* **3**(1), 113–126 (1973)
4. M.T. Almeida, F.D. Carvalho, The k -club problem: new results for $k = 3$. Technical report CIO working paper 3/2008, CIO-Centro de Investigação Operacional, 2008
5. M.T. Almeida, F.D. Carvalho, Integer models and upper bounds for the 3-club problem. *Networks* **60**(3), 155–166 (2012)
6. A.T. Amin, S.L. Hakimi, Upper bounds on the order of a clique of a graph. *SIAM J. Appl. Math.* **22**, 569–573 (1972)
7. D. Applegate, D.S. Johnson, dfmax.c [c program, second dimacs implementation challenge], <ftp://dimacs.rutgers.edu/pub/challenge/graph/solvers/>
8. S. Arora, D. Karger, M. Karpinski, Polynomial time approximation schemes for dense instances of -hard problems. *J. Comput. Syst. Sci.* **58**(1), 193–210 (1999)
9. J. Arquilla, D. Ronfeldt, What next for networks and netwars? in *Networks and Netwars: The Future of Terror, Crime, and Militancy*, ed. by J. Arquilla, D. Ronfeldt (RAND Corporation, Santa Monica, 2001), pp. 311–361
10. Y. Asahiro, K. Iwama, H. Tamaki, T. Tokuyama, Greedily finding a dense subgraph. *J. Algorithm* **34**, 203–221 (2000)
11. Y. Asahiro, R. Hassin, K. Iwama, Complexity of finding dense subgraphs. *Discret. Appl. Math.* **121**(1–3), 15–26 (2002)

12. Y. Asahiro, E. Miyano, K. Samizo, Approximating maximum diameter-bounded subgraphs, in *LATIN 2010: Theoretical Informatics*, ed. by A. López-Ortiz. Lecture Notes in Computer Science, vol. 6034 (Springer, Berlin/Heidelberg, 2010), pp. 615–626
13. L. Babel, Finding maximum cliques in arbitrary and in special graphs. *Computing* **46**(4), 321–341 (1991)
14. G.D. Bader, C.W.V. Hogue, An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinf.* **4**(2), (2003)
15. E. Balas, J. Xue, Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring. *Algorithmica* **15**, 397–412 (1996)
16. E. Balas, C. Yu, Finding a maximum clique in an arbitrary graph. *SIAM J. Comput.* **15**, 1054–1068 (1986)
17. B. Balasundaram, Graph theoretic generalizations of clique: optimization and extensions. Ph.D. thesis, Texas A&M University, College Station, TX, USA, 2007
18. B. Balasundaram, S. Butenko, Network clustering, in *Analysis of Biological Networks*, ed. by B.H. Junker, F. Schreiber (Wiley, New York, 2008), pp. 113–138
19. B. Balasundaram, S. Butenko, S. Trukhanov, Novel approaches for analyzing biological networks. *J. Comb. Optim.* **10**(1), 23–39 (2005)
20. B. Balasundaram, S. Butenko, I.V. Hicks, Clique relaxations in social network analysis: the maximum k -plex problem. *Oper. Res.* **59**(1), 133–142 (2011)
21. B. Balasundaram, S.S. Chandramouli, S. Trukhanov, Approximation algorithms for finding and partitioning unit-disk graphs into co- k -plexes. *Optim. Lett.* **4**(3), 311–320 (2010)
22. A.-L. Barabási, R. Albert, Emergence of scaling in random networks. *Science* **286**(5439), 509–512 (1999)
23. F. Barahona, A.R. Mahjoub, Compositions of graphs and polyhedra II: stable sets. *SIAM J. Discret. Math.* **7**, 359–371 (1994)
24. F. Barahona, A.R. Mahjoub, Compositions of graphs and polyhedra III: graphs with no W_4 minor. *SIAM J. Discret. Math.* **7**, 372–389 (1994)
25. A. Bhaskara, M. Charikar, E. Chlamtac, U. Feige, A. Vijayaraghavan, Detecting high log-densities – an $o(n^{1/4})$ approximation for densest k -subgraph. *CoRR*, abs/1001.2891, 2010
26. A. Billionnet, Different formulations for solving the heaviest k -subgraph problem. *Inf. Syst. Oper. Res.* **43**(3), 171–186 (2005)
27. V. Boginski, Network-based data mining: operations research techniques and applications, in *Encyclopedia of Operations Research and Management Science* (Wiley, New York, 2011, to appear)
28. V. Boginski, S. Butenko, P.M. Pardalos, On structural properties of the market graph, in *Innovation in Financial and Economic Networks*, ed. by A. Nagurney (Edward Elgar, London, 2003)
29. V. Boginski, S. Butenko, P. Pardalos, Statistical analysis of financial networks. *Comput. Stat. Data Anal.* **48**, 431–443 (2005)
30. V. Boginski, S. Butenko, P. Pardalos, Mining market data: a network approach. *Comput. Oper. Res.* **33**, 3171–3184 (2006)
31. I.M. Bomze, M. Budinich, P.M. Pardalos, M. Pelillo, The maximum clique problem, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos (Kluwer Academic, Dordrecht 1999), pp. 1–74
32. F. Bonomo, J. Maranco, D. Saban, N. Stier-Moses, A polyhedral study of the maximum edge subgraph problem. *Discret. Appl. Math.* (2011). doi:10.1016/j.dam.2011.10.011
33. R. Bornsdörfer, *Aspects of Set Packing, Partitioning, and Covering* (Shaker Verlag, Aachen, 1998). Ph.D. thesis, Technische Universität Berlin
34. J.-M. Bourjolly, G. Laporte, G. Pesant, Heuristics for finding k -clubs in an undirected graph. *Comput. Oper. Res.* **27**, 559–569 (2000)
35. J.-M. Bourjolly, G. Laporte, G. Pesant, An exact algorithm for the maximum k -club problem in an undirected graph. *Eur. J. Oper. Res.* **138**, 21–28 (2002)
36. J. Brimberg, N. Mladenović, D. Urosević, E. Ngai, Variable neighborhood search for the heaviest k -subgraph. *Comput. Oper. Res.* **36**(11), 2885–2891 (2009)

37. A. Broido, K.C. Claffy, Internet topology: connectivity of ip graphs, in *Scalability and Traffic Control in IP Networks*, ed. by S. Fahmy, K. Park (SPIE, Bellingham, 2001), pp. 172–187
38. C. Bron, J. Kerbosch, Algorithm 457: finding all cliques on an undirected graph. *Commun. ACM* **16**, 575–577 (1973)
39. M. Brunato, H. Hoos, R. Battiti, On effectively finding maximal quasi-cliques in graphs, in *Learning and Intelligent Optimization*, ed. by V. Maniezzo, R. Battiti, J.-P. Watson. Lecture Notes in Computer Science, vol. 5313 (Springer, Berlin/Heidelberg, 2008), pp. 41–55
40. S. Butenko, O. Prokopyev, On k -club and k -clique numbers in graphs. Technical report, Texas A&M University, 2007
41. S. Butenko, W. Wilhelm, Clique-detection models in computational biochemistry and genomics. *Eur. J. Oper. Res.* **173**, 1–17 (2006)
42. L.M. Camarinha-matos, H. Afsarmanesh, Collaborative networks: a new scientific discipline. *J. Intell. Manuf.* **16**, 439–452 (2005)
43. R. Carraghan, P. Pardalos, An exact algorithm for the maximum clique problem. *Oper. Res. Lett.* **9**, 375–382 (1990)
44. F.D. Carvalho, M.T. Almeida, Upper bounds and heuristics for the 2-club problem. *Eur. J. Oper. Res.* **210**(3), 489–494 (2011)
45. G.J. Chang, G.L. Nemhauser, The k -domination and k -stability problems on sun-free chordal graphs. *SIAM J. Algebr. Discret. Method* **5**, 332–345 (1984)
46. M. Charikar, Greedy approximation algorithms for finding dense components in a graph, in *Approximation Algorithms for Combinatorial Optimization*, ed. by K. Jansens, S. Khuller. Lecture Notes in Computer Science, vol. 1913 (Springer, Berlin/Heidelberg, 2000), pp. 139–152
47. H. Chen, D. Zeng, H. Atabakhsh, W. Wyzga, J. Schroeder, COPLINK: managing law enforcement data and knowledge. *Commun. ACM* **46**(1), 28–34 (2003)
48. H. Chen, W. Chung, J.J. Xu, G. Wang, Y. Qin, M. Chau, Crime data mining: a general framework and some examples. *Computer* **37**(4), 50–56 (2004)
49. E. Cheng, W.H. Cunningham, Wheel inequalities for stable set polytopes. *Math. Program.* **77**, 389–421 (1997)
50. E. Cheng, S. de Vries, On the facet-inducing antiweb-wheel inequalities for stable set polytopes. *SIAM J. Discret. Math.* **15**(4), 470–487 (2002)
51. E.J. Chesler, M.A. Langston, Combinatorial genetic regulatory network analysis tools for high throughput transcriptomic data. Technical report ut-cs-06-575, CS Technical reports, University of Tennessee, Knoxville, 2006
52. F. Chung, L. Lu, *Complex Graphs and Networks*. CBMS Lecture Series (American Mathematical Society, Providence, 2006)
53. V. Chvátal, On certain polytopes associated with graphs. *J. Comb. Theory (B)* **18**, 138–154 (1975)
54. B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs. *Discret. Math.* **86**, 165–177 (1990)
55. D.J. Cook, L.B. Holder, Graph-based data mining. *IEEE Intell. Syst.* **15**(2), 32–41 (2000)
56. W. Cook, W. Cunningham, W. Pulleyblank, A. Schrijver, *Combinatorial Optimization* (Wiley, New York, 1998)
57. D. Corneil, Y. Perl, Clustering and domination in perfect graphs. *Discret. Appl. Math.* **9**, 27–39 (1984)
58. G. Cornuéjols, *Combinatorial Optimization: Packing and Covering*. CBMS-NSF Regional Conference Series in Applied Mathematics (SIAM, Philadelphia, 2001)
59. L. Cowen, R. Cowen, D. Woodall, Defective colorings of graphs in surfaces: partitions into subgraphs of bounded valence. *J. Graph Theory* **10**, 187–195 (1986)
60. R.H. Davis, Social network analysis: an aid in conspiracy investigations. *FBI Law Enforc. Bull.* **50**(12), 11–19 (1981)
61. A. Dessmark, K. Jansen, A. Lingas, The maximum k -dependent and f -dependent set problem, in *Proceedings of the 4th International Symposium on Algorithms and Computation:ISAAC '93*, ed. by K.W. Ng, P. Raghavan, N.V. Balasubramanian, F.Y.L. Chin. Lecture Notes in Computer Science, vol. 762 (Springer, Berlin, 1993), pp. 88–97

62. R. Diestel, *Graph Theory* (Springer, Berlin, 1997)
63. DIMACS, Cliques, coloring, and satisfiability: second DIMACS implementation challenge (1995), Online: <http://dimacs.rutgers.edu/Challenges/>. Accessed Mar 2007
64. H. Djidjev, O. Garrido, C. Levcopoulos, A. Lingas, On the maximum k -dependent set problem. Technical report LU-CS-TR:92-91, Department of Computer Science, Lund University, Sweden, 1992
65. R.G. Downey, M.R. Fellows, Fixed-parameter tractability and completeness II: on completeness for W[1]. *Theor. Comput. Sci.* **141**(1–2), 109–131 (1995)
66. J. Edachery, A. Sen, F.J. Brandenburg, in *Graph Clustering Using Distance-k Cliques*. Lecture Notes in Computer Science, vol. 1731 (Springer, Berlin/New York, 1999), pp. 98–106
67. J. Edmonds, Maximum matching and a polyhedron with 0,1-vertices. *J. Res. Natl. Bur. Stand. B* **69B**, 125–130 (1965)
68. U. Feige, G. Kortsarz, D. Peleg, The dense k -subgraph problem. *Algorithmica* **29**, 410–421 (2001)
69. U. Feige, M. Seltser, On the densest k -subgraph problem. Technical report CS97-16, Weizmann Institute, 1997
70. M.R. Fellows, J. Guo, H. Moser, R. Niedermeier, A generalization of nemhauser and Trotter's local optimization theorem. *J. Comput. Syst. Sci.* **77**(6), 1141–1158 (2011)
71. D.R. Fulkerson, Blocking and anti-blocking pairs of polyhedra. *Math. Program.* **1**, 168–194 (1971)
72. J. Gagneur, R. Krause, T. Bouwmeester, G. Casari, Modular decomposition of protein-protein interaction networks. *Genome Biol.* **5**(8), R57.1–R57.12 (2004)
73. G. Gallo, M.D. Grigoriadis, R.E. Tarjan, A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* **18**(1), 30–55 (1989)
74. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman, New York, 1979)
75. R. Giles, L.E. Trotter, On stable set polyhedra for $k_{1,3}$ -free graphs. *J. Combin. Theory B* **31**(3), 313–326 (1981)
76. L.M. Glass, R.J. Glass, Social contact networks for the spread of pandemic influenza in children and teenagers. *BMC Public Health* **8**, 61 (2008)
77. J. Grossman, P. Ion, R. De Castro, The Erdős number project (1995), Online: <http://www.oakland.edu/enp/>. Accessed Mar 2007
78. R. Hassin, S. Rubinstein, A. Tamir, Approximation algorithms for maximum dispersion. *Oper. Res. Lett.* **21**(3), 133–137 (1997)
79. J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.* **182**, 105–142 (1999)
80. F. Havet, R.J. Kang, J.-S. Sereni, Improper colouring of unit disk graphs. *Electron. Notes Discret. Math.* **22**, 123–128 (2005)
81. F. Havet, R.J. Kang, J.-S. Sereni, Improper colouring of unit disk graphs. Technical report RR-6206, Institute National de Recherche en Informatique et en Automatique (INRIA), France, May 2007. To appear in *Networks*. Available at: http://hal.inria.fr/inria-00150464_v2/
82. S. Hill, F. Provost, C. Volinsky, Network-based marketing: identifying likely adopters via consumer networks. *Stat. Sci.* **22**, 256–275 (2006)
83. D.S. Hochbaum, Efficient bounds for the stable set, vertex cover and set packing problems. *Discret. Appl. Math.* **6**(3), 243–254 (1983)
84. H.B. Hunt, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, R.E. Stearns, NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algor.* **26**(2), 238–274 (1998)
85. D. Iacobucci, N. Hopkins, Modelling dyadic interactions and networks in marketing. *J. Mark. Res.* **24**, 5–17 (1992)
86. T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, Y. Sakaki, A comprehensive two-hybrid analysis to explore the yeast protein interactome. *Proc. Natl. Acad. Sci. USA* **98**(8), 4569–4574 (2001)

87. D. Jiang, J. Pei, Mining frequent cross-graph quasi-cliques. *ACM Trans. Knowl. Discov. Data* **2**(4), 16:1–42 (2009)
88. R. Kang, Improper coloring of graphs. Ph.D. thesis, University of Oxford, 2007
89. J.M. Keil, T.B. Brecht, The complexity of clustering in planar graphs. *J. Combin. Math. Combin. Comput.* **9**, 155–159 (1991)
90. S. Khot, Ruling out ptas for graph min-bisection, densest subgraph and bipartite clique, in *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, Vancouver, pp. 136–145 (2004)
91. C. Komusiewicz, F. Hüffner, H. Moser, R. Niedermeier, Isolation concepts for efficiently enumerating dense subgraphs. *Theor. Comput. Sci.* **410**(38–40), 3640–3654 (2009)
92. G. Kortsarz, D. Peleg, On choosing a dense subgraph, in *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science* (IEEE, Piscataway, 1993), pp. 692–701
93. M. Lazić, I. Milis, V. Zissimopoulos, A constant approximation algorithm for the densest k-subgraph problem on chordal graphs. *Inf. Process. Lett.* **108**(1), 29–32 (2008)
94. R.D. Luce, Connectivity and generalized cliques in sociometric group structure. *Psychometrika* **15**(2), 169–190 (1950)
95. R.D. Luce, A.D. Perry, A method of matrix analysis of group structure. *Psychometrika* **14**(2), 95–116 (1949)
96. C. Lund, M. Yannakakis, The approximation of maximum subgraph problems, in *Proceedings of the 20th International Colloquium on Automata, Languages and Programming, ICALP '93* (Springer, London, 1993), pp. 40–51
97. F. Mahdavi Pajouh, B. Balasundaram, On inclusionwise maximal and maximum cardinality k -clubs in graphs. *Discret. Optim.* **9**(2), 84–97 (2012). doi:10.1016/j.disopt.2012.02.002
98. F. Mahdavi Pajouh, Polyhedral combinatorics, complexity & algorithms for k -clubs in graphs. Oklahoma State University, Stillwater (2012)
99. F. Mahdavi Pajouh, Z. Miao, B. Balasundaram, A branch-and-bound approach for maximum quasi-cliques. *Ann. Oper. Res.* (2012). doi:10.1007/s10479-012-1242-y
100. M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz, Simple heuristics for unit disk graphs. *Networks* **25**, 59–68 (1995)
101. J. Marincek, B. Mohar, On approximating the maximum diameter ratio of graphs. *Discret. Math.* **244**(1–3), 323–330 (2002)
102. B. McClosky, Independence systems and stable set relaxations. Ph.D. thesis, Rice University, 2008
103. B. McClosky, Clique relaxations, in *Wiley Encyclopedia of Operations Research and Management Science* Cochran, ed. by J.J. Cochran, L.A. Cox, P. Keskinocak, J.P. Kharoufeh, J.C. Smith (Wiley, New York, 2010)
104. B. McClosky, J.D. Arellano, I.V. Hicks, Co- k -plex vertex partitions. Rice University, Houston (2012)
105. B. McClosky, I.V. Hicks, The co-2-plex polytope and integral systems. *SIAM J. Discret. Math.* **23**(3), 1135–1148 (2009)
106. B. McClosky, I.V. Hicks, Combinatorial algorithms for the maximum k -plex problem. *J. Combin. Opt.* **23**, 29–49 (2012)
107. S.T. McCormick, Optimal approximation of sparse Hessians and its equivalence to a graph coloring problem. *Math. Program.* **26**, 153–171 (1983)
108. N. Memon, H.L. Larsen, Structural analysis and mathematical methods for destabilizing terrorist networks using investigative data mining, in *Advanced Data Mining and Applications*, ed. by X. Li, O. Zaïane, Z. Li. Lecture Notes in Computer Science, vol. 4093 (Springer, Berlin/Heidelberg, 2006), pp. 1037–1048
109. N. Memon, K.C. Kristoffersen, D.L. Hicks, H.L. Larsen, Detecting critical regions in covert networks: a case study of 9/11 terrorists network, in *The Second International Conference on Availability, Reliability and Security, ARES 2007*, Vienna, 2007, pp. 861–870
110. J. Miao, D. Berleant, From paragraph networks to document networks, in *Proceedings of the International Conference on Information Technology: Coding and Computing, (ITCC 2004)*, Los vegas, vol. 1, 2004, pp. 295–302

111. N. Mladenović, P. Hansen, Variable neighborhood search. *Comput. Oper. Res.* **24**, 1097–1100 (1997)
112. R.J. Mokken, Cliques, clubs and clans. *Qual. Quant.* **13**(2), 161–173 (1979)
113. H. Moser, Finding optimal solutions for covering and matching problems. Ph.D. thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2009
114. H. Moser, R. Niedermeier, M. Sorge, Exact combinatorial algorithms and experiments for finding maximum k -plexes. *J. Combin. Opt.* 1–27 (2011). doi:10.1007/s10878-011-9391-5
115. T.S. Motzkin, E.G. Straus, Maxima for graphs and a new proof of a theorem of Turán. *Canad. J. Math.* **17**, 533–540 (1965)
116. G.L. Nemhauser, L.E. Trotter, Properties of vertex packings and independence system. *Math. Program.* **6**, 48–61 (1974)
117. G.L. Nemhauser, L.E. Trotter, Vertex packing: structural properties and algorithms. *Math. Program.* **8**, 232–248 (1975)
118. N. Nishimura, P. Ragde, D.M. Thilikos, Fast fixed-parameter tractable algorithms for nontrivial generalizations of vertex cover. *Discret. Appl. Math.* **152**(1–3), 229–245 (2005)
119. Office of Technology Assessment U.S. Congress, Technologies for detecting money laundering, in *Information Technologies for the Control of Money Laundering* (U.S. Government Printing Office, Washington, DC, 1995), pp. 51–74. Online: <http://www.wws.princeton.edu/ota/disk1/1995/9529/9529.PDF>. Accessed May 2006
120. P.R.J. Östergård, A fast algorithm for the maximum clique problem. *Discret. Appl. Math.* **120**, 197–207 (2002)
121. M.W. Padberg, On the facial structure of set packing polyhedra. *Math. Programm.* **5**(1), 199–215 (1973)
122. P.M. Pardalos, J. Xue, The maximum clique problem. *J. Global Opt.* **4**, 301–328 (1994)
123. S. Pasupuleti, Detection of protein complexes in protein interaction networks using n -clubs, in *EvoBIO 2008: Proceedings of the 6th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*, Naples. Lecture Notes in Computer Science, vol. 4973 (Springer, Berlin, 2008), pp. 153–164
124. J. Patillo, A. Veremyev, S. Butenko, V. Boginski, On the maximum quasi-clique problem. *Discret. Appl. Math.* **161**(1–2), 244–257 (2013). doi:10.1016/j.dam.2012.07.019
125. J. Patillo, N. Youssef, S. Butenko, Clique relaxation models in social network analysis, in *Handbook of Optimization in Complex Networks*, ed. by M.T. Thai, P.M. Pardalos. Springer Optimization and its Applications, vol. 58 (Springer, New York, 2012), pp. 143–162
126. J. Pei, D. Jiang, A. Zhang, Mining cross-graph quasi-cliques in gene expression and protein interaction data, in *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005*, Tokyo, 2005, pp. 353–356
127. X. Peng, M.A. Langston, A.M. Saxton, N.E. Baldwin, J.R. Snoddy, Detecting network motifs in gene co-expression networks through integration of protein domain information, in *Methods of Microarray Data Analysis V*, ed. by P. McConnell, S.M. Lin, P. Hurban (Springer, New York, 2007), pp. 89–102
128. O.A. Prokopyev, V. Boginski, W. Chaovatitwongse, P.M. Pardalos, J.C. Sackellares, P.R. Carney, Network-based techniques in eeg data analysis and epileptic brain modeling, in *Data Mining in Biomedicine*, ed. by P.M. Pardalos, V. Boginski, A. Vazacopoulos (Springer, Berlin, 2007), pp. 559–573
129. S.S. Ravi, D.J. Rosenkrantz, G.K. Tayi, Heuristics and special case algorithms for dispersion problems. *Oper. Res.* **42**(2), 299–310 (1994)
130. F. Roupin, A. Billionnet, A deterministic approximation algorithm for the densest k -subgraph problem. *Int. J. Oper. Res.* **3**(3), 301–314 (2008)
131. A. Schäfer, Exact algorithms for s-club finding and related problems. Master's thesis, Diplomarbeit, Institut für Informatik, Friedrich-Schiller-Universität Jena, 2009
132. A. Schäfer, C. Komusiewicz, H. Moser, R. Niedermeier, Parameterized computational complexity of finding small-diameter subgraphs. *Opt. Lett.* 1–9 (2011). doi:10.1007/s11590-011-0311-5

133. S.B. Seidman, B.L. Foster, A graph theoretic generalization of the clique concept. *J. Math. Sociol.* **6**, 139–154 (1978)
134. E.C. Sewell, A branch and bound algorithm for the stability number of a sparse graph. *INFORMS J. Comput.* **10**(4), 438–447 (1998)
135. T. Smieszek, L. Fiebig, R.W. Scholz, Models of epidemics: when contact repetition and clustering should be included. *Theor. Biol. Med. Model.* **6**(1), 11 (2009)
136. M.K. Sparrow, The application of network analysis to criminal intelligence: an assessment of the prospects. *Soc. Netw.* **13**, 251–274 (1991)
137. V. Spirin, L.A. Mirny, Protein complexes and functional modules in molecular networks. *Proc. Natl. Acad. Sci.* **100**(21), 12123–12128 (2003)
138. A. Srivastav, K. Wolf, Finding dense subgraphs with semidefinite programming, in *Approximation Algorithms for Combinatorial Optimization*, ed. by K. Jansen, J. Rolim. Lecture Notes in Computer Science, vol. 1444 (Springer, Berlin/Heidelberg, 1998), pp. 181–191. doi: 10.1007/BFb0053974
139. P.-N. Tan, M. Steinbach, V. Kumar, *Introduction to Data Mining* (Addison-Wesley, Boston, 2006)
140. L. Terveen, W. Hill, B. Amento, Constructing, organizing, visualizing collections of topically related, web resources. *ACM Trans. Comput. Human Int.* **6**, 67–94 (1999)
141. E. Tomita, T. Kameda, An efficient branch-and-bound algorithm for finding a maximum clique with computational experiments. *J. Global Opt.* **37**(1), 95–111 (2007)
142. L.E. Trotter, A class of facet producing graphs for vertex packing polyhedra. *Discret. Math.* **12**, 373–388 (1975)
143. S. Trukhanov, Novel approaches for solving large-scale optimization problems on graphs. Ph.D. thesis, Texas A&M University, 2008
144. S. Trukhanov, B. Balasundaram, S. Butenko, Exact algorithms for hard node deletion problems arising in network-based data mining (2011, submitted)
145. A. Veremyev, V. Boginski, Identifying large robust network clusters via new compact formulations of maximum k -club problems. *Eur. J. Oper. Res.* **218**(2), 316–326 (2012)
146. A. Veremyev, V. Boginski, P. Krokhmal, D.E. Jeffcoat, Asymptotic behavior and phase transitions for clique relaxations in random graphs, working paper, 2010
147. B.H. Voy, J.A. Scharff, A.D. Perkins, A.M. Saxton, B. Borate, E.J. Chesler, L.K. Branstetter, M.A. Langston, Extracting gene networks for low-dose radiation using graph theoretical algorithms. *PLoS Comput. Biol.* **2**(7), e89 (2006)
148. T. Washio, H. Motoda, State of the art of graph-based data mining. *SIGKDD Explor. Newsl.* **5**(1), 59–68 (2003)
149. S. Wasserman, K. Faust, *Social Network Analysis* (Cambridge University Press, New York, 1994)
150. D. West, *Introduction to Graph Theory* (Prentice-Hall, Upper Saddle River, 2001)
151. D.R. Wood, An algorithm for finding a maximum clique in a graph. *Oper. Res. Lett.* **21**(5), 211–217 (1997)
152. A.G. Woodside, M.W. DeLozier, Effects of word of mouth advertising on consumer risk taking. *J. Advert.* **5**(4), 12–19 (1976)
153. M. Yannakakis, Node-and edge-deletion NP-complete problems, in *STOC '78: Proceedings of the 10th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1978), pp. 253–264
154. Z. Zeng, J. Wang, L. Zhou, G. Karypis, Coherent closed quasi-clique discovery from large dense graph databases, in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06* (ACM, New York, 2006), pp. 797–802

Greedy Approximation Algorithms

Peng-Jun Wan

Contents

1	Introduction	1600
2	Independence Systems	1601
2.1	Maximum Independent Set	1601
2.2	Maximum-Weight Independent Set	1603
3	Dependence Systems	1608
3.1	Minimum-Cost Submodular Cover	1609
3.2	Minimum CDS	1616
3.3	Minimum-Weight CDS	1621
4	Graph Coloring	1624
5	Conclusion	1627
	Cross-References	1627
	Recommended Reading	1628

Abstract

Greedy strategy is a simple and natural method in the design of approximation algorithms. This chapter presents greedy approximation algorithms for very broad classes of maximization problems and minimization problems and analyzes their approximation bounds. A number of applications of these greedy approximation algorithms are also described in this chapter. For many applications, the greedy approximation algorithm can achieve the best known approximation bound in a straightforward manner. For some applications, they have different formulations, and the corresponding greedy approximation algorithms have different approximation bounds. In addition, some applications can have better approximation algorithm than the greedy approximation algorithm.

P.-J. Wan

Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA
e-mail: wan@cs.iit.edu

1 Introduction

Greedy strategy is one of the major techniques in the design of approximation algorithms for NP-hard optimization problems. A greedy approximation algorithm is an iterative algorithm which produces a partial solution incrementally. Each iteration makes a locally optimal or suboptimal augmentation to the current partial solution, so that a globally suboptimal solution is reached at the end of the algorithm. This chapter presents a number of classes of optimization problems to which the greedy strategy can be applied to produce good approximate solutions.

The following standard terms and notations are adopted throughout this chapter. Consider a ground set E and a real function f defined on 2^E . f is *increasing* if, for any two subsets A and B of E , $A \subset B$ implies $f(A) \leq f(B)$. f is *proper* if, for any $X \subseteq E$,

$$f(X) \leq \sum_{e \in X} f(e).$$

f is *submodular* if, for any two subsets A and B of E ,

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

f is a *polymatroid function* if $f(\emptyset) = 0$ and f is submodular and increasing. The *marginal value* of $B \subseteq E$ with respect to $A \subseteq E$ is defined by

$$\Delta_B f(A) = f(A \cup B) - f(A).$$

Similarly, the *marginal value* of an element $e \in E$ with respect to $A \subseteq E$ is defined by

$$\Delta_e f(A) = f(A \cup \{e\}) - f(A).$$

Then, f is increasing if and only if $\Delta_x f(A) \geq 0$ for any $A \subseteq E$ and $x \in E \setminus A$. f is submodular if and only if for any $A \subseteq E$ and *different* $x, y \in E \setminus A$, the inequality $\Delta_x f(A) \geq \Delta_x f(A \cup \{y\})$ holds (see, e.g., [28]). In addition, the following are equivalent:

- f is increasing and submodular.
- For any $A, B \subseteq E$,

$$f(B) - f(A) \leq \sum_{x \in B \setminus A} \Delta_x f(A).$$

- For any $A \subseteq E$ and $x, y \in E \setminus A$,

$$\Delta_x f(A) \geq \Delta_x f(A \cup \{y\}).$$

2 Independence Systems

Consider a finite ground set E and a collection \mathcal{I} of subsets of E . (E, \mathcal{I}) is called an *independence system* if \mathcal{I} is hereditary (i.e., for any two subsets I and J of E , $I \subseteq J \in \mathcal{I}$ implies $I \in \mathcal{I}$). Each member of \mathcal{I} is called an *independent set* or a *stable set*. A subset I of E is said to be a *maximal independent set* (or *base*) if $I \in \mathcal{I}$, but for any $e \in E \setminus I$, $I + e \notin \mathcal{I}$. A subset I of E is said to be a *circuit* if $I \notin \mathcal{I}$, but for any $e \in I$, $I - e \in \mathcal{I}$. This section presents greedy approximation algorithms for the following two general maximization problems:

- *Maximum Independent Set (MIS)*: given an independent system (E, \mathcal{I}) , compute a set $I \in \mathcal{I}$ maximizing $|I|$.
- *Maximum-Weight Independent Set (MWIS)*: given an independent system (E, \mathcal{I}) and an increasing nonnegative weight (or profit) function f on 2^E , compute a set $I \in \mathcal{I}$ maximizing $f(I)$.

2.1 Maximum Independent Set

Consider an independent system (E, \mathcal{I}) and an ordering $\langle e_1, e_2, \dots, e_n \rangle$ of E ; an independent set I can be computed by a first-fit greedy algorithm *GreedyMIS* described in [Table 1](#).

We derive an upper bound on the approximation ratio of the algorithm *GreedyMIS* in terms of the exchangeability of an ordering of E . Let \prec be an ordering of E . For a subset $X \subset E$ and an element $e \in X$, $X \prec e$ (respectively, $e \prec X$) means for each $x \in X$, $x \prec e$ (respectively, $e \prec x$). The ordering \prec is said to be *q-exchangeable* if the following holds: If $X \subset Y \in \mathcal{I}$, $X \prec e \prec Y \setminus X$, and $X + e \in \mathcal{I}$, then there is a set $Z \subseteq Y \setminus X$ such that $|Z| \leq q$ and $(Y \setminus Z) + e \in \mathcal{I}$.

Theorem 1 *The approximation ratio of the algorithm GreedyMIS in a q-exchangeable ordering is at most q.*

Proof Let x_1, x_2, \dots, x_k be the sequence of elements selected by the greedy algorithm. Denote $I_0 = \emptyset$, and $I_j = \{x_1, \dots, x_j\}$ for each $1 \leq j \leq k$. Let O be an optimal solution. We construct k subsets C_1, C_2, \dots, C_k of O as described in [Table 2](#). We will show that $\{C_1, C_2, \dots, C_k\}$ is a partition of O and $|C_j| \leq q$ for each $1 \leq j \leq k$, from which the theorem follows.

We first claim that for each $1 \leq j \leq k$, $I_j \cup B_j \in \mathcal{I}$ and $I_j \cap B_j = \emptyset$. The claim is proved by induction on j . The claim holds trivially for $j = 0$. Assume that the claim holds for some $j - 1$ with $1 \leq j < k$. We consider two cases:

Case 1: $x_j \in B_{j-1}$. Then, $C_j = \{x_j\}$ and $B_j = B_{j-1} \setminus \{x_j\}$. So

$$I_j \cup B_j = (I_{j-1} + x_j) \cup (B_{j-1} - x_j) = I_{j-1} \cup B_{j-1} \in \mathcal{I},$$

$$I_j \cap B_j = (I_{j-1} + x_j) \cap (B_{j-1} - x_j) \subseteq I_{j-1} \cap B_{j-1} = \emptyset.$$

Table 1 The first-fit selection of a maximal independent set

	<i>Greedy MIS</i>
	$I \leftarrow \emptyset;$
	For $i = 1$ to n do
	if $I + e_i \in \mathcal{I}$ then $I \leftarrow I + e_i$;
	Output I .

Table 2 A partition of O

	$B_0 \leftarrow O;$
	for $j = 1$ to k do
	if $x_j \in B_{j-1}$ then $C_j \leftarrow \{x_j\}$;
	else $C_j \leftarrow$ a smallest subset C of B_{j-1} such that
	$I_j \cup (B_{j-1} \setminus C) \in \mathcal{I};$
	$B_j \leftarrow B_{j-1} \setminus C_j;$
	output $\{C_1, C_2, \dots, C_k\}$.

Case 2: $x_j \notin B_{j-1}$. Then,

$$\begin{aligned} I_j \cup B_j &= I_j \cup (B_{j-1} \setminus C_j) \in \mathcal{I}, \\ I_j \cap B_j &= (I_{j-1} + x_j) \cap B_j \subseteq (I_{j-1} + x_j) \cap B_{j-1} = I_{j-1} \cap B_{j-1} = \emptyset. \end{aligned}$$

Therefore, the claim holds for j as well.

Next, we show that $|C_j| \leq q$ for each $1 \leq j \leq k$ in two cases:

Case 1: $x_j \in B_{j-1}$. Then, $|C_j| = 1 \leq q$.

Case 2: $x_j \notin B_{j-1}$. Since $I_{j-1} \cup B_{j-1} \in \mathcal{I}$, we have $I_{j-1} \prec x_j \prec B_{j-1}$ by the first-fit greedy principle. Since $I_{j-1} + x_j = I_j \in \mathcal{I}$ and $I_{j-1} \cap B_{j-1} = \emptyset$, there is a set $C \subseteq (I_{j-1} \cup B_{j-1}) \setminus I_{j-1} = B_{j-1}$ such that $|C| \leq q$ and $((I_{j-1} \cup B_{j-1}) \setminus C) + x_j = I_j \cup (B_{j-1} \setminus C) \in \mathcal{I}$. Therefore, $|C_j| \leq q$.

Finally, we show that $\{C_1, C_2, \dots, C_k\}$ is a partition of O . By the first-fit greedy principle, I_k is a maximal independent set. Since $I_k \cup B_k \in \mathcal{I}$ and $I_k \cap B_k = \emptyset$, we have $B_k = \emptyset$. As

$$B_k = O \setminus (C_1 \cup C_2 \cup \dots \cup C_k),$$

$\{C_1, C_2, \dots, C_k\}$ is a partition of O . ■

For the application of [Theorem 1](#) to specific problems, the major task is to find an ordering of the ground set with small exchangeability. Two prominent examples are seeking the maximum number of interference-free node transmissions [[37](#), [38](#)] or link transmissions [[32](#), [36](#)] in wireless networks. Certain natural node orderings or link orderings have been shown to have constant-bounded exchangeability.

2.2 Maximum-Weight Independent Set

Consider an independence system (E, \mathcal{I}) and a nonnegative weight (or profit) function f on 2^E . In the general greedy approximation strategy for this instance of *MWIS*, each iteration optimally or suboptimally solves the augmentation problem: For any $I \in \mathcal{I}$, find one $e \in E \setminus I$ with $I + e \in \mathcal{I}$ that maximizes $\Delta_e f(I)$. Let \mathcal{A} be a polynomial μ -approximation algorithm for this augmentation (the case $\mu = 1$ means the exact algorithm). [Table 3](#) describes the greedy algorithm *GreedyMWIS* which utilizes the algorithm \mathcal{A} in each iteration.

To derive the approximation bound of the algorithm *GreedyMWIS*, we introduce some relevant terms. Consider a subset X of E . A subset I of X is said to be a *maximal independent set* (or *base*) of X if $I \in \mathcal{I}$, but for any $x \in X \setminus I$, $I + x \notin \mathcal{I}$. Let $\mathcal{B}(X)$ denote the set of maximal independent subsets of X . The (*upper*) *rank* (or *independence number*) of X is defined by

$$r^+(X) = \max \{|I| : I \in \mathcal{B}(X)\},$$

and the *lower rank* of X is defined by

$$r^-(X) = \min \{|I| : I \in \mathcal{B}(X)\}.$$

The *rank quotient* of (E, \mathcal{I}) is defined by

$$\max \left\{ \frac{r^+(X)}{r^-(X)} : X \subseteq E, \ell(X) > 0 \right\}.$$

Note that an independence system whose rank quotient is one is termed as a *matroid*.

Theorem 2 *Let q be the rank quotient of (E, \mathcal{I}) . The approximation ratio of the algorithm GreedyMWIS is at most $q\mu + 1$ if f is polymatroid and at most $q\mu$ if f is linear.*

The proof of [Theorem 2](#) makes use of the following exchange property of independent systems.

Theorem 3 *Suppose that (E, \mathcal{I}) has rank quotient q . Let X and Y be two bases and $X \setminus Y = \{x_1, x_2, \dots, x_k\}$. Denote $X_0 = X \cap Y$ and $X_i = (X \cap Y) \cup \{x_1, x_2, \dots, x_i\}$ for $1 \leq i \leq k$. Then, there is a partition $\{C_1, C_2, \dots, C_k\}$ of $Y \setminus X$ such that for each $1 \leq i \leq k$, (1) $|C_i| \leq q$, and (2) for each $e \in C_i$, $X_{i-1} + e \in \mathcal{I}$.*

Proof We construct k subsets C_1, C_2, \dots, C_k of $Y \setminus X$ as described in [Table 4](#). Clearly, for each $1 \leq i \leq k$, (1) $|C_i| \leq q$, and (2) for each $e \in C_i$, $X_{i-1} \cup \{e\} \in \mathcal{I}$. So, we only have to show that $\{C_i : 1 \leq i \leq k\}$ is a partition of $Y \setminus X$ or equivalently $Y_0 = \emptyset$. We prove a stronger claim that for each $0 \leq i \leq k$, $|Y_i| \leq iq$ by induction downward on i . As

Table 3 A partition of $Y \setminus X$

<i>GreedyMWIS</i>
$I \leftarrow \emptyset, A \leftarrow E;$
While $A \neq \emptyset$ do
$x \leftarrow$ the augmentation to I output by \mathcal{A} ;
$I \leftarrow I + x;$
$A \leftarrow \{e \in A \setminus I : I + e \in \mathcal{I}\};$
Output I .

Table 4 A partition of $Y \setminus X$

$Y_k \leftarrow Y \setminus X;$
for each $i = k$ down to 1 do
$B_i \leftarrow \{e \in Y_i : X_{i-1} \cup \{e\} \in \mathcal{I}\};$
$C_i \leftarrow$ a subset of B_i of cardinality $\min\{ B_i , q\};$
$Y_{i-1} \leftarrow Y_i \setminus C_i;$
output $\{C_1, C_2, \dots, C_k\}$.

$$|Y_k| = |Y| - |X \cap Y| \leq q |X| - |X \cap Y| \leq q (|X| - |X \cap Y|) = kq,$$

the claim holds for $i = k$. Now, suppose that $|Y_i| \leq iq$ for some $1 < i \leq k$. We show that $|Y_{i-1}| \leq (i-1)q$ in two cases:

Case 1: $|B_i| \geq q$. Then $|C_i| = q$ and $|Y_{i-1}| = |Y_i| - |C_i| \leq (i-1)q$.

Case 2: $|B_i| < q$. Then, $C_i = B_i$ and $Y_{i-1} = Y_i \setminus B_i$. By the definition of B_i , X_{i-1} is a base of

$$X_{i-1} \cup Y_{i-1} = X_{i-1} \cup (Y_i \setminus B_i).$$

Hence, $|Y_{i-1}| \leq q |X_{i-1}| = (i-1)q$. ■

From [Theorem 3](#), we obtain the following weak exchange property of independent systems.

Corollary 1 Suppose that (E, \mathcal{I}) has rank quotient q . Let $X = \{x_1, x_2, \dots, x_k\}$ be a base of E and denote $X_0 = \emptyset$ and $X_j = \{x_1, \dots, x_j\}$ for $1 \leq j \leq k$. Then, for any base Y of E , there is a partition $\{C_1, C_2, \dots, C_k\}$ of Y such that for each $1 \leq i \leq k$, (1) $|C_i| \leq q$ (2) $X_{i-1} \cap C_i = \emptyset$, and (3) for each $e \in C_i$, $X_{i-1} + e \in \mathcal{I}$.

Now, we prove [Theorem 2](#). Let x_1, x_2, \dots, x_k be the sequence of elements selected by the greedy algorithm. Denote $I_0 = \emptyset$, and $I_j = \{x_1, \dots, x_j\}$ for each $1 \leq j \leq k$. Let O be an optimal solution. By [Corollary 1](#), there is a partition $\{C_1, C_2, \dots, C_k\}$ of O such that for each $1 \leq j \leq k$, (1) $|C_j| \leq q$, (2) $I_{j-1} \cap C_j = \emptyset$, and (3) for each $e \in C_j$, $I_{j-1} + e \in \mathcal{I}$. We first consider the case that f is polymatroid. By the greedy rule, for each $e \in C_j$,

$$\Delta_e f(I_{j-1}) \leq \mu \Delta_{x_j} f(I_{j-1}),$$

which implies that

$$\Delta_{C_j} f(I_{j-1}) \leq |C_j| \cdot \mu \Delta_{x_j} f(I_{j-1}) \leq q\mu \Delta_{x_j} f(I_{j-1}).$$

Therefore,

$$\begin{aligned} f(O) - f(I) &\leq \sum_{j=1}^k \Delta_{C_j} f(I) \leq \sum_{j=1}^k \Delta_{C_j} f(I_{j-1}) \\ &\leq q\mu \sum_{j=1}^k \Delta_{x_j} f(I_{j-1}) = q\mu f(I). \end{aligned}$$

So,

$$f(O) \leq (q\mu + 1) f(I).$$

Next, we consider the case that f is linear. In this case, for each $e \in C_j$, $f(e) \leq \mu f(x_j)$, which implies that $f(C_j) \leq q\mu f(x_j)$. So, $f(O) \leq q\mu f(I)$. This completes the proof of [Theorem 2](#).

Remark For linear f and $q = \mu = 1$, [Theorem 2](#) is the classic result on greedy algorithm for matroid due to Rado [29] and Edmonds [9, 10]. For linear f and $\mu = 1$, [Theorem 2](#) was proved by Jenkyns [19] and Korte and Hausmann [22]. For polymatroid f and $\mu = 1$, [Theorem 2](#) was proved by Fisher et al. [12]. For polymatroid f and $\mu \geq 1$, the theorem was proved by Calinescu et al. [4].

A matroid (E, \mathcal{I}) is called a uniform matroid if

$$\mathcal{I} = \{I \subseteq E : |I| \leq k\}$$

for some positive integer k . For uniform matroid (E, \mathcal{I}) , the algorithm *GreedyMWIS* achieves better approximation bound.

Theorem 4 *If (E, \mathcal{I}) is a uniform matroid and f is a polymatroid function, then the approximation ratio of the algorithm GreedyMWIS is at most $\frac{1}{1-e^{-1/\mu}}$.*

The proof of the above theorem utilizes the following algebraic inequality.

Lemma 1 *Suppose that c_1, c_2, \dots, c_k are k positive numbers less than a . Then, for any k nonnegative numbers x_1, x_2, \dots, x_k ,*

$$\min_{i=1}^k \left(\sum_{j=1}^{i-1} c_j x_j + a x_i \right) \leq \frac{\sum_{i=1}^k c_i x_i}{1 - \prod_{i=1}^k \left(1 - \frac{c_i}{a}\right)}.$$

Proof For each $1 \leq i \leq k$, let

$$\lambda_i = \frac{\frac{c_i}{a} \prod_{j=i+1}^k \left(1 - \frac{c_j}{a}\right)}{1 - \prod_{i=1}^k \left(1 - \frac{c_j}{a}\right)}.$$

Then $\sum_{i=1}^k \lambda_i = 1$, and hence,

$$\begin{aligned} \min_{i=1}^k \left(\sum_{j=1}^{i-1} c_j x_j + ax_i \right) &\leq \sum_{i=1}^k \lambda_i \left(\sum_{j=1}^{i-1} c_j x_j + ax_i \right) \\ &= \sum_{i=1}^k \left(\frac{a}{c_i} \lambda_i + \sum_{j=i+1}^k \lambda_j \right) c_i x_i = \frac{\sum_{i=1}^k c_i x_i}{1 - \prod_{i=1}^k \left(1 - \frac{c_j}{a}\right)}. \end{aligned}$$

So, the lemma follows. ■

Now, we prove [Theorem 4](#). Let x_1, x_2, \dots, x_k be the sequence of elements selected by the greedy algorithm. Then, $|O| = k$. By the greedy rule, for each $1 \leq i \leq k$ and each $y \in O \setminus I_{i-1}$,

$$\Delta y f(I_{i-1}) \leq \mu \Delta x_i f(I_{i-1}),$$

and consequently,

$$\begin{aligned} f(O) &\leq f(I_{i-1}) + \sum_{y \in O \setminus I_{i-1}} \Delta y f(I_{i-1}) \\ &\leq f(I_{i-1}) + |O \setminus I_{i-1}| \mu \Delta x_i f(I_{i-1}) \\ &\leq \sum_{j=1}^{i-1} \Delta x_j f(I_{j-1}) + k \mu \Delta x_i f(I_{i-1}). \end{aligned}$$

By [Lemma 1](#), we have

$$f(O) \leq \min_{i=1}^k \left(\sum_{j=1}^{i-1} \Delta x_j f(I_{j-1}) + k \mu \Delta x_i f(I_{i-1}) \right)$$

$$\begin{aligned} &\leq \frac{\sum_{j=1}^k \Delta x_j f(I_{j-1})}{1 - \left(1 - \frac{1}{k\mu}\right)^k} = \frac{f(I_k)}{1 - \left(1 - \frac{1}{k\mu}\right)^k} \\ &\leq \frac{f(I_k)}{1 - e^{-1/\mu}}. \end{aligned}$$

This completes the proof of [Theorem 4](#).

Remark Maximizing submodular functions subject to a uniform matroid constraint appears to have come from an application to a facility location problem in the work of Cornuejols et al. [6]. In [26], Fisher et al. formally proved the special case of [Theorem 4](#) in which $\mu = 1$. For this special case, Nemhauser and Wolsey [27] further showed that any algorithm with better approximation bound requires an exponential number of queries of the values of f , and Feige [11] showed that $\frac{1}{1-e^{-1}}$ is the best approximation ratio unless $\mathbf{P} = \mathbf{NP}$. For $\mu \geq 1$, Wan and Liu [35] obtained the same approximation bound as the one in [Theorem 4](#) of the greedy approximation algorithm for throughput maximization in wavelength-routed optical networks, and Hochbaum and Pathria [18] also derived the same approximation bound of the greedy approximation algorithm for maximum k -coverage.

For the application of [Theorem 2](#) to specific problems, the main task is to estimate the rank quotient of the underlying independence system. [Theorem 1](#) can often be applied for this purpose. An independence system (E, \mathcal{I}) is said to be q -exchangeable if every ordering of E is q -exchangeable. By [Theorem 1](#), the rank quotient of any q -exchangeable system is at most q . It is easy to verify that:

- (E, \mathcal{I}) is q -exchangeable if and only if for any $X \subset Y \in \mathcal{I}$ and $e \in E$ with $X + e \in \mathcal{I}$, there is a set $Z \subseteq Y \setminus X$ such that $|Z| \leq q$ and $(Y \setminus Z) + e \in \mathcal{I}$.
- (E, \mathcal{I}) is q -exchangeable if for any $I \in \mathcal{I}$ and $e \in E$, $I + e$ contains at most q circuits.
- (E, \mathcal{I}) is q -exchangeable if it is the intersection of q matroids.

Below is the list of instances of MWIS with linear weight function and non-matroid independent systems.

Maximum-Weight Hamiltonian Cycle: Given an edge-weighted complete graph, find a Hamiltonian cycle with maximum total weight.

- Ground set: the set of edges
- Independent sets: Hamiltonian cycles or vertex-disjoint paths
- Rank quotient: 2

Maximum-Weight Directed Hamiltonian Cycle: Given an edge-weighted complete digraph, find a Hamiltonian cycle with maximum total weight.

- Ground set: the set of arcs
- Independent sets: directed Hamiltonian cycles or vertex-disjoint paths.
- Rank quotient: 3

Maximum-Weight Stable Set of Unit-Disk Graphs: Given a vertex-weighted unit-disk graph, find a stable set with maximum total weight.

- Ground set: the set of vertices.
- Independent sets: stable sets. An independent set or stable set of a graph is a set of vertices in the graph, no two of which are adjacent.
- Rank quotient: 5.

A number of applications of [Theorem 4](#) were surveyed by Goundan and Schulz [15].

3 Dependence Systems

Consider a finite ground set E and a collection \mathcal{D} of subsets of E . (E, \mathcal{D}) is called a *dependence system* if for any two subsets X and Y of E , $X \subseteq Y$ and $X \in \mathcal{D}$ implies $Y \in \mathcal{D}$. Each member of \mathcal{D} is called a *dependent set*. A subset X of E is said to be a *minimal dependent set* if $X \in \mathcal{D}$, but for any $e \in X$, $X - e \notin \mathcal{D}$. Given an dependent system (E, \mathcal{D}) and an increasing and proper nonnegative cost function c on 2^E , the problem $\min_{X \in \mathcal{D}} c(X)$ is referred to as *Minimum-Cost Dependent Set*. For many instances of *Minimum-Cost Dependent Set*, the dependence system (E, \mathcal{D}) is implicitly defined by an increasing *coverage function* f on 2^E : For each $X \subseteq E$, $X \in \mathcal{D}$ if and only if $f(X) = f(E)$, or it is implicitly defined by a decreasing *coverage deficiency function* (also called *potential function*) f on 2^E ; for each $X \subseteq E$, $X \in \mathcal{D}$ if and only if $f(X) = 0$. If the coverage function is a polymatroid function, the problem *Minimum-Cost Dependent Set* is commonly known as the *Minimum-Cost Submodular Cover*, and each dependent set is also called as a *submodular cover*.

Minimum-Cost Submodular Cover is well behaved. In [Sect. 3.1](#), we present a general greedy approximation algorithm for *Minimum-Cost Submodular Cover* and its approximation bound. For other instances of *Minimum-Cost Dependent Set* which cannot be cast as *Minimum-Cost Submodular Cover*, they lack such unified framework for both design and analysis of greedy approximation algorithms. In [Sects. 3.2](#) and [3.3](#), we present greedy approximation algorithms for *Minimum Connected Dominating Set (MCDS)* and *Minimum-Weight Connected Dominating Set (MWCDS)*, respectively. A connected dominating set (CDS) of a connected graph $G = (V, E)$ is a subset of vertices U such that U is a dominating set (i.e., each vertex $v \in V \setminus U$ is adjacent to at least one vertex in U) and the induced subgraph $G[U]$ is connected. The problem *MCDS* seeks a CDS of the smallest size in a given graph G , and the problem *MWCDS* seeks a CDS of the smallest weight in a given vertex-weighted graph G . They are selected as representative examples for the following reasons:

- *MCDS* is associated with a coverage function which is not submodular but has a special shifted submodularity. Such shifted submodularity can still be exploited to design and analyze the greedy approximation algorithm in the manner similar to those for *Minimum-Cost Submodular Cover*.

- The naive extension of the greedy approximation algorithm for *MCDS* to *MWCDS* has a very poor approximation ratio.
- The design and analysis of the greedy approximation algorithm for *MWCDS* can be slightly modified to some other well-known minimization problems, which are listed in Sect. 3.3.

3.1 Minimum-Cost Submodular Cover

Consider an instance of *Minimum-Cost Submodular Cover* specified by the ground set E , the polymatroid coverage function f , and the increasing and proper cost function c . In the general greedy approximation strategy for this instance, each iteration optimally or suboptimally solves the augmentation problem: For any given $X \subset E$, find one $e \in E \setminus X$ that maximizes $\Delta_e f(X) / c(e)$. Let \mathcal{A} be a polynomial μ -approximation algorithm for this augmentation (the case $\mu = 1$ means the exact algorithm). Table 5 describes the greedy algorithm *GreedyMCSC* which utilizes the algorithm \mathcal{A} in each iteration.

Let x_1, x_2, \dots, x_k be the sequence of elements selected by the algorithm *GreedyMCSC*. Denote $X_0 = \emptyset$, and $X_i = \{x_1, x_2, \dots, x_i\}$ for each $1 \leq i \leq k$. For each $1 \leq i \leq k$, let $p_i = \frac{c(x_i)}{\Delta_{x_i} f(X_{i-1})}$, which is referred to the price of x_i . Let opt be the cost of an optimal solution. When f is integer-valued, c is linear, and \mathcal{A} is optimal, it is well known that $c(X_k) / \text{opt} \leq H(q)$, where $q = \max_{e \in E} f(e)$ and

$$H(q) = 1 + \frac{1}{2} + \dots + \frac{1}{q}$$

is the k th Harmonic number. For the general case, we define the *curvature* to be

$$\rho = \frac{\min \left\{ \sum_{e \in S} c(e) : S \text{ is a cover} \right\}}{\text{opt}}.$$

Note that $\rho = 1$ if c is linear. We have the following two general performance bounds.

Theorem 5 *If f is integer-valued, then $c(X_k) / \text{opt} \leq \rho \mu H(q)$, where $q = \max_{e \in E} f(e)$.*

Theorem 6 *If $f(E) \geq \text{opt}$ and $p_i \leq 1$ for each $1 \leq i \leq k$, then $c(X_k) / \text{opt} \leq 1 + \rho \mu \ln \frac{f(E)}{\text{opt}}$.*

The proof of both theorems utilizes the following algebraic inequalities. Suppose that $\ell_0, \ell_1, \dots, \ell_{k-1}$ are decreasing positive numbers. Then,

$$\sum_{i=1}^{k-1} \frac{\ell_{i-1} - \ell_i}{\ell_{i-1}} \leq \ln \frac{\ell_0}{\ell_{k-1}}.$$

Table 5 Greedy algorithm for minimum submodular cover

<i>GreedyMCSC</i>
$X \leftarrow \emptyset;$
While $f(X) < f(E)$ do
$x \leftarrow$ the augmentation to X output by \mathcal{A} ;
$X \leftarrow X + x;$
Output X .

If all of them are integers, then

$$\sum_{i=1}^{k-1} \frac{\ell_{i-1} - \ell_i}{\ell_{i-1}} \leq H(\ell_0) - H(\ell_{k-1}) \leq \ln \frac{\ell_0}{\ell_{k-1}}.$$

We prove [Theorem 5](#) by a charging argument. Let $O = \{y_1, y_2, \dots, y_m\}$ be a cover satisfying that

$$\sum_{j=1}^m c(y_j) \leq \rho \cdot \text{opt}.$$

Denote $O_0 = \emptyset$, and $O_j = \{y_1, y_2, \dots, y_j\}$ for each $1 \leq j \leq m$. Each x_i charges on y_j with

$$p_i (\Delta_{y_j} f(X_{i-1} \cup O_{j-1}) - \Delta_{y_j} f(X_i \cup O_{j-1})).$$

Then, the total charge by x_i on O is exactly $c(x_i)$ as

$$\begin{aligned} p_i \sum_{j=1}^m (\Delta_{y_j} f(X_{i-1} \cup O_{j-1}) - \Delta_{y_j} f(X_i \cup O_{j-1})) \\ &= p_i [(f(X_{i-1} \cup O) - f(X_{i-1})) - (f(X_i \cup O) - f(X_i))] \\ &= p_i [f(X_i) - f(X_{i-1})] \\ &= p_i \Delta_{x_i} f(X_{i-1}) \\ &= c(x_i). \end{aligned}$$

Thus, the total charge on O is $\sum_{i=1}^k c(x_i)$. On the other hand, we claim that the total charge on each y_j is at most $\mu H(q) c(y_j)$. Indeed, let l be the first i such that $\Delta_{y_j} f(X_i \cup O_{j-1}) = 0$. Then, the charge on y_j by each x_i with $i \geq l$ is 0. For each $1 \leq i \leq l$, $\Delta_{y_j} f(X_{i-1} \cup O_{j-1}) > 0$ which implies that $y_j \notin X_{i-1} \cup O_{j-1}$, and hence by the greedy rule, we have

$$p_i = \frac{c(x_i)}{\Delta_{x_i} f(X_{i-1})} \leq \mu \frac{c(y_j)}{\Delta_{y_j} f(X_{i-1})} \leq \mu \frac{c(y_j)}{\Delta_{y_j} f(X_{i-1} \cup O_{j-1})}.$$

Thus,

$$\begin{aligned}
& \sum_{i=1}^l p_i (\Delta_{y_j} f(X_{i-1} \cup O_{j-1}) - \Delta_{y_j} f(X_i \cup O_{j-1})) \\
& \leq \mu c(y_j) \sum_{i=1}^l \frac{\Delta_{y_j} f(X_{i-1} \cup O_{j-1}) - \Delta_{y_j} f(X_i \cup O_{j-1})}{\Delta_{y_j} f(X_{i-1} \cup O_{j-1})} \\
& \leq \mu c(y_j) H(\Delta_{y_j} f(O_{j-1})) \leq \mu c(y_j) H(\Delta_{y_j} f(\emptyset)) \\
& \leq \mu H(q) c(y_j).
\end{aligned}$$

So, our claim holds. As

$$\begin{aligned}
& \sum_{j=1}^m \mu H(q) c(y_j) = \mu H(q) \sum_{j=1}^m c(y_j) \\
& \leq \mu H(q) \rho \cdot c(O) = \rho \mu H(q) \cdot \text{opt}.
\end{aligned}$$

The total charge on O is at most $\rho \mu H(q) \cdot \text{opt}$. Therefore,

$$c(X) \leq \sum_{i=1}^k c(x_i) \leq \rho \mu H(q) \cdot \text{opt}.$$

This completes the proof for [Theorem 5](#).

Next, we prove [Theorem 6](#). Let O be a cover satisfying that

$$\sum_{y \in O} c(y) \leq \rho \cdot \text{opt}.$$

For each $0 \leq i \leq k$, let $\ell_i = f(E) - f(X_i)$ be the ‘‘coverage deficiency’’ at the end of iteration i . Let t be the subscript satisfying that $\ell_t \geq \text{opt} > \ell_{t+1}$. We claim that for each $1 \leq i \leq k$,

$$p_i \leq \min \left\{ \rho \mu \frac{\text{opt}}{\ell_{i-1}}, 1 \right\}.$$

Indeed, the inequality $p_i \leq 1$ holds by assumption. By the greedy rule, we have

$$\begin{aligned}
\frac{1}{p_i} &= \frac{\Delta_{x_i} f(X_{i-1})}{c(x_i)} \geq \frac{1}{\mu} \max_{y \in O \setminus X_{i-1}} \frac{\Delta_y f(X_{i-1})}{c(y)} \\
&\geq \frac{1}{\mu} \frac{\sum_{y \in O \setminus X_{i-1}} \Delta_y f(X_{i-1})}{\sum_{y \in O \setminus X_{i-1}} c(y)}
\end{aligned}$$

$$\begin{aligned} &\geq \frac{1}{\mu} \frac{f(O) - f(X_{i-1})}{\sum_{y \in O} c(y)} \\ &\geq \frac{1}{\mu} \frac{\ell_{i-1}}{\rho \cdot \text{opt}} = \frac{\ell_{i-1}}{\rho \mu \cdot \text{opt}}. \end{aligned}$$

So, our claim holds. Thus, for each $1 \leq i \leq k$,

$$\begin{aligned} c(x_i) &= p_i \Delta_{x_i} f(X_{i-1}) = p_i (\ell_{i-1} - \ell_i) \\ &\leq \min \left\{ \rho \mu \frac{\text{opt}}{\ell_{i-1}}, 1 \right\} \cdot (\ell_{i-1} - \ell_i) \\ &= \min \left\{ \rho \mu \cdot \text{opt} \frac{\ell_{i-1} - \ell_i}{\ell_{i-1}}, \ell_{i-1} - \ell_i \right\}. \end{aligned}$$

Hence,

$$\begin{aligned} &\sum_{i=1}^t c(x_i) + \frac{\ell_t - \text{opt}}{\ell_t - \ell_{t+1}} c(x_{t+1}) \\ &\leq \rho \mu \cdot \text{opt} \left(\sum_{i=1}^t \frac{\ell_{i-1} - \ell_i}{\ell_{i-1}} + \frac{\ell_t - \text{opt}}{\ell_t} \right) \\ &\leq \rho \mu \cdot \text{opt} \ln \frac{\ell_0}{\text{opt}} \\ &= \text{opt} \cdot \rho \mu \ln \frac{f(E)}{\text{opt}}, \end{aligned}$$

and

$$\frac{\text{opt} - \ell_{t+1}}{\ell_t - \ell_{t+1}} c(x_{t+1}) + \sum_{i=t+2}^k c(x_i) \leq (\text{opt} - \ell_{t+1}) + \sum_{i=t+2}^k (\ell_{i-1} - \ell_i) = \text{opt}.$$

Summing up the above two inequalities, we have

$$c(X) \leq \sum_{i=1}^k c(x_i) \leq \left(1 + \rho \mu \ln \frac{f(E)}{\text{opt}} \right) \text{opt}.$$

This completes the proof for [Theorem 6](#).

Remark For linear c and $\mu = 1$, [Theorem 5](#) was proved by Wolsey [39]. For submodular c and $\mu = 1$, [Theorems 5](#) and [6](#) were proved by Wan et al. [34].

For many specific *Minimum-Cost Submodular Cover* problems, the coverage function f is not given explicitly. For these problems, the main task is to find the proper submodular coverage function. For some *Minimum-Cost Submodular Cover* problems, such as set cover, the submodular potential functions can be easily defined. But for some others, it is far from trivial to find the submodular potential function. A prominent example is *Weighted Connected Vertex Cover* (WCVC). An instance of WCVC is a connected vertex-weighted graph $G = (V, E)$. A solution to WCVC is a connected vertex cover, which is a subset of vertices inducing a connected subgraph and covering all edges. The objective is to find a connected vertex-cover with the minimum total weight. While the unweighted variant of WCVC has a simple 2-approximation algorithm [13], WCVC has the same approximation hardness as weighted set cover [13]. It can be formulated as a special case of *Minimum-Cost Submodular Cover* in which the ground set is V and the coverage of a subset $U \subseteq V$ is the number of edges in G incident to U minus the number of connected components of the subgraph of G induced by U [7]. By Theorem 5, the approximation ratio of greedy approximation algorithm is at most $H(\Delta - 1)$.

For some specific *Minimum-Cost Submodular Cover* problems, they may have different formulations which lead to different approximation bounds. A prominent example is *Minimum-Power Spanning Tree*. Let $G = (V, E; c)$ be an edge-weighted graph with $c(e) > 0$ for each $e \in E$. Any subgraph H induces a power assignment $p_H : V(H) \rightarrow \mathbb{R}_+$ defined by

$$p_H(u) = \max_{e \in \delta_H(u)} c(e), \forall u \in V(H);$$

the power of H is defined by $\sum_{u \in V(H)} p_H(u)$. The problem of finding a spanning tree of minimum power is referred to as *Minimum-Power Spanning Tree* and may have two different formulations into *Minimum-Cost Submodular Cover*. In the first formulation [20], the ground set is the set E of edges; for any subset any $A \subseteq E$, its coverage is the rank of the graphic matroid on (V, A) , and its cost is the power of (V, F) . For this formulation, $q = \mu = 1$ and the curvature is at most 2. So the approximation ratio of the greedy algorithm is at most 2. In the second formulation [1], the ground set consists of all k -restricted trees (i.e., subtrees of G containing at most k vertices) for some fixed integer $k \geq 2$. Let $\text{mst}(G)$ denote the weight of a minimum-weight spanning tree of G . For any subgraph H of G , let $\text{mst}(G : H)$ denote the weight of the minimum-weight spanning forest F such that $H \cup F$ is connected. For any subset of k -restricted trees whose union is H , its coverage is defined to be $2[\text{mst}(G) - \text{mst}(G : H)]$, and its cost is the power of H . It was proved in [24] that the curvature is at most

$$\rho_k \leq 1 + \frac{1}{\lfloor \log k \rfloor - 1 + k/2^{\lfloor \log k \rfloor}}.$$

By [Theorem 6](#), the approximation ratio of the greedy algorithm is at most

$$1 + \left(1 + \frac{1}{\lfloor \log k \rfloor - 1 + k/2^{\lfloor \log k \rfloor}} \right) \ln 2.$$

When k tends to infinity, the above approximation bound approaches to $1 + \ln 2$, which is better than the approximation bound 2 obtained in the first formulation.

The remaining of the subsection provides the formulations of some well-known optimization problems as instances of *Minimum-Cost Submodular Cover*. For all of them, $\mu = 1$. Below is the list of problems with linear cost.

Generalized Set Multi-cover [14]: Given a matrix $A \in N_{m \times n}$ and two vectors $b \in N_{m \times 1}$, compute

$$\min \left\{ \sum_{j=1}^n c_j x_j : Ax \geq b, x \in \{0, 1\}^n \right\}.$$

- Ground set: $\{1, 2, \dots, n\}$.
- Coverage function: for any $S \subseteq \{1, 2, \dots, n\}$,

$$f(S) = \sum_{i=1}^m \min \left\{ \sum_{j \in S} a_{ij}, b_i \right\}.$$

Capacitated Set Cover [39]: Given a bipartite graph $G = (V, W; E)$ and a capacity bound $b : V \rightarrow \mathbb{N}$, find a subgraph $H = (U, W; E)$ such that $\deg_H(u) \leq b(u)$ for each $u \in U$, $\deg_H(w) \geq 1$ for each $w \in W$, and $|U|$ is minimized.

- Ground set: V .
- Coverage function: for any $U \subseteq V$, $f(U)$ is the maximum number of vertices in W that can be covered by U under the capacity constraint.

Weakly Connected Dominating Set: Given a vertex-weighted graph G , find a minimum-weight weakly connected dominating set.

- Ground set: the set of vertices.
- Coverage function: for any $U \subseteq V$, $f(U)$ is the rank of the graphic matroid on the spanning subgraph of G consisting of all edges incident to U .

Minimum-Label Spanning Tree [33]: Given a graph G in which each edge has a label from $\{1, 2, \dots, L\}$, find a spanning tree with the smallest number of different labels.

- Ground set: $\{1, 2, \dots, L\}$.
- Coverage function: For any $S \subseteq \{1, 2, \dots, L\}$, $f(S)$ is the rank of the graphic matroid on the spanning subgraph of G consisting of all edges with label $l \in S$.

Subset Interconnection Design [8]: Given an edge-weighted graph G and m subsets V_1, \dots, V_m of vertices, find a minimum-weight subgraph H of G satisfying that the subgraph of H induced by each V_i for $1 \leq i \leq m$ is connected.

- Ground set: E .
- Coverage function: for any $A \subseteq E$, let $f_i(A)$ be rank of the graphic matroid on the subgraph of (V, A) induced by V_i for each $1 \leq i \leq m$. The coverage of A is given by

$$f(A) = \sum_{i=1}^m f_i(A).$$

Below are two problems with polymatroid cost functions.

Minimum Steiner Tree: Given an edge-weight graph G and a subset S of vertices, find a minimum-weighted subgraph H of G interconnecting S .

- Ground set: fix a positive integer $k \geq 2$. For any subset U of terminals with $2 \leq |U| \leq k$, we compute a minimum-weight Steiner tree and add the output tree to the ground set if it is full.
- Cost function: for any subset of trees in the ground set whose union is H , its cost is the total weight of the edges in H .
- Coverage function: for any subset of k -restricted full trees whose union is H , its coverage is defined to be $\text{mst}(G) - \text{mst}(G : H)$, and its cost is $c(H)$.
- Curvature: it was shown in [3] that the curvature is at most

$$\rho_k = 1 + \frac{1}{\lfloor \log_2 k \rfloor - 1 + k/2^{\lfloor \log_2 k \rfloor}}.$$

Minimum Node-Weighted Steiner Tree in Unit-Disk Graphs: Given a node-weighted unit-disk graph G and a subset S of vertices, find a minimum node-weighted subgraph H of G interconnecting S .

- Ground set: fix a positive integer $k \geq 2$. For any subset U of terminals with $2 \leq |U| \leq k$, we compute a Minimum Node-Weighted Steiner Tree and add the output tree to the ground set if it is full.
- Cost function: for any subset of trees in the ground set whose union is H , its cost is the total weight of the vertices in H other than the terminals.
- Coverage function: the *length* of a simple path in G is the sum of the weights of all its internal nodes. The *distance* between two nodes in G is the length of the shortest path between them and can be computed in polynomial time. The *terminal distance graph* of G is the complete graph K on S in which the length of each edge is the distance between its two endpoints. For any subset of trees in the ground set whose union is H , its coverage is $\text{mst}(K) - \text{mst}(K : H)$.
- Curvature: it was shown in [40] that the curvature is at most

$$\rho_k = 1 + \frac{4}{\lfloor \log_2 k \rfloor - 1 + k/2^{\lfloor \log_2 k \rfloor}}.$$

3.2 Minimum CDS

Consider a connected graph $G = (V, E)$. If the maximum degree Δ of G is either one or two, then a minimum CDS of G can be computed trivially. Indeed, if $\Delta = 1$, then G contains only one edge and a minimum CDS consists of any single vertex. If $\Delta = 2$, G is a path or a cycle. When G is a path, the minimum CDS consists of all internal nodes; when G is a cycle, a minimum CDS can be obtained by deleting two adjacent nodes. So we assume $\Delta \geq 3$ from now on and let γ_c be the size of a minimum CDS and $n = |V|$. This subsection presents a simple greedy $(2 + \ln(\Delta - 2))$ -approximation algorithm [30] for *MCDS*.

We first describe the coverage function used by the greedy algorithm. For any subset U of V , let $G \langle U \rangle$ denote the spanning subgraph of G including all edges incident to U and $f_1(U)$ be the rank of the graphic matroid on $G \langle U \rangle$ (in other words, $f_1(U)$ equals to n minus the number of connected components of $G \langle U \rangle$) and $f_2(U)$ be the number of connected components of the graph $G[U]$. The coverage of U is then defined to be

$$f(U) = f_1(U) + f_2(U) - 1.$$

Clearly, $f(\emptyset) = 0$; for any $U \subseteq V$, $f(U) \leq n - 2 = f(V)$, and the equality holds if and only if U is a CDS. For any $U \subset V$ and any $v \in V \setminus U$, $\Delta_v f_1(U)$ is the number of connected components of $G \langle U \rangle$ adjacent to but not containing v , and $-\Delta_v f_2(U)$ is the number of connected components of $G[U]$ adjacent to v minus one; consequently, $\Delta_v f(U) \geq 0$. Therefore, f is increasing. However, f is not submodular. Therefore, the general greedy approximation algorithm for *Minimum-Cost Submodular Cover* cannot be directly applied to *MCDS*.

While f is not submodular, it has a “shifted submodularity” as stated in the next lemma.

Lemma 2 Suppose that U and W are two subsets of V satisfying that $G[W]$ is connected. Then, for any vertex $v \in V$,

$$\Delta_v f(U \cup W) \leq \Delta_v f(U) + 1.$$

Proof Since f_1 is submodular, we have

$$\Delta_v f_1(U \cup W) \leq \Delta_v f_1(U).$$

As $G[W]$ is connected, the number of connected components in $G[U \cup W]$ adjacent to v is at most one more than the number of connected components in $G[U]$ adjacent to v . Hence,

$$-\Delta_v f_2(U \cup W) \leq -\Delta_v f_2(U) + 1.$$

Summing up the above two inequalities yields the inequality in the lemma. ■

In addition, f still has the following growth property.

Lemma 3 *If $U \subseteq V$ is not a CDS of G , then there exists a vertex v satisfying that*

$$\Delta_v f(U) \geq \max \left\{ 1, \frac{n - 1 - f(U)}{\gamma_c} - 1 \right\}.$$

Proof We first show that there is a vertex v satisfying that $\Delta_v f(U) \geq 1$. We consider three cases:

Case 1: $U = \emptyset$. Let v be the vertex with maximum degree. Then,

$$\begin{aligned}\Delta_v f_1(U) &= \Delta, \\ \Delta_v f_2(U) &= -1, \\ \Delta_v f(U) &= \Delta - 1 \geq 1.\end{aligned}$$

Case 2: U is nonempty but is not a dominating set. Let v be a vertex which is adjacent to both U and some vertex two hops away from U . Then,

$$\begin{aligned}\Delta_v f_1(U) &\geq 1, \\ \Delta_v f_2(\emptyset) &\geq 0, \\ \Delta_v f(\emptyset) &\geq 1.\end{aligned}$$

Case 3: U is a dominating set. Let U_1 and U_2 be two connected components of $G[U]$ separated by the smallest number of hops. Then the number of hops between U_1 and U_2 is either two or three. So, we further consider two subcases:

Subcase 3.1: U_1 and U_2 are separated by two hops. Let v be a vertex adjacent to both U_1 and U_2 . Then,

$$\begin{aligned}\Delta_v f_1(U) &= 0, \\ \Delta_v f_2(\emptyset) &\geq 1, \\ \Delta_v f(\emptyset) &\geq 1.\end{aligned}$$

Subcase 3.2: U_1 and U_2 are separated by three hops. Let v be a vertex in the shortest (3-hop) path joining U_1 and U_2 . Then,

$$\begin{aligned}\Delta_v f_1(U) &\geq 1, \\ \Delta_v f_2(\emptyset) &= 0, \\ \Delta_v f(\emptyset) &\geq 1.\end{aligned}$$

Next, we show that there is a vertex v satisfying that

$$\Delta_v f(U) \geq \frac{f(U)}{\gamma_c} - 1.$$

Consider a MCDS

$$W = \{v_i : 1 \leq i \leq \gamma_c\}$$

sorted in a breadth-first search order. For each $1 \leq i \leq \gamma_c$, let

$$W_j = \{v_i : 1 \leq i \leq j\}.$$

Then, $G[W_j]$ is connected for each $1 \leq j \leq \gamma_c$. By [Lemma 2](#),

$$\begin{aligned} n - 2 - f(U) &= f(U \cup W) - f(U) \\ &= \Delta_{v_1} f(U) + \sum_{j=2}^{\gamma_c} \Delta_{v_j} f(U \cup W_{j-1}) \\ &\leq \Delta_{v_1} f(U) + \sum_{j=2}^{\gamma_c} (\Delta_{v_j} f(U) + 1) \\ &= \gamma_c - 1 + \sum_{j=1}^{\gamma_c} \Delta_{v_j} f(U) \\ &\leq \gamma_c - 1 + \gamma_c \cdot \max_{1 \leq j \leq \gamma_c} \Delta_{v_j} f(U). \end{aligned}$$

Hence,

$$\max_{1 \leq j \leq \gamma_c} \Delta_{v_j} f(U) \geq \frac{n - 1 - f(U)}{\gamma_c} - 1.$$

Let v be the vertex in W which has the largest gain with respect to U . Then,

$$\Delta_v f(U) \geq \frac{n - 1 - f(U)}{\gamma_c} - 1.$$

This completes the proof of the lemma. ■

Now, we are ready to describe the greedy algorithm *GreedyCDS* in [Table 6](#). By [Lemma 3](#), the algorithm *GreedyCDS* runs in at most $n - 2$ iterations, and the output is a CDS.

We prove the approximation bound $2 + \ln(\Delta - 2)$ of the algorithm *GreedyCDS* in the theorem below.

Theorem 7 *The approximation ratio of GreedyCDS is at most $2 + \ln(\Delta - 2)$.*

Table 6 Outline of the algorithm *GCDS*

<i>GreedyCDS</i>	$B \leftarrow \emptyset;$ While $f(B) < n - 2$ do select $v \in V \setminus B$ with maximum $\Delta_v f(B)$; $B \leftarrow B \cup \{v\}$; Output B .
------------------	--

Proof Let γ_c be the size of a minimum CDS. We claim that $\gamma_c \geq \frac{n-2}{\Delta-1}$. Consider a breadth-first search ordering of a MCDS. The first vertex dominates at most $\Delta + 1$ nodes. Each subsequent vertex dominates at most $\Delta - 1$ additional nodes. Thus,

$$n \leq (\Delta + 1) + (\Delta - 1)(\gamma_c - 1) = (\Delta - 1)\gamma_c + 2.$$

Hence, $\gamma_c \geq \frac{n-2}{\Delta-1}$.

Let B be the output of the algorithm. For each $1 \leq i \leq |B|$, B_i denotes the sequence of the first i nodes in B ; furthermore, let $B_0 = \emptyset$. Let k be the first (smallest) nonnegative integer such that

$$f(B_k) \geq n - 2 - 2\gamma_c.$$

We claim that

$$\begin{aligned} |B \setminus B_k| &\leq 2\gamma_c - 1, \\ k - 1 &\leq \gamma_c \ln(\Delta - 2), \end{aligned}$$

These two inequalities imply that

$$|B| \leq (2 + \ln(\Delta - 2))\gamma_c,$$

and so the theorem follows.

We first prove the first inequality in the claim. By Lemma 3, each vertex in $B \setminus B_k$ has gained at least one. If $f(B_k) > n - 2 - 2\gamma_c$, then

$$|B \setminus B_k| \leq n - 2 - f(B_k) < 2\gamma_c.$$

If $f(B_k) = n - 2 - 2\gamma_c$, then the first vertex in $B \setminus B_k$ has gained at least two with respect to B_k by Lemma 3, and hence,

$$2 + (|B \setminus B_k| - 1) \leq n - 2 - f(B_k) = 2\gamma_c,$$

which also implies that

$$|B \setminus B_k| \leq 2\gamma_c - 1.$$

Next, we prove the second inequality in the claim. This inequality holds trivially if $k \leq 1$. So we assume that $k > 1$. For each $0 \leq i < k$, let $\ell_i = n - 1 - f(B_i) - \gamma_c$, which is referred to as the shifted coverage deficiency. Then

$$n - 1 - \gamma_c = \ell_0 > \ell_1 > \dots > \ell_{k-1} \geq \gamma_c + 2.$$

By [Lemma 3](#), for each $0 \leq i < k$,

$$\ell_{i-1} - \ell_i \geq \frac{n - 1 - f(B_{i-1})}{\gamma_c} - 1 = \frac{\ell_{i-1}}{\gamma_c},$$

and hence,

$$\frac{\ell_{i-1} - \ell_i}{\ell_{i-1}} \geq \frac{1}{\gamma_c}.$$

Therefore,

$$\frac{k-1}{\gamma_c} \leq \sum_{i=1}^{k-1} \frac{\ell_{i-1} - \ell_i}{\ell_{i-1}} \leq \ln \frac{\ell_0}{\ell_{k-1}} \leq \ln \frac{n-1-\gamma_c}{\gamma_c+2}.$$

Since $\gamma_c \geq \frac{n-2}{\Delta-1}$, we have $n-2 \leq (\Delta-1)\gamma_c$, and hence,

$$\begin{aligned} n - 1 - \gamma_c &\leq (\Delta - 1)\gamma_c + 1 - \gamma_c \\ &= (\Delta - 2)\gamma_c + 1 \\ &= (\Delta - 2)(\gamma_c + 2) - (2\Delta - 5) \\ &\leq (\Delta - 2)(\gamma_c + 2) - 1 \\ &< (\Delta - 2)(\gamma_c + 2). \end{aligned}$$

Thus,

$$\frac{k-1}{\gamma_c} < \ln(\Delta - 2)$$

which implies

$$k - 1 \leq \gamma_c \ln(\Delta - 2).$$

This completes the proof of the theorem. ■

The approximation bound $2 + \ln(\Delta - 2)$ in [Theorem 7](#) is slightly better than the approximation bound $2 + \ln \Delta$ derived in [\[30\]](#). Du et al. [\[7\]](#) presented a generalization of *GreedyCDS* which chooses at each iteration a set of up to $2k - 1$ vertices with the maximum gain-cost ratio for some fixed positive integer parameter k . The approximation ratio of this generalization is at most $(1 + \frac{1}{k})(1 + \ln(\Delta - 1))$. This implies that for any $\varepsilon > 0$, there is a $(1 + \varepsilon)(1 + \ln(\Delta - 1))$ -approximation

algorithm for *MCDS*. An interesting observation is that for greedy approximation algorithms with submodular coverage functions, the above generalization cannot lead to better performance ratio.

3.3 Minimum-Weight CDS

The algorithm *GreedyCDS* for *MCDS* augments the partial solution by a single vertex in each iteration. While it can be modified for *MWCDS*, the proof of [Theorem 7](#) cannot be extended to prove a logarithmic approximation bound in the weighted case. This subsection presents a different greedy approximation algorithm [16] for *MWCDS*, which can achieve the logarithmic approximation bound. The main difference is that a group of nodes are selected to augment the partial solution in each iteration. This technique is adapted from the greedy approximation algorithm [21] for Minimum Node-Weighted Steiner Tree.

We begin with the definition of the potential (or coverage deficiency) function. Let B be a set of nodes in V . A *piece* with respect to B is either a vertex not in $N[B]$ or a connected component of $G[B]$. Let $f(B)$ denote the number of pieces with respect to B . Then, B is a CDS if and only if $f(B) = 1$. Let D be the arc-weighted digraph obtained from G by replacing each edge uv in G with two arcs (u, v) and (v, u) of weights $c(u)$ and $c(v)$, respectively. The greedy algorithm maintains a set B of black nodes, which is initially empty. Repeat the following iteration while $f(B) > 1$. Choose an arborescence T in D on some greedy manner, add all internal nodes of T to B , and then update $f(B)$. When $f(B) = 1$, B is output as the CDS. The key ingredient of this algorithm is the greedy strategy used by each iteration to select a proper arborescence T . The detail of this greedy strategy is presented below.

For any arborescence T in D , $I(T)$ denotes the set of all internal nodes in T . Fix a subset B of V with $f(B) > 1$. The *head* of a piece with respect to B is defined to be the vertex in this piece with the smallest ID. The *price* of an arborescence T in D with respect to B is defined to be the ratio of $c(I(T))$ to the number of heads with respect to B contained in T . Ideally, one would greedily wish to use a cheapest (i.e., least priced) arborescence in each iteration. While a cheapest arborescence can be computed easily when $f(B)$ is small, it is hard to be computed in general. In general, a cheapest arborescence is selected among a polynomial number of specially defined candidates. Let $\mathcal{T}(B)$ denote the set of candidates. The construction of $\mathcal{T}(B)$ is described below. For each vertex u , let A_u be the shortest-path spanning arborescence in D rooted at u . Suppose that B is a set of black nodes with $f(B) > 1$. Sort all the $f(B)$ heads in the increasing distance from u in A_u . For each $2 \leq j \leq f(B)$, $T_j(B, u)$ is the minimal arborescence in A_u spanning u and the first j heads. Then, $\mathcal{T}(B)$ consists of $(f(B) - 1)n$ candidates:

$$\mathcal{T}(B) = \{T_j(B, u) : u \in V, 2 \leq j \leq f(B)\}.$$

The greedy algorithm *GreedyWCDS* is described in [Table 7](#).

Next, we derive an approximation bound of *GreedyWCDS*.

Table 7 Outline of the algorithm *GreedyWCDS*

<i>GreedyWCDS</i>
$B \leftarrow \emptyset;$
while $f(B) > 1$,
construct $\mathcal{T}(B)$;
find a cheapest $T \in \mathcal{T}(B)$;
$B \leftarrow B \cup I(T);$
output B .

Theorem 8 *The approximation ratio of GreedyWCDS is at most $2(H(n) - 1)$.*

To prove the above theorem, we first give a lower bound on the “gain” of each candidate arborescence in $\mathcal{T}(B)$ in terms of the reduction on the number of pieces.

Lemma 4 *Suppose that $f(B) > 1$. Then for any $T \in \mathcal{T}(B)$ containing h heads with respect to B ,*

$$f(B) - f(B \cup I(T)) \geq h/2.$$

Proof After the addition of $I(T)$, all pieces with respect to B containing a head in T (and possibly some other pieces with respect to B) get merged into some new (and larger) piece with respect to $B \cup I(T)$. Thus,

$$f(B) - f(B \cup I(T)) \geq h - 1 \geq h/2.$$

So, the lemma holds. ■

The next lemma presents an upper bound on the price of the cheapest candidate in $\mathcal{T}(B)$.

Lemma 5 *Suppose that $f(B) > 1$. Then the price of the cheapest tree in $\mathcal{T}(B)$ is at most $\frac{\text{opt}}{f(B)}$.*

Proof We prove the lemma by a decomposition argument. A spider is an arborescence in D in which all nodes except the root have out-degree at most 2. Initialize T^* to be a spanning arborescence of D whose internal nodes form an optimal solution and j to be 0. Repeat the following iteration while T^* contains at least two heads. Increment j by one. Repeatedly prune the sinks of T^* which are not heads until all sinks are heads. Choose a deepest vertex u in T^* such that the maximal u -arborescence contained in T^* (which is the subgraph of T^* induced by u and all its descendants in T^*) contains at least two heads. Set S_j to be the maximal u -arborescence contained in T^* . By the maximum depth of u , every child arborescence of S_j has exactly one head, and hence, S_j is a spider. Delete S_j from T^* . If T^* still contains one head, then augment the last S_j by the path from the root of S_j to this head.

Let S_1, S_2, \dots, S_q be the sequence of spiders obtained by this construction. Then, they are vertex disjoint. In addition, their union contains all heads. For each $1 \leq j \leq q$, let h_j denote the number of heads contained in S_j . Then,

$$\sum_{j=1}^q h_j = f(B),$$

and

$$\sum_{j=1}^q c(I(S_j)) \leq \text{opt}.$$

Thus,

$$\min_{1 \leq j \leq q} \frac{c(I(S_j))}{h_j} \leq \frac{\sum_{j=1}^q c(I(S_j))}{\sum_{j=1}^q h_j} \leq \frac{\text{opt}}{f(B)}.$$

Let S be the cheapest one among S_1, S_2, \dots, S_q . Then, the price of S is at most $\frac{\text{opt}}{f(B)}$.

Let u be the root of S . We show that $T_\ell(B, u)$ is at least as cheaper as S , from which the lemma follows. Let v_1, v_2, \dots, v_ℓ be the heads in S and $v'_1, v'_2, \dots, v'_\ell$ be the heads in $T_\ell(B, u)$. By treating S as a subgraph of D , we have

$$\begin{aligned} & c(I(T_\ell(B, u))) + (\ell - 1)c(u) \\ & \leq \sum_{i=1}^\ell \text{dist}_D(u, v'_i) \leq \sum_{i=1}^\ell \text{dist}_D(u, v_i) \leq \sum_{i=1}^\ell \text{dist}_S(u, v_i) \\ & = c(I(S)) + (\ell - 1)c(u). \end{aligned}$$

So, $c(I(T_\ell(B, u))) \leq c(I(S))$. As $T_\ell(B, u)$ and S have the same number of heads, $T_\ell(B, u)$ is at least as cheaper as S . ■

Finally, we prove [Theorem 8](#) by applying [Lemmas 4](#) and [5](#). Suppose that the algorithm runs in k iterations. Let $h_0 = n$ which is the number of initial pieces. For any $1 \leq i \leq k$, let T_i be the arborescence selected at iteration i and let ℓ_i be the number of pieces just after iteration i . Let b_i be the cost of T_i and h_i be the number of heads in T_i . Then by [Lemma 4](#), for each $1 \leq i \leq k$,

$$\ell_{i-1} - \ell_i \geq h_i/2,$$

and hence,

$$h_i \leq 2(\ell_{i-1} - \ell_i).$$

By [Lemma 5](#), for each $1 \leq i \leq k$,

$$\frac{b_i}{h_i} \leq \frac{\text{opt}}{\ell_{i-1}}$$

and consequently

$$b_i \leq \frac{h_i}{\ell_{i-1}} \text{opt} \leq 2\text{opt} \frac{\ell_{i-1} - \ell_i}{\ell_{i-1}}.$$

So,

$$\begin{aligned} \sum_{i=1}^k b_i &\leq 2\text{opt} \sum_{i=1}^k \frac{\ell_{i-1} - \ell_i}{\ell_{i-1}} \\ &\leq 2\text{opt} \sum_{i=1}^k \sum_{j=\ell_i+1}^{\ell_{i-1}} \frac{1}{j} \\ &= 2\text{opt} \sum_{j=\ell_k+1}^{\ell_0} \frac{1}{j} \\ &= 2(H(\ell_0) - H(\ell_k)) \text{opt} \\ &= 2(H(n) - 1) \text{opt}. \end{aligned}$$

This completes the proof of [Theorem 8](#).

By choosing more sophisticated pool $\mathcal{T}(B)$ of candidates as introduced in [16], better approximation bounds $1.5 \ln n$ and $(1.35 + \varepsilon) \ln n$ for arbitrary constant $\varepsilon > 0$ can be achieved. The same algorithmic techniques have been applied to other problems including *Minimum Node-Weighted Steiner Tree* [16], *Minimum-Power Spanning Arborescence* [5], and *Minimum Node-Weighted Strongly Connected Dominating Set* [23].

4 Graph Coloring

Let $G = (V, E)$ be an undirected graph. A (vertex) *coloring* of G is an assignment of colors to V satisfying that adjacent vertices are assigned with distinct colors. Equivalently, a vertex coloring corresponds to a partition of V into stable sets. The *chromatic number* of G , denoted by $\chi(G)$, is the smallest number of colors required by any vertex coloring of G . The problem *Minimum Vertex Coloring* seeks a vertex coloring of a given graph with the smallest colors. It is NP-hard in general. It can be cast as *Minimum-Cost Submodular Cover* with the following formulation: The ground set is the collection \mathcal{I} of all stable sets of G . For a subcollection of k stable sets whose union is U , its coverage is $|U|$ and its cost is k . For this formulation, the curvature is one, and the augmentation problem is *Maximum Stable Set* in graphs. Suppose that there is a μ -approximation algorithm \mathcal{A} for such *Maximum Independent Set* problem. The adaptation of the general greedy algorithm

Table 8 Greedy link scheduling

<i>GreedyColoring</i>
$U \leftarrow V;$
$i \leftarrow 0;$
While $U \neq \emptyset$ do
$i \leftarrow i + 1$
$I \leftarrow$ the stable set of $G [U]$ output by \mathcal{A} ;
assign the i th color to all vertices in I ;
$U \leftarrow U \setminus I.$

GreedyMCSC for *Minimum Vertex Coloring*, called *GreedyColoring*, is described in Table 8. By Theorem 5, the approximation ratio of *GreedyColoring* is at most $\mu H(\alpha)$, where α is the independence number of G .

This section presents yet another greedy algorithm of first-fit type which can outperform *GreedyColoring* in many graph classes. We introduce some standard notations and terms. Let $G = (V, E)$ be an undirected graph. For any vertex $v \in V$, $\deg_G(v)$ denotes the degree of v in G , and $N_G(v)$ denotes the set of neighbors of v in G . The minimum degree of G is denoted by $\delta(G)$. Consider a vertex ordering \prec of V . For each $v \in V$, $N_G^\prec(v)$ denotes the set of neighbors of v preceding v in the ordering \prec , and $N_G^\prec[v]$ denotes $\{v\} \cup N_G^\prec(v)$. For any subset U of V , $G[U]$ denotes the subgraph of G induced by U . Suppose that $D = (V, A)$ is digraph. For each $v \in V$, $\deg_D^{\text{in}}(v)$ (respectively, $\deg_D^{\text{out}}(v)$) denotes the in-degree (respectively, out-degree) of v in D , and $N_D^{\text{in}}(v)$ (respectively, $N_D^{\text{out}}(v)$) denotes the set of in-neighbors (respectively, out-neighbors) of v in D . For any subset U of V , $D[U]$ denotes the subgraph of D induced by U .

Given a vertex ordering \prec of V given by $\langle v_1, v_2, \dots, v_n \rangle$, a coloring of V with colors represented by natural numbers can be produced in the following first-fit manner: Assign the color 1 to v_1 . For $i = 2$ up to n , assign to v_i with the smallest color which is not used by any vertex in $N_\prec(v_i)$. Such coloring of V is referred to as the *first-fit coloring* in the ordering \prec . It is easy to verify that the number of colors used by the first-fit coloring in the ordering $\langle v_1, v_2, \dots, v_n \rangle$ is at most $1 + \max_{v \in V} |N_\prec(v)|$. The value $\max_{v \in V} |N_\prec(v)|$ is referred to as the *inductivity* of the ordering \prec . The smallest inductivity of all vertex orderings is referred to as the *inductivity* of G and is denoted by $\delta^*(G)$. It was shown in [25] that

$$\delta^*(G) = \max_{U \subseteq V} \delta(G[U]),$$

and it can be achieved by a special vertex ordering known as *smallest-degree-last ordering*, which can be produced iteratively as follows: Initialize H to G . For $i = n$ down to 1, let v_i be a vertex of the smallest degree in H and delete v_i from H . Then the ordering $\langle v_1, v_2, \dots, v_n \rangle$ is a smallest-degree-last ordering. In the next, we derive the approximation bound of the first-fit coloring in the smallest-degree-last ordering.

For any vertex ordering \prec of V , its *backward local independence number* (BLIN) is defined to be the parameter

$$\max_{v \in V} \max_{I \in \mathcal{I}} |I \cap N_G^<(v)|.$$

The *backward local independence number* (BLIN) of G , denoted by $\alpha^*(G)$, is the smallest BLIN of all vertex orderings of G . An *orientation* of G is a digraph obtained from G by imposing an orientation on each edge of G . For any orientation D of G , its *inward local independence number* (ILIN) is defined to be the parameter

$$\max_{v \in V} \max_{I \in \mathcal{I}} |I \cap N_D^{\text{in}}(v)|.$$

The *inward local independence number* (ILIN) of G , denoted by $\beta^*(G)$, is the smallest ILIN of all orientations of G . The following theorem was proved in [32,37].

Theorem 9 *The first-fit coloring of G in the smallest-degree-last ordering has approximation bound $\min\{\alpha^*(G), 2\beta^*(G)\}$.*

Proof Since the first-fit coloring of G in the smallest-degree-last ordering uses at most $1 + \delta^*(G)$ colors, it is sufficient to show that

$$1 + \delta^*(G) \leq \chi(G) \cdot \min\{\alpha^*(G), 2\beta^*(G)\}.$$

Consider an optimal coloring which partitions V into $\chi(G)$ stable sets $\{I_j : 1 \leq j \leq \chi(G)\}$.

Let \prec be a vertex ordering of G whose BLIN is $\alpha^*(G)$. Then, for any vertex $v \in V$,

$$|N_{\prec}(v)| = \sum_{j=1}^{\chi(G)} |N_{\prec}(v) \cap I_j| \leq (\chi(G) - 1) \alpha^*(G).$$

Hence, the of the ordering \prec is at most $\alpha^*(G)(\chi(G) - 1)$. This implies that

$$\delta^*(G) \leq (\chi(G) - 1) \alpha^*(G).$$

Therefore,

$$1 + \delta^*(G) \leq 1 + (\chi(G) - 1) \alpha^*(G) \leq \chi(G) \cdot \alpha^*(G).$$

Let D be an orientation of G whose ILIN is $\beta^*(G)$ and U be the subset of V such that $\delta^*(G) = \delta(G[U])$. Then, there exists at least one vertex $u \in U$ such that

$$\deg_{D[U]}^{\text{in}}(u) \geq \deg_{D[U]}^{\text{out}}(u).$$

Thus,

$$\delta^*(G) = \delta(G[U]) \leq \deg_{G[U]}(u) = \deg_{D[U]}^{\text{in}}(u) + \deg_{D[U]}^{\text{out}}(u)$$

$$\begin{aligned} &\leq 2 \deg_{D[U]}^{\text{in}}(u) \leq 2 \deg_D^{\text{in}}(u) = 2 \sum_{j=1}^{\chi(G)} |N_D^{\text{in}}(u) \cap I_j| \\ &\leq 2(\chi(G) - 1) \beta^*(G). \end{aligned}$$

So,

$$1 + \delta^*(G) \leq 1 + 2(\chi(G) - 1) \beta^*(G) \leq 2\chi(G) \cdot \beta^*(G).$$

This completes the proof of the theorem. \blacksquare

Theorem 9 has found many applications in the wireless scheduling of node transmissions [37, 38] and link transmissions [32, 36]. These wireless scheduling problems can be formulated as the minimum graph coloring of some conflict graphs, which have constant-bounded BLINs and/or constant-bounded ILINs.

5 Conclusion

This chapter presents a number of general greedy approximation algorithms for a broad class of optimization problems. For many specific problems, a straightforward greedy approximation algorithm can achieve the best known approximation bound. But for some problems, they may have different formulations, and the corresponding greedy approximation algorithms have different approximation bounds as illustrated in Sect. 3.1. Furthermore, some problems have approximation algorithms with better approximation bounds than that of the greedy approximation algorithm. Below are some examples of these problems:

- The problem of maximizing the sum of weighted rank functions of matroids over an arbitrary matroid constraint has a $1/(1-e^{-1})$ -approximation algorithm [4], while the greedy approximation algorithm has an approximation ratio 2 by Theorem 2.
- The problem *Maximum-Weight Stable Set of Unit-Disk Graphs* described in Sect. 2.2 admits a polynomial-time approximation scheme [17, 31], while the greedy approximation algorithm has an approximation ratio 5.
- The two problems *Minimum-Weight Vertex Cover* and *Minimum-Weight Feedback Vertex Set* both have 2-approximation algorithms [2], (Bafna et al., 1994, Approximating feedback vertex set for undirected graphs within ratio 2, unpublished), while the greedy approximation algorithms are logarithmic approximation ratios.

Cross-References

- ▶ [A Unified Approach for Domination Problems on Different Network Topologies](#)
- ▶ [Algorithmic Aspects of Domination in Graphs](#)
- ▶ [Connected Dominating Set in Wireless Networks](#)

- Energy Efficiency in Wireless Networks
- On Coloring Problems
- Variations of Dominating Set Problem

Recommended Reading

1. E. Althaus, G. Calinescu, I. Mandoiu, S. Prasad, N. Tchervenski, A. Zelikovsky, Power efficient range assignment in ad-hoc wireless network. *Wireless Network* **12**(3), 287–299 (2006)
2. R. Bar-Yehuda, S. Even, A local-ratio theorem for approximating the weighted vertex cover problem. *Ann. Discrete Math.* **25**, 27–46 (1985)
3. A. Borchers, D.-Z. Du, The k -Steiner ratio in graphs. *SIAM J. Comput.* **26**(3), 857–869 (1997)
4. G. Calinescu, C. Chekuri, M. Pal, J. Vondrak, Maximizing a submodular set function subject to a matroid constraint. *SIAM J. Comput.* **40**(6), 740–1766 (2011)
5. G. Calinescu, S. Kapoor, A. Olshevsky, A. Zelikovsky, Network lifetime and power assignment in ad hoc wireless networks. *Lect. Note Comput. Sci.* **2832**, 114–126 (2003)
6. G. Cornuejols, M. Fisher, G. Nemhauser, Location of bank accounts to optimize float: an analytic study of exact and approximate algorithms. *Manag. Sci.* **23**, 789–810 (1977)
7. D.-Z. Du, R.L. Graham, P.M. Pardalos, P.-J. Wan, W. Wu, W. Zhao, Analysis of greedy approximations with nonsubmodular potential functions, in *SIAM SODA*, 2008, pp. 167–175
8. X. Du, W. Wu, D.F. Kelley, Approximations for subset interconnection designs. *Theor. Comput. Sci.* **207**(1), 171–180 (1998)
9. J. Edmonds, Matroids, submodular functions and certain polyhedra. in *Proceedings of the Calgary International Conference on Combinatorial Structures and Their Applications*, ed. by R. Guy, H. Hanani, N. Sauer, J. Schoenheim (Gordon and Breach, New York, 1970), pp. 69–87
10. J. Edmonds, Matroids and the greedy algorithm. *Math. Program.* **1**, 127–36 (1971)
11. U. Feige, A threshold of $\ln n$ for approximating set cover. *J. ACM* **45**(4), 634–652 (1998)
12. M.L. Fisher, G.L. Nemhauser, L.A. Wolsey, An analysis of approximations for maximizing submodular set functions - II. *Math. Prog. Study* **8**, 73–87 (1978)
13. T. Fujito, On approximability of the independent/connected edge dominating set problems. *Lect. Note Comput. Sci.* **1974**, 117–126 (2000)
14. T. Fujito, Approximation algorithms for submodular set cover with applications. *IEICE Trans. Inf. Syst.* **E83-D**(3), 480–487 (2000)
15. P.R. Goundan, A.S. Schulz, Revisiting the greedy approach to submodular set function maximization. preprint, (2007), available from optimization-online.org
16. S. Guha, S. Khuller, Improved methods for approximating node weighted Steiner trees and connected dominating sets. *Inform. Comput.* **50**(1), 57–74 (1999)
17. D.S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM* **32**(1), 130–136 (1985)
18. D.S. Hochbaum, A. Pathria, Analysis of the greedy approach in problems of maximum k -coverage. *Nav. Res. Logist.* **45**, 615–627 (1998)
19. T.A. Jenkyns, The efficacy of the “greedy” algorithm, in *Proceedings of 7th South Eastern Conference on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica, Winnipeg, 1976, pp. 341–350
20. L.M. Kirousis, E. Kranakis, D. Krizanc, A. Pelc, Power consumption in packet radio networks. *Theor. Comput. Sci.* **243**, 289–305 (2000)
21. P. Klein, R. Ravi, A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. Algorithm* **19**(1), 104–115 (1995)
22. B. Korte, D. Hausmann, An analysis of the greedy algorithm for independence systems. *Ann. Discrete Math.* **2**, 65–74 (1978)
23. D. Li, H. Du, P.-J. Wan, X. Gao, Z. Zhang, W. Wu, Construction of strongly connected dominating sets in asymmetric multihop wireless networks. *Theor. Comput. Sci.* **410**(8–10), 661–669 (2009)

24. M. Li, S. Huang, X. Sun, X. Huang, Performance evaluation for energy efficient topologic control in ad hoc wireless networks. *Theor. Comput. Sci.* **326**(1–3), 399–408 (2004)
25. D.W. Matula, L.L. Beck, Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM* **30**(3), 417–427 (1983)
26. G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of approximations for maximizing submodular set functions - I. *Math. Prog.* **14**, 265–294 (1978)
27. G.L. Nemhauser, L.A. Wolsey, Best algorithms for approximating the maximum of a submodular set function. *Math. Oper. Res.* **3**(3), 177–188 (1978)
28. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988)
29. R. Rado, A theorem on independence relations. *Q. J. Math.* **13**, 83–89 (1942)
30. L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, K. Ko, A Greedy approximation for minimum connected dominating sets. *Theor. Comput. Sci.* **329**, 325–330 (2004)
31. E.J. van Leeuwen, Better Approximation schemes for disk graphs. *Proc. SWAT Springer Lect. Note Comput. Sci.* **4059**, 316–327 (2006)
32. P.-J. Wan, Multiflows in multihop wireless networks. in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, New Orleans, LA, USA, pp. 85–94 (2009)
33. Y. Wan, G. Chert, Y. Xu, A note on the minimum label spanning tree. *Inform. Process. Lett.* **84**(2), 99–101 (2002)
34. P.-J. Wan, D.-Z. Du, P. Pardalos, W. Wu, Greedy approximations for minimum submodular cover with submodular cost. *Comput. Optim. Appl.* **45**(2), 463–474 (2010)
35. P.-J. Wan, L.-W. Liu, Maximal throughput in wavelength-routed optical networks, in *DIMACS Workshop on Optical Networks*, 1998
36. P.-J. Wan, C. Ma, Z. Wang, B. Xu, M. Li, X. Jia, Weighted wireless link scheduling without information of positions and interference/communication radii, in *IEEE INFOCOM*, 2011
37. P.-J. Wan, Z. Wang, H.W. Du, S.C.-H. Huang, Z. Y. Wan, First-fit scheduling for beaconing in multihop wireless networks, in *IEEE INFOCOM*, 2010
38. P.-J. Wan, C.-W. Yi, X. Jia, D. Kim, Approximation algorithms for conflict-free channel assignment in wireless ad hoc Networks. *Wiley J. Wireless Comm. Mobile Comput.* **6**(2), 201–211 (2006)
39. L.A. Wolsey, An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica* **4**(2), 385–393 (1982)
40. X. Xu, Y. Wang, H. Du, P.-J. Wan, F. Zou, X. Li, W. Wu, Approximations for node-weighted Steiner tree in unit disk graphs. *J. Optim. Lett.* **4**(3), 405–416 (2010)

Hardness and Approximation of Network Vulnerability

My T. Thai, Thang N. Dinh and Yilin Shen

Contents

1	Introduction	1632
2	Hardness Results.....	1634
2.1	NP-Completeness of Critical Edge Disruptor.....	1634
2.2	NP-Completeness of Critical Vertex Disruptor.....	1639
2.3	NP-Completeness of β -Edge Disruptor.....	1645
2.4	Hardness of β -Vertex Disruptor.....	1648
3	Approximation Algorithm for β -Edge Disruptor.....	1648
3.1	Balanced Tree Decomposition.....	1649
3.2	Pseudo-approximation Algorithm and Analysis.....	1650
4	Approximation Algorithm for β -Vertex Disruptor.....	1654
4.1	Algorithm Description.....	1655
4.2	Theoretical Analysis.....	1657
5	A Second Look.....	1658
5.1	$O(\sqrt{\log n})$ Pseudo-approximation for β -Edge Disruptor.....	1659
5.2	$O(\sqrt{\log n})$ Pseudo-approximation for β -Vertex Disruptor.....	1662
6	Literature.....	1664
7	Conclusion.....	1665
	Recommended Reading.....	1665

Abstract

Assessing network vulnerability is a central research topic to understand networks structures, thus providing an efficient way to protect them from attacks and other disruptive events. Existing vulnerability assessments mainly focus on investigating the inhomogeneous properties of graph elements, node degree, for example; however, these measures and the corresponding heuristic solutions cannot either provide an accurate evaluation over general network topologies

M.T. Thai (✉) • T.N. Dinh • Y. Shen

Department of Computer Science, Information Science and Engineering, University of Florida,
Gainesville, FL, USA

e-mail: mythai@cise.ufl.edu; tdinh@cise.ufl.edu; yshen@cise.ufl.edu

or performance guarantees to large-scale networks. To this end, this chapter introduces two new optimization models to quantify the network vulnerability, which aim to discover the set of *key node/edge disruptors*, whose removal results in the maximum decline of the global pairwise connectivity. Results presented in this chapter consist of the NP-completeness and inapproximability proofs of these problems along with pseudo-approximation algorithms.

1 Introduction

Complex network systems such as the Internet, WWW, MANETs, and the power grids are often greatly affected by several uncertain factors, including external natural or man-made interferences (e.g., severe weather, enemy attacks, malicious attacks). Additionally, they are extremely vulnerable to attacks, that is, the failures of a few *critical nodes* (*links*) which play a vital role in maintaining the network's functionality can break down its operation [1].

In order to assess the network vulnerability, we need to address several fundamental questions such as the following: How do we quantitatively measure the vulnerability degree of a network? What are the key nodes (*links*) that play a vital role in maintaining the network's functionality? What is the minimum number of nodes (*links*) do we need to take down in order to break down an adversarial network?

Despite numerous methods have been investigated on the network vulnerability, they neither reveal the global damage done on the network when multiple nodes (*links*) fail simultaneously nor correctly identify the key nodes who play a vital role in maintaining the network's functionality [2–8]. (More details of related work are discussed later in Sect. 6.) In this chapter, two optimization models along with their four related problems to assess the network vulnerability are discussed as follows:

Model 1: Critical Vertex (Edge) Disruptor [CVD (CED)] : Given a graph $G = (V, E)$ representing a network (where V is a set of nodes and E is a set of links in the network) and a positive integer $k < |V|$ ($k < |E|$), determine a set of vertices (*edges*) $S \subset V$ ($S \subset E$) with $|S| = k$ so as to minimize the total pairwise connectivity in the induced graph $G[V \setminus S]$ ($G[E \setminus S]$), obtained by removing S from G , where the total pairwise connectivity in G is defined as the total number of connected pairs of vertices in G . A pair of nodes is connected if there is a path between them in graph G .

Note that E is a set of links where links can be physical links or logical links between two endpoints. For example, if there is a communication between nodes u and v in the network, there will be an edge (u, v) in the representing graph G . If G is used to represent the functional dependency between each node in a network, then there is an edge (u, v) in G iff there is a functional relationship between u and v in the network. Therefore, the connectivity discussed in this chapter is not simply a physical connectivity in a given network.

Clearly, this model aims to discover vertices (edges) whose removal maximizes the network fragmented and maximumly destroys the network. The ratio between the size of S and the number of pairwise disconnectivity in $G[V \setminus S]$ determines the degree of vulnerability of G , that is, the smaller this ratio is, the more vulnerable the network is. Thus, this model represents a new paradigm for quantitatively characterizing the vulnerability of an underlying network.

Model 2. β -Vertex (Edge) Disruptor [β -VD (β -ED)] : Given a graph $G = (V, E)$ and $0 \leq \beta < 1$, find a subset of vertices (edges) $S \subseteq V$ ($S \subseteq E$) with the minimum cardinality so that the total pairwise connectivity of $G[V \setminus S]$ ($G[E \setminus S]$) is at most $\beta \binom{n}{2}$.

This model also reveals the vulnerability degree of the networks. That is, the more key nodes (edges) there are (i.e., the more nodes (edges) whose deletion is required to meet the requirement of pairwise connectivity), the less vulnerability the network is. Conversely, the fewer the key nodes (edges), the more vulnerability the network will be to the attacks.

Model 1 can be used to maximally disrupt the network with a given cost, whereas Model 2 can be used in the study of breaking down an adversarial social network to a certain degree with the minimum cost. For example, this concept can be applied to destroy, that is, arrest, a small number of key individuals in an adversarial social network (e.g., terrorist networks) in order to maximally disrupt the networks, ability to deploy a coordinated attack. The set of nodes S obtained from the two models present critical nodes in networks, such as a key person, a commander of military networks, or a key computer components in computer networks.

Moreover, the two models present a deeper meaning and greater potentials on the study of multiple disruption levels (different values of k and β). Several recent studies in the context of wireless networks have aimed to discover the nodes/edges whose removal disconnects the network, regardless of how disconnected it is [9–11]. However, it is not reasonable to limit the possible disruption to only disconnecting the graph, ignoring how fragmented it is since the giant connected component still exists and the network may function well. For example, a scale-free network can tolerate high random failure rates [1], since the destructions to boundary vertices may not significantly decline the network connectivity even though the whole graph becomes disconnected. In addition, different disruption levels may require different sets of disruptors on which these two models can differentiate, whereas existing methods cannot. For example, the node centrality method always returns a set of nodes with nonincreasing degrees regardless of the disruption level.

The study of these models also finds applications in several other fields, such as network immunization, offering control of epidemic outbreaks, and containment of viruses. For example, instead of an expensive mass vaccination, we can vaccinate a small number of individuals (corresponding to key nodes of a network). The immunized nodes cannot propagate the virus, thus minimizing the overall transmissibility of the virus.

This chapter presents the hardness complexity of the four problems, CED, CVD, β -ED, and β -VD, along with their pseudo-approximation solutions. The results are discussed in the following sections.

2 Hardness Results

2.1 NP-Completeness of Critical Edge Disruptor

In this section, the NP-completeness of CED in general graphs and in unit disk graphs (UDGs) is shown. Furthermore, this section presents an NP-completeness of CED in power-law graphs (PLGs), strengthening its intractability. UDGs are often used in the networking research field to model the homogeneous wireless networks where all nodes have the same transmission range where PLGs represent a class of complex networks whose nodes follow the power-law distribution such as the Internet and social and biological networks [12]. That is, the fraction of nodes in the network having degree k is proportional to $k^{-\beta}$, where β is a parameter whose value is typically in the range $2 < \beta < 3$.

The hardness proof of CED in general graphs uses a reduction from the p -multiway cut problem, which is defined as follows:

Definition 1 (p -Multiway Cut) Given an undirected graph $G = (V, E)$ and a subset of terminals $T \subseteq V$ with $|T| = p$, find a minimum set of edges of G whose removal disconnects all vertices in T . That is, each vertex in T belongs to each connected component.

Note that p -multiway cut is NP-complete for any $p \geq 3$ according to E. Dahlhaus et al. [13].

Fact 1 *Given a graph $G = (V, E)$ with p -connected components, the total pairwise connectivity of G is lower bounded by $p\binom{\lfloor n/p \rfloor}{2}$ and upper bounded by $\binom{n-p+1}{2}$ where $n = |V|$.*

Theorem 1 *The critical edge disruptor (CED) problem is NP-complete.*

Proof Consider the decision version of CED that asks whether $G = (V, E)$ contains a set of edges $S \subset E$ of size k such that the total pairwise connectivity of $G[E \setminus S]$ is at most c for a given positive integer c .

It is easy to see that $\text{CED} \in \text{NP}$ since a nondeterministic algorithm needs only guess a subset S of edges and checks in polynomial time whether S has the appropriate size k and the total pairwise connectivity of $G[E \setminus S]$ is at most c by using the depth-first search.

The reduction from the 3 -*multiway cut* (3-MC) problem to CED is used to prove its NP-hardness. Let an undirected graph $G = (V, E)$ where $|V| = n$, a set $T \subseteq V$ of three terminals and a positive integer $k \leq |E|$ be any instance of 3-MC. The graph $G' = (V', E')$ is constructed as follows: For each terminal $v \in T$, $l = n^2$, more vertices are added onto v to construct a clique of size $n^2 + 1$, as illustrated in Fig. 1. We now show that there is a 3-MC cut of size k in G iff G' has a CED

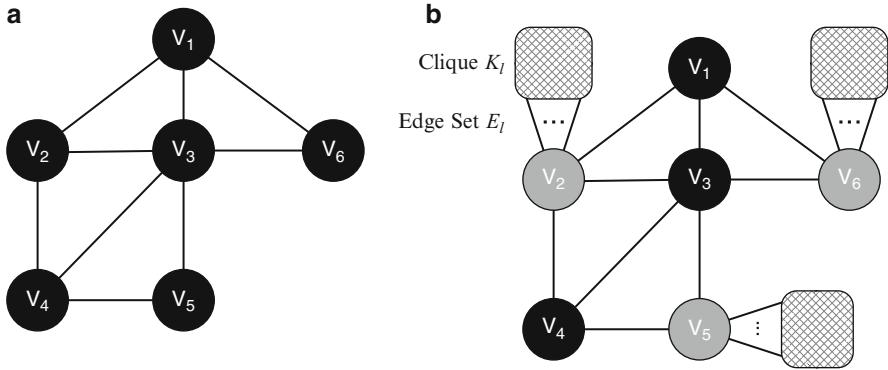


Fig. 1 The reduction from 3-multiway cut instance to a CED instance. (a) Graph G with $T = \{v_2, v_5, v_6\}$. (b) Reduction from 3-multiway cut to CED

S of size k such that the pairwise connectivity of $G[E \setminus S]$ is at most c where $2\binom{n^2+1}{2} + \binom{n^2+n-2}{2} < c < \binom{2n^2+2}{2}$.

First, suppose $S \subseteq E$ is a 3-MC for G with $|S| \leq k$. By our construction and according to Fact 1, the total pairwise connectivity of $G'[E' \setminus S]$ is upper bounded by

$$2\binom{n^2+1}{2} + \binom{n^2+n-2}{2} \quad (1)$$

which is less than c . Thus, S is also a CED of G' .

Conversely, suppose that $S \subseteq E'$ with $|S| = k \leq |E|$ is a CED of G' , that is, the total pairwise connectivity of $G'[E' \setminus S] \leq c$. We show that S indeed is also a 3-MC of G . Note that $S \cap (E' \setminus E) = \emptyset$, that is, for any $e \in S$, e cannot be an edge in the constructed cliques since removing k edges cannot disconnect the cliques; thus, the total pairwise connectivity is $\binom{n+3n^2}{2}$, which is larger than c . Therefore, $S \subseteq E$. Now, we further state that set S also disconnects all the three terminals. Assume that it is not, then the two cliques of these two terminals are connected; thus, the total pairwise connectivity is at least $\binom{2n^2+2}{2} > c$, contradicting to the fact the total pairwise connectivity of $G'[E' \setminus S] \leq c$. \square

We now show that CED still remains NP-complete in UDGs by reducing from the planar independent set (PIS) with maximum degree 3. Given a planar graph $G = (V, E)$, the PIS problem asks us to find a subset $S \subseteq V$ with the maximum size such that no two vertices in S are adjacent.

Lemma 1 (L.G. Valiant [14]) *A planar graph G with maximum degree 4 can be embedded in the plane using $O(|V|)$ area in such a way that its vertices are at integer coordinates and its edges are drawn so that they are made up of line segments of the form $x = i$ or $y = j$, for integers i and j .*

Theorem 2 *The critical edge disruptor (CED) problem is NP-complete in UDGs.*

Proof Consider the decision version of CED that asks whether a UDG $G = (V, E)$ contains a set of edges $S \subset E$ of size k' such that the pairwise connectivity in the induced graph $G[E \setminus S]$ is at most c for a given positive integer c . Clearly, CED \in NP in UDGs.

To prove that CED on UDGs is NP-hard, we reduce the planar independent set (PIS) with maximum degree 3 to it. Let a planar graph $G = (V, E)$ where $|V| = n$ with maximum degree 3 and a positive integer $k \leq n$ be an arbitrary instance of PIS. We must construct in polynomial time a UDG $G' = (V', E')$ and positive integers k' and c such that G has an independent set of size k iff $G' = (V', E')$ has a CED of size k' where the pairwise connectivity of G' after removing this CED is at most c .

The construction of G' is as follows: Given a planar graph G , we first embed it into the plane according to [Lemma 1](#). For each node $v_i \in V$, convert it to a triangle $T_{v_i} = \{v_{i1}, v_{i2}, v_{i3}\}$ (dark gray triangles in [Fig. 2b](#)). Each edge in T_{v_i} is set to be a unit length $r = \frac{\sqrt{3}}{3}$. Since each node v_i in G has at most three neighbors, say x, y , and z , we attach these neighbors to v_{i1}, v_{i2} , and v_{i3} . That is, three edges $(v_i, x), (v_i, y)$, and (v_i, z) in G become $(v_{i1}, x), (v_{i2}, y)$, and (v_{i3}, z) . If the degree is less than 3, we only need to arbitrarily choose one or two from them to attach. Afterwards, for each embedded edge $e_{uv} = (u, v)$, we divide it into $|e_{uv}|$ “pieces” according to its length and replace each piece by a concatenation of two triangles as shown in [Fig. 2b](#). Thus, G' is composed of $2 \sum_{u,v \in V} |e_{uv}| K_3$ cliques where $|e_{uv}|$ is the length of edges between node u and v after embedded onto the plane. Note that G' is a unit disk graph with radius $r = \frac{\sqrt{3}}{3}$. Finally, we set $k' = 2 \sum_{u,v \in V} |e_{uv}| + 3(n - k)$ and $c = 3(\sum_{u,v \in V} |e_{uv}| + k)$.

First, suppose $S \subseteq V$ is an independent set of G with $|S| = k$. For each piece in all edges of G , we remove two edges from two concatenation triangles such that each piece has one triangle left. After doing this, all triangles T_{v_i} corresponding to the node $v_i \in S$ will be disconnected from other nodes in G' . For each node $v_i \notin S$, we remove all edges in the corresponding triangles T_{v_i} in G' . Consider a set S' consisting of all the removed edges (2 edges per piece and 3 edges for each T_{v_i} where $v_i \notin S$). Then clearly $|S'| = k'|$ and the pairwise connectivity in G' after removing S' is c . Therefore, S' is a CED of G' .

Conversely, suppose that $S' \subset V'$ with $|S'| = k'$ is a CED of G' , that is, the pairwise connectivity of $G'[E' \setminus S']$ is at most c . First, it is easy to see that the pairwise connectivity will increase by $\sum_{u,v \in V} |e_{uv}|/2$ if two triangles corresponding to every other pieces are connected since there is one more connected node pair. Thus, either one of the two triangles for each piece in G' will be disconnected. It implies that two edges will be removed from each piece. Apart from this, in order to reduce the pairwise connectivity to c , we remove $3(n - k)$ more edges to avoid the connectivity between the nodes in different triangles. Note that for each triangle T_{v_i} , either all three edges are removed or none since the removal of edges in a

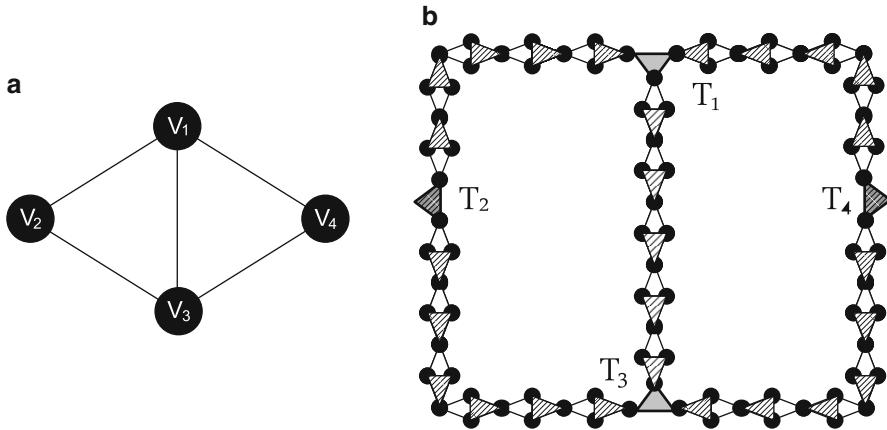


Fig. 2 The reduction from a planar independent set instance to a CED instance. **(a)** An instance G . **(b)** A reduced graph G'

connected component does not help to reduce the pairwise connectivity if it cannot disconnect the component. Some edges belonging to K_2 are exchangeable with the nodes in other triangles within polynomial time. This implies the nodes v_i (in V) corresponding to T_{v_i} which has all their edges remained is an independent set of G . Since the number of T_{v_i} is k , the IS also has size k . \square

Now our attention is turned to the study of CED's NP-completeness in power-law graphs. The $P(\alpha, \beta)$ model mentioned in [15] is used to describe a power-law graph. In this model, the number of vertices of degree d is $\lfloor \frac{e^\alpha}{d^\beta} \rfloor$, where e^α is the normalization factor. To show CED's hardness, we reduce from the clique separator (CS) problem, which is defined as follows:

Definition 2 (Clique Separator (CS)) Given a graph $G = (V, E)$, find a set of edges S such that the induced graph $G[E \setminus S]$ has the connected components to be cliques and each clique has size at least 3. A subgraph of G is called a clique iff all its vertices are pairwise adjacent. The CS problem asks us to find such a clique separator with the minimum size.

Before proceeding, let us first present the following fundamental lemma, which states that a PLG can be constructed from any given simple graph by choosing an appropriate value α :

Lemma 2 (Ferrante et al. [16]) Let $G_1 = (V_1, E_1)$ be a simple graph with n nodes and $\beta \geq 1$. For $\alpha \geq \max\{4\beta, \beta \log n + \log(n+1)\}$, we can construct a power-law

graph $G = G_1 \cup G_2$ with exponential factor β and the number of nodes $e^\alpha \zeta(\beta)$ by constructing a bipartite G_2 as a maximal component in G . Here $\zeta(\gamma) = \sum_{i=1}^{\infty} \frac{1}{i^\gamma}$ is the Riemann zeta function.

Since the hardness proof of CED is based on the reduction from CS, the hardness of CS must be investigated first as follows:

Lemma 3 *The CS problem is NP-hard.*

Proof To prove that CS is NP-hard, we reduce the planar independent set (PIS) problem with maximum degree 3 to it. By using the same reduction in [Theorem 2](#), it can be shown that G has an independent set of size k iff G' has a CS of size $k' = 2 \sum_{u,v \in V} |e_{uv}| + 3(n - k)$.

First, suppose $S \subseteq V$ is an independent set of G with $|S| = k$. To disconnect the triangles from other nodes, two edges from two concatenation triangles for each piece are removed. For each T_{v_i} where $v_i \in S$, since all three nodes will not overlap with other cliques, the triangle T_{v_i} will be selected as a connected component of size 3. For the other T_{v_i} where $v_i \notin S$, T_{v_i} cannot be selected as a clique since at least one of its endpoints overlaps with other triangle. Thus, the removed edges have size k' . And after removing these edges, the obtained connected components are cliques of size 3. That said, the removed edges set is a CS of G' .

Conversely, suppose that $S' \subseteq V'$ is a CS of G' with $S' = k'$. For each piece, at least two edges have to be removed to disconnect the triangle from other nodes. For each T_{v_i} , its edges belongs to S' if at least one has been used by the other triangles. Since $k' = 2 \sum_{u,v \in V} |e_{uv}| + 3(n - k)$, it is easily to modify S' to be an independent set of G . \square

Theorem 3 *The critical edge disruptor (CED) problem is NP-complete in power-law graphs.*

Proof Again, consider the decision version of CED that asks whether a power-law graph $G = (V, E)$ contains a set of links $S \subset V$ of size k' such that the pairwise connectivity in $G[E \setminus S]$ is at most c for a given positive integer c .

To prove that CED in PLGs is NP-hard, we reduce the clique separator (CS) to it. Given G , a power-law graph $G' = G \cup G_b$ where the bipartite graph $G_b = (U_b, V_b; E_b)$ is a maximal component in G' is constructed according to [Lemma 2](#). We show that there is a CS of size k in G iff G' has a CED S' of size k' such that the pairwise connectivity of $G'[E' \setminus S']$ is at most c , where $k' = k + |E_b| - |M_b|$ and $c = |E| - k + |M_b|$. Here M_b denotes the set of edges in the maximum matching of G_b .

First, suppose $S \subseteq V$ is a clique separator of G with $|S| = k$. The total pairwise connectivity of $G[E \setminus S]$ is equal to $|E| - k$ since all components in this graph are cliques. Since the maximum matching on G_b can be found in polynomial time using Hopcroft-Karp algorithm [17], the pairwise connectivity of G' is c after removing additional $E_b \setminus M_b$ edges. Therefore, the set $S' = S \cup (E_b \setminus M_b)$ with size $k' = k + |E_b| - |M_b|$ is a CED of G' .

Conversely, suppose that $S' \subset V'$ is a CED of G' with size k' . Let $S' = A \cup S_b$, where A and S_b are the removed edges (in S') in G and G_b , respectively. We show that $|S_b| = |E_b| - |M_b|$. If $|S_b| < |E_b| - |M_b|$, the pairwise connectivity of G_b is increased at least two when adding one more edge into the maximum matching. On the other hand, the removal of l edges on G can reduce the pairwise connectivity at most l after removing the CS k . Therefore, we have $|S_b| \geq |E_b| - |M_b|$ in order to maximally reduce the number of pairwise connectivity. If $|S_b| > |E_b| - |M_b|$, the pairwise connectivity of G_b is reduced by one when removing one more edge from the maximum matching. Meanwhile, an edge added into the residual graph of G will increase the pairwise connectivity at least one if it connects to two independent nodes and at least 3 if it has one endpoint belonging to some component in the residual graph of G . Therefore, maximally reducing the number of pairwise connectivity requires $|S_b| \leq |E_b| - |M_b|$. Combining both conditions, we have $S_b = |E_b| - |M_b|$. Thus, $|A| = k$, which should be a CS of G in order for $G[E \setminus A]$ having the pairwise connectivity of size $|E| - k$. \square

2.2 NP-Completeness of Critical Vertex Disruptor

In this section, the NP-completeness of CVD on both UDGs and PLGs are shown.

Definition 3 (Minimum Vertex Cover) Given a graph $G = (V, E)$, find a subset $C \subseteq V$ with the minimum size such that for each edge in E at least one end vertex belongs to C .

Lemma 4 (Garey et al. [18]) *Minimum vertex cover remains NP-complete on planar graphs with maximum degree 3.*

Lemma 5 *For any planar graph $G = (V, E)$ with maximum degree 3, it can be embedded into a UDG with the radius $1/2$ such that there exists an independent-space $\Gamma(v)$ for each vertex $v \in V$ satisfying $\Gamma(v) \neq \emptyset$ and any new vertices in $\Gamma(v)$ will be incident to v and independent from all other vertices.*

Proof After mapping $G = (V, E)$ into an integer coordinates based on Lemma 1, a $G_u = (V_u, E_u)$ UDG can be constructed as follows: Each edge $e = (v_i, v_j) \in E$ is replaced by a path $(v_i, w_1, w_2, \dots, w_{2p_e}, v_j)$ where p_e is the length of e and the radius r is set to $1/2$. Now, we prove that such independent-space $\Gamma(v)$ exists for each $v \in V$. Since G has the maximum degree 3, that is, for each vertex $v \in V$, there are at most 3 neighbors (in the form of w_i) in V_u . Thus, there exists a nonempty space $\Gamma(v)$ for each v such that the maximum distance between the vertices in this area to vertex v is at most $1/2$ and the minimum distance between such vertices to all other vertices is larger than $1/2$. As shown in Fig. 3, the shadow area corresponds to $\Gamma(v_1)$. \square

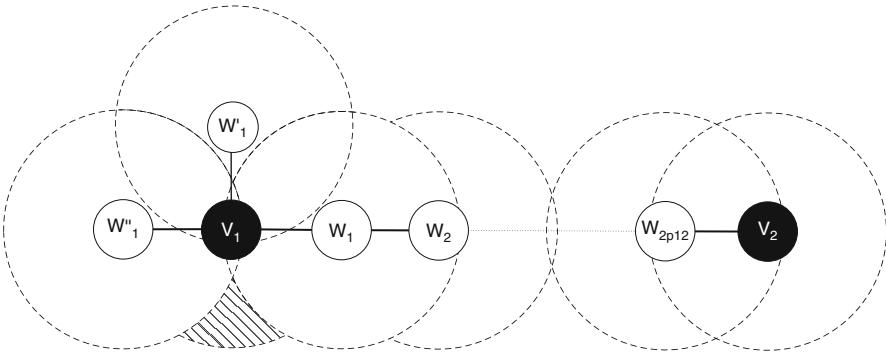


Fig. 3 An independent-space $\Gamma(v_1)$

Theorem 4 *The critical vertex disruptor (CVD) problem is NP-complete in UDGs.*

Proof Consider the decision version of CVD that asks whether a UDG $G = (V, E)$ contains a set of vertices $S \subset V$ of size k' such that the total pairwise connectivity of $G[V \setminus S]$ is at most c for a given positive integer c .

Again, it is easy to see that $\text{CVD} \in \text{NP}$.

Now, to prove that CVD is in NP-hard, we reduce the planar vertex cover (PVC) with maximum degree 3 to it. Let a planar graph with maximum degree 3 $G = (V, E)$ where $|V| = n$ and a positive integer $k \leq n$ be an arbitrary instance of PVC. We must construct in polynomial time a UDG $G' = (V', E')$ and positive integers k' and c such that G has a vertex cover of size at most k iff $G' = (V', E')$ has a CVD of size k' and the pairwise connectivity of G' after removing this CVD is at most c .

Such a construction is described as follows: Given G , construct a G_u according to Lemma 5. On G_u , for each independent-space $\Gamma(v_i)$, a new gray vertex u_i is added such that there is an edge only between v_i and u_i to obtain G' as shown in Fig. 4. Finally, set $k' = k + \sum_{e \in E(G)} p_e$ and $c = n - k$.

First, suppose $S \subseteq V$ is a vertex cover of G with $|S| \leq k$. It is easy to verify that G has a vertex cover S with size at most k iff G_u has a vertex cover S_u with size at most $k + \sum_{e \in E(G)} p_e$. That is, S_u consists of S and a half of nodes w_i on each path $(v_i, w_1, \dots, w_{2p_{(v_i, v_j)}}, v_j)$. We show that S_u is also a CVD of G' . Removing S_u from G' returns all disjoint edges (v_i, u_i) left where $v_i \notin S$. Thus, the pairwise connectivity is $n - k$, which is equal to c .

Conversely, suppose that $S' \subset V'$ with $|S'| = k'$ is a CVD of G' , that is, the total pairwise connectivity of $G'[V' \setminus S']$ is at most c . If $u_i \in S'$, that is, S' contain a gray node, replacing u_i with any v_i or w_i (black or white nodes) will further decrease the pairwise connectivity. Thus, it can be assumed that S' does not contain any gray node. Since $|S'| = k + \sum_{e \in E(G)} p_e$, it is easily to modify the set S' to be a vertex cover of G_u . In this case, the total pairwise connectivity is at most $c = n - k$. Thus, $S' \cap V$ is a VC of G . \square

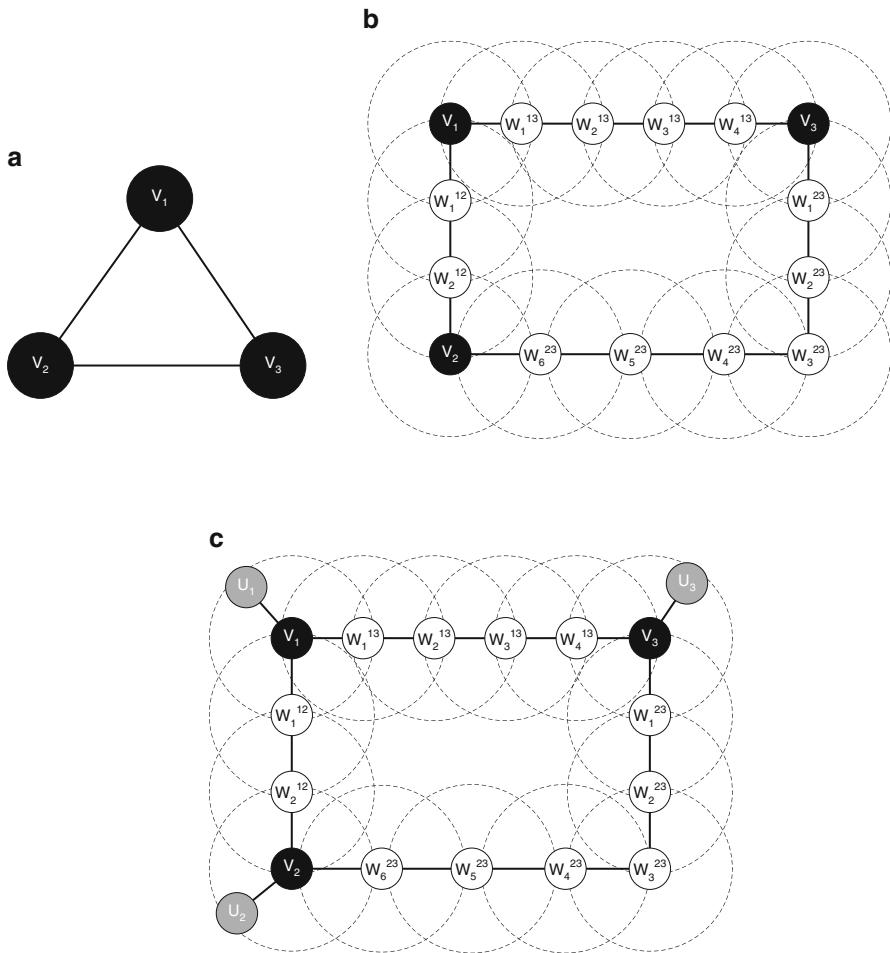


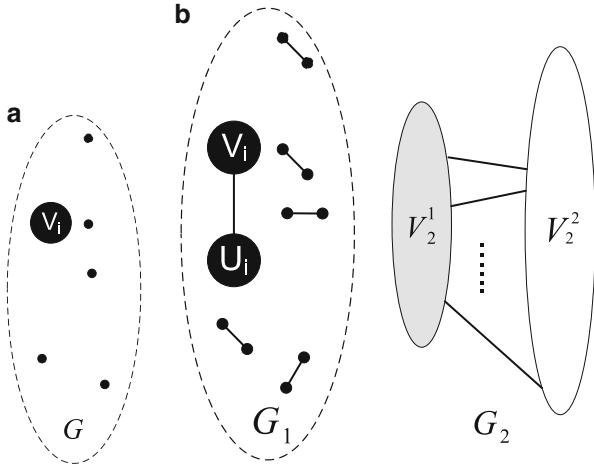
Fig. 4 The reduction for a planar vertex cover instance to a CVD instance. (a) A planar graph G (b) Embed G onto a UDG G_u . (c) Create G' by adding gray modes at independent-spaces

Theorem 5 *The critical vertex disruptor (CVD) problem is NP-complete in power-law graphs.*

Proof Again, consider the decision version of CVD that asks whether a power-law graph $G = (V, E)$ contains a set of nodes $S \subset V$ of size k' such that the pairwise connectivity in $G[V \setminus S]$ is at most c for a given positive integer c .

Now, to prove that CVD in power-law graphs is *NP*-hard, we reduce the vertex cover (VC) to it. Let a graph $G = (V, E)$ where $|V| = n$ and a positive integer $k \leq n$ be any instance of VC. A power-law graph $G' = (V', E')$ is constructed

Fig. 5 The reduction from a vertex cover instance to CVD instance in power-law graphs. For simplicity, all edges in G are omitted. (a) An instance G . (b) Reduced graph $G' = G_1 \cup G_2$



as follows. First, for each node $v_i \in V$, one additional node u_i is added onto it, denoted by G_1 where $V_1 = V \cup U$ and $U = \{u_i\}$. Then according to Lemma 2, a power-law graph $G' = (V', E')$ can be constructed by embedding G_1 and a bipartite graph $G_2 = (V_2^1, V_2^2; E_2)$ where V_2^1 and V_2^2 are two sets of disjoint nodes in G_2 and $\alpha \geq \max\{4\beta, \beta \log(2n) + \log(2n + 1)\}$ with some specific β . Note that V_2^1 and V_2^2 are marked gray and white separately as shown in Fig. 5. We show that there is a VC of size k in G iff G' has a CVD S' of size k' such that the pairwise connectivity of $G'[V' \setminus S']$ is at most c , where $k' = k + \min\{|V_2^1|, |V_2^2|\}$ and $c = n - k$.

First, suppose $S \subseteq V$ is a vertex cover of G with $|S| = k$. Therefore, G has a vertex cover S of size k iff $G \cup G_2$ has a vertex cover S' of size $k + \min\{|V_2^1|, |V_2^2|\}$ since VC is polynomially solvable in any bipartite graphs according to König's theorem [19]. Then after removing S' from G' , only all disjoint links (v_i, u_i) are left where $v_i \notin S$. Therefore, the pairwise connectivity in G' is $n - k$, which is equal to c . That said, S' is a CVD of G' .

Conversely, suppose that $S' \subset V'$ with $|S'| = k'$ is a CVD of G' , that is, the total pairwise connectivity of $G'[V' \setminus S']$ is at most c . First, if $u_i \in S'$, this u_i can be replaced with any v_i without increasing the pairwise connectivity. Since $|S'| = k' = k + \min\{|V_2^1|, |V_2^2|\}$, it is easily to modify the S' to be a vertex cover of $G \cup G_2$, where the total pairwise connectivity on G' is at most $c = n - k$. Thus, $S' \cap V$ is a VC of G . \square

2.2.1 Inapproximability

As the CVD is NP-complete, one will question how tightly one can approximate the solution, leading to the theory of inapproximability. The inapproximability factor gives us the lower bound of near-optimal solution with theoretical performance guarantee. That said, no one can design an approximation algorithm

with a better ratio than the inapproximability factor. In this section, the inapproximability of CVD is further investigated in general graphs by showing the gap-preserving reduction from maximum clique problem, which is defined as follows:

Definition 4 (Maximum Clique) Given a graph $G = (V, E)$, find a clique of maximum size.

Theorem 6 *The CVD problem is NP-hard to be approximated within $\Theta\left(\frac{k}{n^\epsilon}\right)$ for any $\epsilon < 1 - \log_n 2$ in general graphs unless $P = NP$.*

Proof In the proof, a gap-preserving reduction from maximum clique problem to CVD is shown. According to Håstad [20], maximum clique is proven NP-hard to be approximated within $n^{1-\epsilon}$. Let a graph $G = (V, E)$ and a positive integer $k \leq |V| = n$ be an instance of maximum clique problem (MC). Now we construct the graph $G' = (V', E')$ as follows: Firstly, a complement graph $\bar{G} = (V, \bar{E})$ where $\bar{E} = \{(v_i, v_j) | (v_i, v_j) \notin E\}$ is constructed. Next, for each vertex v in \bar{G} , $l \leq n^{1-\epsilon} - 1$, more nodes are added to construct a clique of size $l + 1$ and obtain G' , as shown in Fig. 6. Let ϕ be a feasible solution of MC on G and φ be a feasible solution of CVD on G' where $|\varphi| = n - k$. Define $c = (n - k)\binom{l}{2} + k\binom{l+1}{2}$; the completeness and soundness are shown respectively as follows:

Completeness: In this step, we need to show that if $OPT(\phi) = k$ then $OPT(\varphi) = c$

Let $OPT(\phi)$ be the maximum clique in graph G , then $OPT(\phi)$ is the maximum independent set in \bar{G} . Since the minimum vertex cover and the maximum independent set are complementary on the same graph, removing the set of nodes $V \setminus OPT(\phi)$ in \bar{G} will result into a set of only isolated nodes. Thus, the pairwise connectivity in G' after removing such $n - k$ critical nodes is $(n - k)\binom{l}{2} + k\binom{l+1}{2}$, which is equal to c . It implies that $OPT(\varphi) = c$.

Soundness: In this step, we need to show that if $OPT(\phi) < \frac{1}{n^{1-\epsilon}}k$, then $OPT(\varphi) > \Theta\left(\frac{k}{n^\epsilon}\right)c$.

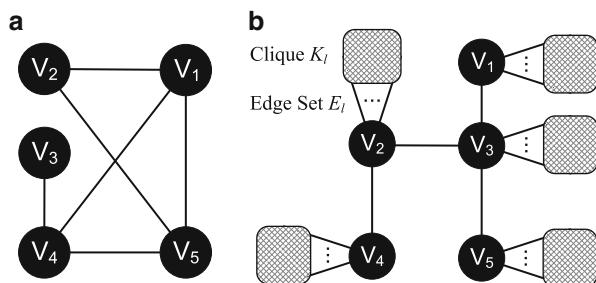


Fig. 6 The gap-preserving reduction from a maximum clique instance to a CVD instance. (a) A graph instance (b) Reduced graph G' from MC

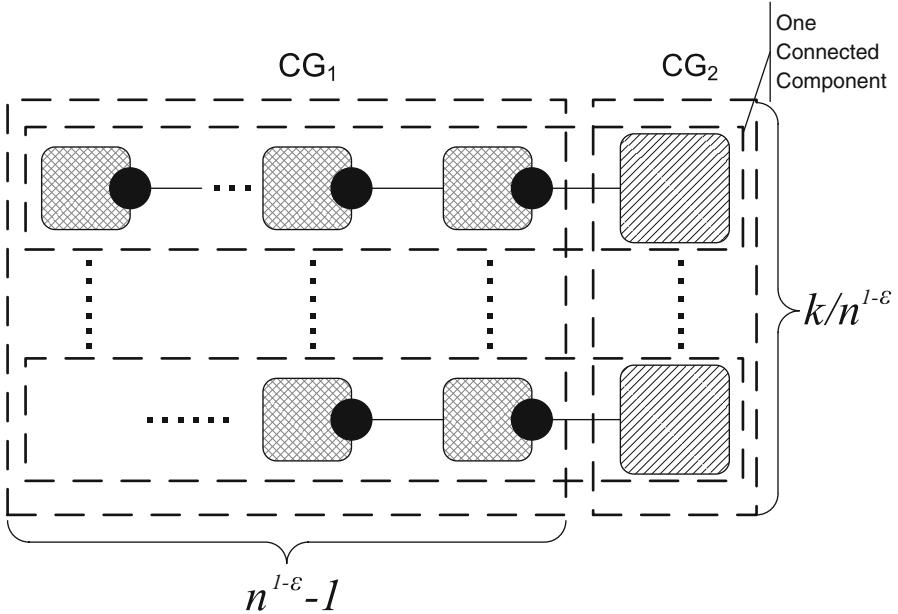


Fig. 7 The lower bound instance of CVD in G' given that $|\varphi| = n - k$ and $OPT(\phi) < \frac{1}{n^{1-\epsilon}}k$

First of all, since $OPT(\phi) < \frac{1}{n^{1-\epsilon}}k$, we cannot remove as many as $n - \frac{1}{n^{1-\epsilon}}k$ critical nodes for any $\epsilon < 1$. Therefore, it is impossible to reduce the pairwise connectivity on G' to $(n - \frac{1}{n^{1-\epsilon}}k) \binom{l}{2} + \frac{1}{n^{1-\epsilon}}k \binom{l+1}{2}$ by fragmenting G' into n components.

For any $\epsilon < 1 - \log_n 2$, it is easy to verify that $1 - \frac{1}{n^{1-\epsilon}} > \frac{1}{n^{1-\epsilon}}$. In this case, the lower bound of the pairwise connectivity in G' after removing $n - \frac{1}{n^{1-\epsilon}}k$ critical nodes is shown to be $\Theta\left(\frac{k}{n^\epsilon}\right)c$. Note that the pairwise connectivity is minimized when the number of connected components are maximized and each connected component has the same size as shown in Fig. 7. That is, since $1 - \frac{1}{n^{1-\epsilon}} > \frac{1}{n^{1-\epsilon}}$, the lower bound is achieved when the same number of cliques CG_1 connects to one CG_2 , where CG_1 and CG_2 denote the group of cliques of size l and cliques of size $l + 1$, respectively, after removing $n - \frac{1}{n^{1-\epsilon}}k$ nodes. Therefore, the following lower bound is obtained:

$$\begin{aligned} & \left(\left\lfloor \frac{\frac{1}{n^{1-\epsilon}} + 1}{2} \right\rfloor (l + 1) \right) \frac{1}{n^{1-\epsilon}} k \\ & \geq \left(\binom{n^{1-\epsilon} - 1}{2} (l + 1) \right) \frac{1}{n^{1-\epsilon}} k \end{aligned}$$

$$\begin{aligned}
&= \frac{\binom{(n^{1-\epsilon}-1)(l+1)}{2} \frac{1}{n^{1-\epsilon}} k}{(n-k)\binom{l}{2} + k\binom{l+1}{2}} c \\
&\geq \frac{\binom{ln^{1-\epsilon}}{2} \frac{1}{n^{1-\epsilon}} k}{n\binom{l}{2} + lk} c \\
&> \frac{\binom{ln^{1-\epsilon}}{2} \frac{1}{n^{1-\epsilon}} k}{n\binom{l+1}{2}} c \\
&= \Theta\left(\frac{k}{n^\epsilon}\right) c
\end{aligned}$$

Step 2 holds since $\lfloor z \rfloor \geq z - 1$ for any real number z ; steps 4 and 5 hold since $l \leq n^{1-\epsilon} - 1$ and $k \leq n$. \square

2.3 NP-Completeness of β -Edge Disruptor

In this section, the NP-completeness of β -ED is shown by reducing from the balanced cut problem. This section begins with several needed definitions as follows:

Definition 5 A cut $\langle S, V \setminus S \rangle$ corresponding to a subset $S \in V$ in G is the set of edges with exactly one endpoint in S . Usually $V \setminus S$ is denoted by \bar{S} . The cost of a cut is the sum of its edges' costs (or simply its cardinality in the case all edges have unit costs) and denoted by $c(S, \bar{S})$.

Finding a min cut in the graph is polynomial solvable [21]. However, if one asks for a somewhat “balanced” cut of minimum size, the problem becomes intractable. A balanced cut is defined as following:

Definition 6 (Balanced cut) Let f be a function from the positive integers to the positive reals. An f -balanced cut of a graph $G = (V, E)$ asks us to find a cut $\langle S, \bar{S} \rangle$ with the minimum size such that $\min\{|S|, |\bar{S}|\} \geq f(|V|)$.

Abusing notations, for $0 < c \leq \frac{1}{2}$, c -balanced cut is used to find the cut $\langle S, \bar{S} \rangle$ with the minimum size such that $\min\{|S|, |\bar{S}|\} \geq c|V|$. The following results on balanced cut shown in [22] will be used in the proofs:

Corollary 1 (Monotony) Let g be a function with

$$0 \leq g(n) - g(n-1) \leq 1$$

Then $f(n) \leq g(n)$ for all n , implies f -balanced cut is polynomially reducible to g -balanced cut.

Corollary 2 (Upper bound) αn^ϵ -balanced cut is NP-complete for $\alpha, \epsilon > 0$.

It follows from Corollaries 1 and 2 that for every $f = O(\alpha n^\epsilon)$, f -balanced cut is NP-complete. It is now ready to prove the NP-completeness of β -edge disruptor:

Theorem 7 β -edge disruptor is NP-complete.

Proof Theorem 7 is proven for a special case when $\beta = \frac{1}{2}$. For other values of β , the proof can go through with a slight modification of the reduction. In the proof, n is considered to be a large enough number, say $n > 10^3$.

Consider the decision version of the problem that asks whether an undirected graph $G = (V, E)$ contains a $\frac{1}{2}$ -edge disruptor of a specified size:

$$\frac{1}{2}\text{-ED} = \{\langle G, K \rangle \mid G \text{ has a } \frac{1}{2}\text{-edge disruptor of size } K\}$$

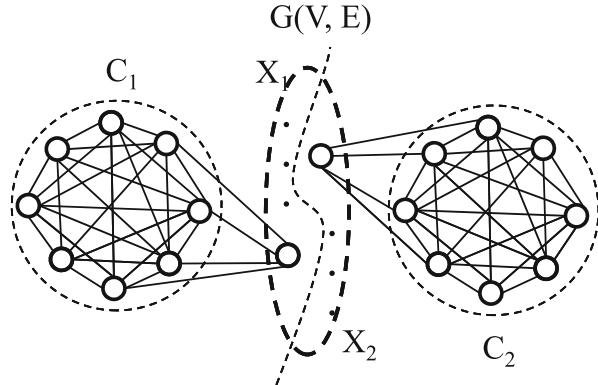
To show that $\frac{1}{2}\text{-ED}$ is in NP-complete, we must show that it is in NP and that all NP-problems are polynomial time reducible to it. The first part is easy; given a candidate subset of edges, it is easily check in polynomial time if it is a $\frac{1}{2}$ -edge disruptor of size K . To prove the second part, we show that f -balanced cut is polynomial time reducible to $\frac{1}{2}\text{-ED}$ where $f = \lfloor \frac{n - \sqrt{2\lfloor \frac{n^2}{3} \rfloor + n}}{2} \rfloor$.

Let $G = (V, E)$ be a graph in which one seeks to find an f -balanced cut of size k . Now construct the graph $H(V_H, E_H)$ as follows: $V_H = V' \cup C_1 \cup C_2$ where $V' = \{v_i \mid i \in V\}$ and C_1, C_2 are two cliques of size $\lfloor \frac{n^2}{3} \rfloor$. The total number of nodes in H is, hence, $N = 2\lfloor \frac{n^2}{3} \rfloor + n$. Beside edges inside two cliques, an edge (v_i, v_j) is added for each edge $(i, j) \in E$. Next each vertex v_i connects to $\lfloor \frac{n^2}{4} \rfloor + 1$ vertices in C_1 and $\lfloor \frac{n^2}{4} \rfloor + 1$ vertices in C_2 so that the degree difference of nodes in the cliques are at most one. The construction of $H(V_H, E_H)$ is illustrated in Fig. 8. Now we need to show that there is a f -balanced cut of size k in G iff H has an $\frac{1}{2}$ -edge disruptor of size $K = n(\lfloor \frac{n^2}{4} \rfloor + 1) + k$ where $0 \leq k \leq \lfloor \frac{n^2}{4} \rfloor$. Note that the cost of any cut $\langle S, \bar{S} \rangle$ in G is at most $|S||\bar{S}| \leq \lfloor \frac{(|S|+|\bar{S}|)^2}{4} \rfloor = \lfloor \frac{n^2}{4} \rfloor$.

On the one hand, an f -balanced cut $\langle S, \bar{S} \rangle$ of size k in G induces a cut $\langle C_1 \cup S, C_2 \cup \bar{S} \rangle$ with size exactly $n(\lfloor \frac{n^2}{4} \rfloor + 1) + k$. If the cut is selected as the disruptor, the pairwise connectivity will be at most $\frac{1}{2}\binom{N}{2}$.

On the other hand, assume that H has a $\frac{1}{2}$ -edge disruptor of size $K = n(\lfloor \frac{n^2}{4} \rfloor + 1) + k$. Because separating n nodes in a clique from the other nodes

Fig. 8 Construction of $H = (V_H, E_H)$ from $G = (V, E)$



in that clique requires cutting at least $n(\lfloor \frac{n^2}{3} \rfloor - n) > n(\lfloor \frac{n^2}{4} \rfloor + 1) + k$ edges, there are at most n nodes separated from both the cliques. A direct consequence is that at least $(\lfloor \frac{n^2}{3} \rfloor - n)$ nodes remain connected in each clique after removing edges in the disruptor. Denote the sets of those nodes by C'_1 and C'_2 , respectively. C'_1 and C'_2 cannot be connected otherwise the pairwise connectivity will exceed $\frac{1}{2} \binom{N}{2}$. Denote by X_1 and X_2 the set of nodes in V' that are connected to C'_1 and C'_2 respectively. Since C'_1 and C'_2 are disconnected we must have $X_1 \cap X_2 = \emptyset$.

The disruptor can be modified without increasing its size and the pairwise connectivity such that no nodes in the cliques are split. For each $u \in C_1 \setminus C'_1$, we remove from the disruptor all edges connecting u to C'_1 and add to the disruptor all edges connecting u to X_2 . This will move u back to the connected component that contains C'_1 while reducing the size of the disruptor at least $(\lfloor \frac{n^2}{3} \rfloor - n) - n$. At the same time, an arbitrary node $v \in X_1$ is selected and added to the disruptor all remaining v 's adjacent edges. This increases the size of the disruptor at most $(\lfloor \frac{n^2}{4} \rfloor + 1) + n$ while making v isolated. By doing so, the size of the disruptor decreases by $(\lfloor \frac{n^2}{3} \rfloor - n) - n - ((\lfloor \frac{n^2}{4} \rfloor + 1) + n) > 0$. In addition, the pairwise connectivity will not increase when connecting u to C'_1 and disconnecting v from C'_1 . If no nodes are left in X_1 , a node $v \in X_2$ can be selected (as in that case $|C'_2 \cup X_2| > |C'_1 \cup X_1|$) to make sure the pairwise connectivity will not be increased. The same process is repeated for every node in $C_2 \setminus C'_2$ and at the end of the process, $C'_1 = C_1$ and $C'_2 = C_2$.

We will prove that $X_1 \cup X_2 = V'$, that is, $\langle X_1, X_2 \rangle$ induces a cut in $V(G)$. Assume not, the cost to separate $C_1 \cup X_1$ from $C_2 \cup X_2$ will be at least $(\lfloor \frac{n^2}{4} \rfloor + 1)(|V' - X_1| + |V' - X_2|) = (\lfloor \frac{n^2}{4} \rfloor + 1)(2n - |X_1| - |X_2|) \geq (\lfloor \frac{n^2}{4} \rfloor + 1)(n + 1) > n(\lfloor \frac{n^2}{4} \rfloor + 1) + k$ that is a contradiction.

Since $X_1 \cup X_2 = V'$, the disruptor induces a cut in G . To have the pairwise connectivity at most $\frac{1}{2} \binom{N}{2}$, both $(C_1 \cup X_1)$ and $(C_2 \cup X_2)$ must have size

at least $\frac{N-\sqrt{N}}{2}$. It follows that X_1 and X_2 must have size at least $f(n) = \lfloor \frac{n-\sqrt{2\lfloor \frac{n^2}{3} \rfloor + n}}{2} \rfloor$. The cost of the cut induced by $\langle X_1, X_2 \rangle$ in G will be $n \left(\lfloor \frac{n^2}{4} \rfloor + 1 \right) + k - n(\lfloor \frac{n^2}{4} \rfloor + 1) = k$. \square

2.4 Hardness of β -Vertex Disruptor

Theorem 8 β -vertex disruptor is NP-complete.

Proof The details of the proof will be ignored. Instead, only the sketch is presented by showing that the vertex cover is polynomial time reducible to β -vertex disruptor. Let $G = (V, E)$ be a graph in which one seeks to find a vertex cover of size k . Note that if we remove nodes in a vertex cover from the graph, the pairwise connectivity in the graph will be zero. Hence, by setting $\beta = 0$, G has a vertex cover of size k iff G has an β -vertex disruptor of size k .

One can also avoid using $\beta = 0$ by replacing each vertex in G by a clique of large enough sizes, say $O(n)$, that ensures no vertices in cliques will be selected in the β -vertex disruptor. \square

Given the NP-hardness of the β -vertex disruptor, the best possible result is a polynomial-time approximation scheme (PTAS) that given a parameter $\epsilon > 0$, produces a $(1 + \epsilon)$ -approximation solution in polynomial time. Unfortunately, this is impossible unless $P = NP$.

Theorem 9 Unless $P = NP$, β -vertex disruptor has no polynomial-time approximation scheme.

Proof In the case $\beta = 0$, the problem is equivalent to finding the minimum vertex cover in the graph. In [23], Dinur and Safra showed that approximating vertex cover within constant factor less than 1.36 is NP-hard. Hence, a PTAS scheme for β -vertex disruptor does not exist unless $P = NP$. \square

3 Approximation Algorithm for β -Edge Disruptor

This section presents an $O(\log^{\frac{3}{2}} n)$ pseudo-approximation algorithm for the β -edge disruptor problem in *directed* graphs. (Note that in directed graphs, a pair (u, v) is said to be connected if there exists a directed path from u to v and vice versa, from v to u .) Formally, the algorithm finds in a directed graph G a β' -edge disruptor whose cost is at most $O(\log^{\frac{3}{2}} n)OPT_{\beta-ED}$, where $\frac{\beta'}{4} < \beta < \beta'$ and $OPT_{\beta-ED}$ is the cost of an optimal β -edge disruptor.

As shown in [Algorithm 1](#), the approximation solution consists of two main steps: (1) constructing a decomposition tree of G by recursively partitioning the

graph into two halves with a directed c -balanced cut and (2) solving the problem on the obtained tree using a dynamic programming algorithm and transform this solution to the original graph. These two main steps are explained in the next two subsections.

3.1 Balanced Tree Decomposition

Let us first introduce some definitions before describing the balanced tree decomposition algorithm.

Definition 7 Given a directed graph $G = (V, E)$ and a subset of vertices $S \subset V$. The set of edges outgoing from S and the set of edges incoming to S are denoted by $\delta^+(S)$ and $\delta^-(S)$, respectively. A cut $\langle S, \bar{S} \rangle$ in G is defined as $\delta^+(S)$. A c -balanced cut is a cut $\langle S, \bar{S} \rangle$ s.t. $\min\{|S|, |\bar{S}|\} \geq c|V|$. The directed c -balanced cut problem is to find the minimum c -balanced cut. And finally, $\mathcal{P}(G)$ denotes the total pairwise connectivity in G .

Note that a cut $\langle S, \bar{S} \rangle$ separates pairs $(u, v) \in S \times (\bar{S})$ so that there is no path u to v , that is, there is no strongly connected component (SCC) containing vertices both in S and \bar{S} .

The tree decomposition procedure is as follows: As shown in [Algorithm 1](#) (lines 3–13), the procedure starts with a tree $T = (V_T, E_T)$ containing only one root node t_0 . t_0 is associated with the vertex set V of G , that is, $V(t_0) = V(G)$. For each node $t_i \in V_T$ whose $V(t_i)$ contains more than one vertex and $V(t_i)$ has not been partitioned, the subgraph $G[V(t_i)]$ induced by $V(t_i)$ in G is partitioned using a c -balanced cut algorithm [24] where constant $c = 1 - \sqrt{\frac{\beta}{\beta'}}$. For each c -balanced cut on $G[V(t_i)]$, two children nodes t_{i1} and t_{i2} of t_i in V_T are created. These two children nodes correspond to two sets of vertices returning by the cut. Finally, each node t_i is assigned a cost $cost(t_i)$ which is equal to the cost of the cut performed on $G[V(t_i)]$. This procedure continues until $V(t_i)$ contains only a single vertex. That is, the leaves of T represent nodes in G . Therefore, T has n leaves and $n - 1$ internal nodes where each internal node in T represents a subset of nodes in V .

Note that the root node t_0 is at level 1. If a node is at level l , all its children are defined to be at level $l + 1$. Note that all collections of vertices corresponding to nodes in a same level form a partition in V .

Lemma 6 *The height of T obtained in the above balanced tree decomposition procedure is at most $O(-\log_{1-c'} n)$*

Proof Note that the directed c -balanced cut algorithm [24] finds in polynomial time a c' -balanced cut within a factor of $O(\sqrt{\log n})$ from the optimal c -balanced cut for $c' = \alpha c$ and fixed constant α . Thus, when separating $G[V(t_i)]$ using c -balanced cut, the size of the larger part is at most $(1 - c')|V(t_i)|$. By induction

method, it can be shown that if a node t_i is on level l , the size of the corresponding collection $V(t_i)$ is at most $|V| \times (1 - c')^{l-1}$. It follows that the tree's height is at most $O(-\log_{1-c'} n)$. \square

3.2 Pseudo-approximation Algorithm and Analysis

This subsection presents the second main step which uses the dynamic programming to search for the right set of nodes in T such that the cuts to partition those corresponding sets of vertices in G have the minimum cost and the obtained pairwise connectivity is at most $\beta' \binom{n}{2}$ where $\beta < \beta' < 1$. The details of this step are shown in [Algorithm 1](#) (lines 14–23).

Denote a set $F = \{t_1, t_2, \dots, t_k\} \subset V_T$ such that $V(t_1), V(t_2), \dots, V(t_k)$ is a partition of $V(G)$, that is, $V(G) = \bigcup_{i=1}^k V(t_i)$ and for any pair t_i and $t_j \in F$, $V(t_i) \cap V(t_j) = \emptyset$. Such a subset F is called G -partition. Denote by $\mathcal{A}(t_i)$ the set

Algorithm 1 β -edge disruptor

```

1: Input: A directed graph  $G = (V, E)$  and  $0 \leq \beta < \beta' < 1$ 
2: Output: A  $\beta'$ -edge disruptor of  $G$ 
   {Construct the decomposition tree}
3:  $c \leftarrow 1 - \sqrt{\frac{\beta}{\beta'}}$ 
4:  $T(V_T, E_T) \leftarrow (\{t_0\}, \phi)$ ,  $V(t_0) \leftarrow V(G)$ ,  $l(t_0) = 1$ 
5: while  $\exists$  unvisited  $t_i$  with  $|V(t_i)| \geq 2$  do
6:   Mark  $t_i$  visited, create new child nodes  $t_{i1}, t_{i2}$  of  $t_i$ 
7:    $l(t_{i1}), l(t_{i2}) \leftarrow l(t_i) + 1$ 
8:    $V_T \leftarrow V_T \cup \{t_{i1}, t_{i2}\}$ 
9:    $E_T \leftarrow E_T \cup \{(t_i, t_{i1}), (t_i, t_{i2})\}$ 
10:  Separate  $G[V(t_i)]$  into two using directed  $c$ -balanced cut
11:  Assign two obtained partitions to  $V(t_{i1})$  and  $V(t_{i2})$ 
12:   $cost(t_i) \leftarrow$  The cost of the balanced cut
13: end while
   {Find the minimum cost  $G$ -partition}
14: for  $t_i \in T$  in reversed BFS order from root node  $t_0$  do
15:   for  $p \leftarrow 0$  to  $\beta' \binom{n}{2}$  do
16:     if  $\mathcal{P}(G[V(t_i)]) \leq p$  then
17:        $cost(t_i, p) \leftarrow 0$ 
18:     else
19:        $cost(t_i, p) \leftarrow \min\{cost(t_{i1}, \pi) + cost(t_{i2}, p - \pi) + cost(t_i) \mid \pi \leq p\}$ 
20:     end if
21:   end for
22: end for
23: Find  $F$  with  $\mathcal{P}(F) = \min\{cost(t_0, p) \mid p \leq \beta' \binom{n}{2}\}$ 
24: Return union of cuts used at  $\mathcal{A}(F)$  during tree construction

```

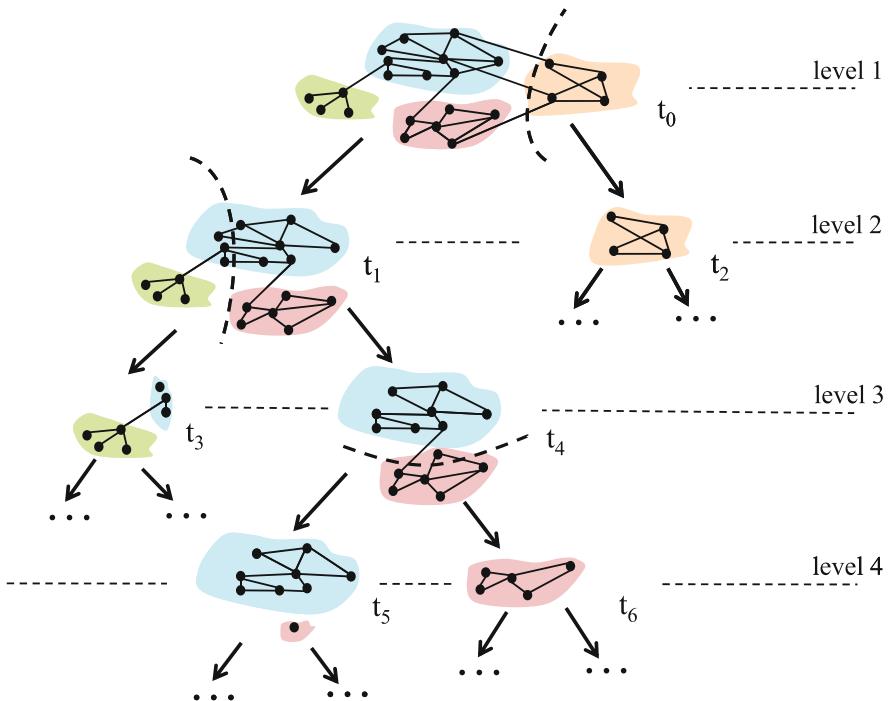


Fig. 9 A part of a decomposition tree. $F = \{t_2, t_3, t_5, t_6\}$ is a G -partition. The corresponding partition $\{V(t_2), V(t_3), V(t_5), V(t_6)\}$ in G can be obtained by using cuts at ancestors of nodes in F , that is, t_0, t_1, t_4

of nodes corresponding to ancestor of t_i in T and $\mathcal{A}(F) = \bigcup_{t_i \in F} \mathcal{A}(t_i)$. It is clear that

an F is G -partition iff F satisfies:

1. $\forall t_i, t_j \in F : t_i \notin \mathcal{A}(t_j)$ and $t_j \notin \mathcal{A}(t_i)$
2. $\forall t_i \in V_T, t_i$ is a leaf: $\mathcal{A}(t_i) \cap F \neq \emptyset$

In case $F = \{t_1, t_2, \dots, t_k\}$ is G -partition, $V(t_1), V(t_2), \dots, V(t_k)$ can be separated by performing the cuts corresponding to ancestors of nodes in F during the tree construction. For example, a decomposition tree with a G -partition set $\{t_2, t_3, t_5, t_6\}$ in Fig. 9. The corresponding partition $\{V(t_2), V(t_3), V(t_5), \text{ and } V(t_6)\}$ in G can be obtained by cutting $G[V(t_0)]$, $G[V(t_1)]$, and $G[V(t_4)]$ successively using c -balanced cuts in the tree construction. The cut cost, hence, will be $cost(t_0) + cost(t_1) + cost(t_4)$. In general, the total cost of all the cuts to separate $V(t_1), V(t_2), \dots, V(t_k)$ is equal to

$$cost(F) = \sum_{t_i \in \mathcal{A}(F)} cost(t_i)$$

And the pairwise connectivity in G is equal to

$$\mathcal{P}(F) = \sum_{t_i \in F} \mathcal{P}(G[V(t_i)])$$

Our goal now is to find a G -partition $F \in V_T$ so that $\mathcal{P}(F) \leq \beta' \binom{n}{2}$ with a minimum $\text{cost}(F)$ since the cut associated to F on T is the β' -edge disruptor of G . Clearly, finding such a set F has an optimal substructure; thus, it can be found in $O(n^3)$ using dynamic programming as described next.

Let $\text{cost}(t_i, p)$ denote the minimum cut cost to make the pairwise connectivity in $G[V(t_i)]$ be less than or equal to p using only c -balanced cuts corresponding to nodes in the subtree rooted at t_i . The minimum cost for a G -partition subset F that induces a β' -edge disruptor of G is then $\min\{\text{cost}(t_0, p) \mid p \leq \beta' \binom{n}{2}\}$ where t_0 is the root node in T .

The value of $\text{cost}(t_i, p)$ can be calculated using following recursive formula:

$$\text{cost}(t_i, p) = \begin{cases} 0 & \text{if } \mathcal{P}(G[V(t_i)]) \leq p \\ \min_{\pi \leq p} \text{cost}(t_{i1}, \pi) + \text{cost}(t_{i2}, p - \pi) + \text{cost}(t_i) & \text{otherwise} \end{cases}$$

where t_{i1}, t_{i2} are children of t_i .

In the first case, when $\mathcal{P}(G[V(t_i)]) \leq p$, no cut is required; thus, $\text{cost}(t_i, p) = 0$. Otherwise, all possible combinations of pairwise connectivity π in $V(t_{i1})$ and $p - \pi$ in $V(t_{i2})$ for all $\pi \leq p$ can be investigated. The combination with the smallest cut cost is then selected.

Based on the above recursive formula, a dynamic programming (shown in lines 14–23 of [Algorithm 1](#)) can find a minimum cost G -partition set F that induces a β' -edge disruptor in G .

The only part left is to prove that such a set F exists in T and the cost of the β' -edge disruptor induced from F found in the dynamic programming algorithm is no more than $O(\log^{\frac{3}{2}} n) \text{Opt}_{\beta}\text{-ED}$. The proof is shown as follows:

Lemma 7 *There exists a G -partition subset F in T that induces a β' -edge disruptor whose cost is no more than $O(\log^{\frac{3}{2}} n) \text{Opt}_{\beta}\text{-ED}$.*

Proof We first show that such a set F exists. That is, we are able to find a set G -partition set F such that $\mathcal{P}(F) \leq \beta' \binom{n}{2}$. Denote D_β an optimal β -edge disruptor in G . Removing D_β from G , we obtain a set of strongly connected components (SCCs), denote as $\mathcal{C}_\beta = \{C_1, C_2, \dots, C_k\}$.

We construct a G -partition subset F based on \mathcal{C}_β as shown in the [Algorithm 2](#). Nodes in T are visited in a top-down manner, that is, every parent must be visited before its children. This can be done by visiting nodes in breadth first search (BFS) order from the root node t_0 . For each node t_i , if there exists some component $C_j \in \mathcal{C}_\beta$ such that $V(t_i)$ contains more than $(1 - c)|V(t_i)|$ nodes in C_j (all leaves in T

satisfies this condition) and no ancestors of t_i has ever been selected into F , then we select t_i as a member of F . It is clear to see that F is a G -partition as it satisfies two conditions mentioned earlier.

The total pairwise connectivity of G induced by F is bounded as

$$\begin{aligned}
\mathcal{P}(F) &\leq \sum_{t_i \in F} \binom{|V(t_i)|}{2} \\
&= \frac{1}{2} \sum_{C_j \in \mathcal{C}_\beta} \sum_{|V(t_i) \cap C_j| \geq (1-c)|V(t_i)|} |V(t_i)|^2 - \frac{n}{2} \\
&\leq \frac{1}{2} \sum_{C_j \in \mathcal{C}} \left(\sum_{|V(t_i) \cap C_j| \geq \sqrt{\frac{\beta'}{\beta'}} |V(t_i)|} |V(t_i)| \right)^2 - \frac{n}{2} \\
&\leq \frac{1}{2} \sum_{C_j \in \mathcal{C}} \left(\sqrt{\frac{\beta'}{\beta}} |C_j| \right)^2 - \frac{n}{2} \\
&< \frac{\beta'}{2\beta} \left(\sum_{C_j \in \mathcal{C}} |C_j|^2 - n \right) \leq \beta' \binom{n}{2}
\end{aligned}$$

Thus, such a set F exists.

Next, we show that $\text{cost}(F) \leq O(\log^{\frac{3}{2}} n) \text{Opt}_{\beta}\text{-ED}$. Let $h(T)$ and L_T^u denote the height of T and the set of nodes at level u of T , respectively. We have

$$\text{cost}(F) = \sum_{u=1}^{h(T)} \sum_{t_i \in (L_T^u \cap \mathcal{A}(F))} \text{cost}(t_i) \tag{2}$$

If $t_i \in \mathcal{A}(F)$, then t_i is not selected to F . Hence, there exists $C_j \in \mathcal{C}_\beta$ so that $|V(t_i) \cap C_j| < (1-c)|V(t_i)|$ (otherwise t_i was selected into F as it satisfies the conditions in the line 3, [Algorithm 2](#)). To guarantee $c < 1-c$ we constrain $c < 1/2$, that is, $\beta > \frac{\beta'}{4}$.

The edges in the optimal β -edge disruptor D_β separate C_j from the other SCCs. Hence, D_β also separates $C_j \cap V(t_i)$ from the $V(t_i) \setminus C_j$ in $G[V(t_i)]$. Denote $\text{sep}(t_i, D_\beta)$ the set of edges in D_β separating $C_j \cap V(t_i)$ from the rest in $G[V(t_i)]$. Clearly, $\text{sep}(t_i, D_\beta)$ is a directed c -balanced cut of $G[V(t_i)]$. Since, the cut algorithm we used in the tree construction has a pseudo-approximation ratio of only $O(\sqrt{\log n})$, we have $\text{cost}(t_u) \leq O(\sqrt{\log n}) |\text{sep}(t_i, D_\beta)|$.

Algorithm 2 Find a good G -partition set F of T that induces a β' -edge disruptor in G

```

 $F \leftarrow \emptyset$ 
for  $t_i \in T$  in BFS order from  $t_0$  do
    if  $(\exists C_j \in \mathcal{C}_\beta : |V(t_i) \cap C_j| \geq (1 - c)|V(t_i)|)$  and  $(\mathcal{A}(t_i) \cap F = \emptyset)$  then
         $F \leftarrow F \cup \{t_i\}$ 
    end if
end for

```

Recall that if two nodes t_i and t_j are on the same level, then $V(t_i)$ and $V(t_j)$ are two disjoint subsets. It follows that $\text{sep}(t_i, D_\beta)$ and $\text{sep}(t_j, D_\beta)$ are also disjoint sets. Therefore, we have

$$\begin{aligned}
& \sum_{t_i \in (L_T^u \cap \mathcal{A}(F))} \text{cost}(t_i) \\
& \leq O(\sqrt{\log n}) \sum_{t_i \in (L_T^u \cap \mathcal{A}(F))} |\text{sep}(t_i, D_\beta)| \\
& \leq O(\sqrt{\log n}) \left| \bigcup_{t_i \in (L_T^u \cap \mathcal{A}(F))} \text{sep}(t_i, D_\beta) \right| \\
& = O(\sqrt{\log n}) \text{Opt}_{\beta\text{-ED}}
\end{aligned}$$

Since $h(T)$ is at most $O(\log n)$ (Lemma 6), it follows from Eq. 2 that $\text{cost}(F) \leq O(\log^{\frac{3}{2}} n) \text{Opt}_{\beta\text{-ED}}$. It completes the proof. \square

Since there exists a G -partition subset of T that induces a β' -edge disruptor whose cost is no more than $O(\log^{\frac{3}{2}} n) \text{Opt}_{\beta\text{-ED}}$ as shown in Lemma 7 and the dynamic programming is always able to find such a set F , the following theorem follows immediately:

Theorem 10 *Algorithm 1* achieves a pseudo-approximation ratio of $O(\log^{\frac{3}{2}} n)$ for the β -edges disruptor problem.

4 Approximation Algorithm for β -Vertex Disruptor

This section presents a polynomial-time algorithm (shown in Algorithm 3) that finds in a directed graph $G = (V, E)$ a β' -vertex disruptor whose the size is at most $O(\log n \log \log n)$ times the optimal β -vertex disruptor where $0 < \beta < \beta'^2$.

At the high level, Algorithm 3 consists of two main phases: (1) vertex conversion and (2) size constraint cut. In the first phase, a given graph G is converted into G' in a way that removing $v \in G$ has the same effects as removing edge in G' . In the second phase, G' is cut into SCCs capping the sizes of the largest component

while minimizing the number of removed edges. The constraint on the size of each component is kept relaxing until the set of cut edges induces a β' -vertex disruptor of G .

4.1 Algorithm Description

Phase 1: Vertex Conversion. Given a directed graph $G = (V, E)$ for which we want to find a small β' -vertex disruptor, a new directed graph $G' = (V', E')$ is constructed as follows:

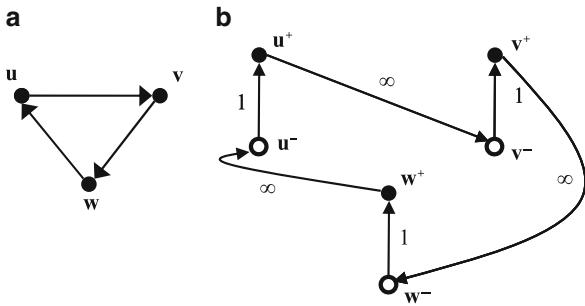
1. *V' construction:* For each vertex $v \in V$, create two vertices v^- and v^+ in V' . Thus, $V' = \{v^-, v^+ \mid v \in V\}$.
2. *E' construction:* For each vertex $v \in V$, add a directed edge $(v^- \rightarrow v^+)$ to E' . And for each directed edge $(u \rightarrow v) \in E$, add a directed edge $(u^+ \rightarrow v^-) \in E'$. That is, $E' = \{(v^- \rightarrow v^+) \mid v \in V\} \cup \{(u^+ \rightarrow v^-) \mid (u \rightarrow v) \in E\}$.
3. *Edge cost assignment:* Assign 1 for all edges $(v^- \rightarrow v^+)$ and $+\infty$ to other edges in E' .

An example of this construction is shown in Fig. 10.

Based on this conversion, a careful edge cut on G' will return a vertex disruptor on G . Indeed, let $E'_V = \{(v^- \rightarrow v^+) \mid v \in V\}$ and consider a cut set $D'_e \subset E'$ that contains only edge in E'_V , we have a one-to-one correspondence between D'_e and $D_v = \{v \mid (v^- \rightarrow v^+) \in D'_e\}$ which is a vertex disruptor set of G . However, G and G' have different maximum pairwise connectivity, $\frac{(n-1)n}{2}$ for G and $\frac{(2n-1)2n}{2}$ for G' , the fractions of pairwise connectivity remaining in G and G' after removing D_v and D'_e are, therefore, not simply related to each other. Thus, we next describe what a “careful edge cut” is and how good a vertex disruptor of G can be obtained.

Phase 2: Size Constraint Cut. Observe that if edges are removed in order to separate a graph into SCCs, then there is a relation between the pairwise connectivity in the remaining graph and the maximum size of SCC. The smaller the maximum size of SCC, the smaller pairwise connectivity in the graph. However, the smaller the maximum size of each SCC, the more edges are needed to be cut. Therefore, a good range of the maximum size of SCC in G' needs to be found in order to return a good vertex disruptor of G with the minimum cut. Along this direction, a binary search can be performed to find a right upper bound $\bar{\beta}|V'|$ and lower bound $\beta|V'|$ of the size of each SCC in G' . As shown in Algorithm 3, at each step, we find in $G' = (V', E')$ a minimum cost edge set whose removal partitions the graph into strongly connected components, each has size at most $\tilde{\beta}|V'|$, where $\tilde{\beta} = \lfloor \frac{\beta + \bar{\beta}}{2\epsilon} \rfloor \times \epsilon$. The value of $\tilde{\beta}$ is rounded to the nearest multiple of ϵ so that the number of steps for the binary search is bounded by $\log \frac{1}{\epsilon}$. The problem of finding a minimum cost edge set to decompose a graph of size n into strongly connected components of size at most ρn is known as ρ -separator problem. At this step, the

Fig. 10 Phase 1: Vertex conversion which converts a graph G into G' such that removing a vertex $v \in G$ has the same effects as removing an edge in G'



Algorithm 3 β' -vertex disruptor

Input: Directed graph $G = (V, E)$ and fixed $0 < \beta' < 1$
Output: A β' -vertex disruptor of G
 $G'(V', E') \leftarrow (\phi, \phi)$
 $\forall v \in V : V' \leftarrow V' \cup \{v^+, v^-\}$
 $\forall v \in V : E' \leftarrow E' \cup \{(v^- \rightarrow v^+)\}, c(v^-, v^+) \leftarrow 1$
 $\forall (u \rightarrow v) \in E : E' \leftarrow E' \cup \{u^+ \rightarrow v^-\}, c(u^+, v^-) \leftarrow \infty$
 $\beta \leftarrow 0, \tilde{\beta} \leftarrow 1$
 $D_V \leftarrow V(G)$
while ($\bar{\beta} - \underline{\beta} > \epsilon$) **do**
 $\tilde{\beta} \leftarrow \lfloor \frac{\bar{\beta} + \underline{\beta}}{2\epsilon} \rfloor \times \epsilon$
 Find $D_e \subset E'$ to separate G' into strongly connected components of sizes at most $\tilde{\beta}|V'|$ using algorithm in [25]
 $D_v \leftarrow \{v \in V(G) \mid (v^+ \rightarrow v^-) \in D_e\}$
 if $\mathcal{P}(G[V \setminus D_v]) \leq \beta \binom{n}{2}$ **then**
 $\underline{\beta} = \tilde{\beta}$
 Remove nodes from D_v as long as $\mathcal{P}(G[V \setminus D_v]) \leq \beta \binom{n}{2}$
 if $|D_V| > |D_v|$ **then**
 $D_V = D_v$
 end if
 else
 $\bar{\beta} = \tilde{\beta}$
 end if
end while
Return D_V

algorithm presented in [25] which finds a ρ -separator in a directed graph G with a pseudo-approximation ratio of $O\left(\frac{1}{\epsilon^2} \cdot \log n \log \log n\right)$ for a fixed $\epsilon > 0$ is used. In our context, $\rho = \tilde{\beta}|V'|$. By the cost assignment step in the vertex construction phase, the edge cut obtained in this step must be a subset of E'_V as other edges e' not in E'_V have a cost of $+\infty$; hence, e' never got selected. Finally, the cut edges in G' is converted to vertices in G to obtain the β' -vertex disruptor. Simply, for each edge $(v^- \rightarrow v^+)$ in a cut set, we have a corresponding vertex $v \in V$.

4.2 Theoretical Analysis

Lemma 8 *Algorithm 3* always terminates with a β' -vertex disruptor.

Proof We show that whenever $\tilde{\beta} \leq \beta'$ then the corresponding D_v found in **Algorithm 3** is a β' -vertex disruptor of G . Consider the edge separator D'_e of G' induced by D_v . We first show the mapping between SCCs in $G[V \setminus D_v]$ and SCCs in $G'[E' \setminus D'_e]$, the graph obtained by removing D'_e from G' . Partition the vertex set V of G into (1) D_v : the set of removed nodes; (2) $V_{\text{singleton}}$: the set of SCCs whose sizes are one, that is, each SCC has only one node; and (3) $V_{\text{connected}}$: the set of remaining SCCs whose sizes are at least two, denote $V_{\text{connected}} = \bigcup_{i=1}^l C_i, |C_i| \geq 2$. Vertices in $V_{\text{connected}}$ belong to at least one cycle in G , whereas vertices in V_{single} are all singleton.

We have the following corresponding SCCs in $G'[E' \setminus D'_e]$:

1. $v \in D_v \Leftrightarrow$ SCCs $\{v^+\}$ and $\{v^-\}$ in $G'[E' \setminus D'_e]$. Because after removing $(v^- \rightarrow v^+)$, v^+ does not have incoming edges and v^- does not have outgoing edges.
2. $v \in V_{\text{singleton}} \Leftrightarrow$ SCCs $\{v^+\}$ and $\{v^-\}$. Assume v^+ belong to some SCC of size at least 2, that is, v^+ lies on some cycle in G' . Because the only incoming edge to v^+ is from v^- , it follows that v^- is preceding v^+ on that cycle. Let u^- and u^+ be the successive vertices of v^+ on that cycle. Then u and v should belong to the same SCC in G which yields a contradiction as $v \in V_{\text{singleton}}$. Similarly, v^- cannot lie on any cycle in G' .
3. SCC $C_i \subset V_{\text{connected}} \Leftrightarrow$ SCC $C'_i = \{v^-, v^+ \mid v \in C_i\}$. This can be shown using a similar argument as above.

Since D'_e is a $\tilde{\beta}$ -separator, the sizes of SCCs in $G'[E' \setminus D'_e]$ are at most $\tilde{\beta} 2n$. It follows that the sizes of SCCs in $G[V \setminus D_v]$ are bounded by $\tilde{\beta} n$.

Denote the set of SCCs in $G[V \setminus D_v]$ by \mathcal{C} with the convention that vertices in D_v become singleton SCC in $G[V \setminus D_v]$. Therefore, we have

$$\begin{aligned} \mathcal{P}(G[V \setminus D_v]) &= \sum_{C_i \in \mathcal{C}} \binom{|C_i|}{2} = \frac{1}{2} \left(\sum_{C_i \in \mathcal{C}} |C_i|^2 - |V| \right) \\ &\leq \frac{1}{2} \left(\sum_{C_i \in \mathcal{C}} \tilde{\beta} |V| |C_i| - |V| \right) \\ &= \frac{1}{2} \left(\tilde{\beta} |V|^2 - |V| \right) \leq \tilde{\beta} \binom{|V|}{2} < \beta' \binom{|V|}{2} \end{aligned}$$

This guarantees that the binary search always finds a β' -vertex disruptor and completes the proof. \square

Theorem 11 *Algorithm 3* achieves a pseudo-approximation ratio of $O(\log n \log \log n)$ for the β -vertex disruptor problem.

Proof We will show that [Algorithm 3](#) always finds a β' -vertex disruptor whose size is at most $O(\log n \log \log n)$ times the optimal β -vertex disruptor for $\beta'^2 > \beta > 0$. First, it follows from the [Lemma 8](#) that [Algorithm 3](#) returns a β' -vertex disruptor D_v . At some step, the size of D_v equals to the cost of $\tilde{\beta}$ -separator D'_e in G' where $\tilde{\beta}$ is at least $\beta' - \epsilon$ according to [Lemma 8](#) and the binary search scheme. The cost of the separator is at most $O(\log n \log \log n)$ times the $Opt_{(\tilde{\beta}-\epsilon)}$ -separator using the algorithm in [25].

Consider an optimal $(\beta'^2 - 9\epsilon)$ -vertex disruptor D'_v of G and its corresponding edge disruptor D'_e in G' . Denote the cost of that optimal vertex disruptor by $Opt_{(\beta'^2-9\epsilon)}\text{-VD}$. If there exists in $G[V \setminus D_v]$ a SCC C_i so that $|C_i| > (\beta' - 2\epsilon)n$, then

$$\mathcal{P}(G[V \setminus D_v]) > \frac{1}{2}((\beta' - 2\epsilon)n - 2)((\beta' - 2\epsilon)n - 1) > (\beta'^2 - 9\epsilon) \binom{n}{2}$$

when $n > \frac{20(\beta'+1)}{\epsilon}$. Hence, every SCC in $G'[V \setminus D'_v]$ has size at most $(\beta' - 2\epsilon)(2n)$, that is, D'_e is an $(\beta' - 2\epsilon)$ -separator in G' . It follows that $Opt_{(\beta'^2-9\epsilon)}\text{-VD} \geq Opt_{(\beta'-2\epsilon)}$ -separator in G' .

Because $\tilde{\beta} - \epsilon \geq \beta' - 2\epsilon$, we have

$$Opt_{(\tilde{\beta}-\epsilon)}\text{-separator} \leq Opt_{(\beta'-2\epsilon)}\text{-separator} \leq Opt_{(\beta'^2-9\epsilon)}\text{-VD}$$

The size of the vertex disruptor $|D_v| = |D'_e|$ is at most $O(\log n \log \log n)$ times $Opt_{(\tilde{\beta}-\epsilon)}$ -separator. Thus, the size of found β' -vertex disruptor D_v is at most $O(\log n \log \log n)$ times the optimal $(\beta'^2 - 9\epsilon)$ -vertex disruptor. As an arbitrary small ϵ can be chosen, setting $\beta = \beta'^2 - 9\epsilon$ completes the proof. \square

5 A Second Look

Another look at the solutions to β -ED and β -VD in order to obtain a better algorithm is discussed in this section. As can be seen earlier, a c -balance cut for a tree decomposition is used to solve the β -ED problem. This results into a $\sqrt{\log n}$ ratio for a c -balance cut and another $\log n$ for the height of the decomposition tree, leading to a $\log^{1.5} n$ ratio. If a better cut without a need of decomposing G into a tree is devised, a $\log n$ factor can be saved, thus obtaining a $\sqrt{\log n}$ ratio. In order to do so, let us first describe the following definitions which are the key concepts to a better algorithm by approximating the best “local” cut at each step:

Cut Ratio. Given a graph $G = (V, E)$, for any cut $\langle S, \bar{S} \rangle$ where $S \subseteq V$ and $\bar{S} = V \setminus S$, the *cut ratio* of $\langle S, \bar{S} \rangle$, denoted by $\alpha(S)$, is defined as

$$\alpha(S) = \frac{c(S, \bar{S})}{|S||\bar{S}|}, \quad (3)$$

where $c(S, \bar{S})$ is the total cost of edges in the cut.

The cut ratio metric intuitively allows to find “natural” partitions: The numerator captures the minimum cost criterion, while the denominator favors a balanced partition. Assume that G is connected, the cut disrupts at least $|S||V \setminus S|$ pairs, those with exact one endpoint in S ; hence, $\alpha(S)$ can also be seen as the average cost to disrupt pairs.

Sparsest Cut. The sparsest cut is a cut with the smallest cut ratio. Let $\alpha(G)$ denote the $\min_{S \subset V} \alpha(S)$, the minimum cut ratio of all cuts in G . Thus, the sparsest cut obtains a cut with the cut ratio $\alpha(G)$. As expected, finding the sparsest cut is also an NP-hard problem [26].

5.1 $O(\sqrt{\log n})$ Pseudo-approximation for β -Edge Disruptor

As the cut ratio can be seen as the average cost to disrupt the pairwise connectivity, iteratively finding and applying the sparsest cut until obtaining a β -edge separator may result in a small cost β -edge disruptor. We describe a new algorithm using the sparsest cut in [Algorithm 4](#) and show this algorithm is indeed a $\sqrt{\log n}$ pseudo-approximation algorithm for the β -ED problem.

The *recursive bisection algorithm* (RBA) iteratively selects a connected component in the graph and calls a subroutine SPARSEST_CUT to cut the component into two smaller ones, until the pairwise connectivity in the graph is no more than $\beta \binom{n}{2}$. The subroutine SPARSEST_CUT function works as follows: Assume that it takes a *connected* graph $G' = (V', E')$ as the input it will find a small ratio cut S' in G' , and returns the pair of $(\langle S', \bar{S}' \rangle, \alpha(S'))$, that is, the edges in the cut and the cut ratio. At each iteration, a component C^* in G that has the minimum cut ratio is selected, and the edges in the sparsest cut of C^* are removed from G . After that, C^* is no longer a component of G , and new components are introduced into G . The SPARSEST_CUT is used again

Algorithm 4 Recursive bisection algorithm (RBA)

```

 $E_\beta \leftarrow \emptyset$ 
for each connected component  $C$  in  $G$ 
   $(C_E, C_\alpha) \leftarrow \text{SPARSEST\_CUT}(C)$  [27]
while  $\mathcal{P}(G) > \beta \binom{n}{2}$ 
  Find a component  $C^*$  of  $G$  with minimum cut ratio  $C_\alpha^*$ 
   $E_\beta \leftarrow E_\beta \cup C_E^*$ 
  Remove edges in  $C_E^*$  from  $G$ 
  for each new component  $C'$  in  $G$ 
     $(C'_E, C'_\alpha) \leftarrow \text{SPARSE\_CUT}(C')$ 
return  $E_\beta$ 

```

to find cuts and the corresponding cut ratios for the newly created components. This procedure is repeated until the pairwise connectivity in G is no more than $\beta \binom{n}{2}$.

Clearly, the performance of the SPARSEST_CUT subroutine directly impacts on the performance of RBA. Here the SPARSEST_CUT is selected as the algorithm that guarantees the best theoretical result in literature for the min ratio cut problem [27]. The result in [27] gives us a polynomial-time algorithm to find a cut (S, \bar{S}) given a graph $G = (V, E)$ with the cut ratio at most $O(\sqrt{\log n})\alpha(G)$. This selection yields an $O(\sqrt{\log n})$ pseudo-approximation algorithm for the β -edge disruptor problem as analyzed below.

Lemma 9 *Given a graph $G = (V, E)$ and a subset of edges $M_\omega \subseteq E$, if $\omega = \mathcal{P}(G) - \mathcal{P}(G[E \setminus M_\omega]) > 0$, then $\frac{c(M_\omega)}{\omega} \geq \alpha_{\min}(G)$, where*

$$\alpha_{\min}(G) = \min\{\alpha(C) \mid C \text{ is a connected component of } G\}.$$

Proof We first prove for the case G is connected, that is, $\alpha_{\min}(G) = \alpha(G)$ and $\mathcal{P}(G) = \binom{n}{2}$. Let C_1, C_2, \dots, C_k be connected components in $G[E \setminus M_\omega]$ and let $C_i(V)$ denote the set of vertices in component C_i . By definition of $\alpha(G)$, we have $\alpha(G) \leq \alpha(C_i(V))$. Thus,

$$c(C_i(V), \overline{C_i(V)}) \geq \alpha(G)|C_i(V)||V \setminus C_i(V)|$$

Take the sum over all components C_i , we have

$$\sum_i c(C_i(V), \overline{C_i(V)}) \geq \alpha(G) \sum_i |C_i(V)|(n - |C_i(V)|)$$

Simplify both sides and note that $\sum_i |C(V_i)| = n$, we have

$$\begin{aligned} 2c(M_\omega) &\geq \alpha(G) \left(n \sum_i |C(V_i)| - \sum_i |C(V_i)|^2 \right) \\ &\geq 2\alpha(G) \left(\binom{n}{2} - \sum_i \mathcal{P}(C_i) \right) \geq 2\alpha(G) \omega \end{aligned}$$

Hence, the lemma holds when G is connected.

Now, if G is not connected, let T_1, T_2, \dots, T_l be connected components of G , $M_\omega^{(j)}$ be the intersection of M_ω and the edges in T_j , and T'_j be the subgraphs obtained from T_j after removing $M_\omega^{(j)}$. Applying the above results to the case that the graph is connected on each connected component, we have

$$\begin{aligned}
c(M_\omega) &= \sum_j c(M_\omega^{(j)}) \geq \sum_j \alpha(T_j) (\mathcal{P}(T_j) - \mathcal{P}(T'_j)) \\
&\geq \alpha_{\min}(G) \sum_j (\mathcal{P}(T_j) - \mathcal{P}(T'_j)) \\
&= \alpha_{\min}(G) (\mathcal{P}(G) - \mathcal{P}(G[E \setminus M_\omega]))
\end{aligned}$$

Thus, the lemma holds for every graph G . \square

Theorem 12 *If SPARSEST_CUT is an approximation algorithm with a factor $f(n)$ for the min cut ratio problem, then RBA returns a β -edge disruptor of cost at most $f(n) \text{OPT}_{\beta'}^e$, for any $0 < \beta' < \beta$, where $\text{OPT}_{\beta'}^e$ denotes the cost of a minimum β' -edge disruptor.*

Proof By Lemma 9, if removing a set of edges $M_\omega \subseteq E$ from G disrupts ω pairs of vertices, then $\frac{c(M_\omega)}{\omega} \geq \alpha_{\min}(G)$. Thus, at any round in the while loop of RBA, since a set of edges $E_{\beta'}^*$ in a minimum β' -edge disruptor, for some $0 < \beta' < \beta$, can disrupt at least $(\beta - \beta') \binom{n}{2}$ more pairs in G , we have

$$\frac{\text{OPT}_{\beta'}^e}{(\beta - \beta') \binom{n}{2}} \geq \alpha_{\min}(G). \quad (4)$$

From the hypothesis that SPARSE_CUT is an $f(n)$ factor approximation algorithm for the min cut ratio problem, the average cost to disrupt a pair by removing C_E^* is upper bounded by $f(n)\alpha_{\min}(G)$. By (4), the average cost to disrupt pairs in the graph at any step is at most

$$f(n) \frac{\text{OPT}_{\beta'}^e}{(\beta - \beta') \binom{n}{2}}$$

Therefore, even when E_β disrupt all $\binom{n}{2}$ pairs in G , the total cost is no more than

$$f(n) \frac{\text{OPT}_{\beta'}^e}{(\beta - \beta') \binom{n}{2}} \binom{n}{2} \leq \frac{f(n)}{\beta - \beta'} \text{OPT}_{\beta'}^e.$$

Hence, the theorem follows. \square

Remarks If the underlying graph is directed, the directed sparsest cut algorithm in [24] can be used in the place of the sparsest cut algorithm in [27] to obtain a pseudo-approximation algorithm with the same ratio. Note that for directed graphs, a pair of vertices (u, v) is said to be connected if there exists a path from u to v and from v to u (i.e., connected in both directions).

5.2 $O(\sqrt{\log n})$ Pseudo-approximation for β -Vertex Disruptor

For the β -vertex disruptor problem, the first phase – vertex conversion – as discussed in Sect. 4 still can be used. However, in order to obtain an $O(\sqrt{\log n})$ pseudo-approximation, a relationship between β -VD and β -ED needs to be defined in order to take advantage of Algorithm 4 instead of using the phase two – size constraint cut. Recall that given a directed graph $G = (V, E)$ for which we want to find a small β -VD, after the vertex conversion, we should obtain a directed graph $G' = (V', E')$ which will have twice the number of vertices in G , that is, $|V'| = 2|V| = 2n$.

Consider a directed edge disruptor set $D'_e \subset E'$ that contains only edge in E'_V . There is a one-to-one correspondence between D'_e to a set $D_v = \{v \mid (v^- \rightarrow v^+) \in D'_e\}$ in $G(V, E)$ which is a vertex disruptor set in G . Since G and G' have different maximum pairwise connectivity, $\frac{(n-1)n}{2}$ for G and $\frac{(2n-1)2n}{2}$ for G' , the fractions of pairwise connectivity remaining in G and G' after removing D_v and D'_e are, however, not exactly equal to each other.

Lemma 10 *A β -edge disruptor set in the directed graph G' induces the same cost β -vertex disruptor set in G .*

Proof Let us denote D_v and D'_e for vertex disruptor in G and edge disruptor in G' .

Given $\mathcal{P}(G'[E' \setminus D'_e]) \leq \beta \binom{2n}{2}$ we need to prove that: $\mathcal{P}(G_{[V \setminus D_v]}) \leq \beta \binom{n}{2}$ where $n = |V|$.

Assume $G_{[V \setminus D_v]}$ has l SCCs of size at least 2, say $C_i, i = 1 \dots l$. The corresponding SCCs in $G'[E' \setminus D'_e]$ will be $C'_i, i = 1 \dots l$ where $|C'_i| = 2|C_i|$.

Since $\frac{\binom{2k}{2}}{\binom{2n}{2}} - \frac{\binom{k}{2}}{\binom{n}{2}} = \frac{k(n-k)}{(n-1)n(2n-1)} \geq 0$, for all $0 \leq k \leq n$. We have

$$\frac{\mathcal{P}(G_{[V \setminus D_v]})}{\binom{n}{2}} = \sum_{i=1}^l \frac{\binom{|C_i|}{2}}{\binom{n}{2}} \leq \sum_{i=1}^l \frac{\binom{|C'_i|}{2}}{\binom{2n}{2}} \leq \beta \quad \square$$

Lemma 11 *A β -vertex disruptor set in G induces the same cost $(\beta + \epsilon)$ -edge disruptor set in G' for any $\epsilon > 0$.*

Proof The same notations in the proof of Lemma 10 are used here. Given $\mathcal{P}(G_{[V \setminus D_v]}) \leq \beta \binom{n}{2}$ we need to prove that $\mathcal{P}(G'[E' \setminus D'_e]) \leq (\beta + \epsilon) \binom{2n}{2}$. We have

$$\begin{aligned} & \frac{\mathcal{P}(G'[E' \setminus D'_e])}{\binom{2n}{2}} \\ &= \sum_{i=1}^l \frac{|C_i|(n - |C_i|)}{(n-1)n(2n-1)} + \frac{\mathcal{P}(G_{[V \setminus D_v]})}{\binom{n}{2}} \end{aligned}$$

$$\begin{aligned}
&= \frac{\mathcal{P}(G_{[V \setminus D_v]})}{\binom{n}{2}} \left(1 - \frac{1}{2n-1}\right) + \frac{\sum_{i=1}^l |C_i|}{n(2n-1)} \\
&< \beta + \frac{1}{2n-1} < \beta + \epsilon
\end{aligned} \tag{5}$$

when $n \geq \lfloor \frac{1+\epsilon}{2\epsilon} \rfloor + 1$. \square

We now prove the main theorem for conversion from directed edge disruptor to vertex disruptor.

Theorem 13 *Given a factor $f(n)$ polynomial-time bicriteria approximation algorithm for β -edge disruptor in directed graphs, there exists a factor $f(n)$ polynomial-time bicriteria approximation algorithm that finds a $(\beta + \epsilon)$ -vertex disruptor whose cost at most $f(n)$ time the cost of the optimal β -vertex disruptor where $\epsilon > 0$ is an arbitrary small constant.*

Proof Let G be a directed graph with uniform vertex costs in which we wish to find a β -vertex disruptor. Construct G' as described at the beginning of this section.

For $\beta' < \beta$, apply the given approximation algorithm to find in G' a β -edge disruptor, denoted by D'_e , with the cost at most $f(n) \cdot \text{Opt}_{\beta'-\text{ED}}(G')$, where $\text{Opt}_{\beta'-\text{ED}}(G')$ is the cost of a minimum β -edge disruptor in G' . From Lemma 10, D'_e induces in G a β -vertex disruptor D_v of the same cost. We shall prove that

$$\text{Opt}_{\beta'-\text{ED}}(G') \leq \text{Opt}_{\beta'-\text{VD}}(G) + \gamma_0,$$

where $\text{Opt}_{\beta'-\text{VD}}(G)$ is the cost of a minimum β' -vertex disruptor in G and γ_0 is some positive constant. It follows that the cost of D_v will be at most

$$f(n) \cdot (\text{Opt}_{\beta'-\text{VD}}(G) + \lambda_0) \leq (1 + \epsilon) f(n) \text{Opt}_{\beta'-\text{VD}}(G)$$

Here, assume that $\text{Opt}_{\beta'-\text{VD}}(G) > \frac{\gamma_0}{\epsilon}$; otherwise, $\text{Opt}_{\beta'-\text{VD}}(G)$ can be found in time $O(n^{\frac{\gamma_0}{\epsilon}+2})$ (polynomial time).

From an optimal β -vertex disruptor of G , construct its corresponding edge disruptor D_e^* in G' . If $\mathcal{P}(G'[E \setminus D_e^*] \leq \beta \binom{2n}{2}$, then $\text{Opt}_{\beta-\text{ED}}(G') \leq \text{Opt}_{\beta-\text{VD}}(G)$, and we yield the proof. Thus, only the case $\mathcal{P}(G'[E \setminus D_e^*] > \beta \binom{2n}{2})$ is considered.

Among SCCs of $G'[E \setminus D_e^*]$, there must be a SCC of size at least $\beta 2n$ or else $G'[E \setminus D_e^*] \leq \beta^{-1} \binom{\beta 2n}{2} \leq \beta \binom{2n}{2}$ (contradiction). Remove $\gamma_0 = \lceil \frac{1}{\beta} \rceil$ vertices from that SCC. The pairwise connectivity in $G'[E \setminus D_e^*]$ will decrease at least $(\beta 2n - \frac{1}{\beta}) \frac{1}{\beta} = 2n - \frac{1}{\beta^2} \geq n$ for sufficient large n . From Lemma 11, the pairwise connectivity after removing vertices will be less than

$$\left(\beta + \frac{1}{2n-1}\right) \binom{2n}{2} - n \leq \beta \binom{2n}{2}$$

Therefore, after removing at most γ_0 vertices from D_e^* , a β -edge disruptor is obtained. Hence,

$$\text{Opt}_{\beta-\text{ED}}(G') \leq \text{Opt}_{\beta-\text{VD}}(G) + \gamma_0.$$

From the above theorem and the $O(\sqrt{\log n})$ pseudo-approximation algorithm for directed β -edge disruptor (shown in [Algorithm 4](#)), we have the following result: \square

Theorem 14 *For any $0 \leq \beta' < \beta$, there is an algorithm that finds a β -vertex disruptor of cost at most $O(\sqrt{\log n})\phi(\text{OPT}_{\beta'}^v)$, where $\text{OPT}_{\beta'}^v$ is the optimal β -vertex disruptor.*

6 Literature

Several existing works on network vulnerability and survivability have been investigated, roughly categorized into two periods. Researches during the early period mainly focused on investigating the node's centrality and prestige, using functions of node degree [3, 5, 28, 29] as the only measure. The main measures of centrality can be classified as degree centrality, betweenness, closeness, and eigenvector centrality. Readers are referred to [30] to find more details about this centrality measurement.

As wireless ad hoc networks came forth, several recent studies have aimed to discover the critical nodes whose removal causes the network disconnected, regardless of how fragmented it is! Several centralized, distributed, and localized heuristics have been proposed. Some representatives are centralized DFS-based search [7–9, 31], distributed disjoint path [10, 32], and localized k -critical [11]. Most of these algorithms suffer from high communication overhead to discover the network partition and could not efficiently locate the critical nodes. In addition, none of these methods have been proven theoretically.

Unfortunately, none of the above work reveals the global damage done on the network in the case of multiple nodes/links failing simultaneously, thus inaccurately assessing the network vulnerability. None of these methods prove the performance bound of their solutions either.

This chapter presents a novel paradigm via two new optimization models for quantitatively characterizing the vulnerability of networks. The proposed approaches *aim to identify the key nodes* who play a vital role in the network overall performance, such as whose removal *maximizes the network fragmentation*. This study defines the objective function differently in terms of the connected components rather than a traditional direct measure of algebraic connectivity, thereby providing an excellent way to assess the network vulnerability.

7 Conclusion

This chapter presents two models along with their complexity hardness and solutions to the network vulnerability. There are a couple of notes that we would like to discuss further as follows:

It is easy to see that when k is large enough, the CVD problem cannot be approximated within a factor of $\alpha(n)$ for any polynomial-time computable function $\alpha(n)$ unless $\mathbf{P} = \mathbf{NP}$ by reducing from the vertex cover problem. To avoid the case of the optimal total pairwise connectivity after deleting S equal to 0, k can be set as $k < |E|/\Delta$ where Δ is the maximum degree of a given graph G . Approximation solutions to this problem are still open.

The two optimization models can be modified to further assess the network vulnerability under other objective functions. For example, in the context of networks, QoS (quality of services) is an important concept. Therefore, we can consider maximizing the QoS instead of total pairwise connectivity. Other objective functions include the following: minimize the number of pairwise nodes such as their shortest paths in $G[V \setminus S] \leq d$ for some distance d minimize the *connected* vertex disruptor. All these problems can be studied on a probabilistic graph $G = (V, E)$ where each edge $e \in E$ has a failure probability $p_e \in [0, 1]$, which models a dynamic network.

As many complex networks such as the Internet, WWW, biological networks, and social networks can specifically be modeled as power-law graphs, where the number of nodes of degree i is $\lfloor e^\alpha / i^\beta \rfloor$ for some constants α and β , it would be interesting to investigate the proposed problems (β -ED, β -VD, CVD, and CED) in power-law networks. As shown early, these problems still remain NP-complete on power-law graphs. Even though, does the power-law distribution play any key factor on the inapproximability? Are we able to obtain a smaller such as a constant approximation algorithm for these problems on power-law graphs?

Recommended Reading

1. R. Albert, H. Jeong, A.-L. Barabasi, Error and attack tolerance of complex networks. *Nature* **406**, 378–482 (2000)
2. R. Ravi, D. Williamson, An approximation algorithm for minimum cost vertex connectivity problems, in *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms* (ACM, New York, 1995), pp. 332–341
3. S.P. Borgatti, Identifying sets of key players in a social network. *Comput. Math. Org. Theory* **12**(1), 21–34 (2006)
4. E. Zio, C.M. Rocco, D.E. Salazar, G. Muller, Complex Networks Vulnerability: a multiple-objective optimization approach, in *Proceedings of Reliability and Maintainability Symposium*, Orlando, 2007, pp. 196–201
5. L.C. Freeman, Centrality in social networks i: conceptual clarification. *Soc. Netw.* **1**(3), 215–239 (1979)
6. P.K. Srimani, Generalized fault tolerance properties of star graphs source, in *Proceedings of the International Conference on Computing and Information*, Ottawa, Orlando, 1991, pp. 510–519
7. M. Duque-Anton, F. Bruyaux, P. Semal, Measuring the survivability of a network: connectivity and rest-connectivity. *Europ. Trans. Telecommun.* **11**(2), 149–158 (2000)

8. R. Tarjan, Depth first search and linear graph algorithms. *SIAM J. Comput.* **1**, 146–160 (1972)
9. D. Goyal, J. Caffery, Partitioning avoidance in mobile ad hoc networks using network survivability concepts, in *Seventh IEEE Symposium on Computers and Communications (ISCC'02)*, Taormina-Giardini Naxos, Orlando, 2002
10. M. Hauspie, J. Carle, D. Simplot, Partition detection in mobile ad hoc networks using multiple disjoint paths set, in *Objects, Models and Multimedia Technology Workshop*, Switzerland, 2003
11. M. Jorgic, I. Stojmenovic, M. Hauspie, D. Simplot-Ryl, Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks, in *Proceedings of 3rd IFIP MED-HOC-NET Workshop*, Bodrum, 2004
12. A.L. Barabasi, H. Jeong, Z. Nada, E. Ravasz, A. Schubert, T. Vicsek, Evolution of the social network of scientific collaborations. *Phys. A* **311**, 590 (2002)
13. E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, M. Yannakakis, The complexity of multiterminal cuts. *SIAM J. Comput.* **23**, 864–894 (1994)
14. L.G. Valiant, Universality considerations in vlsi circuits. *IEEE Trans. Comput.* **30**(2), 135–140 (1981)
15. W. Aiello, F. Chung, L. Lu, A random graph model for massive graphs. in *STOC '00* (ACM, New York, 2000)
16. A. Ferrante, G. Pandurangan, K. Park, On the hardness of optimization in power-law graphs. *Theory. Comput. Sci.* **393**(1–3), 220–230 (2008)
17. http://en.wikipedia.org/wiki/Hopcroft-Karp_algorithm
18. M.R. Garey, D.S. Johnson, The rectilinear steiner tree problem is np complete. *SIAM J. Appl. Math.* **32**, 826–834 (1977)
19. Dénes König, Gráfok és mátrixok. *Matematikai és Fizikai Lapok* **38**, 116–119 (1931)
20. J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science* (IEEE Computer Society, Washington, DC, 1996), p. 627
21. M. Stoer, F. Wagner, A simple min-cut algorithm. *J. ACM* **44**(4), 585–591 (1997)
22. D. Wagner, F. Wagner, Between min cut and graph bisection, in *MFCS '93: Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science* (Springer, London, 1993), pp. 744–750
23. I. Dinur, S. Safra, On the hardness of approximating minimum vertex cover. *Ann. Math.* **162**, 2005 (2004)
24. A. Agarwal, M. Charikar, K. Makarychev, Y. Makarychev, $O(\log n)$ approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems, in *STOC '05: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing* (ACM, New York 2005), pp. 573–581
25. G. Even, J.S. Naor, S. Rao, B. Schieber, Divide-and-conquer approximation algorithms via spreading metrics. *J. ACM* **47**(4), 585–616 (2000)
26. S. Arora, S. Rao, U. Vazirani, Expander flows, geometric embeddings and graph partitioning, in *STOC '04: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing* (ACM, New York, 2004), pp. 222–231
27. S. Arora, E. Hazan, S. Kale, $O(\sqrt{\log n})$ approximation to sparsest cut in $\tilde{O}(n^2)$ time. *SIAM J. Comput.* **39**(5), 1748–1771 (2010)
28. A. Bavelas, A mathematical model for group structure. *Human Org.* **7**, 16–30 (1948)
29. S.P. Borgatti, M.G. Everett, A graph-theoretic perspective on centrality. *Soc. Netw.* **28**(4), 466–484 (2006)
30. D. Koschützki, K.A. Lehmann, S. Peters, L. Richter, D. Tenfelde-Podehl, O. Zlotowski, in *Centrality Indices*. Lecture Notes in Computer Science, vol. 3418 (Springer, Berlin/New York, 2005) pp. 16–61
31. A. Karygiannis, E. Antonakakis, A. Apostolopoulos, Detecting critical nodes for manet intrusion detection systems, in *Proceedings of Security, Privacy and Trust in Pervasive and Ubiquitous Computing*, Lyon, Orlando, 2006
32. M. Sheng, J. Li, Y. Shi, Critical nodes detection in mobile ad hoc network, in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06)*, Vienna, Orlando, 2006

Job Shop Scheduling with Petri Nets

Hejiao Huang, Hongwei Du and Farooq Ahmad

Contents

1	Introduction	1668
2	Deadlock-Free Design of JSSP with Petri Nets.....	1669
2.1	Terminology About Petri Nets.....	1670
2.2	Petri Net Modeling of JSSP.....	1672
2.3	Deadlock-Free Design for JSSP with Multi-resources Sharing.....	1675
2.4	A Case Study.....	1680
3	Modular-Based JSSP Design and Makespan Calculation.....	1684
3.1	Operators and Makespan Optimization.....	1685
3.2	Modular-Based Design and Optimizatioan: A Case Study.....	1693
4	Genetic Algorithm for JSSP.....	1698
4.1	Petri Net-Based Modeling and Problem Formulation.....	1699
4.2	Genetic Algorithm for FJSSP.....	1703
4.3	Simulation Experiment.....	1706
5	Conclusion.....	1708
	Cross-References.....	1708
	Recommended Reading.....	1709

Abstract

In job shop scheduling problem (JSSP), deadlock-free design and scheduling optimization are important issues, which are difficult to tackle in the JSSP having multiple shared resources. To deal with these issues, this chapter introduces a modular-based system design method for modeling and optimizing JSSP. Several operators are presented for handling the combination of JSSP modules and

H. Huang (✉) • H. Du

Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, People's Republic of China

e-mail: hjhuang@hitsz.edu.cn; hwdu@hitsz.edu.cn

F. Ahmad

Information Technology, University of Central Punjab, Lahore, Pakistan

e-mail: farooq190@gmail.com

resource sharing. For handling deadlock issues with multi-resources sharing, the conditions on “dead transitions” and “circular waiting” are considered. Based on a transitive matrix, deadlock detection and deadlock recovery algorithms are developed in order to get a deadlock-free system. For each operator, the partial schedule and makespan of each module are obtained. The optimal scheduling of JSSP can be derived from these theoretical results step by step. With the modular-based system design method, one cannot only design a correct system model which is live, bounded, reversible, and deadlock-free and terminate properly but also find the optimal scheduling of JSSP in a formal way. For handling large-scale JSSP, the traditional optimization technologies such as genetic algorithm, branch and bound method, hybrid methods, and rule-based methods can be applied together with the modular-based approach in this chapter. Based on the Petri net model, a genetic algorithm is also introduced here.

1 Introduction

The job shop scheduling problem (JSSP) considered in this chapter can be stated as follows: There are a finite set of n jobs, each of which consists of a chain of operations, and a finite set of m machines, which are able to process at most one operation at a time. Each operation must be processed during an uninterrupted time on some given machine, and the purpose is to find a schedule to allocate the resources rationally and finish the task in time. The allocation of resources, such as machines, robots, workers, and AGVs, has to be considered for the modeling and functioning of JSSP, which is a challenge for a system designer. Among others, a JSSP design has the following characterizations and challenges:

- *Heavy reliance on resource sharing and deadlock avoidance.* In a JSSP, resources such as robots, machines, assembly lines, and buffers are sometimes shared by several processes. Therefore, handling resource sharing becomes an important aspect during JSSP design. It is noteworthy that “sharing” does not imply simultaneous usage while simultaneous usage can be handled by re-enterable codes in software systems, which is not allowed in JSSP, and “sharing” requires a resource to be occupied by some part(s) exclusively during utilization and to be released afterward. In single resource-sharing scheduling, if a resource is occupied by others, the job has no other choice but waits until the resource is released. For multiple resources systems, a wrong order of resource allocation, when the resources are limited or distributed unreasonably, deadlocks, or overflows may occur. In a JSSP, deadlock may be the state of “circular waiting” which occurs unexpectedly. Thus, deadlocks must be detected and prevented during system design. In this chapter, handling deadlock issues will be discussed in Sect. 2.
- *Strong tendency toward component-based architecture.* The processes and resources of a JSSP are modeled, built, or owned by different unrelated parties, at different times, and under different environments. Therefore, in order not to have severe interference in their individual developments, it is better for a JSSP to adopt a modular-based architecture. The architecture enlarges the reusability

of the system modules and increases the system running speed. However, it faces many difficulties during integration, including operation system, interfaces, and communication. In the literature, net-based approaches concerning integration include place merging, transition merging, path merging, and modular synthesis. The related references can be found in [1, 2]. This chapter handles the integration problem by applying property-preserving composition operators [3–6] such as enable, choice, interleaving, and iteration. For each operator, the partial schedule and makespan of each module are obtained. The optimal scheduling of JSSP can be derived from these theoretical results step by step. With the modular-based system design method, one cannot only design a correct system model which is live, bounded, and reversible and terminate properly but also find the optimal scheduling of JSSP in a formal way. The technical details concerning modular-based design of JSSP with Petri net will be introduced in Sect. 3.

In the literature, much work involved the research on the above features. The description of the related approaches and a brief review can be found in Deniz K. Terry's PhD thesis [7].

2 Deadlock-Free Design of JSSP with Petri Nets

Petri nets are suitable for modeling sequence, concurrency, choice, and mutual exclusion in discrete event systems, and they have been extensively used to model and analyze the JSSP [8–12]. Further, they are useful for allocating resources and detecting or preventing unexpected system behaviors, such as deadlock and capacity overflow. Motivated by the features of Petri nets, this chapter uses timed Petri nets (TPNs) to model the JSSP, while deadlock detection and recovery are performed by considering the transitive matrix of Petri net model [13].

In the literature, Petri net-based deadlock-free design for JSSP considers the following three types of approaches. The first one is relying on the techniques concerning siphons and traps. Elementary siphon invariants in Petri net structure are useful for analyzing deadlock in flexible manufacturing system (FMS) [14–17]. The deadlock-free method is addressed by adding a monitor or controller to avoid deadlock structure [16–18]. Recently siphons, traps, and invariants have been extensively studied to prevent deadlock and analyze the properties of Petri nets. However, deadlock analysis for compositional system that shares common resources could not grab much attention of the researchers because the number of siphons grows rapidly with the size of final composite net.

The second is about heuristic scheduling algorithm, such as genetic algorithm, used to get an optimum and deadlock-free scheduling of FMS, for example, in [19, 20]. This method is limited to some special problems in JSSP with the given system model; it is not suitable for general cases.

The last one is based on the transitive matrix, which is driven from the directed graph in [13]. The place transitive matrix describes the transferring relation from one place to another place through transitions. The analysis of the cyclic scheduling for the determination of the optimal cycle time is studied, and transitive matrix

has efficiently been used to slice off some subnets from the original net in [21]. Deadlock-free conditions are given in [22, 23] based on transitive matrix, and an algorithm for finding deadlock is given in [24] by using the place transitive matrix.

In this chapter, JSSP is studied by using the TPNs. Further, modular-based architecture is adopted and transitive matrix-based deadlock detection and recovery algorithms are presented. Firstly, each job is modeled by a “small” Petri net, and then these “small” Petri nets, representing different jobs to be completed, are merged together by fusing the common resource places. In multi-resources sharing, an efficient algorithm based on place transitive matrix is presented to detect deadlock for each two jobs of JSSP. In addition, a deadlock recovery method is introduced to give the deadlock-free schedule. Finally, a deadlock-free method for system design is obtained based on these two algorithms. A case study is presented to illustrate the efficiency of the given methods.

2.1 Terminology About Petri Nets

In this subsection, the basic definitions about Petri net and its structural classification are presented for completeness. The related terminology is mostly taken from [13, 25–29].

Definition 1 (Petri Net) A *Petri net* (N, M_0) is a net $N = (P, T, F, W)$ with an initial marking M_0 where, P is a finite set of *places*; T is a finite set of *transitions* such that $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$; $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*; and W is a *weight function* such that $W(x, y) \in N^+$ if $(x, y) \in F$ and $W(x, y) = 0$ if $(x, y) \notin F$. M_0 is a function $M_0: P \rightarrow N$ such that $M_0(p)$ represents the number of *tokens* in place $p \in P$.

Definition 2 (Pre-set and Post-set) For $x \in P \cup T$, $\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$ are called the *pre-set (input set)* and *post-set (output set)* of x , respectively. For a set $X \subseteq P \cup T$, $\bullet X = \bigcup_{x \in X} \bullet x$ and $X^\bullet = \bigcup_{x \in X} x^\bullet$.

Definition 3 (Marking) A *marking* $M(k) = [p_1(k), p_2(k), \dots, p_m(k)]^T$ is an m -vector of nonnegative integer, where $p_i(k)$ is the number of tokens in place p_i immediately after firing the k th transition in a firing sequence. Especially, $M(0)$ denotes the initial marking; it is the same as M_0 .

Definition 4 (Incidence Matrix) B^- is an $m \times n$ matrix of input function, where the entry of row i column j is $B^-[i, j] = W(p_i, t_j)$; B^+ is an $m \times n$ matrix of output function, where the entry in row i column j is $B^+[i, j] = W(t_j, p_i)$. $B = B^+ - B^-$ is called *incidence matrix*.

Definition 5 (Transitive Matrix) Let L_P be *place transitive matrix*, and let L_V be *transition transitive matrix* with m rows and m columns, where $L_P = B^-(B^+)^T$

and $L_V = (B^+)^T B^-$. The *labeled place transitive matrix* L_{VP} is an $m \times m$ matrix satisfying $L_{VP} = B^- \text{diag}(t_1, t_2, \dots, t_n) (B^+)^T$, where the elements of L_{VP} describe the direct transferring relation from one place to another through one or more transitions.

Definition 6 (Weighted Place Transitive Matrix) The *Weighted place transitive matrix* is denoted as L_{VP}^* which is a transformation from L_{VP} : If a transition t_k appears s times in the same column of L_{VP} , then replace t_k in L_{VP} by t_k/s in L_{VP}^* ; otherwise the elements in L_{VP}^* are the same as L_{VP} .

Definition 7 (Path) For a Petri net $(N, M_0) = (P, T, F, M_0)$, a *path* $\sigma = x_1 x_2 \dots x_n$ is a sequence of places and transitions such that $(x_i, x_{i+1}) \in F$, $1 \leq i \leq n$; $\sigma = x_1 x_2 \dots x_n$ is said to be *elementary* if $\forall x_i, x_j \in \sigma, i \neq j \Rightarrow x_i \neq x_j$.

Definition 8 (Transition (Resp., Place) Path) For an elementary path $\sigma = x_1 x_2 \dots x_{2n}$ in a Petri net $N = (P, T, F, M_0)$, $\sigma' = x_1 x_3 \dots x_{2n-1}$ is said to be a *transition (resp., place) path* of σ if $x_i \in T$ (resp., $x_i \in P$), $i=1, 3, \dots, 2n-1$.

Definition 9 (Timed Petri Net) A *TPN* (N, M_0) is a Petri net with a time parameter. That is $N = (P, T, F, \tau)$, where P, T, F , and M_0 are the same as given in [Definition 1](#) and τ is a time function $\tau: T \rightarrow N$ (set of positive integers) associated to transitions such that for $t \in T$, $\tau(t)$ represents the time duration of firing t .

Definition 10 (State Machine, Augmented State Machine) A Petri net (PN) is said to be a *state machine* (SM) if and only if $\forall t \in T: |t| = |\dot{t}| = 1$. An *augmented state machine* (ASM) is a Petri net $(N, M_0) = (P \cup R, T, I, O, M_0)$ whose places consist of two disjoint subsets P and R such that the following conditions hold:

1. The net N obtained by removing all the resource places in R is a state machine.
2. Each resource place $r \in R$ is associated with k pairs of transitions (t_i, t_j) , $i, j = 1, 2, \dots, k$, where $r^\cdot = \{t_i | i = 1, 2, \dots, k\}$, $r^\cdot = \{t_j | j = 1, 2, \dots, k\}$.
3. Initially, the place in P without input transitions are marked by M_0 and other places are not marked.
4. $M_0(p) = 1, \forall p \in R$.

Definition 11 (Place Fusion) Consider two Petri nets $(N_1, M_{10}) = (P_1, T_1, F_1, M_{10})$ and $(N_2, M_{20}) = (P_2, T_2, F_2, M_{20})$, where $P_1 \cap P_2 = R \neq \emptyset$. Let $(N, M_0) = (P, T, F, M_0)$ be composed from (N_1, M_{10}) and (N_2, M_{20}) by operation *place fusion*. Then the elements in (N, M_0) are defined as follows: $P = P_1 \cup P_2$, $T = T_1 \cup T_2$, $F = F_1 \cup F_2$, and

$$M_0(p) = \begin{cases} M_{10}(p) & p \in P_1 \\ M_{20}(p) & p \in P_2 \\ \max\{M_{10}(p), M_{20}(p)\} & p \in R \end{cases}$$

Definition 12 (Place Invariant and Minimal Place Invariant) For a Petri net (N, M_0) , a place invariant is an m -vector $y \geq 0$ such that $B^T \cdot y = 0$, where B is $m \times n$ incidence matrix. It is represented by nonzero entries of vector $y \geq 0$, which is called the support of place invariant. Place invariant is said to be *minimal* if no proper subset of the support is also a support.

Definition 13 (Siphon and Minimal Siphon) A set of places $S \subseteq P$ is called a *siphon* if $\bullet S \subseteq S^\bullet$, and a siphon S is called *minimal* if there does not exist S' such that $S' \subset S$.

2.2 Petri Net Modeling of JSSP

In this subsection, JSSP is described, and its modeling through TPN is explained.

Problem Description. JSSP discussed in this chapter can be stated as follows:

- $J = \{J_1, J_2, \dots, J_n\}$ is a finite set of n jobs.
- $O = \{O_1, O_2, \dots, O_n\}$ is a set of operations for the jobs, and $O_i = \{O_{i1}, O_{i2}, \dots, O_{ik}\}$ is a chain of operations for job J_i , $1 \leq i \leq n$.
- $M = \{M_1, M_2, \dots, M_m\}$ is a finite set of m machines.
- Each machine can handle at most one operation at a time.
- $\{\tau_{i1}, \tau_{i2}, \dots, \tau_{ik}\}$ is the uninterrupted period of a given length for each operation processed in one or some given machines.
- The resources in JSSP may be workers, who are responsible for operating the machines, robots, or other equipments.
- The purpose of JSSP is to find a schedule, that is, a reasonable allocation of the operations to time intervals and machines or other resources such that the makespan is minimized.

In this chapter, TPN is used to model the JSSP, and there are two types of places, that is, *operation places* and *resource places*. First of all, take no account of resources in JSSP, the predefined sequence of operations in each job is formulated as the transition sequence $t_{i1} t_{i2} \dots t_{ik}$. Then add a set of places P to connect these transitions together, such that p_{i1} is the input place of t_{i1} and initially marked with one token. Generally, for each transition t_{ij} , its output place $p_{i(j+1)}$ is the input place of $t_{i(j+1)}$, $j = 1, 2, \dots, k$, and $i = 1, 2, \dots, n$. As a result, each job in the problem is modeled as n state machines with the unique path $p_{i1} t_{i1} p_{i2} t_{i2} \dots p_{ij} t_{ij} \dots p_{ik} t_{ik} p_{i(k+1)}$ ($i = 1, 2, \dots, n$), and each net has an initial marking M_{i0} and a final marking M_{if} .

Now, consider the set of resources R . Each resource here is denoted as a resource place r_i initially marked with a token. When an operation t_{ij} requires some resource r_i , the output of r_i is t_{ij} . Moreover, resources' release follows the "hold while waiting" property: Suppose the resource r_j is needed by operation t_{ij} of job J_i , after operation t_{ij} is finished, if the next operation $t_{i(j+1)}$ of job J_i also needs r_j ,

then r_j continue to be held by operation $t_{i(j+1)}$ of job J_i ; otherwise the resource r_j is released. After adding resource place, the model of each job is an ASM.

When some operations are ready to execute, there will be one token in the operation places, which is called *operation enabled*, and if the resources are available to use, there will be one token in the resource places, which is called *resource enabled*. A transition t_i can be fired only when its operation place and resource place are both enabled.

Property 1 Every ASM for an individual job is executable.

Proof Every ASM without resource places is a sequence of operations represented by the transitions in the form of a state machine. Such type of state machine is executable in sequence if the input place of first operation in sequence is marked by a token. Since marked resource places are added to the transitions in such a way that they become output and/or input places of transitions in the form of ASM, furthermore, resources are used exclusively and follow the “hold while waiting” property, every ASM is executable in sequence.

Property 2 For every ASM of an individual job, the final marking is reachable.

Proof The [Property 2](#) is the consequence of the [Property 1](#).

Property 3 For each $r \in R$, which is not on self-loop in the ASM of an individual job, there exists a minimal place invariant with only marked place $r \in R$.

Proof Each resource place, which is not on a self-loop, represents a resource utilized by the sequence of operations. Further, each such type of resource place is added to the sequence of transitions representing the chain of operations in such a way that it become input place for the first transition and output place for last transition in that sequence. Therefore, allocation of each resource place to the chain of operations is represented by the directed cycle with only marked place $r \in R$ representing the availability of a resource. Hence, allocation of resources imposes the existence of minimal place invariant for each $r \in R$ which is not on a self-loop.

Property 4 Every place invariant in the ASM of an individual job is a minimal siphon with only one marked place $r \in R$.

Proof From [Property 3](#), since place invariants in ASM are the only directed cycles, each place invariant stands for a minimal siphon with only marked place $r \in R$.

The cost of the implementation of operation is described as time function $\tau(t)$ in TPN for transition t . As described above, each job is modeled as an ASM. Since there are identical shared resources in each ASM, these ASM can be merged

together by applying place fusion operator, and a compositional net is built for modeling the whole JSSP. Thus, the scheduling purpose is to find a firing sequence in the resultant Petri net model from the initial marking M_0 to the final marking M_f and to get a minimum makespan. If there is no common resource in two jobs, the makespan is the maximal one of these two sub-models since the two jobs can be processed in parallel.

Property 5 The compositional net, obtained by merging the ASMs of individual jobs, is an ASM.

Proof Since the compositional net fulfills the conditions of ASM described in [Definition 10](#), which are given below:

1. The net N obtained by removing all the resource places in R is a state machine.
2. Each resource place $r \in R$ is associated with k pairs of transitions (t_i, t_j) , $i, j = 1, 2, \dots, k$, where $\dot{r} = \{t_i | i = 1, 2, \dots, k\}$, $\dot{r} = \{t_j | j = 1, 2, \dots, k\}$.

Therefore, the compositional net, which is obtained by merging the independent ASMs modeling different jobs to be completed in JSSP, is again an ASM.

Property 6 Every cycle in the compositional ASM is marked.

Proof R is the only set of marked places in ASM. From [Property 3](#), by removing every place $r \in R$, there will be no place invariant and consequently no directed cycle in ASM. Thus, every cycle in compositional ASM contains a place from R , which is marked.

Property 7 For a compositional ASM, for every $r \in R$, there exists a minimal siphon which contains only one marked place r .

Proof According to [Property 6](#), every cycle contains a place from R . From [Property 4](#), every minimal siphon in ASM contains a cycle. Therefore, every siphon in the compositional ASM, which contains at least one minimal siphon, is marked with $r \in R$.

[Figure 1](#) is an example of two jobs of JSSP sharing two machines. Each job contains two operations which are associated with time functions $\tau(t_1), \tau(t_2), \tau(t_3)$, and $\tau(t_4)$ respectively. The operations t_1, t_2 in Job1 are both processed on machine r_1 , and operations t_3, t_4 in Job2 are processed on machines r_1 and r_2 , respectively. The original states of Job1 and Job2 are $M_{10} = (1, 0, 0, 1)^T$ and $M_{20} = (1, 0, 0, 1, 1)^T$, respectively, and the final states are $M_{1f} = (0, 0, 1, 1)^T$ and $M_{2f} = (0, 0, 1, 1, 1)^T$, respectively. Since machine r_1 is occupied by both jobs, these two sub-models are merged together by place fusion, and the resultant Petri net model is shown in [Fig. 2](#). Then the scheduling problem is to find a firing sequence in the resultant Petri net,

Fig. 1 Timed Petri net models for two jobs of JSSP

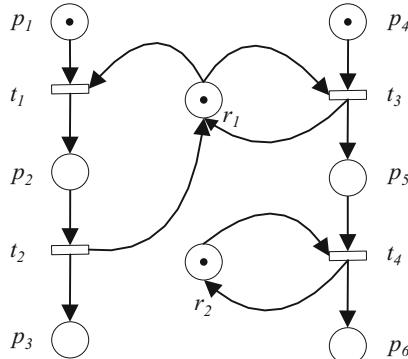
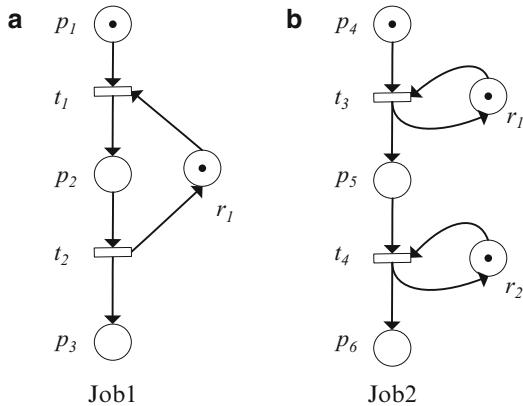


Fig. 2 The merging model of JSSP

from the initial marking $M_0 = (1, 0, 0, 1, 0, 0, 1, 1)^T$ to the final marking $M_f = (0, 0, 1, 0, 0, 1, 1, 1)^T$ such that the makespan is minimized.

2.3 Deadlock-Free Design for JSSP with Multi-resources Sharing

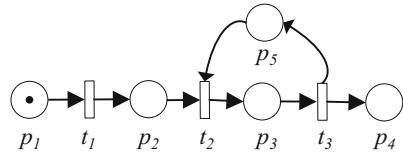
Deadlock is an unexpected state. In a multi-resource-sharing system, deadlock may occur when the resources are limited or distributed unreasonably. In this section, an algorithm is presented to detect deadlock and describe a method to recover deadlock for multi-resource-sharing systems.

2.3.1 Basics of Deadlock

• Dead Marking

In JSSP model, a marking M is said to be a *dead marking* if M is not the final marking M_f , and there is no transition which can be fired at this marking M . Formally, the dead marking can be expressed as $\forall M \in R(N, M_0)$, if $M \neq M_f$ and $\forall t \in T : \neg M[t >]$, then M is a *dead marking*.

Fig. 3 An example to illustrate dead transition



- *Dead Transition*

In JSSP model, a transition t is said to be a *dead transition* if t is operation enabled but not resource enabled. In this chapter, dead transition always appears in the post-set of the operation place with tokens in dead marking.

[Figure 3](#) illustrates the dead transition in a Petri net model. The initial marking is $M_0 = (1, 0, 0, 0, 0)^T$ and the final marking is $M_f = (0, 0, 0, 1, 0)^T$. In this model, after t_1 fires, the marking $M_1 = (0, 1, 0, 0, 0)^T \neq M_f$. Since there are no tokens in p_5 , t_2 cannot be fired and there is no reachable marking for M_1 . Then marking M_1 is a dead marking and transition t_2 is a dead transition.

- *Circular Waiting*

In manufacturing system, “*circular waiting*” is the state that for two or more tasks, each of them holds a machine, but all of them are waiting for each other to release the machine. In this chapter, the state of two tasks is considered. It is one of the inducements of deadlock in manufacturing system.

In Petri net, “*circular waiting*” can be demonstrated as follows: for Petri nets N_i ($i = 1, 2$), r_1 and r_2 are resource places in N_i , and $t_{ia} t_{ib} t_{ic} t_{id}$ ($i = 1, 2$) is a transition path associated with resource place. In N_1 , r_1 is associated with the pairs of transitions (t_{1a}, t_{1c}) , and r_2 is associated with the pairs of transitions (t_{1b}, t_{1d}) ; in addition, $\dot{r}_1 = t_{1a}$, $\dot{r}_1 = t_{1c}$ and $\dot{r}_2 = t_{1b}$, $\dot{r}_2 = t_{1d}$. in N_2 , r_1 is associated with the pairs of transitions (t_{2a}, t_{2c}) , and r_2 is associated with the pairs of transitions (t_{2b}, t_{2d}) ; in addition, $\dot{r}_1 = t_{2b}$, $\dot{r}_1 = t_{2d}$ and $\dot{r}_2 = t_{2a}$, $\dot{r}_2 = t_{2c}$.

[Figure 4](#) is a model of “*circular waiting*” which satisfies $\dot{r}_1 = \{t_{1a}, t_{2b}\}$, $\dot{r}_1 = \{t_{1c}, t_{2d}\}$ and $\dot{r}_2 = \{t_{1b}, t_{2a}\}$, $\dot{r}_2 = \{t_{1d}, t_{2c}\}$.

- *Deadlock*

Based on the place fusion in JSSP, *deadlock* occurs when there are dead transitions that cause “*circular waiting*” in the system.

- *Deadlock-Free*

A Petri net system is said to be *deadlock-free* if there is at least one firing sequence from the initial marking M_0 to the final marking M_f ; otherwise it is said that the system contains *deadlock*.

- *Marking Transformation*

The *transformation of marking* is defined as $M_R(k+1)^T = M(k)^T L_{VP}^*$, where $M_R(k+1)$ is an m -vector of nonnegative integer and is a marking calculated from reachable marking $M(k) = [p_1(k), p_2(k), \dots, p_m(k)]^T$ and L_{VP}^* is the weighted place transitive matrix ([Definition 6](#)) [13].

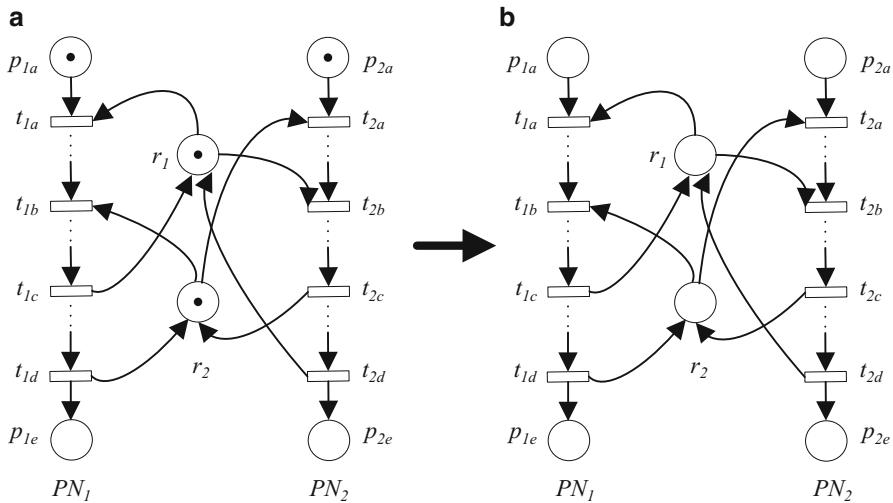


Fig. 4 Circular waiting

The transformation equation shows the flow relation of a token in $p_i(k)$ from $M(k)$ to $M_R(k+1)$ by the weighted place transitive matrix. In this equation, if the coefficient of t_i in the function $\sum_i p_i L_{VP_{ij}}^*$ is an integer, then let $\sum_i p_i L_{VP_{ij}}^* = 1$, which means tokens in $p_i(k)$ can be transferred from $M(k)$ to $M_R(k+1)$; otherwise, let $\sum_i p_i L_{VP_{ij}}^* = 0$, which means tokens in $p_i(k)$ cannot be transferred from $M(k)$ to $M_R(k+1)$.

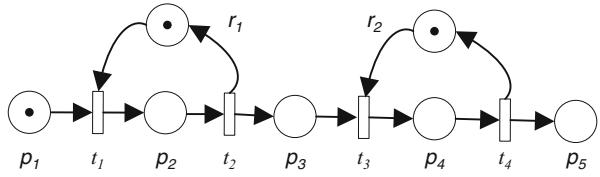
However, $M_R(k+1)$ only presents the flow relation of tokens; it may not correspond to a reachable marking $M(k+1)$. In order to improve this situation, some constrained conditions are added to this equation:

Condition (1) When the coefficient of t_i in the function $\sum_i p_i L_{VP_{ij}}^*$ is an integer, for example, $\sum_i p_i L_{VP_{ij}}^* = 1$ in $M_R(k+1)$, and the tokens in these places appearing in this function are consumed, then these tokens can be transferred from $M(k)$ to $M(k+1)$.

Condition (2) When the coefficient of t_i in the function $\sum_i p_i L_{VP_{ij}}^*$ is not an integer and the tokens in the corresponding places of this entry are not previously consumed, then the tokens in these places are preserved; otherwise let $\sum_i p_i L_{VP_{ij}}^* = 0$ in $M(k+1)$.

From this calculation, a series of reachable marking can be found from M_0 . In addition, if there is no marking reachable from M_i , then there are dead transitions at M_i . Thus, the system contains deadlock. [Figure 5](#) is an

Fig. 5 An example to explain these conditions



example to illustrate these conditions. The initial marking of Fig. 5 is $M_0 = (1, 0, 0, 0, 0, 1, 1)^T$, and the weighted place transitive matrix is $L_{VP}^* =$

$$\begin{array}{l} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ r_1 \\ r_2 \end{array} \left[\begin{array}{cccccc} 0 & t_1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t_2 & 0 & 0 & t_2 & 0 \\ 0 & 0 & 0 & t_3/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & t_4 & 0 & t_4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & t_1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & t_3/2 & 0 & 0 & 0 \end{array} \right].$$

$$M_R(k+1)^T = M(k)^T L_{VP}^* = [0, t_1(p_1(k) + r_1(k))/2, t_2(p_2(k)), t_3(p_3(k) + r_2(k))/2, t_4(p_4(k)), t_2(p_2(k)), t_4(p_4(k))]^T.$$

Since $p_1(0) = 1$, $r_1(0) = 1$ and $r_2(0) = 1$ in the initial marking M_0 , $M_R(1) = M(0)L_{VP}^* = M_0L_{VP}^* = [0, t_1(1+1)/2, t_2*0, t_3(0+1)/2, t_4*0, t_2*0, t_4*0]^T = [0, t_1, 0, t_3/2, 0, 0, 0]^T$.

The second element in $M_R(1)$ is t_1 and the coefficient of t_1 is an integer; at the same time, the tokens in $p_1(0)$ and $r_1(0)$ are consumed, and by Condition (1), the corresponding value in reachable marking M_2 is 1. On the other side, the coefficient of t_3 is 1/2 and the corresponding place is $r_2(0)$ whose tokens are not consumed previously; by Condition (2), the tokens in $r_2(0)$ are preserved. Consequently, the reachable marking of M_0 is $M_1 = (0, 1, 0, 0, 0, 0, 1)^T$. Based on the two constraints, a reachable marking can be obtained by calculating the transformation of marking.

2.3.2 Deadlock Detection

In this subsection, the transformation equation is applied to identify the dead transitions in two Petri nets. Based on the dead transition and the conditions of “circular waiting,” a deadlock detection algorithm is proposed to find out whether there is deadlock or not in the compositional system.

2.3.3 Deadlock Recovery

When there is a deadlock in the net, the system falls into the local deadlock state and even stops running. Therefore, deadlock recovery is important for system design. In this subsection, an efficient method is presented to recover a deadlock for the composition of two Petri nets.

Since the deadlock is caused by “circular waiting” and by removing it, deadlock can be removed. In the following, the recovery method is proposed based on the definition of “circular waiting” and “self-loop” in Petri net.

Algorithm 1: Deadlock detection

Input: $N_1 = (P_1, T_1, F_1, M_{10}, \tau_1)$, $N_2 = (P_2, T_2, F_2, M_{20}, \tau_2)$ (where $P_1 \cap P_2 = R = \{r_1, r_2\}$), the initial marking M_{10} , M_{20} , and the final marking M_{1f} , M_{2f}

Output: Deadlock or deadlock-free

- Step 1: Do place fusion on N_1 and N_2 , and then generate a net N with the initial marking M_0 and the final marking M_f .
- Step 2: Construct the weighted place transitive matrix- L_{VP}^* for compositional net N .
- Step 3: For $k = 0, k < n, k++$:
 - Calculate the next marking $M_R(k+1)$ from the marking $M(k)$ by $M_R(k+1)^T = M(k)^T L_{VP}^*$.
 - If $M_R(k+1) = M_f$, stop and output “there is no deadlock in the net.”
 - Else if there are dead transitions in the net and they form a “circular waiting,” then output “there is a deadlock.”
 - Else output is “there is flaw in the system model.”

Algorithm 2: Deadlock recovery

Input: Dead transitions which cause deadlock (suppose they are t_{1b} and $t_{1b'}$)

Output: A deadlock-free model

- Step 1: Compare #($t_{10}t_{11} \dots t_{1(a-1)}$) and #($t_{20}t_{21} \dots t_{2(a-1)}$), and find out the larger quantity, without loss of generality, suppose it is #($t_{10}t_{11} \dots t_{1(a-1)}$).
 - Step 2: Add two new arcs ($t_{1(b-1)}, r_1$) and (r_1, t_{1b}) into the model, and then the pair (t_{1a}, t_{1c}) is decomposed as ($t_{1a}, t_{1(b-1)}$) and (t_{1b}, t_{1c}) which are both associated with r_1 .
 - Step 3: Detect deadlock by [Algorithm 1](#).
- If output is “there is a deadlock,” do the recovery again by repeating Steps 1 and 2.
Else, output is “there is no deadlock in the net.”

Let N_1 and N_2 be two Petri net sub-models, with two kinds of common resource places r_1 and r_2 . They satisfy the “circular waiting” conditions mentioned in [Sect. 2.3.1](#). Suppose $\sigma_i = t_{i0} t_{i1} \dots t_{im}$ is a transition path in N_i , $\sigma'_i = t_{ia} t_{ib} t_{ic} t_{id}$ is a sub-path of σ_i ($i = \{1, 2\}$), and #($t_{i0} t_{i1} \dots t_{i(a-1)}$) is the makespan of $t_{i0} t_{i1} \dots t_{i(a-1)}$.

After the recovery given above, the subsystem becomes deadlock-free. The method proposed above only changes the input and output of one resource place in one Petri net sub-model. If the complexity of the model is not considered, the similar change is done in the second Petri net sub-model. After this change, the Petri net model can be live and the makespan can also be minimized.

2.3.4 Deadlock-Free Design

Based on the algorithms given above, a compositional deadlock-free model for a multitasking JSSP can be constructed. The detailed process is as shown in [Algorithm 3](#).

Algorithm 3: Deadlock-free design**Input:** n Petri nets**Output:** A compositional deadlock-free model for these n Petri nets

- Step 1: For each pair of Petri nets which have common resource places, run [Algorithm 1](#) to detect deadlock.
- Step 2: If there is a deadlock in any two Petri nets, run [Algorithm 2](#) to recover deadlock, and go to Step 3. Else, if all of these pairs are deadlock-free, go to Step 3.
- Step 3: Merge these n nets together by place fusion to get a deadlock-free model for this JSSP.

2.4 A Case Study

A JSSP example with four jobs and two kinds of resources (e.g., machines and a worker) is shown in [Table 1](#). J_1, J_2, J_3 , and J_4 are four jobs; O_1, O_2, O_3 , and O_4 are operations of each job; M_1 , and M_3 are machines and M_2 is the worker; and the integers following M_i represent the processing time of each operation. The second and the third operations of each job need both machine and worker to process, but the first and the fourth operations of each job need only one resource. In order to obtain a schedule, this problem is illustrated step by step:

Step 1: Construct the model for this JSSP:

- According to [Sect. 2.3](#), the Petri nets (ASMs) for four jobs are depicted in [Fig. 6](#), and $t_{i1}, t_{i2}, t_{i3}, t_{i4}$ are corresponding to the four operations in job i ($i = \{1, 2, 3, 4\}$); r_1, r_2 , and r_3 are corresponding to three resources.
- When there are more than two shared resources in two jobs, deadlock may occur. Therefore, after the system model is built, deadlock detection should be done to make sure the system is deadlock-free. Deadlock detection is performed for each compositional ASM obtained by merging the pair of ASMs of individual jobs. Since there are four Petri nets (ASMs for individual jobs), totally there are six detections.

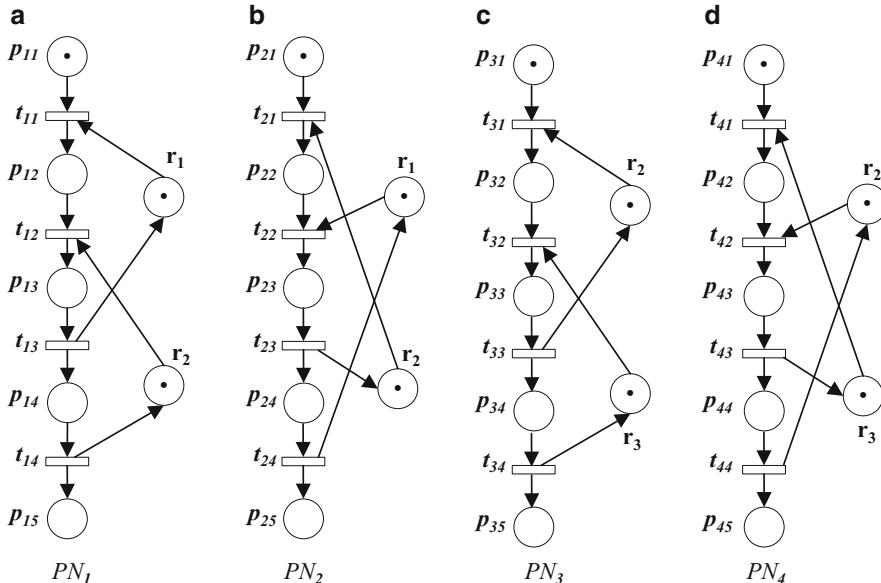
Step 2: Choose two Petri nets with shared resources and detect deadlock; if there exists a deadlock, recover it:

- Consider Petri nets N_1 and N_2 . [Figure 7](#) is the compositional system for N_1 and N_2 . In the model N_{12} , $M(k) = [p_{11}(k), p_{12}(k), p_{13}(k), p_{14}(k), p_{15}(k), p_{21}(k), p_{22}(k), p_{23}(k), p_{24}(k), p_{25}(k), r_1(k), r_2(k)]^T$, the initial marking $M(0) = M_0 = (1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1)^T$ and the final marking $M_f = (0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1)^T$.
- Construct the weighted place transitive matrix- L_{VP}^* for [Fig. 7](#): [Table 2](#) is the weighted place transitive matrix, $M(k) = [p_{11}(k), p_{12}(k), p_{13}(k), p_{14}(k), p_{15}(k), p_{21}(k), p_{22}(k), p_{23}(k), p_{24}(k), p_{25}(k), r_1(k), r_2(k)]^T$, and the initial marking is $M_0 = (1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1)^T$.
- Calculate the reachable marking by $M_R(k+1)^T = M(k)^T L_{VP}^*$, and get the next reachable marking:

$$M_R(k+1)^T = M(k)^T L_{VP}^* = [0, t_{11}(p_{11}(k)+r_1(k))/2, t_{12}(p_{12}(k)+r_2(k))/2, t_{13}(p_{13}(k)+r_3(k))/2, t_{14}(p_{14}(k)+r_3(k))/2, t_{15}(p_{15}(k))/2, t_{21}(p_{21}(k)+r_2(k))/2, t_{22}(p_{22}(k)+r_1(k))/2, t_{23}(p_{23}(k))/2, t_{24}(p_{24}(k))/2, t_{25}(p_{25}(k))/2, r_1(k)t_{21}, r_2(k)t_{22}, r_3(k)t_{23}]^T.$$

Table 1 A job shop scheduling problem

	J_1	J_2	J_3	J_4
O_1	$M_1: 7$	$M_2: 3$	$M_2: 5$	$M_3: 6$
O_2	$M_1, M_2: 4$	$M_1, M_2: 5$	$M_2, M_3: 3$	$M_3, M_2: 4$
O_3	$M_1, M_2: 3$	$M_1, M_2: 4$	$M_2, M_3: 5$	$M_3, M_2: 5$
O_4	$M_2: 4$	$M_1: 3$	$M_3: 4$	$M_2: 3$

**Fig. 6** The Petri net models of the JSSP

$M_R(1) = M(0)L_{VP}^* = [0, t_{11}(p_{11}(k) + r_1(k))/2, t_{12} r_2(k)/2, 0, 0, 0, t_{21}(p_{21}(k) + r_2(k))/2, t_{22} r_1(k)/2, 0, 0, 0, 0]^T$, based on the two conditions presented in Sect. 2.3.1, $M(1) = [0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]^T$.

- Analyze if there exists a deadlock or not: in $M(2)$, transitions t_{12} and t_{22} are dead transitions. Since r_1 and r_2 are associated with transitions t_{12} and t_{22} , which involve in “circular waiting”, the system is confronted a deadlock.
- Recover the deadlock for N_1 and N_2 :
 - Since ($\# t_{11} = 7$) is greater than ($\# t_{21} = 3$), then the first sub-model should be modified.
 - Add two arcs t_{11}, r_1 and (r_1, t_{12}) , and then (t_{11}, t_{11}) and (t_{12}, t_{13}) are the new transition pairs associated with resource r_1 (Fig. 8 is the model after deadlock recovery).
 - Detect deadlock again; since there is no deadlock, stop.

Fig. 7 The model after place fusion

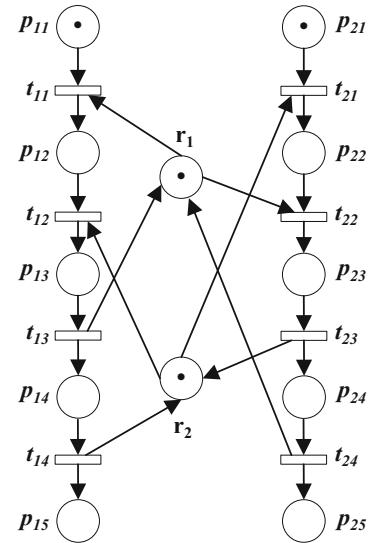


Table 2 Place transitive matrix of Petri net in Fig. 7

$$L_{VP}^* = \begin{bmatrix} p_{11} & 0 & t_{11}/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_{12} & 0 & 0 & t_{12}/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_{13} & 0 & 0 & 0 & t_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{13} \\ p_{14} & 0 & 0 & 0 & 0 & t_{14} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{14} \\ p_{15} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_{21} & 0 & 0 & 0 & 0 & 0 & 0 & t_{21}/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ p_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{22}/2 & 0 & 0 & 0 & 0 & 0 \\ p_{23} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{23} & 0 & 0 & 0 & t_{23} \\ p_{24} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t_{24} & t_{24} & 0 & 0 \\ p_{25} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ r_1 & 0 & t_{11}/2 & 0 & 0 & 0 & 0 & 0 & t_{22}/2 & 0 & 0 & 0 & 0 & 0 \\ r_2 & 0 & 0 & t_{12}/2 & 0 & 0 & 0 & t_{21}/2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Step 3: Choose other pairs of Petri nets and do the same work like Step 2:

- The Petri net pairs are (N_1, N_3) , (N_1, N_4) , (N_2, N_3) , (N_2, N_4) , and (N_3, N_4) .
- For the pairs (N_1, N_3) , (N_1, N_4) , (N_2, N_3) , and (N_2, N_4) , there is no deadlock in each of them. However, there is deadlock in integrating N_3 and N_4 .
- Through analyzing the transformation equation $M_R(k+1)^T = M(k)^T L_{VP}^*$, the dead transitions t_{32} and t_{42} are found, which cause deadlock.
- Recover deadlock on transition t_{32} and t_{42} , and the resultant model is similar to Fig. 8.

Fig. 8 The recovery model for N_1 and N_2

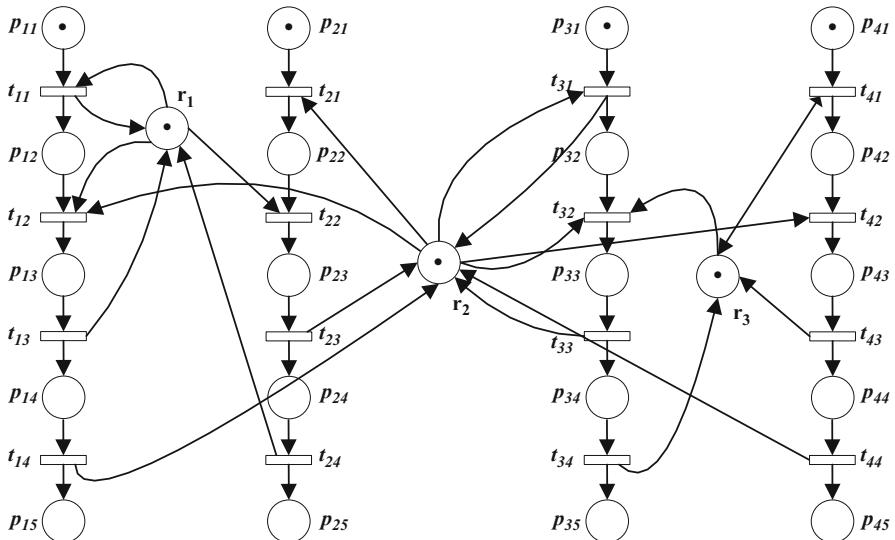
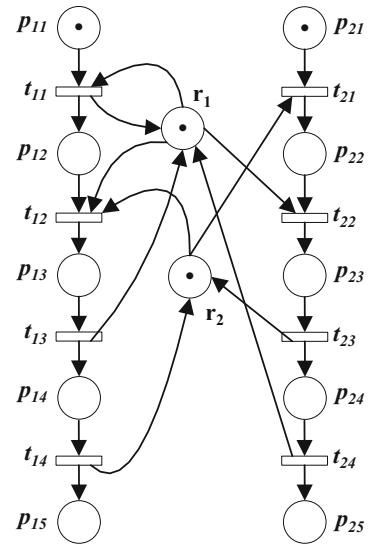


Fig. 9 The deadlock-free model of the given JSSP

Step 4: Deadlock-free design for JSSP

After deadlock detection and recovery, there is no deadlock in each two sub-models. By [Algorithm 3](#), do place fusion on these four Petri nets and a deadlock-free model is obtained ([Fig. 9](#)).

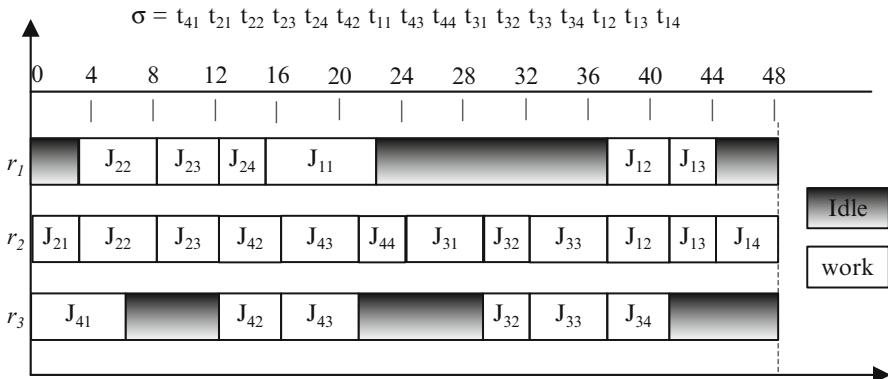


Fig. 10 The Gantt chart of the given JSSP

Step 5: Scheduling solution

Since it is deadlock-free, a scheduling solution for this JSSP can be obtained. Here CPN Tool [30] is applied to simulate the resolutions and choose a best one. The minimum makespan in the report is 48, and one optimal schedule is given in Fig. 10.

3 Modular-Based JSSP Design and Makespan Calculation

In this section, several operators for the design specification of JSSP are presented. For each operator, the property about the makespan calculation is also provided. Based on these operators and properties, a larger and more complex module, even for the whole system, can be deduced from the smaller ones step by step, and the schedule of larger modules can also be calculated.

As a component of a variety of methodologies developed to generate JSSP schedules, Petri net is very powerful for modeling the concurrent processing, resource sharing, routing flexibility, sequential steps, concurrent use of multiple resources, machine setup times, job batch sizes, and so on in JSSP [31–34]. As for schedule optimization, there are two types of Petri net-based methodologies for JSSP: (1) heuristic searching algorithms based on reachability graph of Petri net models [35–37] and (2) GA algorithms for optimizing scheduling [38–40]. Petri nets are attracting the attention of many researchers for the design of JSSP and the optimization of the scheduling of the system [41–43].

However, current Petri net-based methodologies have the following shortages: (1) the Petri net models are put forward by researchers for solving their specified problems. Therefore, the effect of Petri nets cannot be explored efficiently. Although it is good for obtaining better optimization results, they cannot be applied for general problems. (2) the real-time JSSP is usually quite complex and large; consequently,

the size of the Petri net model will be quite large. The current Petri net analysis methods are not applicable because of the state exploding problem of Petri nets.

In order to overcome the hardship mentioned, this chapter handles JSSP problem from two aspects: system design and scheduling optimization. On system design aspect, this chapter applies a modular-based approach for JSSP design specification and verification. The basic module model is an extended Petri net process (EPNP) similar to the Petri net process (PNP) discussed in [2–4]. The difference is that EPNP is an extended Petri net while PNP is a general Petri net. Hence, all the results on property-preserving Petri net process algebra (PPPA) [3–6] will also be true for the JSSP design in this chapter. With modular-based approach, the JSSP is correct from the viewpoint that the system contains some desired properties such as liveness, boundedness, reversibility, and proper termination.

As for system scheduling optimization, based on EPNP, this chapter provides some information on finding the possible schedule (the firing sequences of the Petri net model) and makespan of the composite modules from the constituent modules for the integration operators in PPPA. In order to handle resource-sharing problem, the place-merging operator and the integration operators are combined for system modeling. The firing sequences and the upper bound of the makespan for these combined operators are concluded. The algorithm for finding the makespan and its corresponding scheduling is also presented for the combined operators. With the modular-based method for system design and scheduling optimization, the schedule and makespan of the composite modules can be calculated based on their constituent modules. In order to reduce the calculation time, traditional algorithms such as GAs and branch-and-bound methods can be applied together with the algorithms presented in this chapter for optimizing the schedule of JSSP. For completeness, a GA algorithm based on Petri net model will be introduced in Sect. 4.

The readers are supposed to have some Petri net-related foundations. Most of the terminologies can be found in [3, 4, 45]. The results in this section are mainly derived from [44].

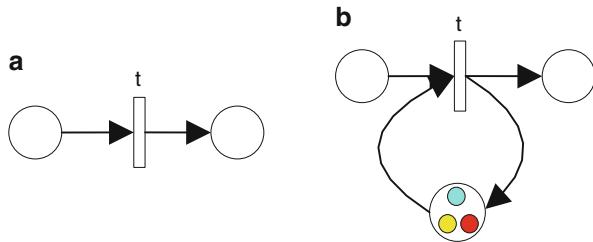
3.1 Operators and Makespan Optimization

Definition 14 An EPNP $B = (N, p_e, p_x)$, where $N = (P, T, F, M_c, C, W, D)$ is a colored TPN (called extended Petri net), p_e is the entry place, and p_x is the exit place of EPNP.

The performance and characteristics, for example, proper initialization, proper termination, and handling resource sharing, of EPNP similar to PNP defined in [3, 4]; the difference is that in EPNP, N is an extended Petri net, while in PNP, N is a general Petri net. Hence, all the theoretical results in [3–6] are also true for EPNP and can be applied to design a correct job shop scheduling system.

Note that since the design specification and verification are similar to [2–4], this chapter will pay much more attention to the design specification and optimization of JSSP.

Fig. 11 The extended Petri net model for operations in JSSP



In job shop scheduling systems, operations either do not need any resources or need resources such as machines, robots, and workers. Hence, the corresponding models for the operations in a JSSP have two different forms.

For the first case, a transition t represents an operation; t has a single input place representing the beginning of the operation and a single output place representing the end of the operation (Fig. 11a). For the second case, one more place representing the resources should be associated to t and a self-loop is formed. Initially, there are some tokens assigned to the resource place. Tokens with different colors are used to represent different resources (Fig. 11b).

Based on the relationship of the operations and the control flow of JSSP, some operators are defined to combine these operations in JSSP.

In this chapter, the basic model is the EPNP. Although EPNP and the general PNP [3, 4] have different structural and behavioral properties and firing rules, the pictorial representation of these operators will be the same as that in [3]. Hence, in this chapter, the same definition and the same pictorial representation as that in Mak's thesis [3] and Huang's thesis [4] are used for the definition of all the operators, that is, action prefix, iteration, enable, choice, interleave, parallel, disable, disable-resume, place merging, and the mix operators.

In the following definition, notation $B = (N, p_e, p_x)$ (resp., $B_i = (N_i, p_{ie}, p_{ix})$) represents an EPNP, where $N = (P, T, F, M_c, C, W, D)$ (resp., $N_i = (P_i, T_i, F_i, M_{ic}, C_i, W_i, D_i)$) is an extended Petri net with an initial static marking M_c , p_e , and p_x (resp., M_{ic} , p_{ie} , and p_{ix}) are the entry and exit places of the process, respectively. The initial marking of B (resp., B_i) is $M_e = p_e + M_c$ (resp., $M_{ie} = p_{ie} + M_{ic}$), and the exit marking of B (resp., B_i) is $M_x = p_x + M_c$ (resp., $M_{ix} = p_{ix} + M_{ic}$).

For each operator defined in this chapter, the firing sequence and the potential partial schedule in the composite process will be concluded based on the possible firing sequences of the constituent processes. These conclusions will be presented as properties. Since the results are trivial, the details of the proof will be omitted. The conclusions can be used to find the optimal scheduling solutions. Some property-preserving results can also be obtained for these operators. Since most property-preserving results can refer to the related results in [3, 4], the behavioral property-preserving results will be omitted in this chapter. While the property-preserving results can be used to specify and verify the JSSP, the work

related to the application of property-preserving technology for system specification and verification can be found in [2, 4].

In JSSP, before processing some operations, a preparation work is usually needed. For example, after a batch of pieces has finished processing, the machine may need repairing before handling the next batch of pieces. In the system modeling, the following formally defined *action prefix* operator is used to combine the preparation action and the operation process. It is a special case of operator *enable* which will be introduced later.

Definition 15 (Action prefix) For a process $B = (N, p_e, p_x)$ and a transition $b \notin T$ representing an action, the operator *action prefix* applied to B , in notation $b; B$, is defined as the process $B' = (N', p_{e'}, p_{x'})$, where $p_{e'}$ is the new entry place, $p_{x'} = p_x$, $P' = P \cup \{p_{e'}\}$, $T' = T \cup \{b\}$, $F' = F \cup \{(p_{e'}, b), (b, p_e)\}$, $M_c' = M_c$, $C' = C$, $W' = W \cup \{w(p_{e'}, b), w(b, p_e)\}$, $D' = D \cup \{d(b)\}$.

Property 8 Suppose $(B', M_{e'})$ is induced from (B, M_e) by applying the operator action prefix. If $M_e[\sigma > M_x]$ in (B, M_e) , then $M_{e'}[b\sigma > M_{x'}]$ in $(B', M_{e'})$. If σ has the minimum makespan in (B, M_e) , then $b\sigma$ has the minimum makespan in $(B', M_{e'})$.

Definition 16 (Iteration) For a process $B = (N, p_e, p_x)$, an operator *iteration* applied to B , in notation B' , is defined as the process $B' = (N', p_{e'}, p_{x'})$, where $P' = P \cup \{p_{e'}, p_{x'}\}$; $T' = T \cup \{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$; $F' = F \cup \{(p_{e'}, \varepsilon_1), (\varepsilon_1, p_e), (\varepsilon_2, p_e), (p_x, \varepsilon_2), (p_x, \varepsilon_3), (\varepsilon_3, p_{x'})\}$, $M_c' = M_c$, $C' = C$, $W' = W \cup \{w(p_{e'}, \varepsilon_1), w(\varepsilon_1, p_e), w(\varepsilon_2, p_e), w(p_x, \varepsilon_2), w(p_x, \varepsilon_3), w(\varepsilon_3, p_{x'})\}$, $D' = D \cup \{d(\varepsilon_1), d(\varepsilon_2), d(\varepsilon_3)\}$.

Iteration modifies a process in such a way that it can be executed repetitively while preserving the option of exiting after any iteration. For example, in JSSP, iteration operator can be used to model the case of processing a batch of pieces in a machine.

Property 9 Suppose $(B', M_{e'})$ is induced from (B, M_e) by applying iteration operator. If $M_e[\sigma > M_x]$ in (B, M_e) , then $M_{e'}[\varepsilon_1 \sigma \varepsilon_2 \sigma \varepsilon_3 \dots \sigma \varepsilon_2 \sigma \varepsilon_3 > M_{x'}]$ in $(B', M_{e'})$. If σ has the minimum makespan in (B, M_e) , then $\varepsilon_1 \sigma \varepsilon_3$ has the minimum makespan in $(B', M_{e'})$.

Definition 17 (Enable) For two processes $B_i = (N_i, p_{ie}, p_{ix})$, $i = 1, 2$, their composition by *enable*, in notation $B_1 \gg B_2$, is defined as the process $B = (N, p_e, p_x)$, where $P = P_1 \cup P_2$, with $p_e = p_{1e}$ and $p_x = p_{2x}$; $T = T_1 \cup T_2 \cup \{\varepsilon\}$; $F = F_1 \cup F_2 \cup \{(p_{1x}, \varepsilon), (\varepsilon, p_{2e})\}$, $M_c = M_{1c} \cup M_{2c}$, $C = C_1 \cup C_2$, $W = W_1 \cup W_2 \cup \{w(p_{1x}, \varepsilon), w(\varepsilon, p_{2e})\}$, $D = D_1 \cup D_2 \cup \{d(\varepsilon)\}$.

Enable $B_1 \gg B_2$ models the execution of two processes B_1 and B_2 in sequence. That is, B_1 is firstly executed and B_2 is executed after the successful termination of B_1 . However, if B_1 does not exit successfully, B_2 will never be activated.

Property 10 Suppose (B, M_e) is induced from (B_1, M_{1e}) and (B_2, M_{2e}) by applying the operator enable. If $M_{1e}[\sigma_1 > M_{1x}]$ in (B_1, M_{1e}) and $M_{2e}[\sigma_2 > M_{2x}]$ in (B_2, M_{2e}) , then $M_e[\sigma_1 \circ \sigma_2 > M_x]$ in (B, M_e) . If σ_1 has the minimum makespan in (B_1, M_{1e}) and σ_2 has the minimum makespan in (B_2, M_{2e}) , then $\sigma_1 \circ \sigma_2$ has the minimum makespan in (B, M_e) .

Definition 18 (Choice) For two processes $B_i = (N_i, p_{ie}, p_{ix})$, $i = 1, 2$, their composition by *choice*, in notation $B_1 [] B_2$, is defined as the process $B = (N, p_e, p_x)$, where $P = (P_1 \cup P_2 - \{p_{1x}, p_{2x}\}) \cup \{p_e, p_x\}$, where p_x is the place merging p_{1x} and p_{2x} ; $T = T_1 \cup T_2 \cup \{\varepsilon_1, \varepsilon_2\}$; $F = F_1' \cup F_2' \cup \{(p_e, \varepsilon_1), (p_e, \varepsilon_2), (\varepsilon_1, p_{1e}), (\varepsilon_2, p_{2e})\}$, where $F'_i = F_i - \{(t, p_{ix}) | t \in {}^* p_{ix}\} \cup \{(t, p_x) | t \in {}^* p_{ix}\}$, $i = 1, 2$, $M_c = M_{1c} \cup M_{2c}$, $C = C_1 \cup C_2$, $W = W_1(F'_1) \cup W_2(F'_2) \cup \{w(p_e, \varepsilon_1), w(p_e, \varepsilon_2), w(\varepsilon_1, p_{1e}), w(\varepsilon_2, p_{2e})\}$, $D = D_1 \cup D_2 \cup \{d(\varepsilon_1), d(\varepsilon_2)\}$.

Choice $B_1 [] B_2$ models the selection for execution between two processes B_1 and B_2 .

Property 11 Suppose (B, M_e) is induced from (B_1, M_{1e}) and (B_2, M_{2e}) by applying the operator choice. If $M_{1e}[\sigma_1 > M_{1x}]$ in (B_1, M_{1e}) or $M_{2e}[\sigma_2 > M_{2x}]$ in (B_2, M_{2e}) , then $M_e[\varepsilon_1 \sigma_1 > M_x]$ and $M_e[\varepsilon_2 \sigma_2 > M_x]$ in (B, M_e) . If σ_1 has the minimum makespan in (B_1, M_{1e}) and σ_2 has the minimum makespan in (B_2, M_{2e}) , then the firing sequence with the minimum makespan in (B, M_e) is $\min\{\varepsilon_1 \sigma_1, \varepsilon_2 \sigma_2\}$.

Definition 19 (Interleave) For two processes $B_i = (N_i, p_{ie}, p_{ix})$, $i = 1, 2$, their composition by *interleave*, in notation $B_1 ||| B_2$, is defined as the process $B = (N, p_e, p_x)$, where $P = P_1 \cup P_2 \cup \{p_e, p_x\}$; $T = T_1 \cup T_2 \cup \{\varepsilon_1, \varepsilon_2\}$; $F = F_1 \cup F_2 \cup \{(p_e, \varepsilon_1), (\varepsilon_1, p_{1e}), (\varepsilon_1, p_{2e}), (p_{1x}, \varepsilon_2), (p_{2x}, \varepsilon_2), (\varepsilon_2, p_x)\}$, $M_c = M_{1c} \cup M_{2c}$, $C = C_1 \cup C_2$, $W = W_1 \cup W_2 \cup \{w(p_e, \varepsilon_1), w(\varepsilon_1, p_{1e}), w(\varepsilon_1, p_{2e}), w(p_{1x}, \varepsilon_2), w(p_{2x}, \varepsilon_2), w(\varepsilon_2, p_x)\}$, $D = D_1 \cup D_2 \cup \{d(\varepsilon_1), d(\varepsilon_2)\}$.

Interleave $B_1 ||| B_2$ models the concurrent but independent execution of two processes B_1 and B_2 with synchronized exit.

Property 12 Suppose (B, M_e) is induced from (B_1, M_{1e}) and (B_2, M_{2e}) by applying the operator interleave. If $M_{1e}[\sigma_1 > M_{1x}]$ in (B_1, M_{1e}) and $M_{2e}[\sigma_2 > M_{2x}]$ in (B_2, M_{2e}) , then $M_e[\varepsilon_1 \circ \sigma_2 > M_x]$ in (B, M_e) , where σ is a transition sequence consisting of all of the transitions in σ_1 and σ_2 such that the appearance order of the transitions conforms to the order in σ_1 and in σ_2 (i.e., the appearance order is preserved). If σ_1 has the minimum makespan in (B_1, M_{1e}) and σ_2 has the minimum makespan in (B_2, M_{2e}) , then σ has the minimum makespan in (B, M_e) and $\#\sigma = \max\{\#\sigma_1, \#\sigma_2\}$, where the notation $\#\sigma$ is the makespan value of the firing sequence σ .

For example, suppose $\sigma_1 = abc$ and $\sigma_2 = de$. Then σ can be any one of the following ten sequences: $deabc, daebc, dabec, dabce, adebc, adbec, adbce, abdec, abdce, and abcde.$

Definition 20 (Composition by the Operator Parallel) For two processes $B_i = (N_i, p_{ie}, p_{ix})$, $i = 1, 2$ with a “common” set of transitions $G = T_1 \cap T_2$ to be synchronized, their composition by *Parallel*, in notation $B_1|G|B_2$, is defined as the process $B = (N, p_e, p_x)$ where $P = P_1 \cup P_2 \cup \{p_e, p_x\}$; $T = T_1 \cup T_2 \cup \{\varepsilon_1, \varepsilon_2\}$, where $T_1 \cap T_2 \neq \emptyset$; $F = F_1 \cup F_2 \cup \{(p_e, \varepsilon_1), (\varepsilon_1, p_{1e}), (\varepsilon_1, p_{2e}), (p_{1x}, \varepsilon_2), (p_{2x}, \varepsilon_2), (\varepsilon_2, p_x)\}$, $M_c = M_{1c} \cup M_{2c}$, $C = C_1 \cup C_2$, $W = W_1 \cup W_2 \cup \{w((p_e, \varepsilon_1), w(\varepsilon_1, p_{1e}), w(\varepsilon_1, p_{2e}), w(p_{1x}, \varepsilon_2), w(p_{2x}, \varepsilon_2), w(\varepsilon_2, p_x))\}$, $D = D_1 \cup D_2 \cup \{d(\varepsilon_1), d(\varepsilon_2)\}$.

Parallel $B_1|G|B_2$ models the concurrent execution of two processes B_1 and B_2 while they have to be synchronized at some points and at their exit points. Synchronization has been one of the most complicated problems in the research of concurrent systems.

Property 13 Suppose (B, M_e) is induced from (B_1, M_{1e}) and (B_2, M_{2e}) by applying the operator parallel. If $M_{1e}[\alpha_1 \tau \beta_1 > M_{1x}$ in (B_1, M_{1e}) and $M_{2e}[\alpha_2 \tau \beta_2 > M_{2x}$ in (B_2, M_{2e}) , then $M_e[\varepsilon_1 \alpha \pi \beta \varepsilon_2 > M_x$ in (B, M_e) , where α is a transition sequence consisting of all of the transitions in α_1 and in α_2 , and β is a transition sequence consisting of all of the transitions in β_1 and β_2 such that the appearance order of the transitions in α conforms to the order in α_1 and in α_2 , and β is a transition sequence consisting of all of the transitions in β_1 and β_2 such that the appearance order of the transitions in β conforms to the order in β_1 and in β_2 . If $\sigma_1 = \alpha_1 \tau \beta_1$ has the minimum makespan in (B_1, M_{1e}) and $\sigma_2 = \alpha_2 \tau \beta_2$ has the minimum makespan in (B_2, M_{2e}) , then $\sigma = \varepsilon_1 \alpha \pi \beta \varepsilon_2$ has the minimum makespan in (B, M_e) and $\#\sigma = \max \{\#\alpha_1, \#\alpha_2\} + \#\tau + \max \{\#\beta_1, \#\beta_2\}$.

Definition 21 (Disable) For two processes $B_i = (N_i, p_{ie}, p_{ix})$, $i = 1, 2$, their composition by *disable*, in notation $B_1 [] B_2$, is defined as the process $B = (N, p_e, p_x)$, where $P = (P_1 - \{p_{1x}\}) \cup (P_2 - \{p_{2x}\}) \cup \{p_e, p_x, p_d\}$, where $p_x = p_{1x} = p_{2x}$; $T = T_1 \cup T_2 \cup \{\varepsilon, t_d\}$; $F = F_1' \cup F_2' \cup \{(p_e, \varepsilon), (\varepsilon, p_{1e}), (\varepsilon, p_{2e}), (p_d, t_d), (t_d, p_{2e})\} \cup \{(p_{2e}, t) | t \in T_1\} \cup \{(t, p_d) | t \in T_1 - \bullet p_{1x}\}$, where $F_i' = F_i - \{(t, p_{ix}) | t \in \bullet p_{ix}\} \cup \{(t, p_x) | t \in \bullet p_{ix}\}$, $i = 1, 2$, $M_c = M_{1c} \cup M_{2c}$, $C = C_1 \cup C_2$, $W = W_1(F_1') \cup W_2(F_2') \cup \{w(p_e, \varepsilon), w(\varepsilon, p_{1e}), w(\varepsilon, p_{2e}), w(p_d, t_d), w(t_d, p_{2e})\} \cup \{w(p_{2e}, t) | t \in T_1\} \cup \{w(t, p_d) | t \in T_1 - \bullet p_{1x}\}$, $D = D_1 \cup D_2 \cup \{d(\varepsilon), d(t_d)\}$.

Disable $B_1 [] B_2$ models a potential interruption of process B_1 by another process B_2 during execution. According to LOTOS, there are three possibilities: (1) If B_2 never begins, B_1 will continue until termination. (2) If B_2 begins while B_1 is executing, B_1 will be stopped and B_2 continues its execution. (3) If B_2 begins first, B_1 will never begin and B_2 will continue its execution until termination. One application of disable is to model task interruption in a JSSP. Although an operation

will not be interrupted during execution, a process may be interrupted since it consists of several operations.

If all initial transitions of B_2 (i.e., outgoing transitions of p_{2e}) are never fired, B_1 will execute until completion. If any initial transition of B_2 is fired when B_1 is executing, B_1 is stopped because any transition of B_1 cannot be fired without a token in p_{2e} and B_2 will execute to completion. If B_2 begins first, the token in p_{2e} will immediately be removed. Then, it is obvious that B_1 can never begin.

Property 14 Suppose (B, M_e) is induced from (B_1, M_{1e}) and (B_2, M_{2e}) by applying the operator disable. If $M_{1e}[\sigma_1 > M_{1x}]$ in (B_1, M_{1e}) and $M_{2e}[\sigma_2 > M_{2x}]$ in (B_2, M_{2e}) , then $M_e[\varepsilon\sigma > M_x]$ in (B, M_e) , where σ satisfies the following: (1) $\sigma = \sigma_1$, (2) $\sigma = \sigma_2$, and (3) σ is a transition sequence consisting of the transitions in a firing sequence α of B_1 and σ_2 ; the appearance order is preserved and $p_{2e} \cdot \cap \alpha = \phi$. If σ_1 has the minimum makespan in (B_1, M_{1e}) and σ_2 has the minimum makespan in (B_2, M_{2e}) , then σ has the minimum makespan in (B, M_e) and $\#\sigma$ equals to one of $\#\sigma_1$, $\#\sigma_2$, and $\#\alpha + \#\sigma_2$.

Definition 22 (Disable-Resume) For two processes $B_i = (N_i, p_{ie}, p_{ix})$, $i = 1, 2$, their composition by the operator *disable-resume*, in notation $B_1 [r] B_2$, is defined as the process $B = (N, p_e, p_x)$, where $P = (P_1 \cup P_2) \cup \{p_e, p_d\}$, where $p_x = p_{1x}$, $T = T_1 \cup T_2 \cup \{\varepsilon_1, \varepsilon_2, t_d\}$; $F = F_1 \cup F_2 \cup \{(p_e, \varepsilon_1), (\varepsilon_1, p_{1e}), (\varepsilon_1, p_{2e}), (p_{2x}, \varepsilon_2), (\varepsilon_2, p_{2e}), (p_d, t_d), (t_d, p_{2e})\} \cup \{(p_{2e}, t) | t \in T_1\} \cup \{(t, p_d) | t \in T_1 - \bullet p_{1x}\}$, $M_c = M_{1c} \cup M_{2c}$, $C = C_1 \cup C_2$, $W = W_1 \cup W_2 \cup \{w(p_e, \varepsilon_1), w(\varepsilon_1, p_{1e}), w(\varepsilon_1, p_{2e}), w(p_{2x}, \varepsilon_2), w(\varepsilon_2, p_{2e}), w(p_d, t_d), w(t_d, p_{2e})\} \cup \{w(p_{2e}, t) | t \in T_1\} \cup \{w(t, p_d) | t \in T_1 - \bullet p_{1x}\}$, $D = D_1 \cup D_2 \cup \{d(\varepsilon_1), d(\varepsilon_2), d(t_d)\}$.

Disable-resume $B_1 [r] B_2$ functions the same as the disable operator $B_1 [] B_2$ except that if B_2 is ever executed and completed, then there will be three options: (1) The entire process terminates. (2) Execution on B_1 resumes from wherever it is left. (3) B_2 is repeated. Similar to the operator disable, there is also a non-pure version of disable-resume.

Property 15 Suppose (B, M_e) is induced from (B_1, M_{1e}) and (B_2, M_{2e}) by applying the operator disable-resume. If $M_{1e}[\sigma_1 > M_{1x}]$ in (B_1, M_{1e}) and $M_{2e}[\sigma_2 > M_{2x}]$ in (B_2, M_{2e}) , then $M_e[\varepsilon_1\sigma > M_x]$ in (B, M_e) , where σ satisfies the following: (1) $\sigma = \sigma_1$, (2) Suppose $\sigma_1 = \alpha\beta$, $p_{2e} \cdot \cap \alpha = \phi$ and $p_{2e} \cdot \cap \beta \neq \phi$. Then $\sigma = \gamma\beta$, where γ is a transition sequence consisting of transitions in α and $\sigma_2\varepsilon_2$ such that the appearance order is preserved. If σ_1 has the minimum makespan in (B_1, M_{1e}) and σ_2 has the minimum makespan in (B_2, M_{2e}) , then σ has the minimum makespan in (B, M_e) and $\#\sigma$ equals to one of $\#\sigma_1$, and $\#\sigma_1 + c \#\sigma_2$, where c is a constant number.

In JSSP, resource sharing is quite necessary; after the composition of the modules, sharing resource should be considered. Therefore, place-merging operator is applied for modeling and analyzing the resource-sharing problem.

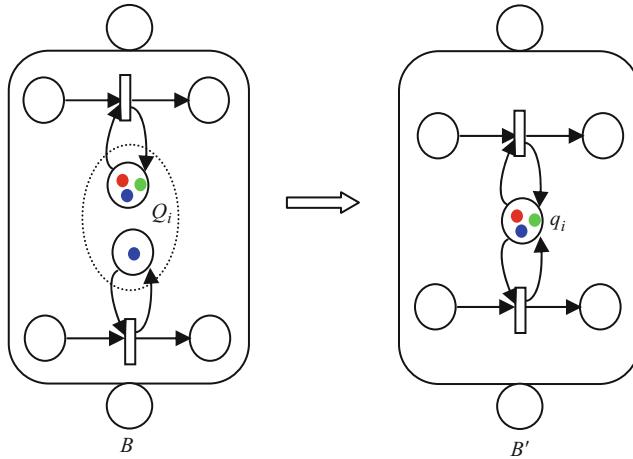


Fig. 12 $B(Q_i \rightarrow q_i)$: the operator place merging in a single process

Definition 23 (Place Merging (Fig. 12)) For a process $B = (N, p_e, p_x)$, suppose $N = (P_0 \cup Q_1 \cup \dots \cup Q_k, T, F, M_c, C, W, D)$ is an extended net satisfying the condition: $\forall i, j \in \{1, 2, \dots, k\}$ and $i \neq j$, $P_0 \cap Q_i = \emptyset$ and $Q_i \cap Q_j = \emptyset$. Let $B' = (N', p_e', p_x')$, denoted as $B(Q_i \rightarrow q_i)$, where $N' = (P_0 \cup Q_0, T', F', M_c', C', W', D')$ and $Q_0 = \{q_1, q_2, \dots, q_k\}$, be obtained from (B, M_e) by merging the places of each Q_i into q_i as follows: $T' = T$; F' is obtained from F by replacing each set of arcs of the form $\{(t, p) | p \in Q_i\}$ with (t, q_i) (resp., $\{(p, t) | p \in Q_i\}$ with (q_i, t))

$$M'_c(p) = \begin{cases} M_c(p) & p \in P_0 \\ \max_{p \in Q_i} \{M_c(p)\} & p = q_i \in Q_0, i = 1, 2, \dots, k \end{cases}$$

$$C' = C, W' = W, D' = D.$$

Property 16 Let (B, M_e) and (B', M_e') be defined in [Definition 10](#). If $M_e[\sigma > M_x]$ in (B, M_e) , then $M_e'[\sigma > M_x']$ in (B', M_e') .

Note that by the modeling technology, each transition t representing an operation forms a self-loop with the corresponding resource places. In such a case, [Property 16](#) is true.

Up to now, several integration operators, including the place-merging operators, are introduced. However, for the place-merging operators, the makespan of the resulted process cannot be obtained from the makespan of the original processes very easily. In order to simplify the computing process and get an exact makespan value, place-merging operator will be applied together with other integration operators if there exists resource sharing in the composition process. For example,

when two processes are composed by *enable* operator, if some operations in these two processes share some common resources, *place-merging* operator will be applied together with *enable* operator. Notations *M-enable*, *M-choice*, *M-interleave*, etc. denote these mixed operators.

Property 17 Suppose $M_{1e}[\sigma_1 > M_{1x}$ in (B_1, M_{1e}) and $M_{2e}[\sigma_2 > M_{2x}$ in (B_2, M_{2e}) . (B, M_e) is resulted from B_1 and B_2 by applying mixed operators and $M_e[\sigma > M_x$ in (B, M_e) . (1) For operator *M-enable*, if σ_1 has the minimum makespan in (B_1, M_{1e}) , and σ_2 has the minimum makespan in (B_2, M_{2e}) , then $\sigma = \sigma_1 \varepsilon \sigma_2$ has the minimum makespan in (B, M_e) and $\#\sigma = \#\sigma_1 + \#\sigma_2$. (2) for *M-choice*, if σ_1 has the minimum makespan in (B_1, M_{1e}) and σ_2 has the minimum makespan in (B_2, M_{2e}) , then the firing sequence with the minimum makespan in (B, M_e) is $\min\{\varepsilon_1 \sigma_1, \varepsilon_2 \sigma_2\}$. (3) for *M-interleave*, denote $\sigma_1 = \alpha_1 t_1 \alpha_2 t_2 \dots \alpha_{k-1} t_{k-1} \alpha_k$ and $\sigma_2 = \beta_1 t'_1 \beta_2 t'_2 \dots \beta_{m-1} t'_{m-1} \beta_m$, where t_1, t_2, \dots, t_{k-1} and $t'_1, t'_2, \dots, t'_{m-1}$ are the transitions sharing some resources in the operators, and α_i and β_j ($i = 1, 2, \dots, k$, $j = 1, 2, \dots, m$) are partial firing sequences. Suppose $\#\sigma_1 \geq \#\sigma_2$. Then the upper bound for the makespan of (B, M_e) is $\#\sigma_2 + \sum_{i=1}^{k-1} \#t_i$.

The proofs for **Property 17** (1) and (2) are trivial. For **Property 17** (3), the worst case would be that when some resources are shared, it is always process B_2 which waits.

By **Properties 12** and **16**, for the mixed operator *M-interleave*, the firing sequence σ of (B, M_e) is a transition sequence consisting of all of the transitions in σ_1 and σ_2 such that the appearance order of the transitions conforms to the order in σ_1 and in σ_2 . Below, an algorithm is introduced for generating a firing sequence σ of (B, M_e) based on σ_1 and σ_2 with the minimum makespan.

Makespan Generating Algorithm for M-Interleave Operator

Input: Two firing sequences $\sigma_1 = \alpha_1 t_1 \alpha_2 t_2 \dots \alpha_{k-1} t_{k-1} \alpha_k$ and $\sigma_2 = \beta_1 t'_1 \beta_2 t'_2 \dots \beta_{m-1} t'_{m-1} \beta_m$ of (B_1, M_{1e}) and (B_2, M_{2e}) , respectively

Output: A firing sequence σ and the corresponding minimum makespan $\#\sigma$ of (B, M_e)

Step 1: Let $\sigma_1 = \alpha_1 t_1 \alpha_2 t_2 \dots \alpha_{k-1} t_{k-1} \alpha_k$, $\sigma_2 = \beta_1 t'_1 \beta_2 t'_2 \dots \beta_{m-1} t'_{m-1} \beta_m$, $\sigma = \phi$, and $\#\sigma = 0$.

Step 2: For $i = 1, i < k$; $j = 1, j < m$,

if there exists a leftmost subsequence $\sigma_i = \alpha_i t_i \dots \alpha_p t_p$ of σ_1 and a leftmost subsequence $\sigma_j = \beta_j t'_j \dots \beta_q t'_q$ of σ_2 satisfying $\#\sigma_i - \#t_p \leq \#\sigma_j - \#t'_q \leq \#\sigma_i$, then $\sigma = \sigma \circ (\beta_j t'_j \dots \beta_q t'_q)$ and $\#\sigma = \#\sigma + \#\sigma_i + \#t'_q$.

If there exists a leftmost subsequence $\sigma_i = \alpha_i t_i \dots \alpha_p t_p$ of σ_1 and a leftmost subsequence $\sigma_j = \beta_j t'_j \dots \beta_q t'_q$ of σ_2 satisfying $\#\sigma_j - \#t'_q \leq \#\sigma_i - \#t_p \leq \#\sigma_j$, then $\sigma = \sigma \circ (\alpha_i t_i \dots \alpha_p t_p)$ and $\#\sigma = \#\sigma + \#\sigma_j + \#t_p$.

Let $\sigma_1 = \sigma_1 - \sigma_i$, $\sigma_2 = \sigma_2 - \sigma_j$, $i = p$ and $j = q$, $i++$, $j++$

Step 3. $\sigma = \max\{\sigma, \sigma_1, \sigma_2\}$ and $\#\sigma = \#\sigma + \max\{\#\sigma_1, \#\sigma_2\}$.

The above algorithm can also be applied to generate the firing sequence and compute the makespan for the mixed operators *M-parallel*, *M-disable*, and *M-disable-resume*. In the case of more than two composition processes, the algorithm can be improved easily such that the input is more than two firing sequences and the comparison is between more than two leftmost subsequences.

3.2 Modular-Based Design and Optimization: A Case Study

This section shows how to apply the composition operators for designing JSSP and finding the optimal scheduling. In order to illustrate the technology more precisely, a real-time JSSP design will be explained as an example.

Suppose there are three jobs to be processed in three workshops. Each job has two or three operations which will be processed in three workshops in a given sequence. In each workshop some machines can be used, and each operation only chooses one machine for processing. Four workers are responsible for these six machines in the three workshops. The details of the operation distribution, the processing time of each operation in different machines, and the worker distribution are shown in [Table 3](#).

3.2.1 System Design

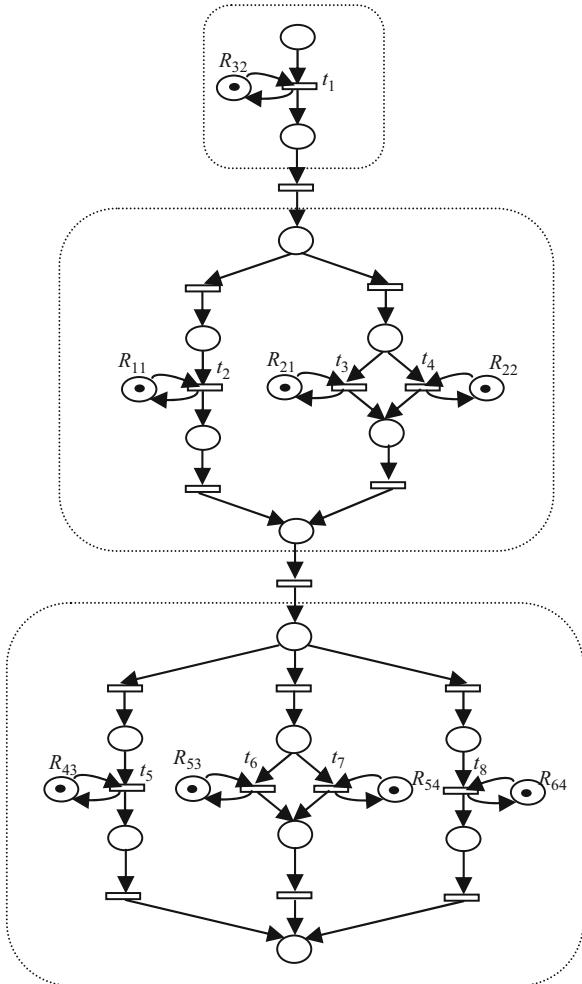
For simplicity, in the figures of Petri net model, only those transitions representing the operations are labeled, and the resource places R_{ij} represent machine M_i which are sponsored by worker W_j .

By the requirement analysis, Job 1 has three operations. In operation 1, machine M_3 is occupied which is sponsored by worker W_2 . Hence, transition t_1 , representing the operation 1, is associated with a resources place R_{32} in the Petri net model ([Fig. 13](#)). For operation 2, it can be processed either in machine M_1 sponsored by worker W_1 or in machine M_2 sponsored by worker W_1 or W_2 . *Choice* operators are applied twice in this model, one is used for choosing machines (M_1 and M_2), and

Table 3 Processing information of jobs

Job	Operation	Workshop	(Machine, time)	W_1	W_2	W_3	W_4
1	1	II	(M_3 , 12)	M_1	O		
	2	I	(M_1 , 9), (M_2 , 10)	M_2	O	O	
	3	III	(M_4 , 5), (M_5 , 6), (M_6 , 8)	M_3		O	
2	1	II	(M_3 , 6)	M_4		O	
	2	III	(M_4 , 6), (M_5 , 9), (M_6 , 8)	M_5		O	O
	3	I	(M_1 , 4), (M_2 , 5)	M_6			O
3	1	I	(M_1 , 5), (M_2 , 4)	<i>Note:</i> “O” means that the worker in the corresponding column is responsible for the machine in the corresponding row			
	2	III	(M_4 , 8), (M_5 , 10), (M_6 , 9)				

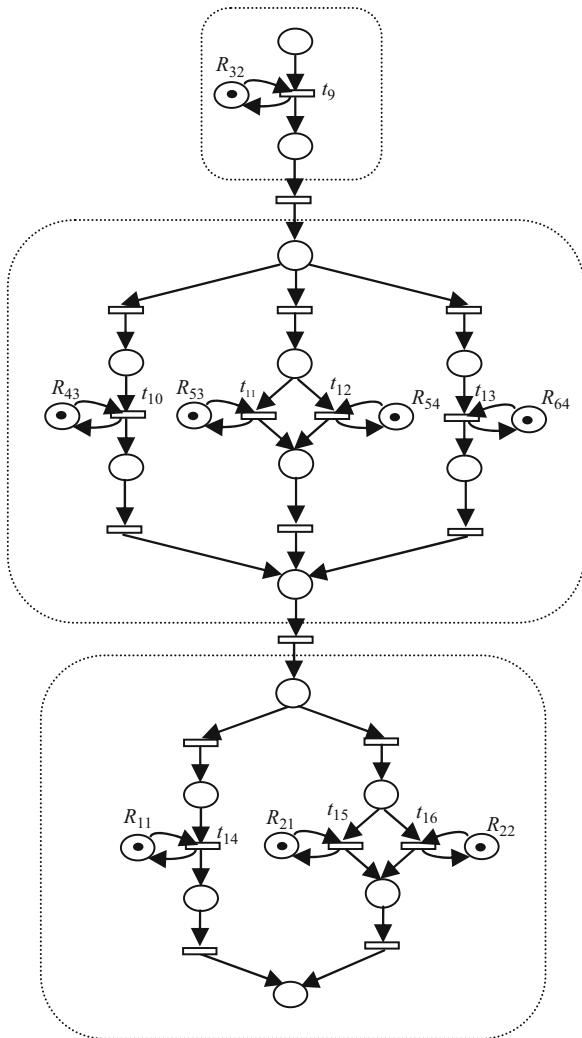
Fig. 13 Petri net model for Job 1



another is used for choosing workers (W_1 and W_2). Similarly, for operation 3, three machines M_4 , M_5 , and M_6 will be chosen, and machine M_5 has a choice between the workers W_3 and W_4 . Since the three operations are in a sequence order, the three modules will be integrated by applying *enable* operator (Fig. 13).

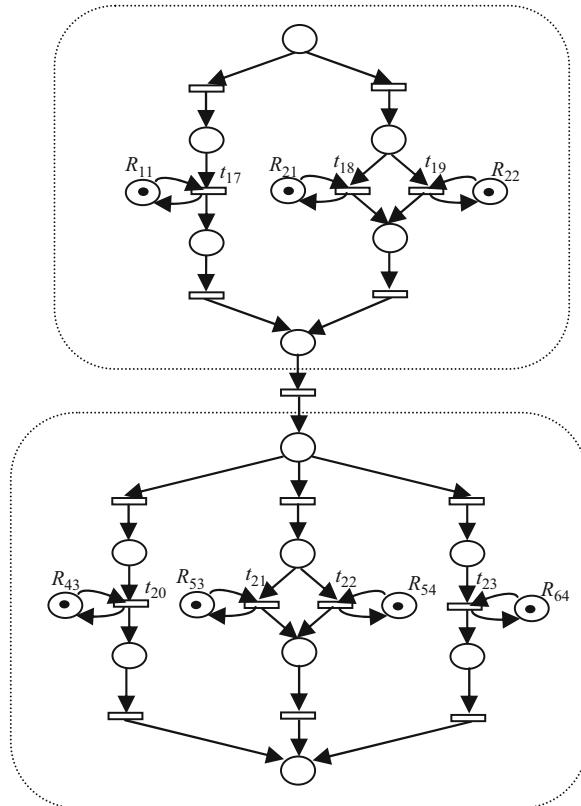
Similarly, the Petri net models for Jobs 2 and 3 can also be obtained (Figs. 14 and 15).

Fig. 14 Petri net model for Job 2



By Properties 8 and 9, the three jobs have the possible firing sequences listed in Fig. 16. In order for simplicity and clarity, only the transitions representing the operations are listed. The processing time and the resources for each operation are assigned to the corresponding transitions. By observing, there are 12 firing sequences for each Petri net model.

Fig. 15 Petri net model for Job 3



Since these three jobs are independent, they can be processed in an interleave relationship. In the three Petri net models, the resource places with the same labels represent the same resources. Hence, a mixed operator M-interleave is applied for composing these three modules, and the system model is obtained. Since the picture is too complex, it is omitted.

Property 18 The job shop scheduling system design is correct.

A system is correct if it is almost live, almost bounded, and almost reversible and terminates properly. Since each of the primary Petri net modules is correct, by the results in [3, 4], the system model resulted by combining the modules by applying the operators defined in this section is also correct. Hence, the design method can guarantee the correctness of the system.

$$\begin{array}{c}
 \left(\begin{array}{l} t_2(9, R_{11}) \\ t_5(5, R_{43}) \\ t_6(6, R_{53}) \\ t_7(6, R_{54}) \\ t_8(8, R_{64}) \\ t_5(5, R_{43}) \\ t_6(6, R_{53}) \\ t_7(6, R_{54}) \\ t_8(8, R_{64}) \\ t_5(5, R_{43}) \\ t_6(6, R_{53}) \\ t_7(6, R_{54}) \\ t_8(8, R_{64}) \end{array} \right) \times \left(\begin{array}{l} t_{17}(5, R_{11}) \\ t_{21}(10, R_{53}) \\ t_{22}(10, R_{54}) \\ t_{23}(9, R_{64}) \\ t_{20}(8, R_{43}) \\ t_{21}(10, R_{53}) \\ t_{22}(10, R_{54}) \\ t_{23}(9, R_{64}) \\ t_{20}(8, R_{43}) \\ t_{21}(10, R_{53}) \\ t_{22}(10, R_{54}) \\ t_{23}(9, R_{64}) \end{array} \right) \times \left(\begin{array}{l} t_{10}(6, R_{43}) \\ t_{11}(7, R_{54}) \\ t_{12}(7, R_{53}) \\ t_{13}(8, R_{64}) \\ t_{14}(4, R_{11}) \\ t_{15}(5, R_{21}) \\ t_{16}(5, R_{22}) \\ t_{14}(4, R_{11}) \\ t_{15}(5, R_{21}) \\ t_{16}(5, R_{22}) \\ t_{14}(4, R_{11}) \\ t_{15}(5, R_{21}) \\ t_{16}(5, R_{22}) \end{array} \right) \\
 t_1(12, R_{32}) \times t_3(10, R_{21}) \times t_9(6, R_{32}) \\
 t_4(10, R_{22}) \times t_{18}(4, R_{21}) \times t_{19}(4, R_{22})
 \end{array}$$

Fig. 16 The possible firing sequences for the three jobs

3.2.2 Scheduling Optimization

By [Property 12](#) and makespan generating algorithm for M-interleave operator, the minimum makespan and the corresponding firing sequence (i.e., schedule) can be obtained. Theoretically, the algorithm should run totally 12^3 times in order to search for the optimal solution for the example. In practice, quite a lot of firing sequences can be deleted from the list.

In order to illustrate the method, two possible schedule examples are illustrated to run the algorithm. For example, let $\sigma_1 = t_1t_2t_5$, $\sigma_2 = t_9t_{10}t_{14}$, and $\sigma_3 = t_{17}t_{20}$. By makespan generating algorithm for M-interleave operator, the schedule is as shown in [Fig. 17a](#), where machines 1, 3, and 4 are used and workers 1, 2, and 3 are needed to sponsor the machines. R_{ij} means machine i sponsored by worker j . J_{ij} means Job i 's j th operation is being processed in the corresponding time slot. In this case, the number of machines is minimum and the makespan equals 29. In another case, let $\sigma_1 = t_1t_4t_6$, $\sigma_2 = t_9t_{10}t_{14}$, and $\sigma_3 = t_{19}t_{21}$. The schedule is shown in [Fig. 17b](#). In this case, five machines are used and three workers are needed. The makespan is 28. There are some more schedules with makespan 28. For example, the following four firing sequences correspond to the optimal schedule: $t_1t_{19}t_{20}t_2t_9t_{10}t_7t_{14}$, $t_{18}t_1t_{20}t_9t_3t_{10}t_6t_{14}$, $t_{17}t_1t_{23}t_9t_4t_{10}t_6t_{14}$, $t_{19}t_1t_{23}t_9t_2t_{10}t_7t_{14}$.

In order to reduce the time complexity, an upper bound for the makespan is given such that for those partial schedules with the possible makespan exceeding the upper bound, they can be deleted from the firing sequence list. For example, in the above example, the makespan of the system will not exceed 28. Hence, the firing sequences $t_1t_2t_8$, $t_1t_3t_8$, and $t_1t_4t_8$ in the Job 1 ([Fig. 13](#)) will be deleted from the list since their makespan have exceeded 28. As for Job 2 ([Fig. 14](#)), before processing, it will wait for 12 time slots because of resource sharing; except the sequence $t_9t_{10}t_{14}$, the makespan for all of other partial schedule will not be less than 28. If an optimal solution is required, only the single firing sequence $t_9t_{10}t_{14}$ needs to be considered. Hence, very possibly, the algorithm only needs to run no more than 9×12 times.

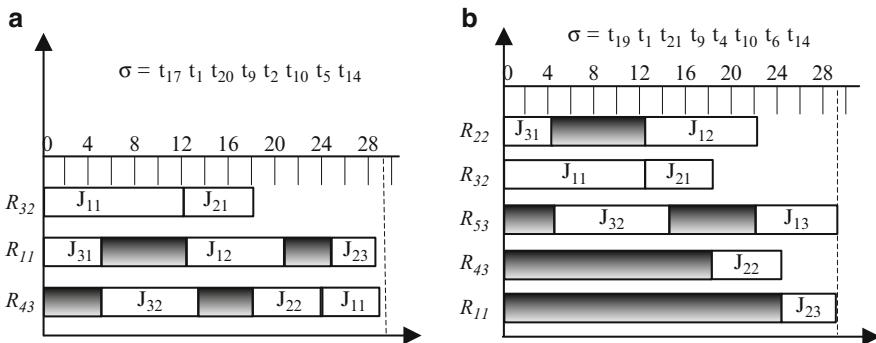


Fig. 17 Some possible schedules

Theoretically, JSSP design method can be applied to find an optimal scheduling and the idea can refer to [2, 4]. In practice, for large and complex systems, the main difficulty is to find the optimal scheduling since there are too many possible partial firing sequences. The makespan generating algorithm should run too many times in order to select the optimal one. In order to reduce the time complexity, one of the possible solutions is to combine the branch and bound method [45] with the algorithm of this chapter. The idea has been introduced in the above example. Another solution is to combine GA algorithm [38] with the algorithm of this chapter, and the idea is as follows: Let the partial firing sequences in each module be genes and the firing sequence of the system be used as the chromosomes. The makespan will be the fitness function and the operators will be newly defined according to the combination operators defined in this article. An efficient GA algorithm is introduced in the next section. The branch and bound method, GA method, and the methods of this chapter can be combined together to find the approximation optimal solution in a much more efficient way.

4 Genetic Algorithm for JSSP

In this section, genetic algorithm is applied to solve JSSP. Different from the existing genetic algorithms, the genetic algorithm applied in this chapter is based on the Petri net model. The chromosomes are composed by sequence of transitions, with crossover and mutation operations based on transition sequences. The experiment result shows that a definite solution to a specific JSSP can be found.

Flexible job shop scheduling problem (FJSSP) is the extension of JSSP, and it is more complex than the classic JSSP. In FJSSP, each operation can be processed on different machines, desiring different period of time. FJSSP is more practical than classic job shop problem and hence attracts much more attentions. There are also many methods used to resolve single-objective flexible job shop problems, such as evolutionary algorithms [46], simulated annealing [47–49], and tabu search [50] which have been proved efficient for solving FJSSP. The combinations of local search and evolutionary algorithm have obtained good solutions for FJSSPs [51].

For both JSSP and FJSSP in a practical application, there are many objectives, e.g., makespan, total tardiness, total earliness, workload of machines, cost, and so on. Sometimes there are conflicts among these objectives; it is nearly impossible to achieve optimization of more objectives simultaneously. Hence, searching for useful methods of finding a counterpoise of the objectives is a challenge and draws many researchers' attention. For instance, Li-Ning Xing and Ying-Wu Chen presented a simulation model to solve the multi-objective FJSSP [52]. The simulation model is proposed for the FJSSP with multi-objectives of minimizing makespan, total workload of machines, and workload of the critical machine [53]. Kazi Shah Nawaz Ripon and Chi-Ho Tsang presented an evolutionary approach for solving multi-objective FJSSP using Jumping Gene Genetic Algorithm that heuristically searches for the near-optimal solutions and optimizes multiple criteria simultaneously [54]. They can achieve near-optimal or optimal solutions to some special problems in their research, but the methods are aimed to solve a given problem, not suitable for the general FJSSP.

In this section, two objectives, that is, makespan and total tardiness, are considered. Firstly, the FJSSP is modeled with Petri nets, and then the problems are solved by a genetic algorithm. As a graphic and mathematic tool, Petri net has been used to model scheduling problems for a long time. Meanwhile, Petri net-based approach will face the state space explosion problem, which may cut down the efficiency of execution.

In order to partially avoid the state explosion problem for Petri net-based approach, genetic algorithm has been used with Petri net to solve scheduling problem. Chung and Fu used timed place Petri net (TPPN) and genetic algorithm to model and schedule FMS [55]. A Petri net with controller is used to model discrete events in flexible job shop scheduling with results obtained based on genetic algorithm and simulated annealing algorithm [56]. Huang et al. applied QCPN (queuing colored Petri net) and GA to model and schedule wafer manufacture [57]. These methods all work effectively, but their encoding related with special case, which sacrificed their general use for good scheduling result. According to the topics talked above, Petri net model is proposed in this section that can visually describe FJSSP. The operations in genetic algorithm such as encode, crossover, and mutation are all operated on elements of Petri net. The method is useful for any FJSSP which can be modeled by Petri net. The results in this section is mainly derived from [58].

4.1 Petri Net-Based Modeling and Problem Formulation

In this section, how to model the scheduling problems with Petri nets is introduced.

The FJSSP is described as follows:

1. $J = \{J_1, J_2, J_3, \dots, J_n\}$ is a set of jobs to be scheduled.
2. Each job consists of a predefined sequence of operations; let O_{ij} be operation j of job i .
3. Each job has a deadline, $D = \{d_1, d_2, d_3, \dots, d_n\}$ is a set of deadline, and d_i is a deadline of job i .

4. $M = \{M_1, M_2, M_3, \dots, M_m\}$ is a set of machines. Each operation is processed on some given machines for a fixed duration without interruption.
5. Each machine can process one operation at a time.

Let $C = \{c_1, c_2, c_3, \dots, c_n\}$ be the set of completion time of jobs, c_i is the completion time of job J_i . Two objectives are considered in this section:

1. Minimization of makespan, $F_1 = \max(c_i | i=1,2,3, \dots, n)$.
2. Minimization of total tardiness, $F_2 = \sum_i \text{tardiness}(i), i=1,2,3,\dots,n$

$$\text{tardiness}(i) = \begin{cases} 0, & c_i \leq d_i \\ c_i - d_i, & c_i > d_i \end{cases}$$

The task is to find a solution which is better than any others when both the objectives are considered. FJSSP, as one of the most difficult problems in combinational optimization, has been proved to be NP-hard problem [59]. Modeling the job shop problem with Petri nets can make the complex scheduling problem visualized. The scheduling problems can be modeled with Petri nets and the rules are as follows:

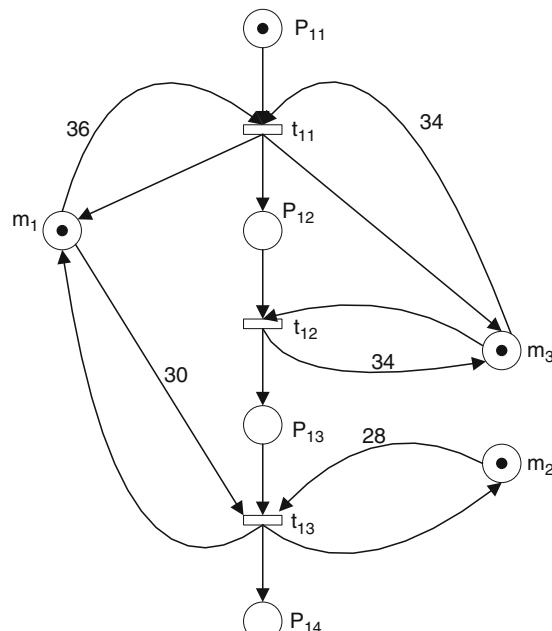
1. Each transition denotes an operation, and the input (resp., output) place of the transition can be interpreted as the beginning (resp., end) of the operation.
2. There are some initially marked places, called resource places, which denote machines. Each resource place is associated with some transitions when the operation is processed on this machine.
3. Each job can be modeled with a small Petri net, where machines are models as resource places in the model without sharing. Then all the small Petri nets are composed by merging their identical resource places if the corresponding machines are shared by different jobs.
4. For the Petri net model with m places and n transitions, an m -by- n matrix is generated as follows: Row i corresponds to place p_i ; column j corresponds to transition t_j ; if p_i is the input place of t_j , then the element in row i column j is -1 ; if p_i is the output place of t_j , then the element in row i column j is 1 ; otherwise, the element in row i column j is 0 ; if p_i is a resource place connecting to t_j , then the element in row i column j is the time pieces that the operation needs to be processed on the machine p_i ; otherwise, the element in row i column j is $1,000$.

At the beginning, tokens only exist in the first place of each job and all the resource places. After firing a sequence of transitions in turn, the tokens are transferred to the last place of each job and resource places, which represents that the job has been finished.

The following is a FJSSP example: three jobs (J_1, J_2, J_3) are to be processed on three machines (M_1, M_2, M_3). Each job consists of three operations (O_1, O_2, O_3), and each operation can be processed on some given machines. Meanwhile, there is a deadline of each job, and if a job is completed after the deadline, extra cost must be charged (Table 4). For example, the first operation (O_1) of J_1 can be operated on machine M_1 for 36-unit time or on machine M_3 for 34-unit time. The second operation (O_2) of J_1 can be operated on machine M_3 for 34-unit time. The third operation (O_3) of J_1 can be operated on machine M_2 for 28-unit time or on machine M_1 for 30-unit time. The deadlines of the three jobs are 98, 84, and 92, respectively.

Table 4 The job shop scheduling problem

	J_1	J_2	J_3
O_1	$M_1(36)/M_3(34)$	$M_2(35)$	$M_3(33)$
O_2	$M_3(34)$	$M_1(29)/M_3(26)$	$M_3(34)$
O_3	$M_2(28)/M_1(30)$	$M_3(22)$	$M_1(28)/M_2(30)$
Deadline	98	84	92

**Fig. 18** Petri net model for J_1

According to the rules of modeling, this section gets the Petri net model for each job and then merges the small Petri nets by sharing machines. Figures 18–20 are the Petri net model of J_1 , J_2 , and J_3 , respectively. After merging the three small Petri nets by sharing machines, the whole Petri net (Fig. 21) is obtained. Take J_1 as an example; t_{11} , t_{12} , and t_{13} are the three predefined operations of J_1 . Add four places to connect the transitions such that except the first place, each place represents the completion state of the last operation and the beginning state of next operation. The first place of a job represents the beginning state of the job. M_1 , M_2 , and M_3 are resource places. If an operation can be processed on a machine, there is an arc from the resource place to corresponding transition and also an arc from transition to the resource place. For instance, operation t_{11} can be operated on machine M_1 for 36-unit time or on machine M_3 for 34-unit time. So there is an arc from M_1 to t_{11} and also an arc from t_{11} to M_1 , both weights are 36, and there is an arc from M_3 to t_{11} and also an arc from t_{11} to M_3 , both weights are 34. If there are tokens in all the input places of an operation, the operation is ready to be processed. And if there are

Fig. 19 Petri net model for J_2

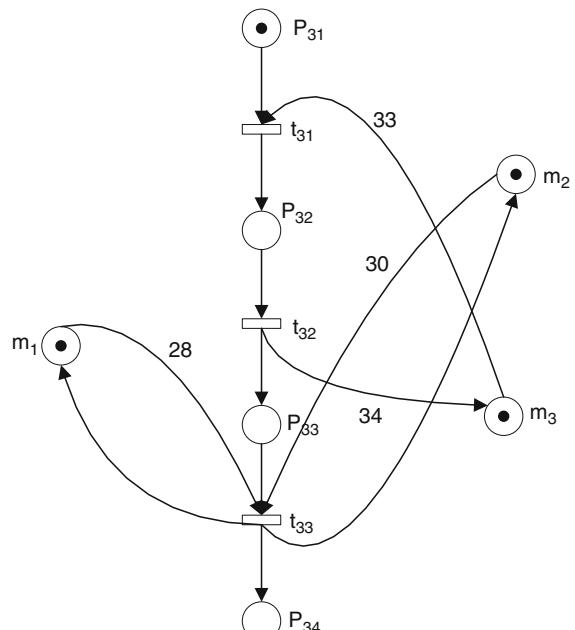
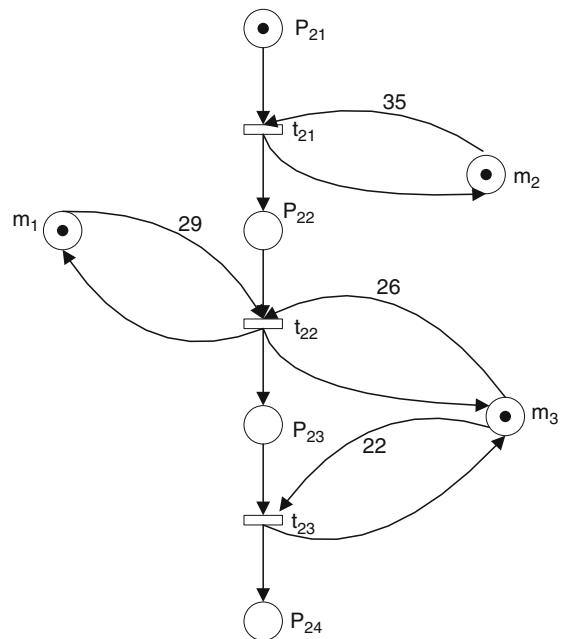


Fig. 20 Petri net model for J_3

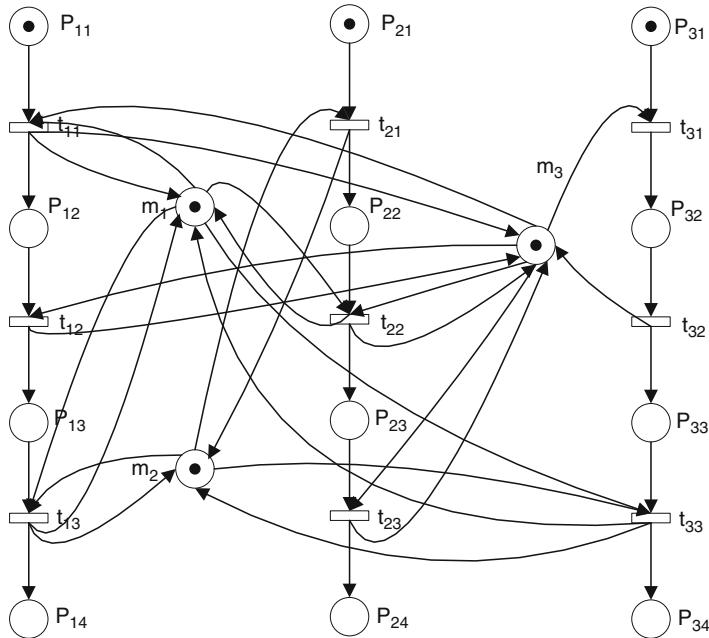


Fig. 21 Petri net model for the FJSSP

tokens in output place of an operation, the operation has been operated. As shown in Fig. 18, operation t_{11} is ready to be operated. With the same method, J_2 and J_3 are modeled.

For a Petri net model, a transition sequence is obtained by their firing time, which corresponds to a solution of the scheduling problem (e.g., $t_{11}t_{21}t_{31}t_{32}t_{22}t_{33}t_{12}t_{13}t_{23}$).

4.2 Genetic Algorithm for FJSSP

Genetic algorithm (GA) has received much attention since it had been introduced, and it has been a powerful tool of solving some NP problems in many areas.

This section uses Petri net to model the scheduling problem. Genetic algorithm is designed on the Petri net model, and a firing sequence as a chromosome, corresponding to a feasible solution.

Chromosome Encoding

For a Petri net model, if there is a firing transition sequence $\tau = t_1 t_2 t_3 \dots t_n$ such that $M_0[N, \tau] M_f$ (M_0 is initial marking and M_f is final marking), then τ is regarded as a chromosome.

Fitness function

The objectives are makespan and total tardiness, which are measured by time. For a chromosome τ , the makespan and the total tardiness of it are computed. Since the importance of the two objectives is equal, the weights are 0.5 and 0.5, respectively.

$$F_1 = \max(c_i | i=1,2,3,\dots,n)$$

$$F_2 = \sum_i \text{tardiness}(i), i = 1, 2, 3, \dots, n$$

$$\text{tardiness}(i) = \begin{cases} 0, & c_i \leq d_i \\ c_i - d_i, & c_i > d_i \end{cases}$$

The fitness function can be denoted as

$$F = 0.5F_1 + 0.5F_2$$

Initial population

Generally, the initial population is randomly produced. The initial population has great influence on finding final solution of JSSP using genetic algorithm. So the proper initial population should be selected in order to guarantee the quality of the final solution.

Since there are orders among the operations of the same job, and unfeasible solutions may be generated when the initial population is generated, the illegal solutions can become feasible by changing some transitions' positions.

Selection operation

Roulette Selection algorithm [60] is used to select parent chromosome to produce chromosome of the next generation by crossover and mutation operations.

Crossover operation

Crossover operation is vital for the genetic algorithm, and the possibility of it has a great influence on the final result. The process of crossover operation is as follows (Fig. 22):

1. Select two chromosomes $\tau_1 = t_1 t_2 t_3 \dots t_i t_{i+1} \dots t_{j-1} t_j \dots t_n$, and $\tau_2 = t'_1 t'_2 t'_3 \dots t'_i t'_{i+1} \dots t'_{j-1} t'_{j+1} \dots t'_n$, and then fire the transitions one by one in both chromosomes. Suppose $M_i[N, t_i] M_{i+1}$ and $M'_i[N, t'_i] M'_{i+1}$, $i = 1, 2, \dots, n$. Then, generate many corresponding temporary markings and get

$$\sigma_1 = \frac{t_1}{M_1} \frac{t_2}{M_2} \frac{t_3}{M_3} \dots \frac{t_i}{M_i} \frac{t_{i+1}}{M_{i+1}} \dots \frac{t_{j-1}}{M_{j-1}} \frac{t_j}{M_j} \dots \frac{t_n}{M_n}$$

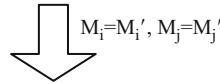
and

$$\sigma_2 = \frac{t'_1}{M'_1} \frac{t'_2}{M'_2} \frac{t'_3}{M'_3} \dots \frac{t'_i}{M'_i} \frac{t'_{i+1}}{M'_{i+1}} \dots \frac{t'_{j-1}}{M'_{j-1}} \frac{t'_{j+1}}{M'_{j+1}} \dots \frac{t'_n}{M'_n}$$

$$\begin{aligned}\tau_1 &= t_1 t_2 t_3 \dots t_i t_{i+1} \dots t_{j-1} t_j \dots t_n \\ \tau_2 &= t_1' t_2' t_3' \dots t_i' t_{i+1}' \dots t_{j-1}' t_j' \dots t_n'\end{aligned}$$

$$\sigma_1 = \frac{t_1}{M_1} \quad \frac{t_2}{M_2} \quad \frac{t_3}{M_3} \quad \dots \quad \frac{t_i}{M_i} \quad \frac{t_{i+1}}{M_{i+1}} \quad \dots \quad \frac{t_{j-1}}{M_{j-1}} \quad \frac{t_j}{M_j} \quad \dots \quad \frac{t_n}{M_n}$$

$$\sigma_2 = \frac{t_1'}{M_1'} \quad \frac{t_2'}{M_2'} \quad \frac{t_3'}{M_3'} \quad \dots \quad \frac{t_i'}{M_i'} \quad \frac{t_{i+1}'}{M_{i+1}'} \quad \dots \quad \frac{t_{j-1}'}{M_{j-1}'} \quad \frac{t_j'}{M_j'} \quad \dots \quad \frac{t_n'}{M_n'}$$



$$\sigma_1' = \frac{t_1}{M_1} \quad \frac{t_2}{M_2} \quad \frac{t_3}{M_3} \quad \dots \quad \frac{t_i}{M_i} \quad \frac{t_{i+1}'}{M_{i+1}''} \quad \dots \quad \frac{t_{j-1}'}{M_{j-1}''} \quad \frac{t_j}{M_j} \quad \dots \quad \frac{t_n}{M_n}$$

$$\sigma_2' = \frac{t_1'}{M_1'} \quad \frac{t_2'}{M_2'} \quad \frac{t_3'}{M_3'} \quad \dots \quad \frac{t_i'}{M_i'} \quad \frac{t_{i+1}''}{M_{i+1}'''} \quad \dots \quad \frac{t_{j-1}''}{M_{j-1}'''} \quad \frac{t_j'}{M_j'} \quad \dots \quad \frac{t_n'}{M_n'}$$

$$\tau_1' = t_1 t_2 t_3 \dots t_i t_{i+1} \dots t_{j-1} t_j \dots t_n$$

$$\tau_2' = t_1' t_2' t_3' \dots t_i' t_{i+1}' \dots t_{j-1}' t_j' \dots t_n'$$

Fig. 22 Process of crossover

$$\begin{aligned}\tau &= t_1 t_2 t_3 \dots t_i \dots t_j \dots t_n \\ \tau' &= t_1 t_2 t_3 \dots t_j \dots t_i \dots t_n\end{aligned}$$

Fig. 23 Process of mutation

2. Search for the identical markings in σ_1 and σ_2 such that $M_i = M'_i$, randomly select two pairs such as $M_i = M'_i$, $M_j = M'_j$, then exchange the gene piece $t_i t_{i+1} \dots t_{j-1} t_j$ of σ_1 and gene piece $t'_i t'_{i+1} \dots t'_{j-1} t'_j$ of σ_2 , and get two child sequences σ'_1 and σ'_2 . At last two child chromosomes τ'_1 and τ'_2 are obtained.

Mutation operation

Mutation operation is used to maintain the variety of populations, and it is operated on some genes of a chromosome. Mutation operation can renovate and make up genes that are lost in the selection and crossover operation. Crossing over mutation is applied to change two gene positions of a chromosome (e.g., exchange positions t_i and t_j in Fig. 23).

For the chromosome τ , positions of t_i and t_j can be interchanged only when the two operations can be processed on the same machine, and they do not belong to the same job.

4.3 Simulation Experiment

According to the topics talked above, the whole process of genetic algorithm is as follows:

- Step 1: Encode with the firing transition sequence.
- Step 2: Initialize the populations.
- Step 3: Calculate the fitness value of each chromosome of the population.
- Step 4: Judge whether the population meets the given condition: YES, go to Step 5; NO, go to Step 7.
- Step 5: Run roulette selection algorithm to select parent chromosome. Take crossover operation with a possibility and obtain a new population.
- Step 6: Take mutation operation on the population achieved in Step 5, to optimize the population, and then go to step 3.
- Step 7: Output the best solution.

For the example discussed in Sect. 4.1, firstly the Petri net model is transferred to a matrix F below (Fig. 24), and then all the genetic algorithm operations are all taken on the matrix. According to Table 4, there are three jobs and three machines and each job has three operations. So there are 9 transitions and 15 places (12 places and 3 resource places). A 15×9 matrix F is constructed. The elements of matrix F are defined as follows:

$F(p, t) = 1$, there is an arc from p to t ; $F(p, t) = -1$, there is an arc from t to p ; $F(p, t) = 0$, there is no arc between p and t ; $F(m, t) = T$, operation t can be processed on machine m for T -unit time; $F(m, t) = 1,000$, operation t cannot be processed on machine m .

Set the size of population 100 and the number of generation 50. Crossover possibility is 0.4 and mutation possibility is 0.2. The objective is to schedule

	t_{11}	t_{12}	t_{13}	t_{21}	t_{22}	t_{23}	t_{31}	t_{32}	t_{33}
P_{11}	1	0	0	0	0	0	0	0	0
P_{12}	-1	1	0	0	0	0	0	0	0
P_{13}	0	-1	1	0	0	0	0	0	0
P_{14}	0	0	-1	0	0	0	0	0	0
P_{21}	0	0	0	1	0	0	0	0	0
P_{22}	0	0	0	-1	1	0	0	0	0
P_{23}	0	0	0	0	-1	1	0	0	0
P_{24}	0	0	0	0	0	-1	0	0	0
P_{31}	0	0	0	0	0	0	1	0	0
P_{32}	0	0	0	0	0	0	-1	1	0
P_{33}	0	0	0	0	0	0	0	-1	1
P_{34}	0	0	0	0	0	0	0	0	-1
m_1	36	1000	30	1000	29	1000	1000	1000	28
m_2	1000	1000	28	35	26	1000	1000	1000	30
m_3	34	34	1000	1000	26	22	33	34	1000

Fig. 24 Matrix of the 3×3 FJSSP

Fig. 25 Solution of 3×3 FJSSP

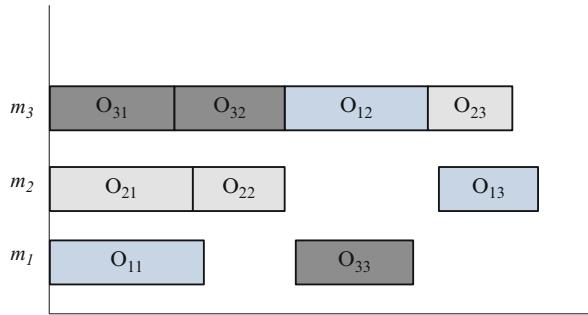


Table 5 5×4 FJSSP

	J_1	J_2	J_3	J_4	J_5
O_1	$M_1(4)/M_2(5)/$ $M_3(6)$	$M_3(8)$	$M_2(5)/$ $M_3(6)$	$M_2(4)/M_3(4)/$ $M_4(5)$	$M_2(7)/$ $M_3(9)$
O_2	$M_3(7)/M_4(6)$	$M_2(3)/$ $M_3(4)$	$M_1(7)/$ $M_4(5)$	$M_1(6)/M_4(6)$	$M_4(5)$
O_3	$M_2(5)/M_3(4)/$ $M_4(7)$	$M_1(8)/$ $M_2(9)$	$M_1(6)/$ $M_3(4)/M_4(7)$	$M_3(4)/M_4(5)$	$M_1(3)/$ $M_2(4)$
Deadline	15	17	18	14	15

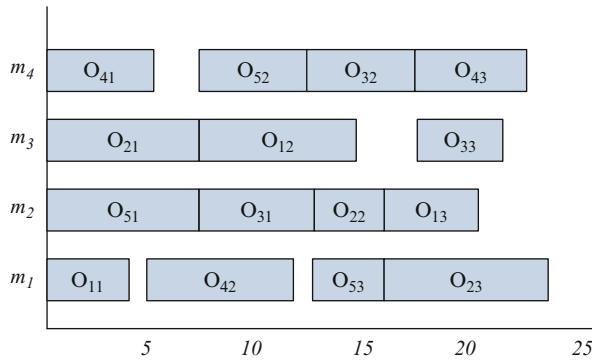


Fig. 26 Solution of 5×4 FJSSP

the three jobs on the three machines to minimize makespan and total tardiness. **Figure 25** is the solution for the 3-job-3-machine FJSSP.

There is another case in **Table 5**: five jobs (J_1, J_2, J_3, J_4, J_5) (each has three operations (O_1, O_2, O_3)) are scheduled on four machines (M_1, M_2, M_3, M_4), each job has a deadline. The numbers in brackets are time needed on related machines. The objective is to complete the jobs before their deadline and minimize the makespan. The parameters stay the same as last example. Since genetic algorithm has randomicity, the experiment carried out 30 times, and there are 20 times such

that the objective value is smaller than 70. The objective value of the optimal solution is 68. The final schedule is shown in Fig. 26.

5 Conclusion

Traditionally, Petri net-based methodologies for JSSP focused on two aspects: One is to model the system with Petri nets and then find the possible schedule by simulation. Another one is based on Petri net model, searching for algorithms for the optimization of the schedule. The common point of current methods is that the Petri net model is assumed to exist. For a real-time complex and large JSSP, generating the Petri net model is in fact a hard work. How to design a correct Petri net model such that it satisfies the requirement by itself is a research topic. Even when the model is given, the simulation and approximation algorithms are not applicable since the state space would be quite large.

This chapter introduces a modular-based approach for JSSP design and optimization. It applies the operators in PPPA for specifying and verifying the JSSP such that a correct Petri net model is generated. Based on the approach, a deadlock-free JSSP is designed. Furthermore, the scheduling of a composite module can be induced from the partial scheduling of the constituent modules. Hence, the schedule of the whole system can be induced from the original modules step by step.

Theoretically, the modular-based approach can be used to find the optimal solution for JSSP in a formal way. For real-time systems, due to its complexity, the algorithm may have a very large time complexity. In order to reduce the time complexity and to find an approximation optimal solution, the traditional optimization technologies such as GAs, branch and bound methods, hybrid methods, and rule-based methods can be applied together with the modular-based approach in this chapter. Based on the Petri net model, a genetic algorithm is introduced in this chapter to solve the problems. The firing sequence of transitions is regarded as a chromosome; genetic operations such as crossover and mutation are based on the elements of the Petri net model.

In order to solve more general JSSP, more integration operators should be considered, and the conclusions about the scheduling-preservation and makespan calculation should also be considered in the future research.

Acknowledgements This work was financially supported by the National Natural Science Foundation of China under grant Nos. 11071271 and 61100191 and the Fundamental Research Funds for the Central Universities under grant Nos. HIT.NSRIF.2009127 and HIT.NSFIR.2011128.

Cross-References

- ▶ [Advances in Scheduling Problems](#)
- ▶ [Online and Semi-online Scheduling](#)
- ▶ [Resource Allocation Problems](#)

Recommended Reading

1. L. Jiao, H.J. Huang, T.Y. Cheung, Property-preserving composition by place merging. *J. Circ. Syst. Comput.* **14**(4), 793–812 (2005)
2. H.J. Huang, T.Y. Cheung, X.L. Wang, Applications of property-preserving algebras to manufacturing system design. *J. Inform. Sci. Eng.* **23**, 167–181 (2007)
3. W.M. Mak, Verifying property preservation for component-based software systems (a Petri-net based methodology). PhD Dissertation, City University of Hong Kong, 2001
4. H.J. Huang, Enhancing the property-preserving Petri net process algebra for component-based system design (with application to designing multi-agent systems and manufacturing systems). PhD Dissertation, City University of Hong Kong, 2004
5. H.J. Huang, T.Y. Cheung, W.M. Mak, Structure and behaviour preservation by Petri-net-based refinements in system design. *Theor. Comput. Sci.* **328**(3), 245–269 (2004)
6. H.J. Huang, L. Jiao, T.Y. Cheung, Property-preserving subnet reductions for designing manufacturing systems with shared resources. *Theor. Comput. Sci.* **332**(1–3), 461–485 (2005)
7. D.K. Terry, A Petri net-based on-line scheduling system for a general manufacturing job shop. PhD Dissertation, Rensselaer Polytechnic Institute, Troy, 1996
8. M.C. Zhou, K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing: A Petri Net Approach* (World Scientific Publishing Company Incorporated, Singapore, 1999)
9. J. Colom, The resource allocation problem in flexible manufacturing systems. *Lect. Note Comput. Sci.* **2679**, 23–35 (2003)
10. M. Attaran, Flexible manufacturing systems, *Information Systems Management*, **9**(2), 44–47 (1992)
11. Hruz, M.C. Zhou, *Modeling and Control of Discrete Event Dynamic Systems* (Springer, London, 2007)
12. N. Wu, M.C. Zhou, Deadlock-free scheduling for semiconductor track systems based on resource oriented Petri net. *OR Spectrum* **29**(3), 421–443 (2007)
13. J. Liu, Y. Itoh, I. Miyazawa, T. Sekiguchi, A research on Petri net properties using transitive matrix, in *Proceedings of the IEEE International Conference on System, Man, and Cybernetics*, 1999, pp. 888–893, Tokyo, Japan
14. Z.W. Li, M.C. Zhou, Elementary siphons of Petri nets and their application to deadlock prevention in flexible manufacturing systems. *IEEE Trans. Syst. Man Cybern.* **34**, 38–51 (2004)
15. H. Hu, Z.W. Li, A.R. Wang, On the optimal set of elementary siphons in Petri nets for deadlock control in FMS, in *Proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control*, 2006, pp. 244–247, Florida, USA
16. Z.W. Li, N. Wei, Deadlock control of flexible manufacturing systems via invariant-controlled elementary siphons of Petri nets. *Int. J. Adv. Manuf. Technol.* **33**, 24–35 (2007)
17. E. Roszkowska, Supervisory control for deadlock avoidance in compound processes. *IEEE Trans. Syst. Man Cybern.* **34**, 52–64 (2004)
18. K.L. Xing, X.J. Jin, Y. Feng, Deadlock avoidance Petri net controller for manufacturing systems with multiple resource service, in *Proceedings of the IEEE Conference on Robotics and Automation*, 2005, pp. 4757–4761, Barcelona, Spain
19. G. Xu, Z.M. Wu, A kind of deadlock-free scheduling method based on Petri net, in *Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering*, 2002, pp.195–200, Tokyo, Japan
20. G. Xu, Z.M. Wu, Deadlock-free scheduling method using Petri net model analysis and GA search, in *Proceedings of the 2002 International Conference on Control Application*, vol. 2, 2002, pp. 1153–1158, Glasgow, UK
21. J.K. Lee, Scheduling analysis with resources share using the transitive matrix based on P-invariant, in *Proceedings of the 41st SICE Annual Conference*, vol. 2, 2002, pp. 1359–1364, Osaka, Japan
22. Y.J. Song, J.K. LEE, Deadlock analysis of Petri nets using the transitive matrix, in *Proceedings of the 41st SICE Annual Conference*, vol. 2, 2002, pp. 689–694, Osaka, Japan

23. S. Kim, S. Lee, J. Lee, Deadlock analysis of Petri nets based on the resource share places relationship, in *IMACS Multiconference on Computational Engineering in Systems Applications*, 2006, pp. 59–64, Beijing, China
24. J. Lee, Deadlock find algorithm using the Transitive Matrix, in *Proceedings of the CIE'04*, 2004
25. H.J. Huang, L. Jiao, T.Y. Cheung, W.M. Mak, *Property-Preserving Petri Net Process Algebra in Software Engineering* (World Scientific Publishing, Singapore, 2012)
26. T. Murata, Petri net: properties, analysis, and applications. Proc. IEEE **77**, 541–580 (1985)
27. J. Desel, J. Esparza, *Free Choice Petri Nets* (Cambridge University Press, Cambridge, 1995)
28. J. Wang, *Timed Petri Nets: Theory and Application* (Kluwer Academic, Boston, 1998)
29. L. Jiao, H.J. Huang, T.Y. Cheung, Handling resource sharing problem using property-preserving place fusions of Petri nets. J. Circ. Syst. Comput. **17**(3), 365–387 (2008)
30. CPN Tool, maintained by CPN Group, University of Aarhus, Denmark. http://wiki.daimi.au/dk/cpntools/_home.wiki
31. J.M. Proth, I. Minis, *Planning and Scheduling Based on Petri Nets: In Petri Nets in Flexible and Agile Automation*, ed. by M.C. Zhou (Kluwer Academic, Boston, 1995), pp. 109–148
32. J.M. Proth. Petri nets. *Handbook on Architectures of Information Systems*, 133–151 (2006)
33. M.C. Zhou, F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems* (Kluwer Academic, Boston, 1993)
34. H.H. Xiong, M.C. Zhou, Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search. IEEE Trans. Semicond. Manuf. **11**(3), 384–393 (1998)
35. H. Shih, T. Sekiguchi, A timed Petri net and beam search based on-line FMS scheduling systems with routing flexibility, in *Proceedings of IEEE International Conference on Robotic and Automation*, 1991, pp. 2548–2553, Sacramento, CA
36. D.Y. Lee, F. DiCesare, FMS scheduling using Petri nets and heuristic search. IEEE Trans. Robot. Autom. **10**(2), 123–132 (1994)
37. T.H. Sun, C.W. Cheng, L.C. Fu, A Petri net based approach to modeling and scheduling for an FMS and a case study. IEEE Trans. Ind. Electron. **41**(6), 593–601 (1994)
38. Y.M. Wang, N.F. Xiao, H.L. Yin, A two-stage genetic algorithm for large size job shop scheduling problems. Int. J. Adv. Manuf. Technol. **39**, 813–820 (2008)
39. S.Y. Lin, L.C. Fu, T.C. Chiang, Y.S. Shen, Colored timed Petri net and GA based approach to modeling and scheduling for wafer probe center, in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 1434–1439, Taipei, Taiwan
40. S.M. Kamrul Hasan, R. Sarker, D. Cornforth, Hybrid genetic algorithm for solving job-shop scheduling problem, in *IEEE/ACIS International Conference on Computer and Information Science*, 2007, pp. 519–524, Melbourne, Qld
41. L. Xue, Y. Hao, Petri net based scheduling for integrated circuits manufacturing. ACTC Electron. Sinca **29**(8), 1064–1067 (2001)
42. Y.J. Gua, X.N. Zhang, Y. Cao, R.J. Zhao, T.Q. Lin, Workshop scheduling based on RCPN and system development on Internet/Intranet. Mini-Micro Syst. **24**(7), 1285–1288 (2003)
43. J. Chen, F.F. Chen, Performance modeling and evaluation of dynamic tool allocation in flexible manufacturing systems using colored Petri nets: an object-oriented approach. Int. J. Adv. Manuf. Technol. **21**(2), 98–109 (2003)
44. H.J. Huang, Component-based design and optimization for job-shop scheduling systems. Int. J. Adv. Manuf. Technol. **45**(9), 958–967, 2009
45. M.C. Zhou, H. Chiu, H.H. Xiong, Petri net scheduling of FMS using branch and bound method, in *Industrial Electronics, Control, and Instrumentation, Proceedings of the 1995 IEEE IECON*, 1995, pp. 211–216, Orlando, FL
46. J.C. Tay, D. Wibowo, An effective chromosome representation for evolving flexible job shop schedules. Proc. Genet. Evol. Comput. **3103**, 210–221 (2004)
47. W. Xia, Z. Wu, An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput. Ind. Eng. **48**, 409–425 (2005)

48. M. Mastrolilli, L.M. Gambardella, Effective neighborhood functions for the flexible job shop problem. *J. Scheduling* **3**, 3–20 (2000)
49. P.J.M. Van Laarhoven, E.H.L. Aarts, K.J. Lenstra, Job shop scheduling by simulation annealing. *Oper. Res.* **40**, 113–125 (1992)
50. M. Dell'Amico, M. Trubian, Applying tabu search to the job shop scheduling problem. *Ann. Oper. Res.* **40**, 231–252 (1993)
51. N.B. Ho, J.C. Tay, Solving multiple-objective flexible job shop problems by evolution and local search. *IEEE Trans. Syst. Man Cybern. C Appl. Rev.* **38**(5), 674–685 (2008)
52. L.N. Xing, Y.W. Chen, K.W. Yang, Multi-objective flexible job shop schedule: design and evaluation by simulation modeling. *Appl. Soft Comput.* **9**, 362–376 (2009)
53. K.J. Shaw, P.L. Lee, H.P. Nott, Genetic algorithm for multi-objective scheduling of combined batch/continuous process plants, in *Proceedings of the 2000 Congress on Evolutionary Computation*, 2000, pp. 293–300, La Jolla, CA
54. K.S.N. Ripon, C.H. Tsang, S. Kwong, Multi-objective evolutionary job-shop scheduling using jumping genes genetic algorithm. *International Joint Conference on Neural Networks*, 2006, pp. 3100–3107, Vancouver, BC
55. Y.Y. Chung, L.C. Fu, M.W. Lin, Petri net based modeling and GA based scheduling for a flexible manufacturing system, in *Proceedings of the 37th IEEE Conference on Decision and Control*, vol. 4, 1998, pp. 4346–4347, Tampa, FL
56. Z. Tao, T.Y. Xiao, Petri net and GASAs based approach for dynamic JSSP, in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, 2007, pp. 3888–3893, Harbin, China
57. T.C. Chiang, A.C. Huang, L.C. Fu, Modeling, scheduling, and performance evaluation for wafer fabrication: a queueing colored Petri-net and GA-based approach. *IEEE Trans. on Auto. Sci. and Eng.* **3**(3), 330–338 (2006)
58. H.J. Huang, T.P. Lu, Solving a multi-objective flexible job shop scheduling problem with timed Petri nets and genetic algorithm. *Discr. Math. Algorithms Appl.* **2**(2), 221–237 (2010)
59. M. Garey, D. Johnson, R. Sethi, The complexity of flow shop and job shop scheduling. *Math. Oper. Res.* **24**(1), 117–129 (1976)
60. J. Baker, Adaptive selection methods for genetic algorithms, in *Proceedings of the 1st International Conference on Genetic Algorithms*, 1985, pp. 100–111, Hillsdale, NJ, USA

Key Tree Optimization

Minming Li

Contents

1	Introduction	1714
2	Batch Update with Probability on Membership Change.....	1715
2.1	Uniform Probability.....	1716
2.2	Loyal Users.....	1721
2.3	Heterogeneous Probability.....	1724
3	Batch Update with a Fixed Number of Membership Changes.....	1724
3.1	1-Replacement.....	1727
3.2	k -Replacement.....	1727
4	Other Variants.....	1750
4.1	Minimize sequential update cost for a sequence of actions.....	1750
4.2	Minimize Cost for the Next Update.....	1750
4.3	Optimization with Underlying Network Structure.....	1750
5	Conclusion.....	1751
	Cross-References.....	1751
	Recommended Reading.....	1752

Abstract

In the applications where a group of people share some service such as teleconferencing and online TV, the most important security problem is to ensure that only the authorized users can enjoy the service. One popular way to enforce the security is to encrypt the service content using a group key and update it whenever users leave or join. The key tree model is proposed for the purpose of managing key updates. Sometimes, the system manager knows the user behavior related to the group and therefore hopes to reduce the number of encryptions needed to update keys. For example, in applications where the resource is limited and there are always users waiting to join the group, the system manager may

M. Li

Department of Computer Science, City University of Hong Kong, Hong Kong, People's Republic of China

e-mail: minmli@cs.cityu.edu.hk

decide to update the keys whenever k users leave (and hence k new users join) the group. There are also scenarios where there are some users who never leave the group (loyal users), but others will leave the group with a certain probability before the system manager decides to rekey. In all these and other similar scenarios, finding a tree which requires minimum number of encryptions to update the keys will be of great interest to the system manager because encryptions usually take quite some time. In this chapter, the relevant results in this area will be surveyed.

1 Introduction

Secure group communication is one of the major problems in wireless security. How to distribute the keys over insecure wireless channels is quite a challenging problem. There are several fundamentally different approaches to dealing with key management.

Key pre-distribution schemes refer to methods where a trusted authority distributes secret information to users so that only privileged users can get the group key. The concept of probabilistic key pre-distribution was proposed by Eschenauer and Gligor in their pioneering work [13]. Chan et al. [7] generalized this idea by considering q -compositeness, increasing the security and robustness against node loss. Liu and Ning [23], extending the ideas of [7, 13], later designed a polynomial-based framework by using random subsets and grids. Zhu et al. [46] designed LEAP protocol which uses different methods for different types of transmission.

Contributory group key agreement schemes require that each member contribute an equal share of the group key. The Diffie-Hellman key exchange protocol [11] is the first important contributory key management protocol, and is also the pioneer of every other subsequent protocol of this kind. Later, the following generalizations are proposed. Ingemarsson et al. [19], Burmester and Desmedt [5], and Steiner et al. [34–36] arrange users in a logical ring or chain structure and accumulate the keying material while traversing group members one by one. Ateniese et al. [1] and Becker and Wille [3] also ask each node to contribute an input to establish a common secret through successive pairwise message exchanges among the nodes in a secure manner using two-party D-H exchange [11]. In [12, 39, 40], logical tree structures are introduced, and the number of rounds for establishing the group key is reduced to the logarithm of the group size.

Group key management schemes use a trusted third party to perform all bookkeeping tasks, and only one group key is maintained and used by everybody. Regarding functionalities, they can be further divided into two types: broadcast encryption, in which join is not supported, and group key distribution, in which both join and leave are supported.

Broadcast encryption schemes are created by Fiat and Naor [14] to handle dynamic membership changes in secure communication groups. Naor et al. [27] later designed an efficient stateless broadcast encryption scheme SD. SD users' storage requirement $O(\log^2 n)$ was reduced by Halevy et al. [17], making the scheme more practical to large-scale networks. Aside from them, perfect hash families were also used to construct different broadcast encryption schemes, such as the work of Safavi-Naini et al. [30] and Fiat and Naor [14].

Group key distribution protocols are mainly built on top of logical trees. We give more details below since it is the focus of the chapter. Other variants can be found in [2, 4, 6, 24–26, 29, 33, 37, 38].

In the group key distribution problem, there are n subscribers and a group controller (GC) that periodically broadcasts messages (e.g., a video clip) to all the subscribers over an insecure channel. To guarantee that only the authorized users can decode the contents of the messages, the GC will dynamically update the group key for the whole group. Whenever some user leaves or joins, the GC will generate a new group key and in some way notify the remaining users in the group. The update of keys basically provides two kinds of securities. One is *Future Security* which prevents a deleted user from accessing the future content; the other is *Past Security* which prevents a newly joined user from accessing the past content.

Key tree model, which was proposed by Wong et al. [42] and Wallner et al. [41], is one of the most popular models used to achieve the goal above. In the key tree model, the group controller (GC) maintains a tree structure for the whole group. The root of the tree stores a Traffic Encryption Key (TEK) which is used to encrypt the content that should be broadcast to the authorized users. Every leaf node of the key tree represents a user and stores his individual key. Every internal node stores a Key Encryption Key (KEK) shared by all leaf descendants of the internal node. Every user possesses all the keys along the path from the leaf node (representing the user) to the root node. To prevent revoked users from knowing future message contents and also to prevent new users from knowing past message contents, the GC updates a set of keys, whenever a new user joins or a current user leaves, as follows. So long as there is a user change among the leaf descendants of an internal node v , the GC will (1) replace the old key stored at v with a new key and (2) broadcast (to all users) the new key encrypted with the key stored at each child node of v . Note that only users represented by leaf descendants of v can get useful information from the broadcast. Furthermore, this procedure must be done in a bottom-up fashion (i.e., starting with the lowest v whose key must be updated.) to guarantee that a revoked user will not know the new keys. The cost of the above procedure counts the number of encryptions used in step 2 (or equivalently, the number of broadcasts made by the GC).

Because updating keys for each user change is too frequent in some applications, [21] proposed *batch rekeying model* where keys are only updated after a certain period of time. A good survey for key tree management can be found in [15]. In this chapter, only optimization problems in batch updating scenario will be considered. Other works can be found in [8, 18, 20, 31], etc.

2 Batch Update with Probability on Membership Change

When the service is not so frequently broadcasted but the user change is frequent, updating keys after every user change will be too costly and unnecessary. Thus, a batch rekeying strategy was proposed in [21] whereby rekeying is done only periodically instead of immediately after each member change. They designed a

marking algorithm for processing the batch updates. It was shown by simulation that, using their algorithm, among the totally balanced key trees (where internal nodes of the tree have branching degree 2^i), degree 4 is the best when the number of requests (leave/join) within a batch is not large. For large number of requests, using a star (a tree of depth 1) to organize the users outperforms all the balanced key trees mentioned above in their simulation. According to how users behave, the following three scenarios are discussed here: uniform probability, loyal user, and heterogeneous probability.

2.1 Uniform Probability

Zhu et al. [45] proposed the uniform probability model where each of the n positions has the same probability p to independently experience subscriber change during a batch rekeying period. In this model, an internal node v with N_v leaf descendants will have probability $1 - q^{N_v}$ that its associated key k_v requires updating, where $q = 1 - p$. The updating incurs $d_v \cdot (1 - q^{N_v})$ expected broadcast messages by the procedure described above. Thus, the expected updating cost $C(T)$ of a key tree T is defined by $C(T) = \sum_v d_v \cdot (1 - q^{N_v})$ where the sum is taken over all the internal nodes v of T . $C(T)$ can also be viewed as an edge weight summation: For each tree edge $e = (u, v)$ where u is a child of v , its edge weight is defined to be $1 - q^{N_v}$. The optimization problem can be formulated as follows.

Given two parameters $0 \leq p \leq 1$ and $n > 0$, let $q = 1 - p$. For a rooted tree T with n leaves and edge set E , define a weight function $c(e)$ on the tree edges as follows. For every edge $e = (u, v)$ where u is a child of v and v has N_v leaf descendants, let $c(e) = 1 - q^{N_v}$. Define the cost of T as $C(T) = \sum_{e \in E} c(e)$. Find a T for which $C(T)$ is minimized. We say that such a tree is (p, n) -optimal and denotes its cost by $OPT(p, n)$.

Zhu et al. [45] characterized the optimal tree with the restriction that each internal node on level i has 2^{a_i} children and the total number of users is 2^k . Graham et al. [16] extend the search of the optimal key tree for batch updates from restricted classes of trees to all trees.

Below are some results and observations from [16]. The main idea of the algorithm to find the optimal tree is to prove some properties of the optimal tree by changing the structure of an arbitrary tree to one with better properties without increasing the total cost. This way, search space can be greatly reduced and make polynomial-time algorithm possible.

2.1.1 When the Star Is Optimal

First, several definitions are given. A tree is of *depth k* if the longest leaf-root path consists of k edges. A tree of depth 2 is also referred to as a *two-level tree*. A tree of depth 1 is called a *k-star* if it has k leaves. A tree edge (u, v) where u is a child of v is said to be at *depth k* if the path from u to the root consists of k edges. The *branching degree* of a node v is the number of children of v ; the *subtree size* of v , denoted by N_v , refers to the number of leaf descendants of v .

Lemma 1 If the n -star can achieve $OPT(p, n)$, then the $(n-1)$ -star can also achieve $OPT(p, n-1)$.

Proof The lemma can be proved by contradiction. Suppose the $(n-1)$ -star cannot achieve $OPT(p, n-1)$. Let the degree of the root in a $(p, n-1)$ -optimal tree be k where $k < n-1$. Write the optimal cost as $OPT(p, n-1) = k(1-q^{n-1}) + C$, where C represents the contribution to the cost by edges at depth ≥ 2 . Thus, $(n-1)(1-q^{n-1}) > k(1-q^{n-1}) + C$, which implies $n(1-q^n) > (k+1)(1-q^n) + C$. This means the cost of $OPT(p, n)$ can be reduced by adopting the same structure of the $(p, n-1)$ -optimal tree but with root degree $k+1$, a contradiction. \square

Lemma 2 When $0 \leq q \leq 3^{-1/3}$, the n -star is strictly better than any two-level tree.

Proof A two-level tree can be obtained from a star by successively grouping certain nodes together to form a subtree of the root. To prove the lemma, one only needs to show that the above operation always increases the cost of the tree, i.e., for any grouping size k where $1 < k < n$, one needs to show that $1-q^n < \frac{1}{k}(1-q^n) + 1-q^k$. This is trivially true when $k=1$ or $q=0$, so one can assume that $k \geq 2$ and $q > 0$.

With fixed $q > 0$, define $f(k)$ for integer k by $f(k) = k \log_k(1/q)$. Notice that for any fixed q where $0 < q < 1$, the value of $f(k)$ is minimized when $k=3$. Thus, when $0 < q \leq 3^{-1/3}$, one has $k \log_k(1/q) \geq 1$ which implies $kq^k \leq 1$. Hence, for $0 < q \leq 3^{-1/3}$, the following deductions are correct:

$$\begin{aligned} 1 - q^n - \left(\frac{1}{k}(1 - q^n) + 1 - q^k \right) &= \frac{1}{k}(kq^k - 1 - (k-1)q^n) \\ &< \frac{1}{k}(kq^k - 1) \\ &\leq 0. \end{aligned}$$

\square

Lemma 3 Let p and n be given. Suppose that the n -star is strictly better than any two-level tree, then the n -star is the (p, n) -optimal tree.

Proof If the n -star is not a (p, n) -optimal tree, then one can transform the (p, n) -optimal tree from the bottom up, every time combining two levels into a star. By Lemma 1, the m -star is strictly better than any two-level tree for $1 < m < n$ and p . Since one level is always better than two levels, one can eventually transform the optimal tree into the n -star without increasing the cost. \square

Theorem 1 When $1 \geq p \geq 1 - 3^{-1/3}$, the n -star is the (p, n) -optimal tree for any n . For $2 \leq n \leq 4$, the n -star is the (p, n) -optimal tree for any $p > 0$.

Proof The first part of the theorem follows from Lemmas 2 and 3. The cases of $2 \leq n \leq 4$ can be verified easily. \square

2.1.2 Properties of an Optimal Tree

By [Theorem 1](#), the structure of an (p, n) -optimal tree is uniquely determined for $0 \leq q \leq 3^{-1/3} \approx 0.693$ (or $1 \geq p \geq 0.307$). In the following, some properties of the optimal trees will be derived which will be used for constructing an (p, n) -optimal tree in the remaining range $1 \geq q > 3^{-1/3}$. Note that the following [Lemmas 4](#) and [5](#) as well as [Theorem 2](#) are true for all (p, n) -optimal trees where $0 \leq p \leq 1$ and $n > 0$.

For a tree T , associate a value $t_v = q^{N_v}$ with every node v (thus, $t_v = q$ if v is a leaf). The subtree rooted at u is denoted by T_u . T_u is called a subtree of v if u is a child of v .

Lemma 4 *For a non-root internal node v with branching degree k in a (p, n) -optimal tree, every child u of v satisfies $t_u \geq \frac{k-1}{k}$.*

Proof If $t_u < \frac{k-1}{k}$, one can move u up to become a sibling of v , as shown in [Fig. 1](#). In this way, the total cost of the tree is increased by

$$\begin{aligned}\Delta C &= (1 - q^{N_w}) + (k - 1)(1 - q^{N_v - N_u}) - k(1 - q^{N_v}) \\ &< 1 + (k - 1)(1 - q^{N_v - N_u}) - k(1 - q^{N_v - N_u} t_u) \\ &= q^{N_v - N_u} (k t_u - (k - 1)) \\ &< 0\end{aligned}$$

where w is the parent of v and N_v represents the value before the transformation. This contradicts the cost optimality of the original tree. \square

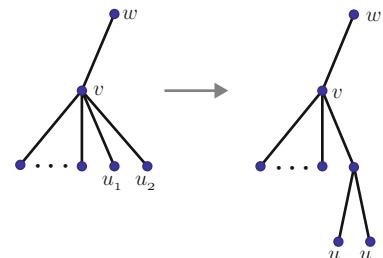
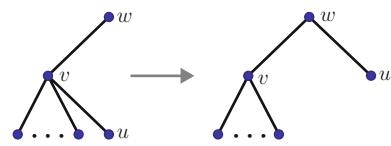
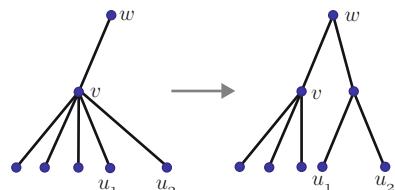
Lemma 5 *Every non-root internal node in a (p, n) -optimal tree has branching degree ≤ 5 .*

Proof By [Lemma 4](#), if a non-root internal node v in the optimal tree has branching degree $k \geq 6$, then $t_u \geq \frac{k-1}{k}$ for any child u of v . One can group together two children of v , with the largest and the second largest t_u values, to form a single subtree of v as shown in [Fig. 2](#). By this transformation, the total cost is increased by

$$\begin{aligned}\Delta C &= 2(1 - t_{u_1} t_{u_2}) - (1 - t_v) \\ &= t_v - 2t_{u_1} t_{u_2} + 1.\end{aligned}$$

Note that t_v is the product of t_u over all children u of v . Thus, $t_v < (t_{u_1} t_{u_2})^{k/2}$, which implies $\Delta C < z^k - 2z^2 + 1$ where $\frac{k-1}{k} \leq z \leq 1$.

It is easy to verify that $z^6 - 2z^2 + 1 < 0$ for $5/6 \leq z \leq 1$. Because the value of $z^k - 2z^2 + 1$ decreases with k for fixed z , the inequality $\Delta C < 0$ holds for any $k \geq 6$ and $\frac{k-1}{k} \leq z \leq 1$, which proves the lemma. \square

Fig. 1 Tree transformation 1**Fig. 2** Tree transformation 2**Fig. 3** Tree transformation 3

Theorem 2 *In an (p, n) -optimal tree:*

1. *Any internal node other than the root must have branching degree ≤ 4 .*
2. *The size of any subtree T_v , where v is a child of the root, must be upper bounded by $\max\{4 \log_q \frac{1}{2}, 1\}$.*

Proof By using Lemma 5, to prove property 1, it is sufficient to show that the optimal tree does not have any internal node with branching degree 5.

Assume that there is a non-root internal node v with five children u_1, \dots, u_5 . For simplicity, write t_{u_i} as t_i and assume $t_1 \geq \dots \geq t_5$. First observe that $z^5 - 2z^2 + 1 < 0$ when $z \geq 0.86$. According to Lemma 4 and the proof of Lemma 5, both conditions $t_1 t_2 < (0.86)^2 = 0.7396$ and $0.8 \leq t_i < 0.86$ for $2 \leq i \leq 5$ must hold. One can now prove that under these conditions, another tree transformation will reduce the total cost which contradicts the tree's optimality. The optimal tree can be transformed into tree T' as shown in Fig. 3. By doing so, the total cost is increased by

$$\begin{aligned}\Delta C &= 3(1 - t_3 t_4 t_5) + 2(1 - t_1 t_2) + (1 - t_w) - 5(1 - t_v) \\ &< -2t_1 t_2 - 3t_3 t_4 t_5 + 5t_v + 1 \\ &= (5t_3 t_4 t_5 - 2)t_1 t_2 - (3t_3 t_4 t_5 - 1)\end{aligned}$$

where t_v represents the value before transformation. By using the fact that $t_i \geq 0.8$ for $1 \leq i \leq 5$ and $t_1 t_2 < 0.7396$, it can be verified that $\Delta C < 0$. This completes the proof of property 1.

For property 2, as is shown in [Lemma 4](#), any child u of v satisfies $q(u) = q^{N_u} > 1/2$. Thus, $N_u < \log_q \frac{1}{2}$. Since v has branching degree at most 4 by property 1 and also $N_v \geq 1$, one has $N_v \leq \max\{4 \log_q \frac{1}{2}, 1\}$. This completes the proof of the theorem. \square

2.1.3 Algorithm for Constructing the Optimal Tree

A (p, n) -optimal tree can be constructed by assembling a forest of suitable subtrees. Firstly, the cost function needs to be generalized from trees to forests as follows.

Definition 1 For $L \leq n$, define a (p, n, L) -forest to be a forest of key trees with L leaves in total. The cost of the tree edges in the forest is defined as before, while the cost of the forest is the sum of individual tree costs plus $k \cdot (1 - q^n)$, where k is the number of trees in the forest. The (p, n, L) -forest with minimum cost is referred to as the optimal (p, n, L) -forest, and the corresponding minimum cost is denoted by $F(p, n, L)$.

Theorem 3 For any fixed p , [Algorithm 1](#) computes the (p, n) -optimal tree cost in $O(n)$ time.

Proof By [Theorem 2](#), in a (p, n) -optimal tree, any subtree T_v where v is a child of the root satisfies (1) its size is at most $\max\{4 \log_q \frac{1}{2}, 1\}$ and (2) the branching degree of any internal node in T_v is at most 4. For fixed q , $4 \log_q \frac{1}{2}$ can be viewed as a constant K . For each i where $2 \leq i \leq K$, the minimum cost $R(i)$ of any tree T_v with size i and subject to the degree restriction stated in (2) will be the focus of the calculations. The value of $R(i)$ can be computed in constant time as follows. First, define (k_1, k_2, k_3, k_4) to be an i -quadruple if $k_1 + k_2 + k_3 + k_4 = i$, $0 \leq k_1, k_2, k_3, k_4 \leq i$, and $k^* \geq 2$ where k^* is the number of positive elements in (k_1, k_2, k_3, k_4) . Then, $R(i)$ is the minimum value of $R(k_1) + R(k_2) + R(k_3) + R(k_4) + (1 - q^i) \cdot k^*$ over all i -quadruples (k_1, k_2, k_3, k_4) , and it can be computed in $O(K^3)$ time using dynamic programming. Now, one can obtain the true (p, n) -optimal tree also by dynamic programming, by computing optimal (p, n, L) -forests as subproblems of size L for $1 \leq L \leq n$ as given in [Algorithm 1](#). Thus, the total running time of the algorithm is $O(n \cdot K + K^4)$ which is $O(n)$ for fixed p . [Algorithm 1](#) focuses on computing the optimal tree cost; the tree structure can be obtained by keeping track of the optimal branching at every dynamic programming iteration. \square

2.1.4 Optimal Trees as $p \rightarrow 0$

[Algorithm 1](#) has running time $O(n \cdot K + K^4)$ where K is upper bounded by $4 \log_q \frac{1}{2}$ (and also by n). We can regard $O(K^4)$ as a constant term for fixed value of p , but its value gets large as $p \rightarrow 0$. Therefore, the structure of (p, n) -optimal trees as $p \rightarrow 0$ is worth exploring. The following theorem is proved in [16].

Algorithm 1 Computing optimal key tree

Input: n and p ($0 \leq p \leq 1$)**Output:** Optimal tree cost $OPT(n, p)$

```

 $q = 1 - p$ 
 $K = \min\{4 \log_q \frac{1}{2}, n\}$ 
if  $q \leq 3^{-1/3}$  or  $2 \leq n \leq 4$  then
     $OPT(p, n) \leftarrow n \cdot (1 - q^n)$ 
    Return  $OPT(p, n)$ 
end if
 $R(1) = 0$ 
for  $i = 2$  to  $4$  do
     $R(i) = i * (1 - q^i)$ 
end for
 $i = 5$ 
while  $i < K$  do
    Compute  $R(i)$ , cost of the restricted  $(p, i)$ -optimal tree.
     $i = i + 1$ 
end while
 $F(p, n, 0) \leftarrow 0$ 
for  $L = 1$  to  $n$  do
     $F(p, n, L) \leftarrow \min_{1 \leq j \leq \min\{K, L\}} (R(j) + 1 - q^n + F(p, n, L - j))$ .
end for
 $OPT(n, p) \leftarrow F(p, n, n)$ 
return  $OPT(n, p)$ 

```

Theorem 4 As $p \rightarrow 0$, the (p, n) -optimal tree $T^*(n)$ always has root degree 3 except for n of the form $4 \cdot 3^t$, in which case $T^*(4 \cdot 3^t)$ has root degree 4, and for $n = 2$, when $T^*(2)$ has root degree 2. When $2 \cdot 3^t \leq n \leq 3^{t+1}$, then $T^*(n)$ is as close to a balanced ternary tree with n leaves as possible. Namely, all subtrees (as well as $T^*(n)$ itself) have root degrees 3, except for the very bottom level, where subtrees of size 2 can occur. However, when $3^t \leq n < 2 \cdot 3^t$, $T^*(n)$ can deviate substantially from a balanced ternary tree. For example, $T^*(39)$ has three subtrees of sizes 9, 12, and 18 (see Fig. 4).

It is observed by Graham et al. [16] to be difficult to predict the sizes of the subtrees in the optimal tree $T^*(n)$ for certain values of n . For example, for $n = 1,252$, the sizes are 280, 486, and 486, while for $n = 1,253$, the sizes are 324, 443, and 486.

In [22], the authors studied the approximate optimal tree. They design a simple $O(n)$ algorithm whose approximation ratio is nearly 2. They also design a refined algorithm and show that it achieves an approximation ratio of $1 + \epsilon$.

2.2 Loyal Users

The concept of “loyal users” is first proposed in [28]. They restructure the key tree scheme and design a modified protocol considering the membership duration. Two kinds of users, volatile and permanent members, are separately grouped into two

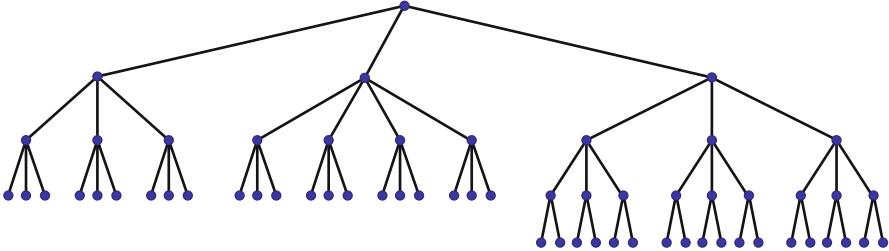


Fig. 4 An optimal $T^*(39)$

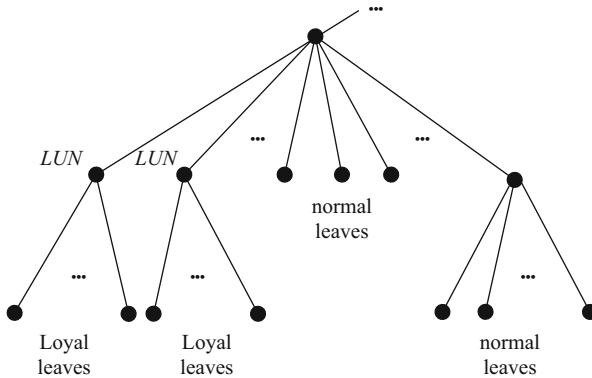


Fig. 5 An example tree with LUN

key trees. The newly joined user is first regarded as a volatile member; once the duration of the user exceeds a threshold, it is moved to the permanent member group. Chan et al. [9] extends the uniform probability batch update key tree model by considering two types of users in the same key tree. Users belonging to the first type have probability p to issue a leave request (the vacant place will be taken by the newly joined user that belongs to this type), while users belonging to the second type are considered loyal (with probability zero to leave the group) to the group. The objective is to theoretically study the optimal key tree structure to minimize expected rekeying cost. Note that loyal users have zero probability to leave the group. To control the tree cost, the loyal users should be appropriately arranged. They define a special internal node, Loyal User Node (LUN), whose children all are loyal users. [Figure 5](#) shows an example.

For a key tree T with n leaves, N_v (and L_v) is used to denote the number of “normal” (and “loyal”) leaf descendants of node v . The normal users have probability p to leave the group, while the loyal users have probability zero. Note that in this new model, the expected updating cost for every internal node v is $d_v \cdot (1 - (1 - p)^{N_v})$. Thus, one can apply the distribution trick again as follows.

Let V be the node set of T , including all internal nodes, normal and loyal leaves. Given p and n and let $q = 1 - p$, the weight $c'(u)$ of node u over V can be defined as

$$c'(u) = \begin{cases} 0 & \text{if } u \text{ is the root} \\ 1 - q^{N_v} & \text{if } u \text{ is a child of internal node } v \end{cases}$$

The cost of T is defined as $C'(T) = \sum_{u \in V} c'(u)$. Accordingly, T is a $(p, n)'$ -optimal tree if $C'(T)$ is minimum among all possible tree structures. The cost of the $(p, n)'$ -optimal tree is denoted as $OPT'(p, n)$. Our objective is to find the structure of $(p, n)'$ -optimal trees.

They first showed the following property of loyal users.

Lemma 6 *For any tree with loyal users, removing loyal leaves strictly decreases the tree cost.*

Proof Consider the subtree T_v rooted at an internal node v which has l_v ($l_v \geq 1$) children to be loyal leaves. Let N_v be the number of normal leaves on T_v . Removing all loyal children from v will decrease the tree cost by $\Delta C = l_v(1 - q^{N_v})$. It is obvious that $\Delta C \geq 0$ as $l_v \geq 1$. The cost on the remaining nodes does not change. This finishes the proof. \square

Let r be the root of the tree T and L_r be the number of loyal leaves ($L_r \geq 1$). One can obtain a tree with no larger cost by doing the following three steps:

1. Arbitrarily remove $L_r - 1$ loyal leaves from T .
2. Replace the last loyal leaf u with a LUN v .
3. Add all loyal leaves removed in step 1 including u to be children of v .

By Lemma 6, step 1 strictly decreases the tree cost. For steps 2 and 3, the new LUN v will not increase the tree cost because all nodes inserted to v are loyal leaves which do not issue any join/leave request. Thus, the transformation does not increase the tree cost. In fact, the newly joined loyal users will be placed under the LUN. This does not make any change to the tree structure excluding the LUN. In other words, the LUN can represent all loyal leaves. Thus, optimal trees can be found by focusing on trees with n normal leaves and one LUN. Similar properties can be proved about the optimal key tree based on the proof paradigm in [16]. The final dynamic programming algorithm to calculate the optimal key tree with loyal users is summarized below.

Theorem 5 *Algorithm 2 computes a $(p, n)'$ -optimal tree in $O(n \cdot K + K^4)$ time where $K = \min\{4(\log q^{-1})^{-1}, n\}$.*

Proof Define a $(p, n, L)'$ -forest to be a forest of key trees with L normal leaves and one LUN. Similarly define (p, n, L) -forest to be a forest of key trees with L normal leaves and no LUN. The cost of the forest of the trees is the cost of the individual trees plus $k(1 - q^n)$ where k is the number of trees. Use $F'(p, n, L)$ to denote the minimum cost of a $(p, n, L)'$ -forest and $F(p, n, L)$ to denote the minimum cost of a (p, n, L) -forest. \square

Algorithm 2 Computing optimal key tree with a LUN

Input: n : Number of normal users ($n \geq 1$)
 p : Probability of normal users to update ($0 < p < 1$)
Output: Cost of $(p, n)'$ -optimal tree $OPT'(p, n)$

```

 $q = 1 - p; K = \min\{4(\log q^{-1})^{-1}, n\}$ 
//Optimal  $(n)'$ -star tree
if  $q \leq 0.57$  then
     $OPT'(p, n) = (n + 1)(1 - q^n)$ 
    return  $OPT'(p, n)$ 
end if
// Optimal tree for  $n \leq 4(\log q^{-1})^{-1}$ 
 $R(0) = R(1) = 0; R(2) = 2(1 - q^2); R'(0) = 0; R'(1) = 2(1 - q); R'(2) = 3(1 - q^2);$ 
 $i = 3;$ 
while  $i \leq K$  do
     $R(i) = \min\{R(k_1) + R(k_2) + R(k_3) + R(k_4) + (1 - q^i)k^*\};$ 
     $R'(i) = \min\{R'(k_1) + R(k_2) + R(k_3) + R(k_4) + (1 - q^i)k^*\};$ 
     $i = i + 1;$ 
end while
if  $n \leq 4(\log q^{-1})^{-1}$  then
    return  $OPT'(p, n) = R'(n).$ 
end if
// Optimal tree when  $n > 4(\log q^{-1})^{-1}$ 
 $F(p, n, L) = 0; F'(p, n, L) = 0;$ 
for  $L = 1$  to  $n$  do
     $F(p, n, L) = \min_{1 \leq j \leq \min\{K, L\}}\{R(j) + 1 - q^n + F(p, n, L - j)\};$ 
     $F'(p, n, L) = \min_{1 \leq j \leq \min\{K, L\}}\{R'(j) + 1 - q^n + F(p, n, L - j)\};$ 
end for
 $OPT'(p, n) = F'(p, n, n)$ 
return  $OPT'(p, n)$ 

```

2.3 Heterogeneous Probability

When a different user has different probability of leaving the group, then the problem becomes much complicated. [Lemmas 4](#) and [5](#) and [Theorem 2](#) are still true. The size limit for each subtree connecting to the root is now only related to the smallest possible probability of leaving. However, with all these good features preserved, one still needs to guess the number of different probabilities in each branch, therefore making polynomial-time algorithm impossible if one still follows the dynamic programming framework.

3 Batch Update with a Fixed Number of Membership Changes

In this section, it is assumed that k users will leave the group during the batch period. A key tree which can minimize the worst case number of encryptions caused during the batch period is the target.

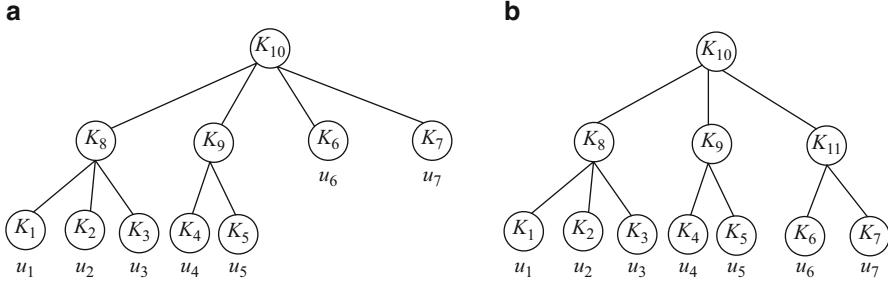


Fig. 6 Two structures of a group with seven users

As shown in Fig. 6a, there are seven users in the group. We take the deletion of user u_4 as an example. Since u_4 knows k_4 , k_9 , and k_{10} , the GC needs to update the keys k_9 and k_{10} (the node that stores k_4 disappears because u_4 is already deleted from the group). GC will encrypt the new k_9 with k_5 and broadcast it to notify u_5 . Note that only u_5 can decrypt the message. Then, GC encrypts the new k_{10} with k_6 , k_7 , and k_8 and the new k_9 , respectively, and then broadcasts the encrypted messages to notify the users. Since all the users and only the users in the group can decrypt one of these messages, the GC can safely notify the users except user u_4 about the new TEK. The deletion cost measured as the number of encryptions is 5 in this example.

In popular servers, since there are always users on the waiting list, the batch rekeying model will update the TEK by replacing constant k revoked users with k waiting users. The newly joined k users will take the k positions which are vacant due to the leave of k users, thus keeping the structure of the tree unchanged for each update. This reduces the update frequency. The updating cost equals the summation of degrees of all ancestors of the k old users (leaves). Thus, the objective is to find an optimal tree where the worst case updating cost incurred by the k -user membership change is minimum. This problem is denoted as *k-replacement problem*. In fact, [32] showed that 2–3 tree is the optimal tree for 1-replacement problem. Wu et al. [43] further denote the problem to find the optimal tree when k users are deleted from the tree as *k-deletion problem*. As shown in Fig. 6b, the tree has a worst case cost 5 for 1-deletion problem and worst case cost 6 for 1-replacement problem.

In the following, a node u is said to have degree/branching degree d if it has d children in the tree.

Wu et al. [43] first define the *k-replacement problem* formally as follows.

Definition 2 Given a tree T , the number of encryptions incurred by replacing u_{i_1}, \dots, u_{i_k} with k new users is denoted as $C_T(u_{i_1}, \dots, u_{i_k}) = \sum_{v \in (\bigcup_{1 \leq j \leq k} ANC(u_{i_j}))} d_v$ where $ANC(u)$ is the set of u 's ancestor nodes and d_v is v 's degree. *k-replacement cost* refers to the maximum cost among all possible combinations and is written as $C_k(T, n) = \max_{i_1, i_2, \dots, i_k} C_T(u_{i_1}, u_{i_2}, \dots, u_{i_k})$.

An optimal tree $T_{n,k,opt}$ (abbreviated as $T_{n,opt}$ if the context is clear) for the k -replacement problem is a tree which has the minimum k -replacement cost over all trees with n leaves, i.e., $C_k(T_{n,opt}, n) = \min_T C_k(T, n)$. This optimal cost is denoted as $OPT_k(n)$. The k -replacement problem is to find the $OPT_k(n)$ and $T_{n,k,opt}$. The k -deletion cost and the k -deletion problem are defined similarly by using the cost incurred by permanently deleting the leaves instead of the cost incurred by replacing the leaves. Note that some keys do not need to be updated if all its leaf descendants are deleted and the number of encryptions needed to update that key is also reduced if some branches of that node totally disappear due to deletion. Notice that the k -deletion problem and the k -replacement problem are not trivially equivalent.

Wu et al. [43] showed that the k -deletion problem can be solved by handling the k -replacement problem. Below are major deductions from [43].

Definition 3 Define a node v to be a pseudo-leaf node if its children are all leaves. In the following two lemmas, t is used to denote the number of pseudo-leaf nodes in a tree T .

Lemma 7 *If $t \leq k$, then the k -deletion cost of T is at least $n - k$.*

Proof When $t \leq k$, in order to achieve k -deletion cost, at least one leaf needs to be deleted from each pseudo-leaf node. Suppose on the contrary there exists one pseudo-leaf node v where none of its children belong to the k leaves deleted. The discussion can be divided into two cases.

First, if each of the k leaves is a child of the remaining $t - 1$ pseudo-leaf nodes, then there exists one pseudo-leaf node u with at least two children deleted. In this case, a larger deletion cost can be achieved if one deletes one child of v while keeping one more child of u undeleted.

Second, if some of the k leaves are not from the remaining $t - 1$ pseudo-leaf nodes, then one can assume that u is one of them whose parent is not a pseudo-leaf. Then there exists one of u 's siblings w that has at least one pseudo-leaf w' (w' can be w itself) as its descendant. If no children of w' belongs to the k leaves, then deleting a child of w' while keeping u undeleted incurs larger deletion cost. If at least one child of w' belongs to the k leaves, then deleting a child of v while keeping u undeleted incurs larger deletion cost.

It can be seen that in the worse case deletion, each pseudo-leaf node has at least one child deleted, which implies that all the keys in the remaining $n - k$ leaves should be used once as the encryption key in the updating process. Hence the k -deletion cost of T is at least $n - k$. \square

Lemma 8 *If $t > k$, then the k -deletion cost is $C_k(T, n) - k$ where $C_k(T, n)$ is the k -replacement cost.*

Proof Using similar arguments as in the proof of Lemma 7, one can prove that when $t > k$, the k -deletion cost can only be achieved when the k deleted leaves are from k different pseudo-leaf nodes. Then it is easy to see that the k -deletion cost is $C_k(T, n) - k$ where $C_k(T, n)$ is the k -replacement cost. \square

Although worst case costs between these two problems do not always differ by a constant k given some tree T , they show that the worst case costs between their optimal tree structures always differ by k .

Theorem 6 *When considering trees with n leaves, the optimal k -deletion cost is $OPT_k(n) - k$ where $OPT_k(n)$ is the optimal k -replacement cost.*

Proof Note that in the tree where all n leaves have the same parent (denoted as one-level tree), the k -deletion cost is $n - k$. By Lemma 7, any tree with the number of pseudo-leaf nodes at most k has the k -deletion cost at least $n - k$. Hence, one only needs to search the optimal tree among the one-level tree and the trees with the number of pseudo-leaf nodes larger than k . Moreover, in the one-level tree T , the k -deletion cost is $n - k = C_k(T, n) - k$ where $C_k(T, n)$ is the k -replacement cost. Further, by Lemma 8, all the trees in the scope for searching the optimal tree have k -deletion cost $C_k(T, n) - k$, which implies that the optimal k -deletion cost is $OPT_k(n) - k$ where $OPT_k(n)$ is the optimal k -replacement cost. \square

The above analysis implies that the optimal tree for the k -deletion problem can be obtained by solving the k -replacement problem. Therefore, one only needs to consider the k -replacement problem in the following.

3.1 1-Replacement

In [32], the authors investigate the optimal tree structure with n leaves where the worst case single-deletion cost is minimum. Their result shows that the optimal tree is a special kind of 2–3 tree defined below.

Definition 4 2–3 tree is a tree with n leaves constructed in the following way:

1. When $n \geq 5$, the root degree is 3 and the number of leaves in three subtrees of the root differs by at most 1. When $n = 4$, the tree is a complete binary tree. When $n = 2$ and $n = 3$, the tree has root degree 2 and 3, respectively. When $n = 1$, the tree only consists of a root.
2. Each subtree of the root is a 2–3 tree defined recursively.

3.2 k -Replacement

The following discussion also comes from [43].

3.2.1 First Step: Loose Degree Bound for the k -Replacement Problem

Given a constant k , k -replacement problem can be computed in polynomial time by deducing a loose degree bound. In the following proofs, the following procedure is often done: First, choose a template tree T and then construct a tree T' by removing from T some leaves together with the exclusive part of leaf-root paths of those leaves. Here, the exclusive part of a leaf-root path includes those edges that are not on the leaf-root path of any of the remaining leaves. In this case, T is called a template tree of T' . By the definition of the k -replacement cost, the following fact holds.

Fact 1 *If T is a template tree of T' , then the k -replacement cost of T' is no larger than that of T .*

Lemma 9 $OPT_k(n)$ is nondecreasing when n increases.

Proof Suppose on the contrary $OPT_k(n_1) > OPT_k(n_2)$ when $n_1 \leq n_2$, then there exist two trees T_1 and T_2 satisfying $C_k(T_1, n_1) = OPT_k(n_1)$ and $C_k(T_2, n_2) = OPT_k(n_2)$. One can take T_2 as a template tree and delete the leaves until the number of leaves decreases to n_1 . The resulting tree T'_2 satisfies $C_k(T'_2, n_1) \leq C_k(T_2, n_2) < OPT_k(n_1)$ by Fact 1, which contradicts the definition of $OPT_k(n_1)$. The lemma is then proved. \square

Lemma 10 $T_{n,opt}$ has root degree upper bounded by $(k + 1)^2 - 1$.

Proof The value of root degree $d \geq (k + 1)^2$ can be divided into two sets, $\{d|(k + t)^2 \leq d < (k + t)(k + t + 1), d, k, t \in N, t \geq 1\}$ and $\{d|(k + t - 1)(k + t) \leq d < (k + t)^2, d, k, t \in N, t \geq 2\}$. Take $k = 2$, for instance, the first set is $\{9, 10, 11, 16, 17, 18, 19, 25, \dots\}$, while the other is $\{12, 13, 14, 15, 20, 21, 22, 23, \dots\}$.

Case 1: $(k + t)^2 \leq d < (k + t)(k + t + 1)$ ($t \geq 1$).

Write d as $(k + t)^2 + r$ where $0 \leq r < k + t$. Given a tree T , one can transform it into a tree with root degree $k + t$ as Fig. 7 shows. In the resulting tree T' , subtrees $T_{u_1}, \dots, T_{u_{k+t}}$ are $k + t$ subtrees where the root u_1, \dots, u_{k+t} are on level one. Among the $k + t$ subtrees, there are r subtrees with root degree $k + t + 1$ and $k + t - r$ subtrees with root degree $k + t$. Suppose that the k -replacement cost of T' is incurred by replacing k_1, k_2, \dots, k_s users from subtrees $T_{i_1}, T_{i_2}, \dots, T_{i_s}$ respectively where $k_1 + k_2 + \dots + k_s = k$ and $s \leq k$. The corresponding cost is $C_k(T', n) = \sum_{j=1}^s C_{k_j}(T_{i_j}, n_{i_j}) + D_0$ where n_{i_j} is the number of leaves in T_{i_j} and D_0 is the cost incurred in the first two levels.

In the original tree T , the corresponding cost for replacing those leaves is $C_k(T, n) = \sum_{j=1}^s C_{k_j}(T_{i_j}, n_{i_j}) + d = \sum_{j=1}^s C_{k_j}(T_{i_j}, n_{i_j}) + (k + t)^2 + r$. In the following, one can show $C_k(T, n) \geq C_k(T', n)$, i.e., $D_0 \leq (k + t)^2 + r$ when $t \geq 1$.

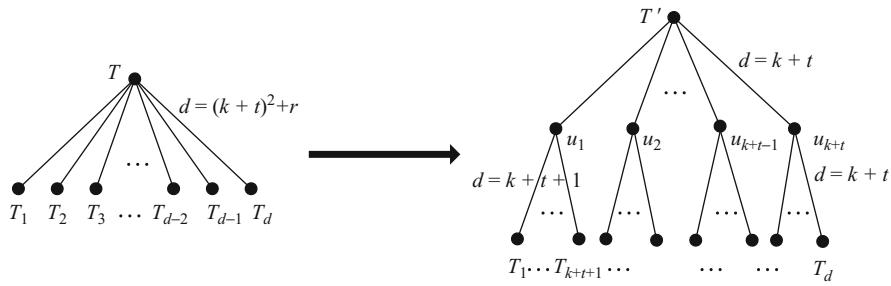


Fig. 7 Transformation of the tree which has root degree $d = (k+t)^2 + r$

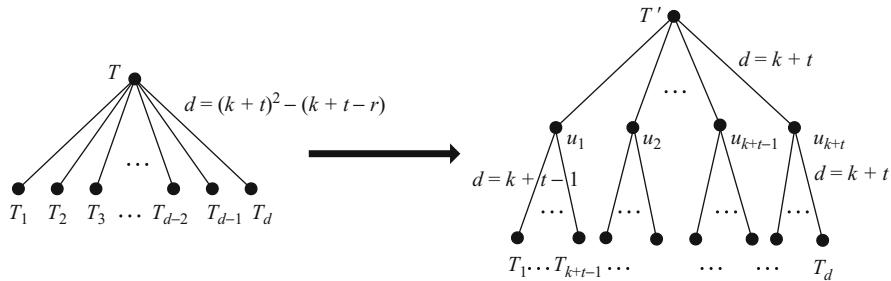


Fig. 8 Transformation of the tree which has root degree $d = (k+t)^2 - (k+t-r)$

Firstly, if $r \leq k$, the cost D_0 is at most $r(k+t+1) + (k-r)(k+t) + k+t$ where there are r users coming from r subtrees with root degree $k+t+1$ and $k-r$ users coming from $k-r$ subtrees with root degree $k+t$. Therefore, one has

$$D_0 \leq (k+t+1)r + (k+t)(k-r) + k+t = (k+t)(k+1) + r \leq (k+t)^2 + r.$$

Secondly, if $r > k$, the cost D_0 is at most $(k+t) + (k+t+1)k$ where the k users are all from k subtrees which have root degree $k+t+1$. Therefore, one has

$$D_0 \leq (k+t) + (k+t+1)k \leq (k+t)(k+1) + k \leq (k+t)^2 + r.$$

Hence, in both situations, the condition $t \geq 1$ ensures that the transformation from \$T\$ to \$T'\$ does not increase the k -replacement cost.

Case 2: $(k+t-1)(k+t) \leq d < (k+t)^2$ ($t \geq 2$).

In this case, d can be written as $(k+t)^2 - (k+t-r)$ where $0 \leq r < k+t$. One can transform \$T\$ into a tree with root degree $k+t$ where there are r subtrees with root degree $k+t$ and $k+t-r$ subtrees with root degree $k+t-1$, as Fig. 8 shows. Similarly for $r \leq k$, one has

$$D_0 \leq (k+t)r + (k+t-1)(k-r) + k + t = (k+t)(k+1) - (k-r) \leq (k+t)^2 - (k+t-r).$$

The last inequality holds because $t \geq 2$.
While for $r > k$, one has

$$D_0 \leq (k+t)k \leq (k+t)^2 - (k+t-r).$$

Thus, $t \geq 2$ ensures that the transformation from T to T' does not increase the k -replacement cost.

Therefore, for any root degree $d \geq (k+1)^2$, through a series of transformations, the original tree can be transformed into a tree with root degree less than $(k+1)^2$ without increasing the k -replacement cost. For example, when $d = 120$ and $k = 2$, since $120 = (k+9)^2 - 1$, firstly one can transform it into a tree with root degree $k+9 = 11$. Since $11 = (k+1)^2 + 2$ is still greater than $(k+1)^2$, one can further transform the resulting tree into a tree with root degree $k+1 = 3$. Because $3 < (2+1)^2$, no further transformation will take place. The lemma is finally proved. \square

[Lemma 10](#) suggests that one can find an optimal tree for the k -replacement problem among trees whose root degree is at most $(k+t)^2 - 1$. Note that the degree bound in [Lemma 10](#) is only for the root. One can also extend this property to all the internal nodes. This leads to a polynomial-time algorithm to compute the optimal k -replacement tree given a constant k .

Theorem 7 *Any internal node in $T_{n,opt}$ has degree upper bounded by $(k+1)^2 - 1$. Given a constant k , the k -replacement problem can be computed in $O(n^{(k+1)^2})$ time.*

Proof Suppose T is an optimal tree and has an internal node v which has degree $d_v \geq (k+1)^2$, as shown in [Fig. 9](#). One can transform the subtree T_v rooted at v into T'_v in the similar way as in [Lemma 10](#) (The resulting tree is denoted by T'). One can prove that the k -replacement cost in the resulting tree T' does not increase. Consider all the three possible relative positions of those k leaves whose membership change achieves k -replacement cost in T' . First, if none of the leaves are from T'_v , then the replacement cost of the corresponding leaves in T will be the same as the replacement cost in T' . If there are m ($m < k$) leaves from T'_v while the others are not, then replacing the corresponding leaves in T incurs no smaller replacement cost compared to that in T' , because replacing m leaves of T_v incurs no smaller cost than replacing the corresponding m leaves in T'_v due to similar analysis shown in [Lemma 10](#). If all the k leaves are from T'_v , then replacing the corresponding leaves in T again incurs no smaller replacement cost than T' by [Lemma 10](#). In all the above three cases, T' has replacement cost no greater than T , which enables us to transform the degree of every internal node to be equal to or less than $(k+1)^2 - 1$ without increasing the k -replacement cost. By enumerating all the possible degrees of the internal node, one can design a $O(n^{(k+1)^2})$ time dynamic programming algorithm to iteratively compute the optimal

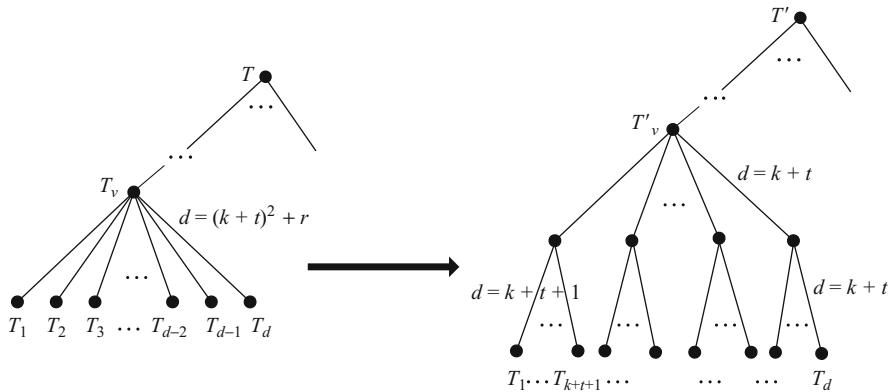


Fig. 9 Transformation of tree T which has an internal node with degree $d \geq (k+t)^2$

k -replacement cost $OPT_k(i)$ ($1 \leq i \leq n$). By keeping the branching information for each iteration, the optimal k -replacement tree can be constructed with the same complexity. \square

3.2.2 Tight Degree Bound for 2-Replacement Problem

From this subsection on, 2-replacement problem will be the major focus.

Definition 5 Denote the maximum cost to delete a single leaf in a tree T as S_T and the maximum cost to delete two leaves in T as D_T , i.e., $S_T = C_1(T, n)$ and $D_T = C_2(T, n)$.

According to [Theorem 7](#), for the 2-replacement problem, $T_{n, \text{opt}}$ has degree upper bounded by 8. Furthermore, root degree $d = 1$ is not possible in the optimal tree because merging the root and its single child into one node can strictly decrease the worst case updating cost.

Fact 2 For the 2-replacement problem, suppose that a tree T has root degree d where $d \geq 2$ and the d subtrees are T_1, T_2, \dots, T_d . One has

$$D_T = \max_{1 \leq i, j \leq d} \{D_{T_i} + d, S_{T_i} + S_{T_j} + d\}.$$

Proof It is easy to see that replacing any two leaves from a subtree T_i will incur a cost at most $D_{T_i} + d$, while replacing two leaves from two different subtrees T_i and T_j will incur a cost at most $S_{T_i} + S_{T_j} + d$. The 2-replacement cost comes from one of the above cases, and therefore the fact holds. \square

Wu et al. [43] further remove the possibility of degree 8 by the following lemma.

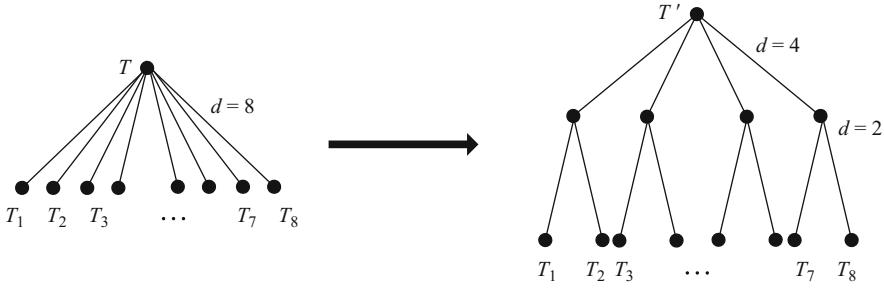


Fig. 10 Transformation of tree T which has an internal node with degree 8

Lemma 11 *For the 2-replacement problem, one can find an optimal tree among the trees with node degrees bounded between 2 and 7.*

Proof By [Theorem 7](#), one only needs to remove the possibility of degree 8 for any internal node. Firstly, any tree with root degree 8 can be transformed into a tree with a smaller root degree and no larger 2-replacement cost. Given a tree T with root degree 8 and subtrees T_1, T_2, \dots, T_8 , one can transform the structure at the root v into degree 4, and each child of v has degree 2, as shown in [Fig. 10](#). In the original tree, 2-replacement cost is $D_T = \max\{D_{T_i} + 8, S_{T_i} + S_{T_j} + 8\}$ ([Fact 2](#)). In the resulting tree T' , the cost of replacing two leaves from different subtrees T_i and T_j is at most $S_{T_i} + S_{T_j} + 8$, and the cost of replacing two leaves from one subtree T_i is at most $D_{T_i} + 6$. Hence, 2-replacement cost in T' is at most $\max\{D_{T_i} + 6, S_{T_i} + S_{T_j} + 8\} \leq D_T$.

Note that the 1-replacement cost of T also does not increase in the transformation. One can then extend this transformation to any internal node as has been shown in the proof of [Theorem 7](#). The lemma is then proved. \square

Based on this bound, the optimal tree can be computed in $O(n^8)$ time by enumerating all possible degrees that are at most 7 using a dynamic programming algorithm. However, $O(n^8)$ is still a bit unaffordable in real computations. Therefore, [43] reduced the complexity to $O(n)$ by accurately determining the structure of the optimal tree. They showed two important properties of the optimal tree (monotone property and 2–3 tree property) and then further remove the possibility of root degree 2 and 3 to reduce the scope of trees within which the optimal tree is searched for.

Lemma 12 (monotone property) *For the 2-replacement problem, suppose a tree T has root degree d where $d \geq 2$ and d subtrees are T_1, T_2, \dots, T_d . Without loss of generality, assume that T has a nonincreasing leaf descendant vector (n_1, n_2, \dots, n_d) , where n_i is the number of leaves in subtree T_i . Then, there exists an optimal tree where $S_{T_1} \geq S_{T_2} \geq \dots \geq S_{T_d}$ and T_i is a template of T_{i+1} for $2 \leq i \leq d - 1$.*

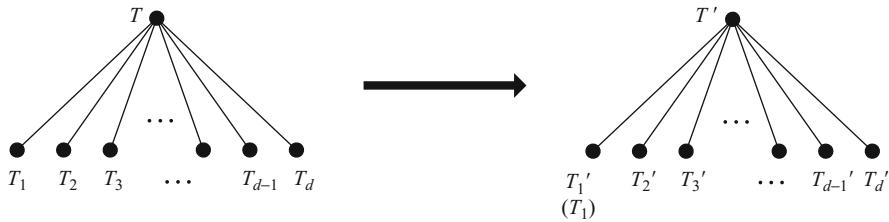


Fig. 11 Transformation of a tree which has $S_{T_1} < S_{T_2}$

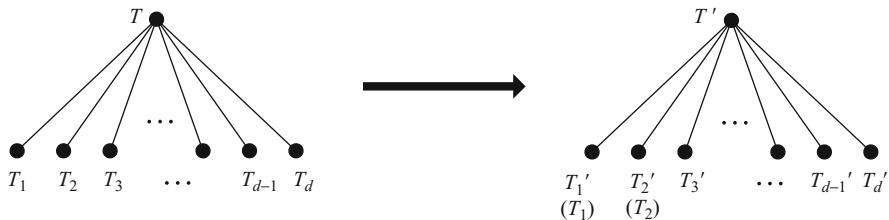


Fig. 12 Transformation of a tree T which has $S_{T_j} < S_{T_{j+1}}$ where $j \geq 2$

Proof If a given optimal tree contradicts the lemma, one can always change this tree into a tree satisfying the monotone property but with equal or less 2-replacement cost.

Case 1: $S_{T_1} < S_{T_2}$.

Let T'_1 be the same as T_1 , and then choose T'_1 to be the template tree and construct a subtree T'_2 by removing $n_1 - n_2$ leaves from T_1 . Similarly, one can construct the subtrees T'_i ($3 \leq i \leq d$) by removing $n_{i-1} - n_i$ leaves from T'_{i-1} , as shown in Fig. 11. Since each subtree T'_i is a template of T'_{i+1} , the final set of new trees satisfies $S_{T'_1} \geq S_{T'_2} \geq \dots \geq S_{T'_d}$ and $D_{T'_1} \geq D_{T'_2} \geq \dots \geq D_{T'_d}$. Then one can prove that 2-replacement cost is not increased in the resulting tree T' . Note that $S_{T'_1} = S_{T_1} < S_{T_2}$ and $D_{T'_1} = D_{T_1}$. Since $D_{T'} = \max_{1 \leq i < j \leq d} \{D_{T'_i} + d, S_{T'_i} + S_{T'_j} + d\} = \max\{D_{T_1} + d, S_{T'_1} + S_{T'_2} + d\}$ and $D_T \geq \max\{D_{T_1} + d, S_{T_1} + S_{T_2} + d\}$, $D_{T'} < D_T$ holds because $S_{T'_1} + S_{T'_2} < S_{T_1} + S_{T_2}$. Hence, the lemma holds in this case.

Case 2: $S_{T_1} \geq S_{T_2} \dots \geq S_{T_j}$ and $S_{T_j} < S_{T_{j+1}}$.

Let T'_1, T'_2 be the same as T_1, T_2 , respectively, and choose T'_2 to be the template tree and construct a subtree T'_3 by removing $n_2 - n_3$ leaves from T_2 . Similarly, one can construct subtrees T'_i ($4 \leq i \leq d$) by removing $n_{i-1} - n_i$ leaves from T'_{i-1} , as shown in Fig. 12. Since each subtree T'_i is a template of T'_{i+1} , the final set of new trees satisfies $S_{T_1} \geq S_{T_2} \geq S_{T'_3} \dots \geq S_{T'_d}$ and $D_{T_2} \geq D_{T'_3} \dots \geq D_{T'_d}$.

Therefore, one has $D_{T'} = \max_{1 \leq i, j \leq d} \{D_{T'_i} + d, S_{T'_i} + S_{T'_j} + d\} = \max\{D_{T_1} + d, D_{T_2} + d, S_{T_1} + S_{T_2} + d\}$, which implies $D_{T'} \leq D_T$.

The 2-replacement cost is not increased in T' in both cases, and therefore the lemma is proved. \square

Fact 3 *For trees satisfying the monotone property (Lemma 12), one has*

$$D_T = \max\{D_{T_1} + d, S_{T_1} + S_{T_2} + d\}.$$

Proof By Fact 2 one has $D_T = \max_{1 \leq i, j \leq d} \{D_{T_i} + d, S_{T_i} + S_{T_j} + d\}$. Lemma 12 further ensures that $\max_{1 \leq i, j \leq d} \{D_{T_i} + d, S_{T_i} + S_{T_j} + d\} = \max\{D_{T_1} + d, D_{T_2} + d, S_{T_1} + S_{T_2} + d\}$. Then $D_{T_2} < 2S_{T_2} \leq S_{T_1} + S_{T_2}$ implies $D_T = \max\{D_{T_1} + d, S_{T_1} + S_{T_2} + d\}$. \square

Lemma 13 (2–3 tree property) *For a tree T satisfying Lemma 12, subtrees T_2, \dots, T_d can be transformed into 2–3 trees without increasing the 2-replacement cost.*

Proof Given a tree T satisfying Lemma 12 and Fact 3, one can transform subtrees T_2, T_3, \dots, T_d into 2–3 trees T'_2, T'_3, \dots, T'_d to get a new tree T' . For $2 \leq i \leq d$, since $S_{T'_i} = OPT_1(n_i)$, one has $S_{T'_d} \leq \dots \leq S_{T'_3} \leq S_{T'_2} \leq S_{T_2}$ (Lemma 9) and $D_{T'_i} \leq 2S_{T'_i} \leq 2S_{T'_2} \leq S_{T_1} + S_{T'_2}$ ($2 \leq i \leq d$). Thus, $D_{T'} = \max\{D_{T_1} + d, D_{T'_2} + d, S_{T_1} + S_{T'_2} + d\} \leq \max\{D_{T_1} + d, S_{T_1} + S_{T_2} + d\} = D_T$, which implies that the transformation does not increase 2-replacement cost. The lemma is then proved. \square

Denote the trees satisfying Lemma 13 as *candidate trees*. By Lemma 13, an optimal tree can be found among all the candidate trees. For a candidate tree T with root degree d , the union of T_i and the edge connecting the root of T with the root of T_i is defined as branch B_i . Branch B_1 is called the *dominating branch* and other branches B_2, \dots, B_d are called *ordinary branches*. To make the discussion self-contained, [43] introduce the following two facts:

Fact 4 *Suppose that $c(u)$ is the ancestor weight of leaf u , i.e., $c(u) = \sum_{v \in ANC(u)} d_v$ where $ANC(u)$ is the set of u 's ancestor nodes and d_v is v 's degree, then the cost of replacing two leaves u_1 and u_2 is $c(u_1, u_2) = c(u_1) + c(u_2) - c(v) - d_v$ where v is the nearest common ancestor of u_1 and u_2 .*

Fact 5 *In a tree T , assume that u_1, u_2 are the two leaves which incur the 2-replacement cost ($c(u_1, u_2) = D_T$) and v is the nearest common ancestor of u_1 and u_2 , then*

1. *One of the leaves satisfies $c(u) = S_T$ (let this leaf be u_1), and therefore $D_T = c(u_1) + c(u_2) - c(v) - d_v = S_T + c(u_2) - c(v) - d_v$.*
2. *$c(u_2) - c(v) - d_v \leq S_T - 1$.*

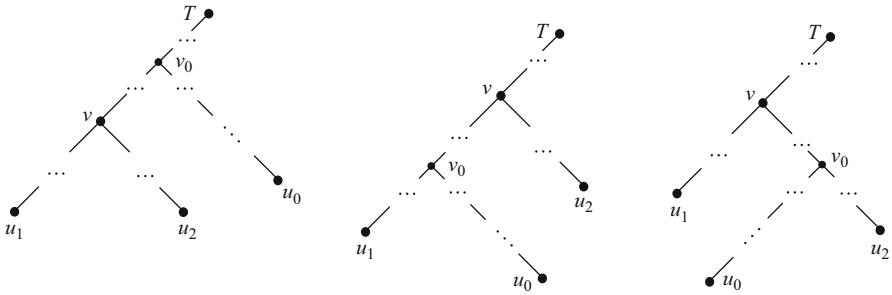


Fig. 13 The three cases of u_0 in the proof of Fact 5

Proof This fact implies that if two-leaf replacement cost in T is maximum, then at least one of the leaves has the maximum ancestor weight S_T . Suppose on the contrary that two leaves u_1, u_2 satisfy $c(u_1, u_2) = D_T$ but $c(u_1) < S_T$ and $c(u_2) < S_T$. Then, for a leaf u_0 which has $c(u_0) = S_T$, one can prove $c(u_0, u_1) > c(u_1, u_2)$ or $c(u_0, u_2) > c(u_1, u_2)$, which contradicts $c(u_1, u_2) = D_T$. Figure 13 shows the transformations for the following three cases:

Case 1: u_0, u_1 's nearest common ancestor $v_0 \in ANC(v)$.

In this case, one has $c(u_0, u_2) = c(u_0) + c(u_2) - c(v_0) - d_{v_0}$, and therefore $c(u_0, u_2) - c(u_1, u_2) = c(u_0) - c(u_1) + c(v) + d_v - c(v_0) - d_{v_0}$. Since $c(u_0) > c(u_1)$ and $c(v_0) + d_{v_0} \leq c(v)$ (v_0 is v_1 's ancestor), one has $c(u_0, u_2) > c(u_1, u_2)$; the lemma is proved in this case.

Case 2: u_0, u_1 's nearest common ancestor $v_0 \in ANC(u_1)$ but $v_0 \notin ANC(v)$.

In this case, one has $c(u_0, u_2) - c(u_1, u_2) = c(u_0) - c(u_1) > 0$.

Case 3: u_0, u_2 's nearest common ancestor $v_0 \in ANC(u_2)$ but $v_0 \notin ANC(v)$.

In this case, one has $c(u_0, u_1) - c(u_1, u_2) = c(u_0) - c(u_2) > 0$.

In all the three cases, either $c(u_0, u_1) > c(u_1, u_2)$ or $c(u_0, u_2) > c(u_1, u_2)$. Therefore, Item (1) of the fact is proved.

Item (2) is correct because $c(u_2) \leq S_T$. □

Through a series of lemmas below, [43] proved the following theorem to further remove the possibility of root degrees 2 and 3 in the optimal tree. While removing large degree $d \geq 8$ is easy, it is comparatively complicated to remove small degrees 2 or 3.

Fig. 14 Transform a tree T where the two subtrees are 2–3 trees into a 2–3 tree T'



Theorem 8 For the 2-replacement problem, a tree T with root degree 2 or 3 can be transformed into a tree with root degree 4 without increasing the 2-replacement cost.

Lemma 14 Given a tree T with root degree 2 where two subtrees T_1, T_2 are both 2–3 trees and $|n_1 - n_2| \leq 1$, when T is transformed into 2–3 tree T' with root degree 3 as shown in Fig. 14, one has $S_T \geq S_{T'}$, $D_T \geq D_{T'}$.

Proof Without loss of generality, assume that $n_1 \geq n_2$. According to the definition and optimality of 2–3 tree, when the 1-replacement cost is r , the maximum possible number of leaves on the tree is $f(\cdot)$ where

$$f(r) = \begin{cases} 3 \cdot 3^{i-1} & \text{if } r = 3i \\ 4 \cdot 3^{i-1} & \text{if } r = 3i + 1 \\ 6 \cdot 3^{i-1} & \text{if } r = 3i + 2 \end{cases}$$

Notice that T_1 and T_2 are 2–3 trees which satisfy function $f(\cdot)$. When $n_1 \in (3 \cdot 3^{i-1}, 4 \cdot 3^{i-1}]$, one has $S_{T_2} \leq S_{T_1} = 3i + 1$ and $S_T = S_{T_1} + 2 = 3i + 3$. Since $n_1 + n_2 \leq 8 \cdot 3^{i-1} < 3 \cdot 3^i$, one has $S_{T'} \leq 3i + 3 = S_T$. Similarly for $n_1 \in (4 \cdot 3^{i-1}, 6 \cdot 3^{i-1}]$, one has $S_T = S_{T_1} + 2 = 3i + 4$ and $S_{T'} \leq 3i + 4$. And for $n_1 \in (6 \cdot 3^{i-1}, 9 \cdot 3^{i-1}]$, one has $S_T = S_{T_1} + 2 = 3i + 5$ and $S_{T'} \leq 3i + 5$. Therefore, the first inequality of the lemma holds.

Suppose that the three subtrees of T' are T'_1, T'_2, T'_3 which are also 2–3 trees and satisfy $S_{T'_1} \geq S_{T'_2} \geq S_{T'_3}$. Since $S_T = S_{T_1} + 2$ and $S_{T'} = S_{T'_1} + 3$, one has $S_{T'_1} \leq S_{T_1} - 1$ and $D_{T'} \leq 2S_{T'_1} + 3 \leq 2S_{T_1} + 1$. Note that one has $S_{T_2} \geq S_{T_1} - 1$ because both subtrees are 2–3 trees with the number of leaves differed by at most 1. Hence, $D_{T'} \leq S_{T_1} + S_{T_2} + 2 \leq D_T$. The lemma is finally proved. \square

Lemma 15 For the 2-replacement problem, a tree T with root degree 2 can be transformed into a tree with root degree 3 or 4 without increasing the 2-replacement cost.

Proof Lemma 13 implies that one can transform T into a candidate tree where T_2 is a 2–3 tree. The discussion can be divided into three cases.

Fig. 15 Transform a tree T from root degree 2 into degree 4

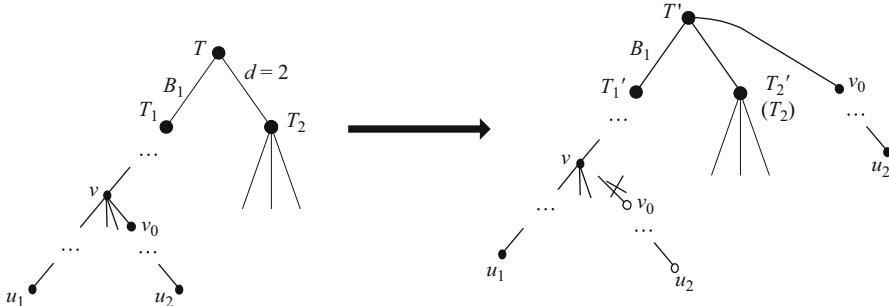
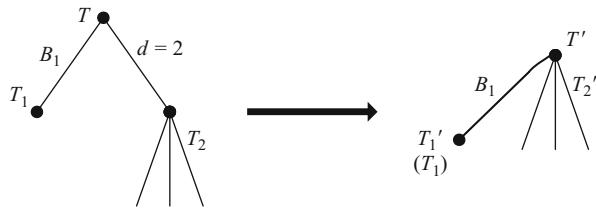


Fig. 16 Transform a tree T from root degree 2 to degree 3

Case 1: $D_{T_1} < S_{T_1} + S_{T_2}$

According to Fact 5, in subtree T_1 , one has $D_{T_1} = S_{T_1} + c(u_2) - c(v) - d_v < S_{T_1} + S_{T_2}$ which implies $c(u_2) - c(v) - d_v \leq S_{T_2} - 1$ where v is the nearest common ancestor of u_1, u_2 . Two subcases are discussed below: $c(u_2) - c(v) - d_v \leq S_{T_2} - 2$ and $c(u_2) - c(v) - d_v = S_{T_2} - 1$.

When $c(u_2) - c(v) - d_v \leq S_{T_2} - 2$, tree T can be transformed into a tree T' which is composed of the dominating branch B_1 and T'_2 (T'_2 is the same as T_2) as shown in Fig. 15. In T' , consider three subcases according to the positions of the two leaves whose replacement cost reaches $D_{T'}$. For the subcase where two leaves are both from T'_2 , one has $D_{T'} = D_{T'_2} + 1 = D_{T_2} + 1 < D_{T_2} + 2 \leq D_T$. For the subcase where one of the leaves is from T'_1 and the other is from T'_2 , one has $D_{T'} = S_{T'_1} + S_{T'_2} + 1 = S_{T_1} + S_{T_2} + 1 < S_{T_1} + S_{T_2} + 2 \leq D_T$. For the subcase where both leaves u_1 and u_2 are from T'_1 , one has $D_{T'} = D_{T'_1} + 4 = D_{T_1} + 4$. Since $c(u_2) - c(v) - d_v \leq S_{T_2} - 2$, one has $D_{T_1} \leq S_{T_1} + S_{T_2} - 2$, which implies $D_{T'} \leq S_{T_1} + S_{T_2} + 2 \leq D_T$.

When $c(u_2) - c(v) - d_v = S_{T_2} - 1$, consider another transformation for T . Notice that $S_{T_1} = c(u_1) - 2 \geq c(u_2) - c(v) = S_{T_2} - 1 + d_v \geq S_{T_2} + 1$, which means $S_{T_2} \leq S_{T_1} - 1$. Assume that T_{v_0} is the subtree of v which contains leaf u_2 , then one can move T_{v_0} to the root to produce a tree T' , as Fig. 16 shows. One also needs to consider three subcases for T' according to the positions of the two leaves whose replacement cost reaches $D_{T'}$. Firstly, if the two leaves are originally from T_1 , then one has $D_{T'} \leq D_{T_1} + 3 \leq S_{T_1} + S_{T_2} + 2 = D_T$. Secondly, if one of the two leaves is originally from T_1 while the other is not, then one can

Fig. 17 u belongs to a leaf descendant of \hat{v}

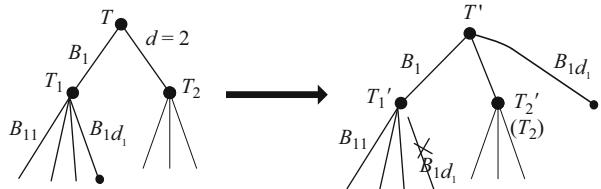
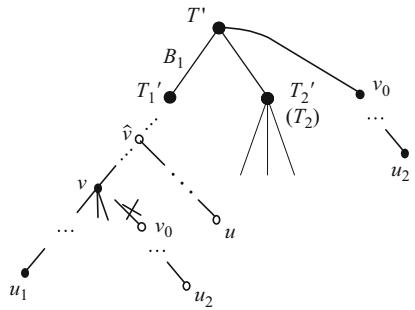


Fig. 18 Transform a tree T from root degree 2 to degree 3

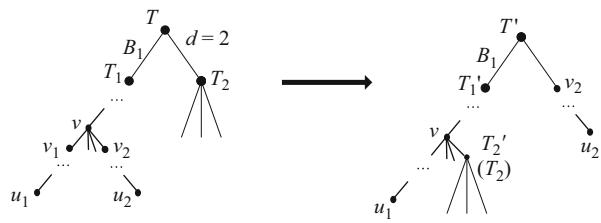
show that replacing a leaf from T'_1 or T_{v_0} incurs cost at most $S_{T_1} - 1$ which implies $D_{T'} \leq (S_{T_1} - 1) + S_{T_2} + 3 = D_T$. Suppose first the leaf u is from subtree $T_v \setminus T_{v_0}$ which is rooted at v . Since d_v decreases by 1 in the resulting tree, the cost to delete u in T'_1 is at most $S_{T_1} - 1$. If the leaf u is from T_{v_0} , then the cost to delete u in T_{v_0} is at most $S_{T_1} - d_v$. Otherwise, if the leaf is not from T_v , then suppose the leaf u is a leaf descendant of \hat{v} where \hat{v} is an ancestor of v , as shown in Fig. 17. Note that $c(u_1) + c(u) - c(\hat{v}) - d_{\hat{v}} \leq c(u_1) + c(u_2) - c(v) - d_v = c(u_1) + S_{T_2} - 1$; thus, one has $c(u) \leq S_{T_2} - 1 + c(\hat{v}) + d_{\hat{v}} \leq S_{T_2} - 1 + c(v) = c(u_2) - d_v \leq c(u_1) - d_v = S_{T_1} + 2 - d_v \leq S_{T_1}$, which implies $c(u) \leq S_{T_1}$. Therefore, replacing u from T'_1 is at most $S_{T_1} - 2$. Finally, if both leaves are originally from T_2 , then one has $D_{T'} < 2S_{T_2} + d + 1 \leq S_{T_1} - 1 + S_{T_2} + d + 1 = S_{T_1} + S_{T_2} + d \leq D_T$.

Hence, when $D_{T_1} < S_{T_1} + S_{T_2}$, one can transform T into a better tree with root degree 3 or 4.

Case 2: $D_{T_1} = S_{T_1} + S_{T_2}$

Suppose the root degree of T_1 is d_1 . One can move a branch B_{1d_1} of T_1 to the root (Fig. 18) to obtain a new tree T' . One can prove that 2-replacement cost of T' does not change compared to that of T . One also needs to consider three subcases for T' according to the positions of the two leaves whose replacement cost reaches $D_{T'}$. If the two leaves are originally from T_1 , one has $D_{T'} \leq D_{T'_1} + d + 1 = D_{T_1} - 1 + d + 1 = D_{T_1} + d \leq D_T$. If one of the leaves is originally from T_1 while the other is not, the cost $D_{T'}$ is $S_{T'_1} + S_{T_2} + d + 1 = S_{T_1} - 1 + S_{T_2} + d + 1 = S_{T_1} + S_{T_2} + d \leq D_T$ or $S_{T'_2} + S_{T_{1d_1}} + d + 1 = S_{T_2} + S_{T_{1d_1}} + d + 1 < S_{T_2} + S_{T_1} + d \leq D_T$. If both leaves are originally from T_2 , the cost $D_{T'}$ is no more than $2S_{T_2} + d + 1$. Furthermore, by Fact 5, $D_{T_1} = S_{T_1} + c(u_2) - c(v) - d_v \leq 2S_{T_1} - 1$. Since $D_{T_1} = S_{T_1} + S_{T_2}$, one

Fig. 19 Exchange two subtrees for the case that $D_{T_1} > S_{T_1} + S_{T_2}$



has $S_{T_2} \leq S_{T_1} - 1$. Therefore, $D_{T'} \leq 2S_{T_2} + d + 1 \leq S_{T_1} - 1 + S_{T_2} + d + 1 = S_{T_1} + S_{T_2} + d \leq D_T$.

Hence, $D_{T'} \leq D_T$.

Case 3: $D_{T_1} > S_{T_1} + S_{T_2}$.

In this case, replacing two leaves from T_1 incurs the cost D_T . First, transform T into a tree satisfying case 1 or case 2. In subtree T_1 , by Fact 5, if $c(u_1, u_2) = D_T$, one can assume $c(u_1) = S_T$. Suppose v is the nearest common ancestor of u_1 and u_2 and v 's children v_1 and v_2 are ancestors of u_1 and u_2 , respectively (v_i can be u_i itself), then $c(u_1) - c(v_1) \geq c(u_2) - c(v_2)$. One can exchange subtree T_{v_2} which is rooted at v_2 with subtree T_2 , as shown in Fig. 19. Because $S_{T_1} + S_{T_2} < D_{T_1} = S_{T_1} + c(u_2) - c(v_2) = S_{T_1} + S_{T_{v_2}}$, T_{v_2} has larger 1-replacement cost than T_2 . After the exchange, one has $D_{T'} \leq D_T$ because replacing one leaf from T_{v_1} and one leaf from T_{v_2} incurs a cost at most $S_{T'_1} + c(u_2) - c(v_2) + d = S_{T_1} + c(u_2) - c(v_2) + d = D_T$; replacing two leaves from T'_2 of T' incurs a cost at most $2S_{T_2} + c(v) + d_v < S_{T_2} + S_{T_{v_2}} + c(v) + d_v \leq S_{T_1} + S_{T_2} + d < D_T$, and other combinations of two replaced leaves in T' incur at most the same cost as that of T correspondingly. Note that T' has degree 2 and belongs to cases 1 or 2. Hence, T' can be further transformed into a tree with degree 3 or 4 according to cases 1 and 2.

Since in all cases, one can transform a tree with root degree 2 into a tree with root degree 3 or 4 without increasing 2-replacement cost; the lemma is finally proved. \square

Lemma 16 For 2-replacement problem, a tree T with root degree 3 can be transformed into a tree with root degree 4 without increasing the 2-replacement cost.

Proof Case 1: $D_{T_1} < S_{T_1} + S_{T_2}$.

Lemma 13 implies that one can transform T into a candidate tree where T_2, T_3 are 2–3 trees without increasing 2-replacement cost. One can then reallocate the leaves in T_2, T_3 so that T_2 is a template tree of T_3 with $0 \leq n_2 - n_3 \leq 1$ and both subtrees are still 2–3 trees as Fig. 20 shows. First, it is easy to see that this transformation will not increase 2-replacement cost. Then one can prove that such a tree can be further transformed into a tree with root degree 4 without increasing 2-replacement cost. Consider T 's two branches B_2 and B_3 (Suppose these two

Fig. 20 Transform a tree T from root degree 3 to root degree 4



branches compose a new tree \tilde{T}) in T ; according to the proof of [Lemma 14](#), one can transform \tilde{T} into tree \tilde{T}' which is a 2-3 tree and denote the whole new tree as T' . The transformation ensures that $S_{\tilde{T}} \geq S_{\tilde{T}'}, D_{\tilde{T}} \geq D_{\tilde{T}'}$. One can prove that the transformation does not increase the 2-replacement cost. One still needs to consider three subcases for T' according to the positions of the two leaves whose replacement cost reaches $D_{T'}$. If two leaves are originally from T_1 , one has $D_{T'} = D_{T'_1} + d + 1 = D_{T_1} + d + 1 \leq S_{T_1} + S_{T_2} + d = D_T$. If one of the leaves is originally from T_1 while the other is from \tilde{T} , one has $D_{T'} = S_{T_1} + S_{\tilde{T}'} + 1 \leq S_{T_1} + S_{\tilde{T}} + 1 = S_{T_1} + S_{T_2} + d = D_T$. If both leaves are from \tilde{T} , one has $D_{T'} = D_{\tilde{T}'} + 1 \leq D_{\tilde{T}} + 1 \leq D_T$. In all the subcases $D_{T'} \leq D_T$.

Case 2: $D_{T_1} = S_{T_1} + S_{T_2}$

The transformation and proof for this case is similar to the case 2 in the proof of [Lemma 15](#). The only difference is that the original tree has root degree $d = 3$ instead of 2 in this case.

Case 3: $D_{T_1} > S_{T_1} + S_{T_2}$

The transformation and proof for this case is similar to the case 3 in the proof of [Lemma 15](#). The only difference is that the original tree has root degree $d = 3$ instead of 2 in this case.

Since, in all cases, the tree with root degree 3 is transformed into a tree with root degree 4 without increasing 2-replacement cost, the lemma is finally proved. \square

3.2.3 Linear Time Algorithm: Balanced Structure of 2-Replacement Problem

Note that the optimal structure can be computed in $O(n^8)$ time by enumerating all possible degrees that are at most 7 by the result of [Sect. 3.2.2](#). Although the possibility of the root degrees 2 and 3 has been removed in [Sect. 3.2.2](#) and the structure of the ordinary branches has been fixed, to exactly compute the optimal structure is still not efficient enough because one needs to enumerate all the possible structures of the dominating branch. In this subsection, [43] showed that among the candidate trees with n leaves, a balanced structure can achieve 2-replacement cost $OPT_2(n)$. This reduces the complexity of the algorithm substantially to $O(n)$.

The basic idea is to first investigate the *capacity* $g(R)$ defined below for candidate trees with 2-replacement cost R ([Theorem 9](#)). Note that the optimal tree

has the minimum 2-replacement cost with n leaves, which reversely implies that if one wants to find a tree with 2-replacement cost R and at the same time has the maximum possible number of leaves, then computing the optimal tree for increasing n until $OPT_2(n) > R$ will produce one such solution. Wu et al. [43] then analyze and calculate the exact value for the capacity (maximum number of leaves) given a fixed 2-replacement cost R ([Theorem 10](#)). Finally, they prove that certain balanced structure can always be the optimal structure that minimizes the 2-replacement cost ([Theorem 11](#)).

Definition 6 Define capacity to be the maximum number of leaves that can be placed in a certain type of trees given a fixed replacement cost. According to [32], function $f(r)$ defined below is the capacity for 1-replacement cost r (among all the possible trees). Wu et al. [43] uses function $g(R)$ to denote the capacity for 2-replacement cost R (among all the possible trees). In other words, when $g(R - 1) < n \leq g(R)$, one has $OPT_2(n) = R$:

$$f(r) = \begin{cases} 3 \cdot 3^{i-1} & \text{if } r = 3i \\ 4 \cdot 3^{i-1} & \text{if } r = 3i + 1 \\ 6 \cdot 3^{i-1} & \text{if } r = 3i + 2 \end{cases}$$

To facilitate the discussion, according to [Fact 3](#), one can divide the candidate trees with 2-replacement cost R and root degree d into two categories as summarized in the following definition.

Definition 7 Candidate trees of category 1: The two leaves whose replacement cost achieves 2-replacement cost are from different branches, i.e., $D_T = S_{T_1} + S_{T_2} + d$, which implies $S_{T_1} + S_{T_2} \geq D_{T_1}$.

Candidate trees of category 2: The two leaves whose replacement cost achieves 2-replacement cost are both from the dominating branch B_1 , i.e., $D_T = D_{T_1} + d$, which implies $S_{T_1} + S_{T_2} < D_{T_1}$.

Correspondingly, denote the capacity of the candidate trees belonging to category 1 with 2-replacement cost R by $g_1(R)$, and denote the capacity of the candidate trees belonging to category 2 with 2-replacement cost R by $g_2(R)$. Note that one can find the optimal tree among the candidate trees according to [Lemma 13](#), which implies that with the same 2-replacement cost R , the best candidate tree can always have equal or larger number of leaves than the general trees. That is, $g(R) = \max\{g_1(R), g_2(R)\}$. Thus, in the following discussions, only the candidate trees will be discussed. On the other hand, because trees with the maximum number of leaves are the targets, it is easy to see that one can assume the numbers of leaves in ordinary branches are all the same (Otherwise, the tree can be bigger without affecting the 2-replacement cost).

In all candidate trees with 2-replacement cost R , by [Fact 3](#), at most one of the two leaves whose replacement cost achieves 2-replacement cost is from an

ordinary branch. Suppose each ordinary branch has 1-replacement cost r^- , and correspondingly T_1 has 1-replacement cost r^+ where $r^+ \leq R - d - r^-$ (otherwise, $D_T \geq r^+ + r^- + d > R$, a contradiction). For fixed cost R , [Lemma 12](#) (monotonous property) implies that $r^+ \geq r^-$. The following capacity bound can be proved.

Theorem 9 $g_i(R) \leq (R - 2r^-) \cdot f(r^-)$ ($i = 1, 2$).

To prove this theorem, it is sufficient to prove $g_1(R) \leq (R - 2r^-) \cdot f(r^-)$ and $g_2(R) \leq g_1(R)$. Wu et al. [43] used critical node defined below to denote a special kind of nodes in the following proofs.

Definition 8 For a selected node u , define the path from u to the root to be a critical path. Correspondingly, u 's ancestor nodes on the path are named critical nodes.

First, investigate the capacity of the candidate tree T in category 1 which has 2-replacement cost R . In [Lemmas 17–20](#), assume that the number of leaves in T is the maximum possible and the goal is to fix the $r_{\max}(\cdot)$ ([Definition 9](#)) of some critical nodes and their siblings to prove the capacity bound $g_1(R)$.

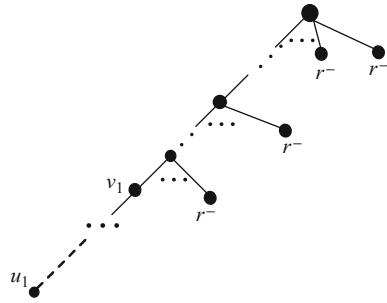
Definition 9 For the subtree T_v rooted at node v , the maximum ancestor weight with respect to T_v is denoted as $r_{\max}(v)$. Correspondingly, the maximum number of leaves T_v can hold given the 1-replacement cost $r_{\max}(v)$ is denoted as $\text{cap}(v)$.

Given a candidate tree in category 1 where $D_T = S_{T_1} + S_{T_2} + d$, there exists at least one leaf $u \in B_1$ satisfying $c(u) = r^+ + d = R - r^-$. Choose such a leaf u to obtain a critical path; correspondingly u 's ancestors are critical nodes. Notice that critical node v has $r_{\max}(v) = R - r^- - c(v)$. Furthermore, without loss of generality, suppose that v_1 is the nearest ancestor of u which satisfies $r_{\max}(v_1) \geq r^-$ (which implies $r_{\max}(v_0) < r^-$ where critical node v_0 is v_1 's child). The following discussions investigate the capacity for the critical node and their siblings.

Lemma 17 *If node v is a critical node, then $r_{\max}(v') = \min\{r^-, r_{\max}(v)\}$ for any sibling v' of v .*

Proof By the definition of r^+ , one has $r_{\max}(v') + c(v') - d \leq r^+ = r_{\max}(v) + c(v) - d$, which implies $r_{\max}(v') \leq r_{\max}(v)$. On the other hand, if one chooses to delete a leaf u in T_v and a leaf w in $T_{v'}$, the replacement cost should be at most R , which means $r_{\max}(v) + r_{\max}(v') + c(v) \leq R$. Note that $r_{\max}(v) = R - r^- - c(v)$. Therefore, $r_{\max}(v') \leq r^-$. The possibility $r_{\max}(v') < \min\{r^-, r_{\max}(v)\}$ can be removed because otherwise the capacity of $T_{v'}$ is not maximum which contradicts the assumption that the number of leaves in T is the maximum possible. In other words, if $r_{\max}(v') < \min\{r^-, r_{\max}(v)\}$, one can enlarge the tree by increasing $r_{\max}(v')$ to $\min\{r^-, r_{\max}(v)\}$ while ensuring that the tree still belongs to candidate trees in category 1 and has 2-replacement cost R . This ends the proof. \square

Fig. 21 $spine(v_1, root)$ that belongs to candidate trees in category 1



Lemma 18 Given a critical node v which is on the path from v_1 to the root, if v' is a sibling of v , then $r_{\max}(v') = r^-$.

Proof Firstly, $r_{\max}(v) = R - r^- - c(v)$ for the critical node v . Furthermore, since v is v_1 's ancestor or v_1 itself, one has $r_{\max}(v) \geq r_{\max}(v_1) \geq r^-$. Hence, the sibling of v has $r_{\max}(v') = \min\{r^-, r_{\max}(v)\} = r^-$ ([Lemma 17](#)). The lemma is proved. \square

Consider the structure composed by the critical nodes on the path from v_1 to the root and the siblings of these critical nodes, as shown in [Fig. 21](#). This structure is denoted as $spine(v_1, root)$ according to the definition below. Suppose that w is a leaf in this structure. The value of $r_{\max}(w)$ is r^- by [Lemma 18](#).

Definition 10 Suppose that v_a and v_b (v_a is a descendent of v_b) are critical nodes on the selected critical path. Denote the structure composed by the critical nodes from v_a to v_b and their siblings as $spine(v_a, v_b)$ (v_b 's siblings are not included).

Consider two cases of $spine(v_1, root)$ for candidate trees in category 1. One can prove $g_1(R) \leq (R - 2r^-) \cdot f(r^-)$ in the following cases:

Case 1: $r_{\max}(v_1) = r^-$.

Case 2: $r_{\max}(v_1) > r^-$.

Lemma 19 $g_1(R) \leq (R - 2r^-) \cdot f(r^-)$ when $r_{\max}(v_1) = r^-$.

Proof In this case, besides node v_1 , any other leaf w of $spine(v_1, root)$ also has $r_{\max}(w) = r^-$ according to [Lemma 18](#).

On the other hand, the total degrees from the parent of v_1 to the root are $c(v_1) = R - r^- - r_{\max}(v_1) = R - 2r^-$, which equals the number of leaves on $spine(v_1, root)$. Therefore, the capacity is bounded by $(R - 2r^-) \cdot f(r^-)$. \square

Lemma 20 $g_1(R) \leq (R - 2r^-) \cdot f(r^-)$ when $r_{\max}(v_1) > r^-$.

Proof Theorem 7 implies that $d_{v_1} \leq 7$; thus, $r_{\max}(v_1) = r_{\max}(v_0) + d_{v_1} \leq r^- + 6$ (because $r_{\max}(v_0) < r^-$).

Similar with the proof in the previous lemma, for any leaf v except v_1 on $\text{spine}(v_1, \text{root})$, one has $\text{cap}(v) \leq f(r^-)$, which results in at most $(R - r^- - r_{\max}(v_1) - 1) \cdot f(r^-)$ leaves. To prove the capacity bound, one only needs to show that $\text{cap}(v_1) \leq (r_{\max}(v_1) + 1 - r^-) \cdot f(r^-)$. If $r_{\max}(v_1)$ is written as $r^- + m$ where $m \leq 6$, then it is equivalent to proving $(m + 1) \cdot f(r^-) - f(r^- + m) \geq 0$. When $1 \leq m \leq 5$, one can verify that $f(r^- + m) < (m + 1)f(r^-)$. When $m = 6$, the above relation does not hold because $f(r^- + 6) = 9f(r^-) > 7f(r^-)$. But when $r_{\max}(v_1) = r^- + m = r^- + 6$, one has $d_{v_1} = 7$. Lemma 17 implies that the sibling u of v_0 has $r_{\max}(u) = \min\{r^-, r_{\max}(v_0)\} = r_{\max}(v_0) = r^- + 6 - 7 = r^- - 1$. Hence, $\text{cap}(v_1) \leq 7f(r^- - 1) \leq 7f(r^-)$. This completes the proof. \square

By Lemmas 19 and 20, the capacity bound of candidate trees in category 1 is proved, i.e., $g_1(R) \leq (R - 2r^-) \cdot f(r^-)$. The following part is to show that candidate trees in category 2 have a capacity no greater than that of category 1.

Note that the following fact holds for candidate trees T in category 2.

Fact 6 Given a candidate tree T in category 2, suppose that u_1, u_2 are the two leaves (from subtree T_1) whose replacement cost equals R , i.e., $c(u_1) + c(u_2) - c(v) - d_v = R$ where v is the nearest common ancestor of u_1, u_2 and $c(u_1) = S_T$. In order to find an optimal tree, among candidate trees in category 2, one only needs to search among those trees satisfying $c(u_2) - c(v) - d_v < OPT_1(n_1)$ where n_1 is the number of leaves in T_1 .

Proof If on the contrary in the optimal tree, one has $c(u_2) - c(v) - d_v \geq OPT_1(n_1)$, then $D_T = c(u_1) + c(u_2) - c(v) - d_v \geq 2OPT_1(n_1) + d$ since $c(u_1) = S_{T_1} + d \geq OPT_1(n_1) + d$. In this case, if T is transformed into T' by replacing subtree T_1 with a 2–3 tree with the same number of leaves, one has $D_{T'} \leq D_T$ because $S_{T'_1} = OPT_1(n_1) \leq S_{T_1}$ and $D_{T'_1} + d < 2OPT_1(n_1) + d \leq c(u_1) + c(u_2) - c(v) - d_v = R$. Notice that T' is either a candidate tree in category 1 or a candidate tree in category 2 satisfying $c(u_2) - c(v) - d_v < OPT_1(n_1)$. Thus, the fact follows. \square

Lemma 21 A candidate tree in category 2 has capacity $g_2(R) \leq g_1(R)$.

Proof Given a candidate tree T in category 2, one has $S_{T_1} + S_{T_2} + d < D_{T_1} + d = R = D_T$. By the monotone property in Lemma 12, the d subtrees satisfy $S_{T_1} \geq S_{T_2} \geq \dots \geq S_{T_d}$. If $S_{T_1} = S_{T_2}$, then $D_{T_1} + d \leq 2S_{T_1} + d = S_{T_1} + S_{T_2} + d$, which leads to a contradiction. Thus, $S_{T_1} \geq S_{T_2} + 1$. Note that the ordinary branches are 2–3 trees; one can do a tree expansion by adding one leaf to each of the ordinary branches (where each resulting subtree is still a 2–3 tree). In this way, one can increase the number of leaves in the tree without affecting the 2-replacement cost.

The resulting subtree's 1-replacement cost will increase by at most 1. Suppose that two leaves in the resulting tree T' whose replacement cost achieves the 2-replacement cost are u_1 and u_2 . We discuss three cases. If u_1, u_2 are both from T_1 ,

then replacing u_1, u_2 incurs a cost $D_{T_1} + d = R$. If one of the two leaves is from T_1 while the other is from T_i , then replacing u_1, u_2 incurs a cost at most $S_{T_1} + S_{T_2} + 1 + d \leq R$. If u_1, u_2 are both from the ordinary branches, then replacing u_1, u_2 incurs a cost at most $2(S_{T_2} + 1) + d \leq S_{T_2} + 1 + S_{T_1} + d \leq R$.

The above tree expansion can be applied until $S_{T_1} + S_{T_i} + d = R$ (The tree becomes a candidate tree of category 1), and the final tree is denoted as T'' . One can prove that T'' is a candidate tree of category 1. To show this, one only needs to prove that in T'' subtree T''_2 has the number of leaves less than T_1 .

This follows because $OPT_1(n''_2) = S_{T''_2} = R - S_{T_1} - d = c(u_2) - c(v) - d_v < OPT(n_1)$ implies $n''_2 < n_1$.

Therefore, with 2-replacement cost R fixed, there is always a candidate tree of category 1 which has more leaves than any candidate tree of category 2. This finishes the proof of $g_2(R) \leq g_1(R)$. \square

Lemma 21 implies that the maximum capacity can always be achieved by the candidate trees in category 1. Thus, the capacity bound is $(R - 2r^-) \cdot f(r^-)$ and **Theorem 9** is then proved.

In the following, among these candidate trees one can study the optimal structure which achieves the maximum capacity for different values of 2-replacement cost R . Only the proof for $R = 6i + 6$ will be elaborated due to the similarity of other proofs. The correctness of other proofs can be referred to in [Table 1](#). The following fact is used in the proofs.

Fact 7

$$(s + 2t) \cdot f(r - t) \leq s \cdot f(r) \text{ when } \begin{cases} r = 3i, t \geq 1, s \geq 4 \\ r = 3i + 1, t \geq 2, s \geq 4 \\ r = 3i + 1, t \geq 1, s \geq 6 \\ r = 3i + 2, t \geq 1, s \geq 4 \end{cases}$$

Proof One only needs to prove $\frac{f(r-t)}{f(r)} \leq \frac{s}{s+2t}$. Since $\frac{s}{s+2t}$ is a monotonously increasing relative to s , thus when $s \geq 4$ it is sufficient to prove $\frac{f(r-t)}{f(r)} \leq \frac{4}{4+2t}$. By the definition of function $f(\cdot)$, one needs to discuss about three cases.

Case 1: $r = 3i$.

$$\frac{f(3i-t)}{f(3i)} = \begin{cases} \frac{f(3i-1-3j)}{f(3i)} = \frac{6 \cdot 3^{i-2}/3^j}{3^i} = \frac{2}{3} \cdot \frac{1}{3^j} \leq \frac{4}{4+2(3j+1)} & \text{if } t = 3j + 1 \\ \frac{f(3i-2-3j)}{f(3i)} = \frac{4 \cdot 3^{i-2}/3^j}{3^i} = \frac{4}{9} \cdot \frac{1}{3^j} < \frac{4}{4+2(3j+2)} & \text{if } t = 3j + 2 \\ \frac{f(3i-3-3j)}{f(3i)} = \frac{3 \cdot 3^{i-2}/3^j}{3^i} = \frac{1}{3} \cdot \frac{1}{3^j} < \frac{4}{4+2(3j+3)} & \text{if } t = 3j + 3 \end{cases}$$

Thus, in this case $(s + 2t) \cdot f(r - t) \leq s \cdot f(r)$ when $s \geq 4$ and $t \geq 1$.

Table 1 Capacity bound for different values of R

R	$d = 7$	$d = 6$	$d = 5$	$d = 4$
6i + 6	$(8 + 2t)f(3i - 1 - t) \leq 8 \cdot f(3i - 1) = 16 \cdot 3^{i-1}$	$(6 + 2t)f(3i - t) \leq 6 \cdot f(3i) = 18 \cdot 3^{i-1}$	$(6 + 2t)f(3i - t) \leq 6 \cdot f(3i) = 18 \cdot 3^{i-1}$	$(4+2t)f(3i+1-t) \leq 4 \cdot f(3i+1) = 16 \cdot 3^{i-1}$
6i + 5	$(7 + 2t)f(3i - 1 - t) \leq 7 \cdot f(3i - 1) = 14 \cdot 3^{i-1}$	$(7 + 2t)f(3i - 1 - t) \leq 7 \cdot f(3i - 1) = 14 \cdot 3^{i-1}$	$(5 + 2t)f(3i - t) \leq 5 \cdot f(3i) = 15 \cdot 3^{i-1}$	$(5 + 2t)f(3i - t) \leq 5 \cdot f(3i) = 15 \cdot 3^{i-1}$
6i + 4	$(8 + 2t)f(3i - 2 - t) \leq 8 \cdot f(3i - 2) = \frac{32}{3} \cdot 3^{i-1}$	$(6 + 2t)f(3i - 1 - t) \leq 6 \cdot f(3i - 1) = 12 \cdot 3^{i-1}$	$(6 + 2t)f(3i - 1 - t) \leq 6 \cdot f(3i - 1) = 12 \cdot 3^{i-1}$	$(4 + 2t)f(3i - t) \leq 4 \cdot f(3i) = 12 \cdot 3^{i-1}$
6i + 3	$(7 + 2t)f(3i - 2 - t) \leq 7 \cdot f(3i - 2) = \frac{28}{3} \cdot 3^{i-1}$	$(7 + 2t)f(3i - 2 - t) \leq 7 \cdot f(3i - 2) = \frac{28}{3} \cdot 3^{i-1}$	$(5 + 2t)f(3i - 1 - t) \leq 5 \cdot f(3i - 1) = 10 \cdot 3^{i-1}$	$(5 + 2t)f(3i - 1 - t) \leq 5 \cdot f(3i - 1) = 10 \cdot 3^{i-1}$
6i + 2	$(8 + 2t)f(3i - 3 - t) \leq 8 \cdot f(3i - 3) = 8 \cdot 3^{i-1}$	$(6 + 2t)f(3i - 2 - t) \leq 6 \cdot f(3i - 2) = 8 \cdot 3^{i-1}$	$(6 + 2t)f(3i - 2 - t) \leq 6 \cdot f(3i - 2) = 8 \cdot 3^{i-1}$	$(4 + 2t)f(3i - 1 - t) \leq 4 \cdot f(3i - 1) = 8 \cdot 3^{i-1}$
6i + 1	$(7 + 2t)f(3i - 3 - t) \leq 7 \cdot f(3i - 3) = 7 \cdot 3^{i-1}$	$(7 + 2t)f(3i - 3 - t) \leq 7 \cdot f(3i - 3) = 7 \cdot 3^{i-1}$	$(5+2t)f(3i-2-t) \leq 5 \cdot f(3i-2) = \frac{20}{3} \cdot 3^{i-1}$	$(5+2t)f(3i-2-t) \leq 5 \cdot f(3i-2) = \frac{20}{3} \cdot 3^{i-1}$

Case 2: $r = 3i + 2$.

$$\frac{f(3i + 2 - t)}{f(3i + 2)} = \begin{cases} \frac{2}{3} \cdot \frac{1}{3^j} \leq \frac{4}{4+2(3j+1)} & \text{if } t = 3j + 1 \\ \frac{1}{2} \cdot \frac{1}{3^j} \leq \frac{4}{4+2(3j+2)} & \text{if } t = 3j + 2 \\ \frac{1}{3} \cdot \frac{1}{3^j} < \frac{4}{4+2(3j+3)} & \text{if } t = 3j + 3 \end{cases}$$

Hence, $(s + 2t) \cdot f(r - t) \leq s \cdot f(r)$ when $s \geq 4$ and $t \geq 1$.

Case 3: $r = 3i + 1$.

$$\frac{f(3i + 1 - t)}{f(3i + 1)} = \begin{cases} \frac{3}{4} \cdot \frac{1}{3^j} < \frac{4}{4+2(3j+1)} & \text{if } t = 3j + 1, t \neq 1 \\ \frac{1}{2} \cdot \frac{1}{3^j} \leq \frac{4}{4+2(3j+2)} & \text{if } t = 3j + 2 \\ \frac{1}{3} \cdot \frac{1}{3^j} < \frac{4}{4+2(3j+3)} & \text{if } t = 3j + 3 \end{cases}$$

In this case, when $s \geq 4, t = 1$, the inequality $(s + 2t) \cdot f(r - t) \leq s \cdot f(r)$ fails. However, one still has $(s + 2t) \cdot f(r - t) \leq s \cdot f(r)$ when $s \geq 4$ and $t \geq 2$. For similar reasons, because

$$\frac{f(3i+1-t)}{f(3i+1)} = \begin{cases} \frac{3}{4} \cdot \frac{1}{3^j} \leq \frac{6}{6+2(3j+1)} & \text{if } t = 3j+1 \\ \frac{1}{2} \cdot \frac{1}{3^j} < \frac{6}{6+2(3j+2)} & \text{if } t = 3j+2 \\ \frac{1}{3} \cdot \frac{1}{3^j} < \frac{6}{6+2(3j+3)} & \text{if } t = 3j+3 \end{cases}$$

one has $(s + 2t) \cdot f(r - t) \leq s \cdot f(r)$ when $s \geq 6$ and $t \geq 1$. \square

Lemma 22 *When $R = 6i + 6$, the maximum capacity is $18 \cdot 3^{i-1}$.*

Proof Only root degree $4 \leq d \leq 7$ needs to be considered according to the root degree bound derived in Sect. 3.2.2. In the following cases, the capacity of candidate trees with the same root degree is firstly compared. One can prove that the tree at certain balance point ($r^+ - r^- \leq 1$) has larger capacity in each case.

Case 1: Candidate trees with root degree $d = 7$.

All these candidate trees have $r^+ \geq r^-$ and $r^+ + r^- + d = 6i + 6$; thus, $1 \leq r^- \leq 3i - 1$. Given such a candidate tree, when writing r^- as $3i - 1 - t$ where $0 \leq t \leq 3i - 1$, one has capacity bound $(R - 2(3i - 1 - t)) \cdot f(3i - 1 - t) = (8 + 2t) \cdot f(3i - 1 - t)$ by Theorem 9. One can prove that this capacity is maximum when $t = 0$. Hence, one only needs to prove $(8 + 2t) \cdot f(3i - 1 - t) \leq 8f(3i - 1)$. Let $r = 3i - 1, s = 8$. Obviously for any $t \geq 1$, Fact 7 implies $(8 + 2t) \cdot f(3i - 1 - t) \leq 8f(3i - 1) = 16 \cdot 3^{i-1}$.

Case 2: Candidate trees with root degree $d = 6$.

All these candidate trees have $r^+ \geq r^-$ and $r^+ + r^- + d = 6i + 6$; thus, $1 \leq r^- \leq 3i$. Given such a candidate tree, when writing r^- as $3i - t$ where $0 \leq t \leq 3i$, one has capacity bound $(R - 2(3i - t)) \cdot f(3i - t) = (d + 2t) \cdot f(3i - t)$. One can prove that this capacity is maximum when $t = 0$. That is, one needs to prove $(6 + 2t) \cdot f(3i - t) \leq 6 \cdot f(3i)$ when $t \geq 1$. Let $r = 3i, s = 6$. Fact 7 implies $(6 + 2t) \cdot f(3i - t) \leq 6 \cdot f(3i)$. Hence, the balance point $t = 0, r^- = 3i, r^+ = 3i$ achieves maximum capacity in this case.

Case 3: Candidate trees with root degree $d = 5$.

In this case $1 \leq r^- \leq 3i$. Write r^- as $3i - t$. Similarly, let $r = 3i, s = 6$. Fact 7 implies $(6 + 2t) \cdot f(3i - t) \leq 6f(3i)$, which further implies that the capacity is maximized to be $18 \cdot 3^{i-1}$ when $t = 0$.

Case 4: Candidate trees with root degree $d = 4$.

In this case $1 \leq r^- \leq 3i + 1$. Let $r = 3i + 1, s = 4$. When $t \geq 2$, Fact 7 implies $(4 + 2t) \cdot f(3i + 1 - t) \leq 4 \cdot f(3i + 1)$. Thus, the capacity of candidate tree which has $r^- = 3i + 1 - t$ and $t \geq 2$ is at most $4f(3i + 1) = 16 \cdot 3^{i-1}$. While for $t = 1$, the inequality fails as $(4 + 2) \cdot f(3i) = 18 \cdot 3^{i-1} > 4 \cdot f(3i + 1) = 16 \cdot 3^{i-1}$. However, in this special case, the capacity $6 \cdot f(3i)$ is equal to that of case 2. Finally, note that when $t = 0$ the capacity bound $4f(3i + 1) = 16 \cdot 3^{i-1}$ is less than that in case 2.

Since in all the four cases the maximum capacity is $18 \cdot 3^{i-1}$, the lemma is proved. \square

Considering other values of R , the following theorem can be proved.

Theorem 10 *For 2-replacement cost R , the maximum capacity is*

$$g(R) = \begin{cases} 6 \cdot 3^{i-1} & \text{if } R = 6i \\ 7 \cdot 3^{i-1} & \text{if } R = 6i + 1 \\ 8 \cdot 3^{i-1} & \text{if } R = 6i + 2 \\ 10 \cdot 3^{i-1} & \text{if } R = 6i + 3 \\ 12 \cdot 3^{i-1} & \text{if } R = 6i + 4 \\ 15 \cdot 3^{i-1} & \text{if } R = 6i + 5 \end{cases}$$

Proof Due to the similarity of the proofs for different values R , [43] put the key inequality and capacity bound in Table 1 for $R = 6i + 1, 6i + 2, 6i + 3, 6i + 4, 6i + 5$. Notice that in all cases, when $t \geq 2$ the inequality holds. In the special case that $t = 1$, when $R = 6i + 6, d = 4$, $R = 6i + 1, d = 5$, or $R = 6i + 1, d = 4$, the inequality fails. The analysis of the case where $R = 6i + 1, d = 5$ or $R = 6i + 1, d = 4$ is similar to that of the case $R = 6i + 6, d = 4$ as shown in the proof of Lemma 22. For example, when $R = 6i + 1, d = 5$, and $t = 1$, the capacity bound $(6i + 1 - 2(3i - 2 - t)) = 7f(3i - 3)$ is the capacity that can be achieved in the case $R = 6i + 1, d = 7$. \square

After obtaining the capacity for the 2-replacement cost R , one can finally prove that among the candidate trees with n leaves, the optimal cost can be achieved by some balanced structure as shown below.

Definition 11 We use the balanced tree to denote a tree with root degree d where each subtree is 2–3 tree and has number of leaves differed by at most 1.

Theorem 11 *Among trees with n leaves,*

1. *When $n \in (15 \cdot 3^{i-1}, 18 \cdot 3^{i-1}]$, the optimal tree is a balanced tree which has root degree 6 and 2-replacement cost $6i+6$.*

2. When $n \in (12 \cdot 3^{i-1}, 15 \cdot 3^{i-1}]$, the optimal tree is a balanced tree which has root degree 5 and 2-replacement cost $6i+5$.
3. When $n \in (10 \cdot 3^{i-1}, 12 \cdot 3^{i-1}]$, the optimal tree is a balanced tree which has root degree 6 and 2-replacement cost $6i+4$.
4. When $n \in (8 \cdot 3^{i-1}, 10 \cdot 3^{i-1}]$, the optimal tree is a balanced tree which has root degree 5 and 2-replacement cost $6i+3$.
5. When $n \in (7 \cdot 3^{i-1}, 8 \cdot 3^{i-1}]$, the optimal tree is a balanced tree which has root degree 6 and 2-replacement cost $6i+2$.
6. When $n \in (6 \cdot 3^{i-1}, 7 \cdot 3^{i-1}]$, the optimal tree is a balanced tree which has root degree 7 and 2-replacement cost $6i+1$.

Proof When $n \in (15 \cdot 3^{i-1}, 18 \cdot 3^{i-1}]$, one has $OPT_2(n) = 6i + 6$. One can prove that the balanced tree with root degree 6 can always achieve this optimal cost. In the balanced tree, each subtree T_j ($1 \leq j \leq 6$) has the number of leaves $n_j = \lceil \frac{n-j+1}{6} \rceil \in [\lfloor \frac{5}{2} \cdot 3^{i-1} \rfloor, 3 \cdot 3^{i-1}]$. By function $f(\cdot)$, one has $S_{T_i} \leq 3i$. Thus, any two leaves from the tree will incur a replacement cost at most $2 \cdot 3i + 6 = 6i + 6$.

When $n \in (12 \cdot 3^{i-1}, 15 \cdot 3^{i-1}]$, one has $OPT_2(n) = 6i + 5$. Then one can prove that the balanced tree with root degree 5 can always achieve the optimal cost. In the balanced tree, each subtree T_j ($1 \leq j \leq 5$) has the number of leaves $n_j = \lceil \frac{n-j+1}{5} \rceil \in [\lfloor \frac{12}{5} \cdot 3^{i-1} \rfloor, 3 \cdot 3^{i-1}]$. By function $f(\cdot)$, one has $S_{T_i} \leq 3i$. Thus, any two leaves from the tree will incur a replacement cost at most $2 \cdot 3i + 5 = 6i + 5$.

When $n \in (10 \cdot 3^{i-1}, 12 \cdot 3^{i-1}]$, one has $OPT_2(n) = 6i + 4$ and $n_j = \lceil \frac{n-j+1}{6} \rceil \in [\lfloor \frac{5}{3} \cdot 3^{i-1} \rfloor, 2 \cdot 3^{i-1}]$. By function $f(\cdot)$, one has $S_{T_i} \leq 3i - 1$. Thus, any two leaves from the tree will incur a replacement cost at most $2 \cdot (3i - 1) + 6 = 6i + 4$.

When $n \in (8 \cdot 3^{i-1}, 10 \cdot 3^{i-1}]$, one has $OPT_2(n) = 6i + 3$ and $n_j = \lceil \frac{n-j+1}{5} \rceil \in [\lfloor \frac{8}{5} \cdot 3^{i-1} \rfloor, 2 \cdot 3^{i-1}]$. By function $f(\cdot)$, one has $S_{T_i} \leq 3i - 1$. Thus, any two leaves from the tree will incur a replacement cost at most $2 \cdot (3i - 1) + 5 = 6i + 3$.

When $n \in (7 \cdot 3^{i-1}, 8 \cdot 3^{i-1}]$, one has $OPT_2(n) = 6i + 2$ and $n_j = \lceil \frac{n-j+1}{6} \rceil \in [\lfloor \frac{7}{6} \cdot 3^{i-1} \rfloor, \frac{4}{3} \cdot 3^{i-1}]$. By function $f(\cdot)$, one has $S_{T_i} \leq 3i - 2$. Thus, any two leaves from the tree will incur a replacement cost at most $2 \cdot (3i - 2) + 6 = 6i + 2$.

When $n \in (6 \cdot 3^{i-1}, 7 \cdot 3^{i-1}]$, one has $OPT_2(n) = 6i + 1$. The balanced tree with root degree 7 where $n_j = \lceil \frac{n-j+1}{7} \rceil \in [\lfloor \frac{6}{7} \cdot 3^{i-1} \rfloor, 3^{i-1}]$ can always achieve the optimal cost. By function $f(\cdot)$, one has $S_{T_i} \leq 3i - 3$. Thus, any two leaves from the tree will incur a replacement cost at most $2 \cdot (3i - 3) + 7 = 6i + 1$. \square

Finally, the structure of the dominating branch is fixed and the optimal tree structure for the 2-replacement problem is obtained. Wu et al. [43] conjecture the general result for the k -replacement problem as follows.

The general form of capacity $G_k(R)$ which denotes the maximum number of leaves that can be placed in a tree given the k -replacement cost R in the k -replacement problem is conjectured to be of the form shown in Eq.(1). They believe such a capacity function is generated by some balanced structure of the trees. Furthermore, if the conjecture is proved to be correct, it is also possible to obtain

the optimal structure in a similar way as in the proof of [Theorem 11](#) and construct the optimal tree in $O(n)$ time:

$$G_k(R) = \begin{cases} (3k + \alpha) \cdot 3^{i-1} & \text{if } R = 3k \cdot i + \alpha, \alpha \in [0, k), \\ (4k + 2(\alpha - k)) \cdot 3^{i-1} & \text{if } R = 3k \cdot i + \alpha, \alpha \in [k, 2k), \\ (6k + 3(\alpha - 2k)) \cdot 3^{i-1} & \text{if } R = 3k \cdot i + \alpha, \alpha \in [2k, 3k). \end{cases} \quad (1)$$

4 Other Variants

4.1 Minimize sequential update cost for a sequence of actions

When the key tree carries out sequential update instead of the batch update, [\[10\]](#) and [\[44\]](#) studied the structure of the optimal trees together with algorithms for computing them. When the user change is n consecutive deletions, [\[10\]](#) prove that there is an optimal tree in which each internal node has at most five children and each internal node with at least one non-leaf child has exactly three children. Based on these characterizations, they present a dynamic programming algorithm that computes an optimal key tree in $O(n^2)$ time. Later in [\[44\]](#), the authors prove a semi-balance property for the optimal tree and use it to reduce the running time to $O(\log n)$ multiplications of $O(\log n)$ —bit integers. Then they study the optimal tree structure when insertion cost is also considered. They show that the optimal tree is such a tree where any internal node has degree at most 7 and children of nodes with degree not equal to 2 or 3 are all leaves. Based on this result, they give a dynamic programming algorithm with $O(n^2)$ time to compute the optimal tree.

4.2 Minimize Cost for the Next Update

Selcuk et al. [\[31\]](#) studied the key tree optimization when the objective is to minimize the expected cost of the next rekey operation (caused by a leave or compromise event). They define the expected cost as $\sum(p_i \cdot d_i)$ where p_i is the probability that member i will be the next to be evicted/compromised and d_i is its depth in the tree. They pointed out that although this objective is the same as data compression trees, there is a major difference. In the key tree, changing a member's location will cause extra cost, while in the data compression tree, such a movement will not cause extra overhead. However, theoretical results on optimization is lacking because of the complicated nature of this objective function.

4.3 Optimization with Underlying Network Structure

If the users themselves have certain underlying network connection, the optimization problem is called key hierarchy problem and defined by Chan et al. [\[8\]](#).

An instance of the key hierarchy problem is given by the tuple (S, w, G, c) , where S is the set of group members, $w : S \rightarrow Z$ is the weight function capturing the update probability of each member, $G = (V, E)$ is the underlying communication network with $V \supseteq S \cup \{r\}$ where r is the group controller (GC), and $c : E \rightarrow Z$ specifies the cost of the edges in G .

For a given instance, a *hierarchy* on a set $X \subseteq S$ is a rooted tree H whose leaves are the elements of X . For a hierarchy T over X , the cost of a member $x \in X$ with respect to T is given by $\sum_{\text{ancestor } u \text{ of } x} \sum_{\text{child } v \text{ of } u} M(T_v)$ where T_v is the set of leaves in the subtree of T rooted at v , and for any set $Y \subseteq S$, $M(Y)$ is the cost of multicasting from root r to Y in G . The cost of a hierarchy T over X is the sum of the weighted costs of all the members of X with respect to T . The goal is to determine a hierarchy of minimum cost.

Within that model, [8] presented a polynomial-time approximation scheme for minimizing the average number of multicast messages needed for an update. They further showed that when routing costs are considered, the problem is NP-hard even if the underlying routing network is a tree (by a reduction from three Partition) or if every member has the same update frequency (by a reduction from 3D matching). Then for the general case when routing network is an arbitrary weighted graph and group members have nonuniform update frequencies, they designed a polynomial-time constant-factor approximation algorithm.

5 Conclusion

This chapter summarizes the recent progress in the key tree optimization problem, especially in the domain of the batch update model. When different batch decisions are adopted based on different user behaviors, optimal solutions can be computed in polynomial time in most of the cases. In the models considered in this chapter, there are quite some open problems to be solved. Firstly, in the model where each user has a certain probability to get replaced, the structural properties of the optimal tree for the uniform probability case when the probability tends to 0 are still well worth exploring. For the heterogeneous probability version, one can try to look for efficient algorithms to calculate the optimal tree. Secondly, when a certain number of users leave the group during the batch period, the conjecture proposed in [43] for the general k is to be verified or disproved. Finally, one can also consider other user patterns and other cost models for the key tree optimization problems.

Cross-References

- ▶ [Advanced Techniques for Dynamic Programming](#)
- ▶ [Network Optimization](#)
- ▶ [Optimization Problems in Data Broadcasting](#)
- ▶ [Optimizing Data Collection Capacity in Wireless Networks](#)

Recommended Reading

1. G. Ateniese, M. Steiner, G. Tsudik, New multiparty authentication services and key agreement protocols. *IEEE J. Sel. Areas Commun.* **18**(4), 628–640 (2000)
2. M. Bechler, H.-J. Hof, D. Kraft, F. Pahlke, L. Wolf, A cluster-based security architecture for ad hoc networks, in *Proceedings of the 23rd IEEE International Conference on Computer Communications*, Hong Kong, 2004, pp. 2393–2403
3. K. Becker, U. Wille, Communication complexity of group key distribution, in *Proceedings of the 5th ACM Conference on Computer and Communications Security*, San Francisco, CA, 1998
4. C. Blundo, P. D'Arco, M. Listo, A new self-healing key distribution scheme, in *Proceedings of the IEEE International Symposium on Computers and Communication*, Kemer-Antalya, Turkey, 2003, pp. 803–808
5. M. Burmester, Y. Desmedt, A secure and efficient conference key distribution system, in *Advances in Cryptology – EUROCRYPT'94*, ed. by A. Santis. Lecture Notes in Computer Science, vol. 950 (Springer, Berlin/Heidelberg, 1995), pp. 275–286
6. A.C.-F. Chan, Distributed symmetric key management for mobile ad hoc networks, in *Proceedings of the 23rd IEEE International Conference on Computer Communications*, Hong Kong, 2004, pp. 2414–2424
7. H. Chan, A. Perrig, D. Song, Random key predistribution schemes for sensor networks, in *Proceeding of 2003 Symposium on Security and Privacy*, Berkeley, CA, 2003, pp. 197–213
8. A. Chan, R. Rajaraman, Z. Sun, F. Zhu, Approximation algorithms for key management in secure multicast, in *Proceedings of the Fifteenth Annual International Conference on Computing and Combinatorics (COCOON)*, Niagara Falls, NY, 2009, pp. 148–157
9. Y.-K. Chan, M. Li, W. Wu, Optimal tree structure with loyal users and batch updates. *J. Comb. Optim.* **22**(4), 630–639 (2011)
10. Z.Z. Chen, Z. Feng, M. Li, F.F. Yao, Optimizing deletion cost for secure multicast key management. *Theor. Comput. Sci.* **401**, 52–61 (2008)
11. W. Diffie, M.E. Hellman, New directions in cryptography. *IEEE Trans. Inf. Theory* **22**, 644–654 (1976)
12. L.R. Dondeti, S. Mukherjee, A. Samal, DISEC: a distributed framework for scalable secure many-to-many communication, in *Proceedings of IEEE Symposium on Computers and Communications*, Antibes, France, 2000, pp. 693–698
13. L. Eschenauer, V.D. Gligor, A key-management scheme for distributed sensor networks, in *Proceedings of the 9th ACM Conference on Computer and Communication Security*, Washington, DC, 2002, pp. 41–47
14. A. Fiat, M. Naor, Broadcast encryption, in *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, Santa Barbara, CA, 1993, pp. 480–491
15. M.T. Goodrich, J.Z. Sun, R. Tamassia, Efficient tree-based revocation in groups of low-state devices, in *Proceedings of the Twenty-Fourth Annual International Cryptology Conference (CRYPTO)*, Santa Barbara, CA, 2004, pp. 511–527
16. R.L. Graham, M. Li, F.F. Yao, Optimal tree structures for group key management with batch updates. *SIAM J. Discret. Math.* **21**(2), 532–547 (2007)
17. D. Halevy, A. Shamir, The LSD broadcast encryption scheme, in *Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, Santa Barbara, CA, 2002, pp. 47–60
18. S.C.-H. Huang, F. Yao, M. Li, W. Wu, Lower bounds and new constructions on secure group communication schemes. *Theor. Comput. Sci.* **407**(1–3), 511–523 (2008)
19. I. Ingemarsson, D. Tang, C. Wang, A conference key distribution system. *IEEE Trans. Inf. Theory* **28**, 714–720 (1982)
20. D.-W. Kwak, S.J. Lee, J.W. Kim, E. Jung, An efficient key tree management algorithm for LKH group key management, in *ICOIN'06 Proceedings of the 2006 International Conference on Information Networking: Advances in Data Communications and Wireless Networks*, Sendai, Japan. Lecture Notes in Computer Science, vol. 3961/2006, 2006, pp. 703–712

21. X.S. Li, Y.R. Yang, M.G. Gouda, S.S. Lam, Batch re-keying for secure group communications, in *Proceedings of the Tenth International Conference on World Wide Web (WWW)*, Hong Kong, China, 2001, pp. 525–534
22. M. Li, Z. Feng, R.L. Graham, F.F. Yao, Approximately optimal trees for group key management with batch updates, in *Proceedings of the Fourth Annual Conference on Theory and Applications of Models of Computation (TAMC)*, Shanghai, China, 2007, pp. 284–295
23. D. Liu, P. Ning, Establishing pairwise keys in distributed sensor networks, in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington, DC, 2003, pp. 52–61
24. D. Liu, P. Ning, K. Sun, Efficient self-healing group key distribution with revocation capability, in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington, DC, 2003, pp. 231–240
25. W. Lou, W. Liu, Y. Fang, SPREAD: enhancing data confidentiality in mobile ad hoc networks, in *Proceedings of the 23rd IEEE International Conference on Computer Communications*, Hong Kong, 2004, pp. 2404–2413
26. S. Mittra, Iolus: a framework for scalable secure multicasting, in *ACM SIGCOMM*, Cannes, France, 1997, pp. 277–288
27. D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, Santa Barbara, CA, 2001, pp. 41–62
28. M. Onen, R. Molva, Reliable group rekeying with a customer perspective, in *Proceedings of the Global Telecommunications Conference (GLOBECOM)*, Dallas, TX, vol. 4, 2004, pp. 2072–2076
29. A. Perrig, D. Song, D. Tygar, Elk, a new protocol for efficient large-group key distribution, in *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, CA, 2001, pp. 247–262
30. R. Safavi-Naini, H. Wang, New constructions for multicast re-keying schemes using perfect hash families, in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, 2000, pp. 228–234
31. A. Selcuk, C. McCubbin, D. Sidhu, Probabilistic optimization of LKH-based multicast key distribution schemes, INTERNET-DRAFT draft-selcuk-probabilistic-lkh-00.txt
32. J. Snoeyink, S. Suri, G. Varghese, A lower bound for multicast key distribution, in *Proceedings of the Twentieth Annual IEEE Conference on Computer Communications*, Alaska, 2001, pp. 422–431
33. J. Staddon, S. Miner, M. Franklin, Self-healing key distribution with revocation, in *Proceedings of IEEE Symposium on Security and Privacy*, Oakland, California, 2002, pp. 241–257
34. M. Steiner, G. Tsudik, M. Waidner, Diffie-Hellman key distribution extended to group communication, in *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, New Delhi, India, 1996, pp. 31–37
35. M. Steiner, G. Tsudik, M. Waidner, Cliques: a new approach to group key agreement, in *Proceedings of the 18th International Conference on Distributed Computing Systems*, Amsterdam, The Netherlands, 1998, pp. 380–387
36. M. Steiner, G. Tsudik, M. Waidner, Key agreement in dynamic peer groups. *IEEE Trans. Parallel Distrib. Syst.* **11**(8), 769–780 (2000)
37. Y. Sun, K.J.R. Liu, Scalable hierarchical access control in secure group communications, in *Proceedings of the 23rd IEEE International Conference on Computer Communications*, Hong Kong, 2004, pp. 1296–1306
38. Y. Sun, K.J.R. Liu, Securing dynamic membership information in multicast communications, in *Proceedings of the 23rd IEEE International Conference on Computer Communications*, Hong Kong, 2004, pp. 1307–1317
39. W. Trappe, Y. Wang, K.J.R. Liu, Establishment of conference keys in heterogeneous networks, in *Proceedings of IEEE International Conference on Communications*, New York, NY, 2002, pp. 2202–2205
40. G. Tsudik, Y. Kim, A. Perrig, Simple and fault-tolerant key agreement for dynamic collaborative groups, in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, 2000, pp. 235–244

-
41. D. Wallner, E. Harder, R.C. Agee, Key management for multicast: issues and architectures, RFC 2627, June 1999
 42. C.K. Wong, M.G. Gouda, S.S. Lam, Secure group communications using key graphs. *IEEE/ACM Trans. Netw.* **8**(1), 16–30 (2000)
 43. W. Wu, M. Li, E. Chen, Optimal key tree structure for deleting two or more leaves, in *Proceedings of the Nineteenth International Symposium on Algorithms and Computation (ISAAC)*, Gold Coast, Australia, 2008, pp. 77–88
 44. W. Wu, M. Li, E. Chen, Optimal tree structures for group key tree management considering insertion and deletion cost. *Theor. Comput. Sci.* **410**(27–29), 2619–2631 (2009)
 45. F. Zhu, A. Chan, G. Noubir, Optimal tree structure for key management of simultaneous join/leave in secure multicast, in *Proceedings of Military Communications Conference*, Boston, 2003, pp. 773–778
 46. S. Zhu, S. Setia, S. Jajodia, LEAP: efficient security mechanisms for large-scale distributed sensor networks, in *Proceedings of the 10th ACM Conference on Computer and Communications Security*, Washington, DC, 2003, pp. 62–72

Linear Programming Analysis of Switching Networks

Hung Q. Ngo and Thanh-Nhan Nguyen

Contents

1	Introduction	1756
2	Preliminaries	1757
2.1	Switching Networks	1758
2.2	The 3-Stage Clos Networks	1761
2.3	The d -Ary Multilog Networks	1762
3	Clos Network Analysis Examples	1765
3.1	Two Classic Examples in Circuit Switching	1765
3.2	Multirate Wide-Sense Nonblocking	1767
4	Strictly Nonblocking Unicast Multilog Networks for Photonic Switching Under the General Crosstalk Constraint	1770
4.1	The Linear Programming Duality Formulation	1771
4.2	The Case When n Is Odd	1774
4.3	The Case When n Is Even	1778
5	Strictly Nonblocking f -Casting Multilog Networks Under the General Crosstalk Constraint	1780
6	Analyzing f -Cast Wide-Sense Nonblocking Multilog Networks	1789
6.1	Setting Up the Linear Program and Its Dual	1790
6.2	Specifying a Family of Dual-Feasible Solutions	1793
6.3	Selecting the Best Dual-Feasible Solution	1798
6.4	Some Quick Consequences of Theorem 9	1802
7	Analyzing Crosstalk-Free f -Cast Wide-Sense Nonblocking Multilog Networks	1803
7.1	Setting Up the Linear Program and Its Dual	1803
7.2	Specifying a Family of Dual-Feasible Solutions	1805
7.3	Selecting the Best Dual-Feasible Solution	1808
8	Conclusion	1811
	Cross-References	1812
	Recommended Reading	1812

H.Q. Ngo (✉) • T.-N. Nguyen

Computer Science and Engineering, University at Buffalo, The State University of New York,
Buffalo, NY, USA
e-mail: hungngo@buffalo.edu; nguyen9@buffalo.edu

Abstract

The analysis of nonblocking switching networks often centers around a natural combinatorial optimization problem. The constraints capture the limited physical capacities and/or capabilities of the inputs, outputs, the internal switching components, and their interconnection patterns. The objective of the optimization problem is often to minimize the number of certain switching components.

Recent works have pointed to a simple strategy to derive useful bounds for the optimization objective. The bounds give us sufficient conditions for a switching network design to be nonblocking in some desired sense. The strategy, which applies in a variety of settings, works roughly as follows. The optimization problem is first rewritten as an (integer) linear program. By exploiting the structure of the primal linear program or by specifying a family of dual-feasible solutions, a sufficient condition for the switching network to be nonblocking can quickly be derived.

This chapter illustrates the above analysis technique with many examples on two fundamental switching network topologies: the Clos network and the multilog network. The examples shall cover a wide range of switching network environments: from unicast to multicast, from circuit- to multirate- to optical-switching, and from wide-sense to strictly nonblocking networks. Many known results are derived as direct consequences of these examples.

1 Introduction

The two most important architectures for designing nonblocking switching networks are Clos-type [7] and Banyan-type [16]. The Clos network not only played a central role in classical circuit-switching theory [1, 21] but also was the bedrock of multirate switching [6, 15, 19, 29, 35, 37, 50] (e.g., in time-division switching environments where connections are of varying bandwidth requirements) and photonic-switching [17, 38, 45, 46, 55, 59, 60]. The Banyan network is isomorphic to various other “bit-permutation” networks such as Omega, baseline, etc. [3]; they are called *Banyan-type* networks and have been used extensively in designing electronic and optical switches, as well as parallel processor architectures [12]. In particular, the multilog design which involves the vertical stacking of a number of inverse Banyan planes has been used in circuit- and photonic-switching environments because they have small depth ($\log N$), self-routing capability, and absolute signal loss uniformity [20, 24, 25, 27, 28, 33, 47, 53]. Clos and the multilog networks are formally defined in the next section.

In analyzing Clos networks, the most basic task is to determine the minimum number of middle-stage crossbars so that the network satisfies a given nonblocking condition. This holds true in space-, multirate-, and photonic-switching; in unicast, f -cast and multicast, and broadcast traffic patterns; and in all nonblocking types (strict-sense, wide-sense, and rearrangeable). Similarly, analyzing multilog networks often involves determining the minimum number of Banyan planes so that the network satisfies some requirements.

This chapter aims to illustrate that a simple and effective two-step strategy based on linear programming can be employed in the analysis:

- First, the minimum value being sought (e.g., the number of middle-stage crossbars in a Clos network or the minimum number of Banyan planes in a multilog network) is upperbounded by the optimum value of a linear program (LP) of the form $\max\{\mathbf{c}^T \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$. The maximization objective often comes from some form of worst-case analysis. For example, one might want the maximum number of middle-stage crossbars in a Clos network which is insufficient to carry a new request. The constraints of the linear program are used to express the fact that no input or output can generate or receive connection requests totaling more than its capacity.
- Second, by specifying *any* feasible solution, say \mathbf{y}^* , to the dual program $\min\{\mathbf{b}^T \mathbf{y} \mid \mathbf{A}^T \mathbf{y} \geq \mathbf{c}\}$ and applying weak duality, the dual-objective value $\mathbf{b}^T \mathbf{y}^*$ can be used as an upperbound for the minimum value being sought.

In some cases, the second step may not be necessary because the primal linear program is small with only a few variables. In most cases, however, the linear program and its dual are very general, dependent on various parameters of the switch design (e.g., number of inputs and switching elements' dimensions). In such cases, it would be difficult to come up with a closed-form primal-optimal solution. Fortunately, by weak duality, a dual-feasible solution can quickly “certify” the bound. In fact, as shall be seen later, it is often the case that a family of dual-feasible solutions is needed and the best one is chosen (i.e., the one giving the smallest objective value) in accordance with the parameters of the problem at hand.

The linear programming duality technique was first used in a recent paper [40] to analyze the (unicast) strictly nonblocking multilog architecture in the photonic-switching case, subject to general crosstalk constraints. This chapter demonstrates that the technique can be applied to a wider range of switching network analysis problems, presenting a subset of follow-up works in [41, 42]. Clos and multilog networks are used as canonical examples to illustrate the analysis technique.

The rest of this chapter is organized as follows. [Section 2](#) presents notations and terminologies. [Section 3](#) illustrates the strength of the LP-duality technique on analyzing several problems on the Clos networks. [Sections 4–7](#) present examples in the multilog architecture. [Section 4](#) is on strictly non-blocking photonic unicast switching under a general crosstalk constraint; [Sect. 5](#) deals with the strictly non-blocking multicast case under the same constraint. [Section 6](#) presents results on the f -cast wide-sense non-blocking case, and [Sect. 7](#) analyzes the crosstalk-free f -cast wide-sense non-blocking multilog networks. Finally, [Sect. 8](#) concludes the chapter with some remarks on open problems.

2 Preliminaries

The following notations will be used throughout this chapter. For any positive integers l, d , let $[l]$ denote the set $\{1, \dots, l\}$, \mathbb{Z}_d denote the set $\{0, \dots, d - 1\}$ which can be thought of as d -ary “symbols,” \mathbb{Z}_d^l denote the set of all d -ary strings of length

l , b^l denote the string with symbol $b \in \mathbb{Z}_d$ repeated l times (e.g., $3^4 = 3333$), $|\mathbf{s}|$ denote the length of any d -ary string \mathbf{s} (e.g., $|31| = 2$), and $\mathbf{s}_{i..j}$ denote the substring $s_i \dots s_j$ of a string $\mathbf{s} = s_1 \dots s_l \in \mathbb{Z}_d^l$. If $i > j$, then $\mathbf{s}_{i..j}$ is the empty string.

2.1 Switching Networks

The reader is referred to [1, 18, 19, 21, 36] for more detailed coverage on switching networks, their practical usage, and related concepts. The discussion on switching network background will be relatively brief.

Roughly, an $N \times N$ switching network is used to interconnect N inputs to N outputs. Figure 1 shows a switching network with 32 inputs and 32 outputs. Depending on the application at hand, the inputs and outputs represent different objects. For example, in the old days, telephone calls are circuit-switched from inputs to outputs simultaneously in a one-to-one fashion. In parallel computer architectures, inputs may be processors, and outputs are memories. Inside a modern Internet router, there often is a switching network (or *switching fabric*) connecting input linecards to output linecards. Internal to the switching network are various types of *switching components* or *switching elements*. For example, in Fig. 1, the most basic switching element is used: 2×2 switching elements. The key functionality of a switching network is to transfer (electronic, optical) signals from the inputs to outputs, realizing as many connection patterns as possible while keeping the hardware cost as low as possible. Hardware cost is often measured by the number of switching components used.

2.1.1 Connection Patterns

The connection patterns to be realized could be one-to-one (*unicast*) or one-to-many (*multicast*). The one-to-all connection pattern is called a *broadcast*. Multicast-capable (photonic)-switching architectures will likely play a central role in the near future because most bandwidth-intensive applications require multicast support. Many current multicast switch designs focus on the broadcast case [13, 26, 57]. Although broadcast switches are certainly capable of supporting multicast with any *fanout* requirements, they are not scalable due to their prohibitively high hardware cost. (The fanout of a multicast request is size of the output set the request asks for.) Moreover, most multicast applications are restricted to a group of users, where broadcasting is rarely required. Hence, allocating expensive broadcast capability to each network switch is cost-inefficient for most practical purposes. Moreover, from the viewpoints of resource fairness and network security (e.g., limiting virus and worm propagation), there are other good reasons to impose a restriction on the maximum fanout of each request. Consequently, there have been many research efforts on designing and analyzing *f*-*cast* switches, in which the maximum fanout of each request is upperbounded by the parameter f [4, 14, 21, 23, 41–43, 54]. An *f*-*cast* switch usually requires significantly lower hardware cost than its broadcast counterpart. Furthermore, a good design of an $N \times N$ *f*-*cast* switch covers both the unicast design ($f = 1$) and the broadcast design ($f = N$) as special cases.

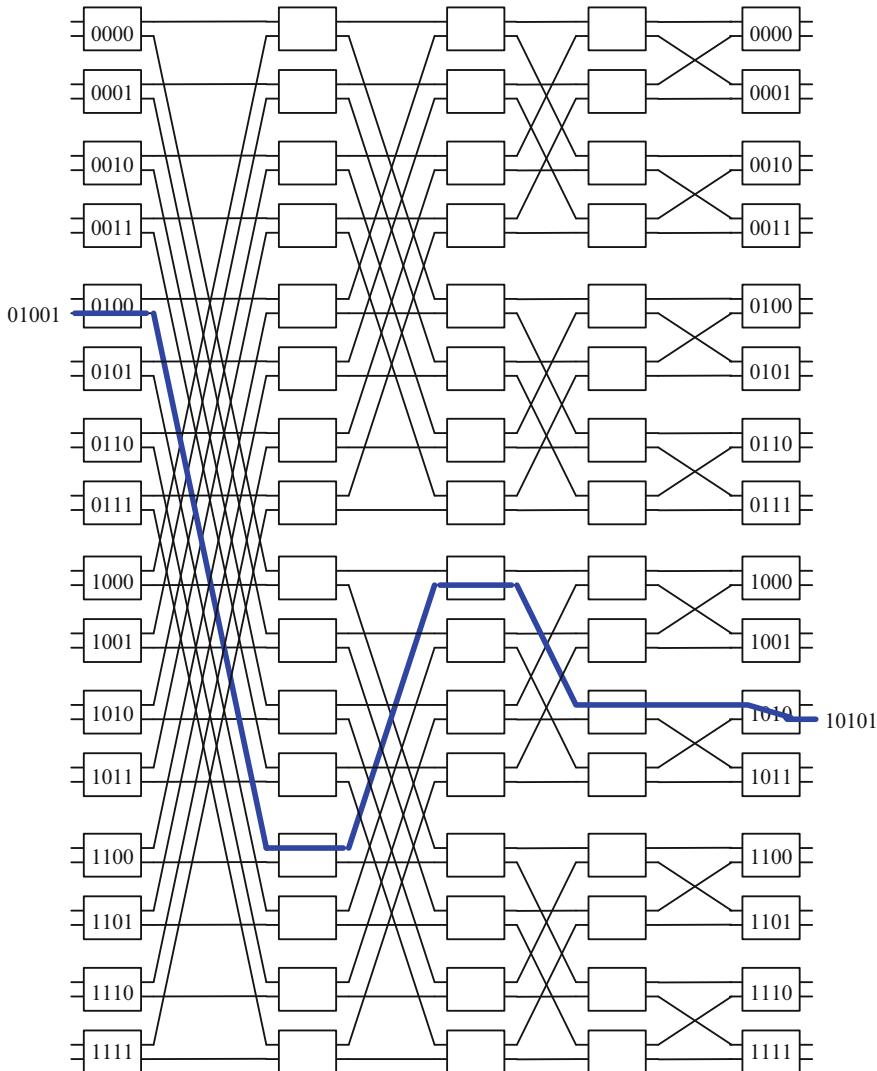


Fig. 1 The inverse Banyan network $\text{BY}^{-1}(5)$

Consequently, studying general f -cast switches is both mathematically pleasing and practically useful, as the results potentially can offer network designers more flexibility in selecting architectures for future multicast-intensive networks.

2.1.2 Switching Environments

Traditionally, each link can only carry one connection at a time. This setting is called the *circuit-switching* or *space-switching* environment, which served as the theoretical foundation for telephone switching [1, 21]. When time-division

multiplexing (TDM) is applicable, multiple connections can time-share a link as long as the total bandwidth of the connections does not exceed the link's capacity. This setting is called the *multirate* environment, which is the theoretical foundation for the development of most Asynchronous Transfer Mode (ATM) switching systems from major ATM equipment manufacturer [8, 50, 51]. With the advances of dense wavelength division multiplexing (DWDM) technology [34, 44, 48], connections can also share (optical) links by being carried on different wavelengths. The number of wavelengths in a wavelength division multiplexed (WDM) network can be up to hundreds or more per fiber, and each wavelength operates at 10 Gbps (OC-192) or higher [30–32].

In circuit switching, a request is a pair (\mathbf{a}, \mathbf{b}) where \mathbf{a} is an unused input and \mathbf{b} is an unused output. A route $R(\mathbf{a}, \mathbf{b})$ realizes the request if it does not share any internal link with existing routes. In an f -cast switching network, each multicast request is of the form (\mathbf{a}, B) where \mathbf{a} is some input and B is a subset of at most f outputs. The number f is called the *fanout restriction*. An $N \times N$ multicast network without fanout restriction is equivalent to an N -cast network.

In the multirate case, each link has a capacity (e.g., bandwidth). All inputs and outputs have the same capacity normalized to 1. An input cannot request more than its capacity, neither can outputs. A request is of the form $(\mathbf{a}, \mathbf{b}, w)$ where \mathbf{a} is an input, \mathbf{b} is an output, and $w \leq 1$ is the requested *rate*. If existing requests have used up to x and y units of \mathbf{a} 's and \mathbf{b} 's capacity, respectively, then the new requested rate w can only be at most $\min\{1 - x, 1 - y\}$. An internal link cannot carry requests with total rate more than 1. In practice, internal links might have higher capacities than 1 because *internal speedup* is needed for throughput guarantees [9]. A simpler picture is presented here for the sake of clarity, even though the analysis technique works in the speedup case just as well.

2.1.3 Nonblockingness

As switching elements and links have limited capacities, one connection can *block* another from being realized. For example, the switching network shown in Fig. 1 is a unique-path network: there is only one path between an input and an output. If each internal link can only carry one connection at a time, then the connection shown in the figure blocks a connection from the other input of switching element 0100 to the other output of switching element 1010. In some switching environment such as photonic switching, it may not even be desirable for two distinct paths to share a switching element. For example, the so-called *crosstalk-free* constraint requires as much. In this case, one route blocks another route if they share a switching element. On the other hand, in a multirate switching environment, routes are allowed to share both links and nodes as long as the total rates of those routes do not exceed the shared element's capacity. Thus, the notion of blockingness intimately depends on the switching environment that the switching architecture is designed for.

One of the main problems in switching network theory is to design and analyze switching network which are nonblocking in a specific sense. There are three levels of nonblockingness of a switching network. A network is *rearrangeably*

nonblocking (RNB) if it can realize any one-to-one mapping between inputs and outputs simultaneously; it is *wide-sense nonblocking* (WSNB) if a new request from a free input to a free output can be realized without disturbing existing connections, as long as all requests are routed according to some algorithm; finally, it is *strictly nonblocking* (SNB) if a new request from a free input to a free output can always be routed no matter how existing connections were arranged. In the multicast case, the three nonblocking concepts are defined similarly.

2.2 The 3-Stage Clos Networks

An $n \times m$ crossbar can be thought of as a switching network with n inputs and m outputs which is strictly nonblocking: a free input can always communicate with a free output without disturbing existing connections between inputs and outputs. This definition of crossbars is sufficient for the purposes of this chapter. How specifically a crossbar is designed is not particularly important.

The *Clos network* $C(n_1, r_1, m, n_2, r_2)$ is a 3-stage interconnection network, where the first stage consists of r_1 crossbars of size $n_1 \times m$, the last stage has r_2 crossbars of dimension $m \times n_2$, and the middle stage has m crossbars of dimension $r_1 \times r_2$ (see Fig. 2). Each input crossbar I_i ($i = 1, \dots, r_1$) is connected to each middle crossbar M_j ($j = 1, \dots, m$). Similarly, the middle stage and the last stage are fully connected. When $n_1 = n_2 = n$ and $r_1 = r_2 = r$, the network is called the *symmetric 3-stage Clos network*, denoted by $C(n, m, r)$. The Clos architecture was proposed by Clos in 1953 [7] for telephone circuit switching. The idea was very influential in the design of modern switching architectures.

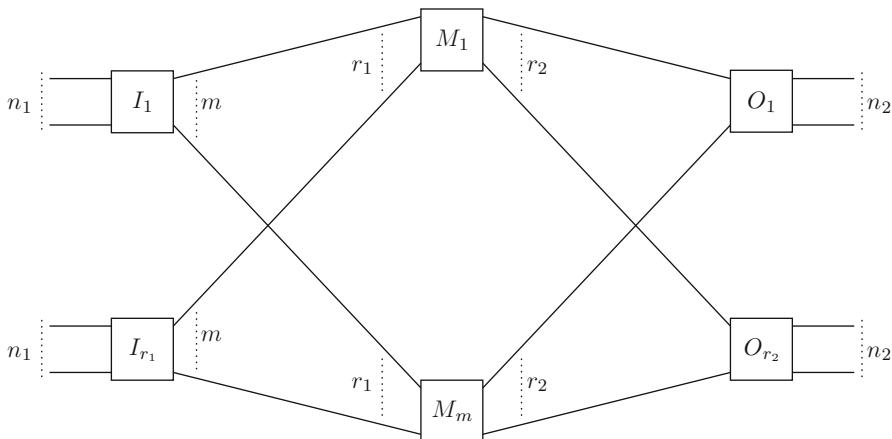


Fig. 2 The 3-stage Clos network $C(n_1, r_1, m, n_2, r_2)$

2.3 The d -Ary Multilog Networks

Let $N = d^n$. The $\log_d(N, 0, m)$ network denotes the stacking of m copies of the d -ary inverse Banyan network $\text{BY}^{-1}(n)$ with N inputs and N outputs. [Figure 3](#) shows a $\log_d(N, 0, 3)$ network. The inputs are connected to a $1 \times m$ demultiplexor (or equivalently a $1 \times m$ crossbar). The output side is the mirror image of the input side, namely, outputs of the network are outputs of multiplexors (or equivalently $m \times 1$ crossbars). [Figure 4](#) illustrates the architecture of a specific multilog network with two planes.

Each of the copies of the inverse Banyan network $\text{BY}^{-1}(n)$ is called a $\text{BY}^{-1}(n)$ -plane or a (inverse) Banyan plane of the multilog network. A d -ary $\text{BY}^{-1}(n)$ -plane is itself a switching network with N inputs and N outputs. [Figures 1](#) and [5](#) illustrate two Banyan planes. Label the inputs and outputs of a $\text{BY}^{-1}(n)$ -plane with d -ary strings of length n . Specifically, each input $\mathbf{x} \in \mathbb{Z}_d^n$ and output $\mathbf{y} \in \mathbb{Z}_d^n$ have the form $\mathbf{x} = x_1 \cdots x_n$, $\mathbf{y} = y_1 \cdots y_n$, where $x_i, y_i \in \mathbb{Z}_d$, $\forall i \in [n]$.

Also, label the $d \times d$ switching elements in each of the n stages of a $\text{BY}^{-1}(n)$ -plane with d -ary strings of length $n - 1$. An input \mathbf{x} (respectively, output \mathbf{y}) is connected to the switching element labeled $\mathbf{x}_{1..n-1}$ in the first stage (respectively, $\mathbf{y}_{1..n-1}$ in the last stage). A switching element labeled $\mathbf{z} = z_1 \cdots z_{n-1}$ in stage $i \leq n - 1$ is connected to d switching elements in stage $i + 1$ numbered $z_1 \cdots z_{i-1} * z_{i+1} \cdots z_{n-1}$, where $*$ is any symbol in \mathbb{Z}_d .

For the sake of clarity, let us first consider a small example. Consider the unicast request $(\mathbf{x}, \mathbf{y}) = (01001, 10101)$ when $d = 2, n = 5$. The input $\mathbf{x} = 01001$ is connected to the switching element labeled 0100 in the first stage, which is connected to two switching elements labeled 0100 and 1100 in the second stage

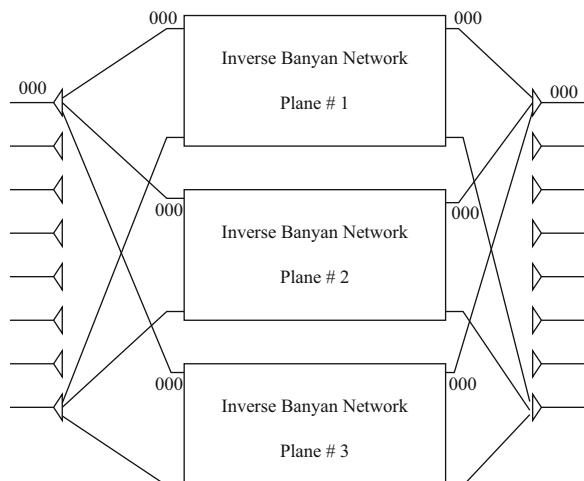


Fig. 3 A multilog network with 3 inverse Banyan planes

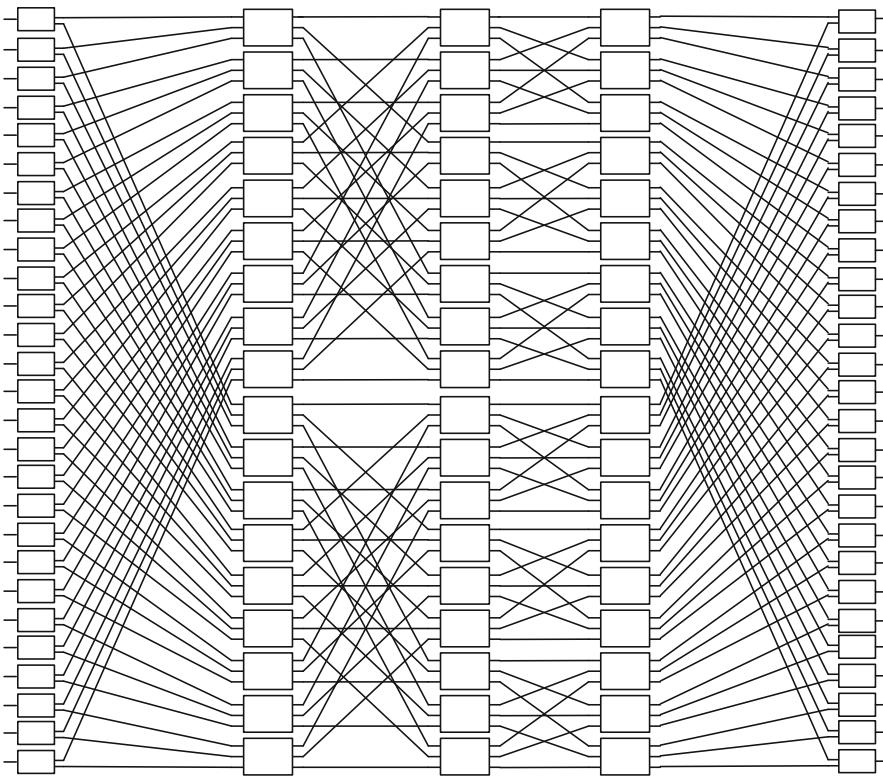


Fig. 4 The $\log_3(27, 0, 2)$ network

and so on. The unique path from \mathbf{x} to \mathbf{y} in the $\text{BY}^{-1}(n)$ -plane can be explicitly written out (see Fig. 1):

input \mathbf{x}	01001
stage-1 switching element	0100
stage-2 switching element	1100
stage-3 switching element	1010
stage-4 switching element	1010
stage-5 switching element	1010
output \mathbf{y}	10101

A pattern emerges from the above example: the prefixes of $\mathbf{y}_{1..n-1}$ are “taking over” the prefixes of $\mathbf{x}_{1..n-1}$ on the path from \mathbf{x} to \mathbf{y} . In general, the unique path $R(\mathbf{x}, \mathbf{y})$ in a $\text{BY}^{-1}(n)$ -plane from an arbitrary input \mathbf{x} to an arbitrary output \mathbf{y} is exactly the following:

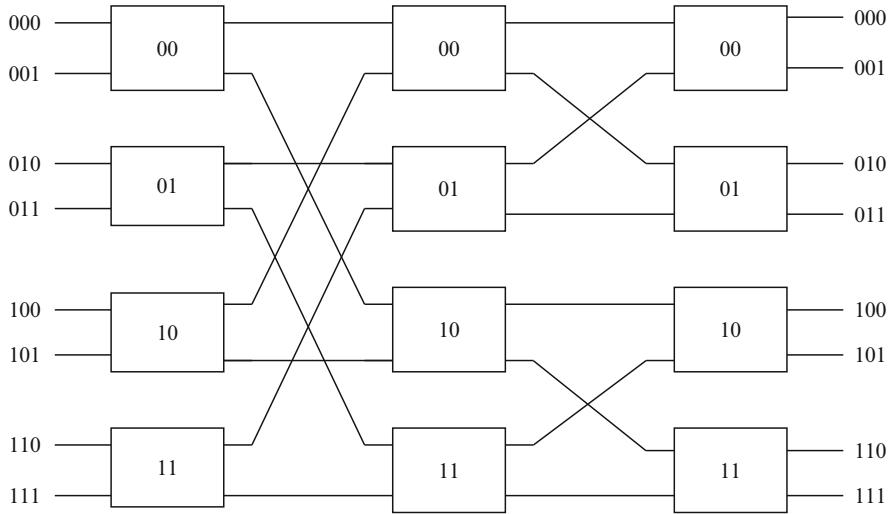


Fig. 5 The inverse Banyan network $\text{BY}^{-1}(3)$

input \mathbf{x}	$x_1 x_2 \dots x_{n-1} x_n$
stage-1 switching element	$x_1 x_2 \dots x_{n-1}$
stage-2 switching element	$y_1 x_2 \dots x_{n-1}$
stage-3 switching element	$y_1 y_2 \dots x_{n-1}$
\vdots	\vdots
stage- n switching element	$y_1 y_2 \dots y_{n-1}$
output \mathbf{y}	$y_1 y_2 \dots y_{n-1} y_n$

Now, consider two unicast requests (\mathbf{a}, \mathbf{b}) and (\mathbf{x}, \mathbf{y}) . From the observation above, on the same $\text{BY}^{-1}(n)$ -plane, the two routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ share a switching element (also called a node) if and only if there is some $j \in [n]$ such that $\mathbf{b}_{1..j-1} = \mathbf{y}_{1..j-1}$ and $\mathbf{a}_{j..n-1} = \mathbf{x}_{j..n-1}$. In this case, the two paths intersect at a stage j switching element. It should be noted that two requests' paths may intersect at more than one switching element.

For any two d -ary strings $\mathbf{u}, \mathbf{v} \in \mathbb{Z}_d^l$, let $\text{PRE}(\mathbf{u}, \mathbf{v})$ denote the *longest common prefix*, and $\text{SUF}(\mathbf{u}, \mathbf{v})$ denote the *longest common suffix* of \mathbf{u} and \mathbf{v} , respectively. For example, if $\mathbf{u} = 0100110$ and $\mathbf{v} = 0101010$, then $\text{PRE}(\mathbf{u}, \mathbf{v}) = 010$ and $\text{SUF}(\mathbf{u}, \mathbf{v}) = 10$. The following proposition immediately follows:

Proposition 1 Let (\mathbf{a}, \mathbf{b}) and (\mathbf{x}, \mathbf{y}) be two unicast requests. Then their corresponding routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ in a $\text{BY}^{-1}(n)$ -plane share at least a common switching element if and only if

$$|\text{SUF}(\mathbf{a}_{1..n-1}, \mathbf{x}_{1..n-1})| + |\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| \geq n - 1. \quad (1)$$

Moreover, the routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ intersect at exactly one switching element if and only if

$$|\text{SUF}(\mathbf{a}_{1..n-1}, \mathbf{x}_{1..n-1})| + |\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| = n - 1, \quad (2)$$

in which case, the common switching element is a switching element at stage $|\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| + 1$ of the $\text{BY}^{-1}(n)$ -plane.

Similarly, on the same Banyan plane, the two routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ share a link if and only if there is some $j \in [n - 1]$ such that $\mathbf{b}_{1..j-1} = \mathbf{y}_{1..j-1}$, $\mathbf{a}_{j..n-1} = \mathbf{x}_{j..n-1}$, $\mathbf{b}_{1..j} = \mathbf{y}_{1..j}$, and $\mathbf{a}_{j+1..n-1} = \mathbf{x}_{j+1..n-1}$. In this case, the two routes share the link from a stage j switching element to a stage $j + 1$ switching element. From the analysis, the following proposition follows:

Proposition 2 *Let (\mathbf{a}, \mathbf{b}) and (\mathbf{x}, \mathbf{y}) be two unicast requests. Then their corresponding routes $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ in a $\text{BY}^{-1}(n)$ -plane share at least a common link if and only if*

$$|\text{SUF}(\mathbf{a}_{1..n-1}, \mathbf{x}_{1..n-1})| + |\text{PRE}(\mathbf{b}_{1..n-1}, \mathbf{y}_{1..n-1})| \geq n. \quad (3)$$

3 Clos Network Analysis Examples

3.1 Two Classic Examples in Circuit Switching

The following two simple examples have become classic textbook materials [1, 21].

Example 1 (Strictly nonblocking 3-stage Clos networks) Consider the symmetric Clos network $C(n, m, r)$. Consider a new request from an input of input crossbar I to an output of output crossbar O . Such request is called an I, O -request. A middle-crossbar M cannot carry this request if it already carried some request from I (the link IM is busy) or some request to O (the link MO is busy). Let x and y be the number of middle crossbars which are already carrying some requests from I and to O , respectively. Because the number of existing requests from I or to O is at most $n - 1$, it follows that $x \leq n - 1$ and $y \leq n - 1$. The number of unavailable middle crossbars is thus bounded above by the optimal value of the linear program

$$\begin{aligned} & \max x + y \\ \text{s.t. } & x \leq n - 1 \\ & y \leq n - 1 \\ & x, y \geq 0 \end{aligned}$$

Its optimal objective value $2n - 2$ is an upperbound on the number of unavailable middle crossbars. It follows that $m \geq 2n - 1$ is sufficient for $C(n, m, r)$ to be strictly nonblocking.

The above example is the standard strictly nonblocking analysis of the Clos network, casted in terms of linear programming. Because the example is too simple, it might feel a little forced to employ the heavy linear programming machinery. However, it should give the reader a quick sense of how the constraints and objectives of the linear programs are like in analyzing Clos networks and multilog networks. The next example is less trivial.

Example 2 (Wide-sense nonblocking 3-stage Clos networks) This example is a classic result by Moore, quoted in Beneš [1]. Consider the $C(n, m, 2)$ network. Requests are routed using the so-called *packing algorithm*, which tries to reuse a busy middle crossbar whenever possible. For any $i, j \in \{1, 2\}$, let M_{ij} be the set of middle crossbars carrying an I_i, O_j -request. The sets M_{ij} certainly change over time as requests come and go. However, it can be shown by induction on time that the routing rule ensures $|M_{11} \cup M_{22}| \leq n$ and $|M_{12} \cup M_{21}| \leq n$ at all times. For example, to show that $|M_{11} \cup M_{22}| \leq n$ at time t , without loss of generality, consider a new request from I_1 to O_1 at time t . If $M_{22} \setminus M_{11} \neq \emptyset$, then the new request can be routed using any middle crossbar in M_{22} . In this case, the quantity $|M_{11} \cup M_{22}|$ is unchanged from time $t - 1$ to time t . When $M_{22} \setminus M_{11} = \emptyset$, it follows that $M_{22} \subseteq M_{11}$. And, due to the fact that there are at most $n - 1$ existing requests from I_1 , it follows that $|M_{11} \cup M_{22}| = |M_{11}| \leq n - 1$ at time $t - 1$, which means $|M_{11} \cup M_{22}| \leq n$ at time t after the new I_1, O_1 -request is routed.

Now, consider a new request from I_1 to O_1 . Again, if $M_{22} \setminus M_{11} \neq \emptyset$, then there is a busy crossbar to reuse, and the total number of middle crossbars needed is not changed from time $t - 1$ to time t . Otherwise, the number of unavailable middle crossbars for this new request is precisely

$$|M_{11} \cup M_{12} \cup M_{21}| = |M_{11}| + |M_{12} \cup M_{21}|.$$

Just before the arrival of this new request, the number of existing requests from I_1 or to O_1 is at most $n - 1$, i.e.,

$$|M_{11} \cup M_{12}| = |M_{11}| + |M_{12}| \leq n - 1,$$

and

$$|M_{11} \cup M_{21}| = |M_{11}| + |M_{21}| \leq n - 1.$$

The number of unavailable middle crossbars is thus bounded by the optimal value of the following linear program (think of set cardinalities as variables):

$$\begin{aligned} \max \quad & |M_{11}| + |M_{12} \cup M_{21}| \\ \text{s.t.} \quad & |M_{11}| + |M_{12}| \leq n - 1 \end{aligned}$$

$$\begin{aligned} |M_{11}| + |M_{21}| &\leq n - 1 \\ |M_{12}| + |M_{21}| &\leq n \\ |M_{12} \cup M_{21}| - |M_{12}| - |M_{21}| &\leq 0 \end{aligned}$$

The last inequality is the straightforward union bound. Obviously, all cardinalities are nonnegative, so nonnegativity constraints are added to the above linear program. The dual linear program is

$$\begin{array}{llll} \min & (n-1)(y_1 + y_2) + ny_3 & & \\ \text{s.t.} & y_1 + y_2 & \geq 1 & \\ & y_2 + y_3 - y_4 & \geq 0 & \\ & y_1 + y_3 - y_4 & \geq 0 & \\ & y_4 & \geq 1 & \\ & y_1, y_2, y_3 & \geq 0 & \end{array}$$

The solution $y_1 = y_2 = y_3 = 1/2$ and $y_4 = 1$ is certainly dual feasible with objective value $3n/2 - 1$. Hence, by weak duality, the number of unavailable middle crossbars for the new I_1, O_1 -request is at most $\lfloor 3n/2 \rfloor - 1$, which means $m \geq \lfloor 3n/2 \rfloor$ is sufficient for $C(n, m, 2)$ to be wide-sense nonblocking.

Remark 1 It is known that $m \geq \lfloor 3n/2 \rfloor$ is also necessary [1]. This ($r = 2$) is the only case for which a necessary and sufficient condition is known for the Clos network $C(n, m, r)$ to be wide-sense nonblocking. In [58], Yang and Wang gave an interesting analysis of the so-called *packing algorithm* for $r > 2$, also based on linear programming. Their formulation is different from the one presented here, however.

3.2 Multirate Wide-Sense Nonblocking

The next example is an example from the multirate environment. In this case, each link has unit capacity. Each request has an associated *rate* or *weight* $w \in (0, 1]$. Total rate of requests from an input cannot exceed 1, neither does the total rate of requests to an output. It is known that $C(n, m, r)$ is multirate wide-sense nonblocking when $m \geq 5.75n$ [15]. This section uses the linear programming technique to improve this bound. The result and its linear programming analysis first appeared in [41]. The proof of the following theorem does not need to involve the dual program, partly because the primal program is sufficiently small to be solved directly.

Theorem 1 *The Clos network $C(n, m, r)$ is multirate wide-sense nonblocking if $m \geq 5.6355n + 4$.*

Proof For the sake of presentation clarity, a slightly weaker sufficient condition of $m \geq 5.675n + 1.8$ will be proved. Then, the straightforward way to obtain the better condition $m \geq 5.6355n + 4$ is discussed. The two proofs are basically identical, but the one for the weaker condition is slightly easier to follow.

Requests are classified into one of four types, depending on their weights. A request is of *type 1*, *type 2*, *type 3*, or *type 4*, if its weight belongs to the interval $(\frac{1}{2}, 1]$, $(\frac{2}{5}, \frac{1}{2}]$, $(\frac{1}{3}, \frac{2}{5}]$, or $(0, \frac{1}{3}]$, respectively. The algorithm maintains four *disjoint* sets of middle-stage crossbars, denoted by C_1, C_2, C_3 , and C_4 . Their cardinalities are $|C_i| = \lceil x_i n \rceil$, where x_1, x_2, x_3 , and x_4 are constants to be determined. Thus, the total number of middle-stage crossbars used by the algorithm is

$$\lceil x_1 n \rceil + \lceil x_2 n \rceil + \lceil x_3 n \rceil + \lceil x_4 n \rceil.$$

The routing algorithm is as follows:

- The crossbars in C_1 are used exclusively for requests of type 1.
- For requests of type i , $2 \leq i \leq 4$, the following strategy is applied. When a type- i request arrives, find a crossbar in C_i to route it. If no crossbar in C_i can accommodate this new request, try a crossbar in C_{i+1} , and so on, up to C_4 .

The next step is to show that if the constants x_i are feasible solutions to a certain linear program, then it is always possible to route a new request. Toward this end, define the *IO-load* of a middle-crossbar M to be the total weight of all existing connections (at time $t - 1$) which are either routed through the link IM or the link MO .

Now, suppose a type-1 request from input crossbar I to output crossbar O arrives at time t . If the algorithm cannot find a crossbar in C_1 for the new request, then every crossbar in C_1 must be carrying a type-1 connection either from I or to O at time $t - 1$. The total number of type-1 connections involving either I or O at time $t - 1$ is at most $(n - 1) + (n - 1) = 2n - 2$. (This reasoning is similar to that of [Example 1](#).) Hence, $\lceil x_1 n \rceil = |C_1| \leq 2n - 2$. Consequently, as long as $x_1 \geq 2$, the new type-1 request is not blocked.

Next, consider a request of weight w from input switch I to output switch O of type 2 which arrives at time t , and suppose the algorithm cannot find a crossbar in $C_2 \cup C_3 \cup C_4$ to route it. For a middle-crossbar $M \in C_2$ to be unavailable for the newly arrived (I, O, w) request, M must be carrying either two (existing) type-2 connections from I or two type-2 connections to O . Thus, the total *IO-load* of crossbars in C_2 must be at least $\frac{4}{5}|C_2|$. Similarly, for a middle-stage crossbar $M \in C_3$ to be unavailable, the total load M carries for either I or O must exceed $1/2$. Note that a middle-stage crossbar $M \in C_3$ only carries type-2 or type-3 requests. Hence, either the link IM or the link MO must be carrying either two type-2 connections (with total weight $> \frac{4}{5}$) or one type-2 connection and one type-3 connection (with total weight $> \frac{1}{3} + \frac{2}{5}$) or two type-3 connections (with total weight $> \frac{2}{3}$). The total *IO-load* for M must thus be at least $\frac{2}{3}$. Overall, the total *IO-load* that all crossbars in C_3 carry must be at least $\frac{2}{3}|C_3|$. Finally, if no crossbar in C_4 is

available, then the total *IO*-load that crossbars in C_4 are carrying must be at least $\frac{1}{2}|C_4|$.

Consequently, because the total *IO*-load overall is strictly smaller than $2n$ (at time $t - 1$), the new request is not blocked if

$$\frac{4}{5}|C_2| + \frac{2}{3}|C_3| + \frac{1}{2}|C_4| \geq 2n.$$

The inequality would hold if

$$\frac{4}{5}x_2 + \frac{2}{3}x_3 + \frac{1}{2}x_4 \geq 2.$$

In a similar fashion, it is easy to argue that newly arriving type-3 requests can be routed if

$$\frac{2}{3}x_3 + \frac{3}{5}x_4 \geq 2$$

and a new type-4 request can be routed if

$$\frac{2}{3}x_4 \geq 2.$$

Overall, the routing algorithm works if the variables x_i are feasible for the following linear program:

$$\begin{aligned} & \min x_1 + x_2 + x_3 + x_4 \\ \text{s.t. } & x_1 \geq 2 \\ & \frac{4}{5}x_2 + \frac{2}{3}x_3 + \frac{1}{2}x_4 \geq 2 \\ & \frac{2}{3}x_3 + \frac{3}{5}x_4 \geq 2 \\ & \frac{2}{3}x_4 \geq 2 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

The solution $x_1 = 2, x_2 = 3/8, x_3 = 3/10, x_4 = 3$ is certainly feasible. The total number of middle crossbars used is

$$\sum_{i=1}^4 \lceil x_i n \rceil \leq (x_1 + x_2 + x_3 + x_4)n + 1.8 = 5.675n + 1.8.$$

To prove the better bound of $5.6355n$, divide the rates into five types belonging to the intervals $(1/2, 1]$, $(2/5, 1/2]$, $(1/3, 2/5]$, $(11/43, 1/3]$, and $(0, 11/43]$. \square

Remark 2 The above strategy can also give a better sufficient condition than the best known in [15] for the case when there is internal speedup in the Clos network.

4 Strictly Nonblocking Unicast Multilog Networks for Photonic Switching Under the General Crosstalk Constraint

When the multilog architecture is employed to design a photonic switch, each 2×2 switching element (SE) needs to be replaced by a functionally equivalent optical component. For instance, when $d = 2$, the so-called *directional couplers* can be used as SEs [34, 48, 56]. However, directional couplers and many other optical switching elements suffer from optical crosstalk between interfering channels, which is one of the major obstacles in designing cost-effective switches [5, 11, 52].

To cope with crosstalk, the *general crosstalk constraint* was introduced by Vaez and Lea [53], which forces each new route to share at most c SEs with other active routes in the switch, where c is a parameter of the design. When $c = 0$, no two routes are allowed to share a common SE, and the switch becomes a *crosstalk-free* switch [5, 11, 25, 27, 33, 53, 54]. On the other hand, when $c = n = \log_d N$, a route can share any number of SEs with other routes (but still, no shared link) because each route in a $\log_d(N, 0, m)$ network contains exactly n SEs. In this case, only link blocking takes effect, and it is commonly referred to as the *link-blocking case*, which is relevant to electronic switches [4, 13, 14, 20, 21, 23, 26, 28, 43, 47]. A switching network which is strictly nonblocking (SNB) under this general crosstalk constraint shall be referred to as a *c-SNB network*.

This section presents the linear programming analysis first derived in [40] for analyzing strictly nonblocking d -ary multilog networks under this general crosstalk constraint. Prior to [40], the only known results are sufficient conditions for the binary $\log_2(N, 0, m)$ network to be c -SNB [53].

One important advantage of the LP-duality approach is the straightforwardness of verifying the sufficiency proofs. All one has to do is to check that a solution is indeed dual feasible, and the dual-objective value automatically gives us a sufficient condition. Earlier works on this problem relied on combinatorial arguments which are quite intricate and somewhat error-prone.

Section 4.1 shows that the number of planes blocking a new request is bounded by the objective value of a general linear program which is dependent on the parameters of the problem (c, d, n) but independent from the request under consideration. By weak duality, in order to bound the number of planes blocking any new request, the objective value of *any* dual-feasible solution can be used.

In Sects. 4.2 and 4.3, families of solutions which are dual feasible are constructed. Then, depending on the relationships between the problem's parameters (c, d, n), the best dual-feasible solution from the constructed families is selected for bounding the number of blocking planes. Although verifying the LP-based proof is straightforward, constructing these families of dual-feasible solutions involves a good deal of experimentation, educated guesses, and even some Maple coding. This “cost” is hidden from the reader; thus, perhaps some insight is lost along the way as to why the bounds hold.

4.1 The Linear Programming Duality Formulation

Let (\mathbf{a}, \mathbf{b}) be an arbitrary request. For each $i \in \{0, \dots, n - 1\}$, let A_i be the set of inputs \mathbf{x} other than \mathbf{a} , where $\mathbf{x}_{1..n-1}$ shares a *suffix* of length exactly i with $\mathbf{a}_{1..n-1}$. Formally, define

$$A_i := \{\mathbf{x} \in \mathbb{Z}_d^n - \{\mathbf{a}\} \mid \text{SUF}(\mathbf{x}_{1..n-1}, \mathbf{a}_{1..n-1}) = i\}.$$

Similarly, for each $j \in \{0, \dots, n - 1\}$, let B_j be the set of outputs \mathbf{y} other than \mathbf{b} which share a *prefix* of length exactly j with \mathbf{b} , namely,

$$B_j := \{\mathbf{y} \in \mathbb{Z}_d^n - \{\mathbf{b}\} \mid \text{PRE}(\mathbf{y}_{1..n-1}, \mathbf{b}_{1..n-1}) = j\}.$$

Note that $|A_i| = |B_i| = d^{n-i} - d^{n-1-i}$ for all $0 \leq i \leq n - 1$. Suppose the network $\log_d(N, 0, m)$ already had some routes established. Consider a $\text{BY}^{-1}(n)$ -plane which blocks the new request (\mathbf{a}, \mathbf{b}) (i.e., (\mathbf{a}, \mathbf{b}) cannot be routed through the plane). There can only be three cases for which this happens:

Case 1: There is a request (\mathbf{x}, \mathbf{y}) routed in the plane for which $R(\mathbf{x}, \mathbf{y})$ and $R(\mathbf{a}, \mathbf{b})$ share a link. By [Proposition 2](#), it must be the case that $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j \geq n$. Such a request (\mathbf{x}, \mathbf{y}) shall be called a *link-blocking request* with respect to (\mathbf{a}, \mathbf{b}) . The qualifier “with respect to” will be dropped when it is unambiguous from the current context which request is being blocked.

Case 2: There is a request (\mathbf{x}, \mathbf{y}) whose route $R(\mathbf{x}, \mathbf{y})$ already intersects c other routes at c distinct SEs on the same plane, and adding (\mathbf{a}, \mathbf{b}) would introduce an additional intersecting SE to the route $R(\mathbf{x}, \mathbf{y})$. These c other requests shall be called *secondary requests* accompanying (\mathbf{x}, \mathbf{y}) , and the request (\mathbf{x}, \mathbf{y}) shall be called a *node-blocking request of type 1*.

In particular, by [Proposition 1](#), it must be the case that $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j = n - 1$. Note that the common SE between $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ is at stage $j + 1$. The routes for secondary requests must thus intersect $R(\mathbf{x}, \mathbf{y})$ at stages strictly less than $j + 1$ or strictly greater than $j + 1$.

If the route $R(\mathbf{u}, \mathbf{v})$ for the secondary request (\mathbf{u}, \mathbf{v}) intersects $R(\mathbf{x}, \mathbf{y})$ at stage s with $1 \leq s < j + 1$, then it follows that $\text{PRE}(\mathbf{y}, \mathbf{v}) = s - 1 < j$ and $\text{SUF}(\mathbf{x}, \mathbf{u}) = n - 1 - (s - 1) = n - s > n - 1 - j = i$. Hence, $(\mathbf{u}, \mathbf{v}) \in A_i \times B_{s-1}$. Such a request (\mathbf{u}, \mathbf{v}) shall be called a *left secondary request*.

If the route $R(\mathbf{u}, \mathbf{v})$ of secondary request (\mathbf{u}, \mathbf{v}) intersects $R(\mathbf{x}, \mathbf{y})$ at stage s where $j + 1 < s \leq n$, then it follows that $\text{PRE}(\mathbf{y}, \mathbf{v}) = s - 1 > j$ and $\text{SUF}(\mathbf{x}, \mathbf{u}) = n - 1 - (s - 1) = n - s < n - 1 - j = i$. Hence, $(\mathbf{u}, \mathbf{v}) \in A_{n-s} \times B_j$. Such a request (\mathbf{u}, \mathbf{v}) shall be called a *right secondary request*.

The Primal LP

Maximize

$$\sum_{i+j \geq n} x_{ij} + \sum_{i+j=n-1} y_{ij} + \frac{1}{c+1} \sum_{i+j=n-1} z_{ij} \quad (4)$$

Subject to

$$\sum_{j: i+j \geq n} x_{ij} + \sum_{j: i+j=n-1} (y_{ij} + z_{ij}) + \sum_{j: i+j < n-1} (l_{ij} + r_{ij}) \leq d^{n-i} - d^{n-1-i} \forall i \quad (5)$$

$$\sum_{i: i+j \geq n} x_{ij} + \sum_{i: i+j=n-1} (y_{ij} + z_{ij}) + \sum_{i: i+j < n-1} (l_{ij} + r_{ij}) \leq d^{n-j} - d^{n-1-j} \forall j \quad (6)$$

$$cy_{ij} - \left(\sum_{i' < i} r_{i'j} + \sum_{j' < j} l_{ij'} \right) = 0 \quad i+j = n-1 \quad (7)$$

$$l_{ij'} - y_{ij} \leq 0 \quad i+j = n-1, j' < j \quad (8)$$

$$r_{i'j} - y_{ij} \leq 0 \quad i+j = n-1, i' < i \quad (9)$$

$$x_{ij}, y_{ij}, z_{ij}, l_{ij}, r_{ij} \geq 0 \quad \forall i, j \quad (10)$$

Fig. 6 The primal LP for analyzing unicast c -SNB multilog networks

To summarize, there are two types of secondary requests accompanying (\mathbf{x}, \mathbf{y}) : the *left secondary requests* are the requests $(\mathbf{u}, \mathbf{v}) \in A_i \times B_{j'}$ for some $j' < j$, and the *right secondary requests* are the requests $(\mathbf{u}, \mathbf{v}) \in A_{i'} \times B_j$ for some $i' < i$. For each $i' < i$, there is at most one right secondary request in $A_{i'} \times B_j$. Similarly, for each $j' < j$, there is at most one left secondary request in $A_i \times B_{j'}$.

Case 3: There are $c+1$ requests in the plane each of whose routes intersects (\mathbf{a}, \mathbf{b}) at exactly one SE. These will be called *node-blocking requests of type 2*. If (\mathbf{x}, \mathbf{y}) is such a request, then $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i+j = n-1$.

Theorem 2 *The number of blocking planes is the objective value of a feasible solution to the primal linear program as shown in Fig. 6.*

Proof Define the following variables. For each pair i, j such that $i+j \geq n$, let x_{ij} be the number of link-blocking requests in $A_i \times B_j$. For each pair i, j such that $i+j = n-1$, let y_{ij} be the number of node-blocking requests of type 1 and z_{ij} be the number of node-blocking requests of type 2 in $A_i \times B_j$. For each pair i, j such that $i+j < n-1$, let l_{ij} and r_{ij} be the number of left and right secondary requests in $A_i \times B_j$. The number of blocking planes is thus expressed in the objective function (4).

The next step is to verify that the variables satisfy all the constraints. Recall that $|A_i| = d^{n-i} - d^{n-1-i}$, $\forall i$. Thus, the number of requests out of A_i is at most $d^{n-i} - d^{n-1-i}$, justifying constraint (5). Similarly, bounding the number of requests

The Dual LP

Minimize

$$\sum_{i=0}^{n-1} (d^{n-i} - d^{n-1-i}) u_i + \sum_{j=0}^{n-1} (d^{n-j} - d^{n-1-j}) v_j \quad (11)$$

Subject to

$$u_i + v_j \geq 1, \quad i + j \geq n \quad (12)$$

$$u_i + v_j + cw_{ij} - \sum_{i' < i} s_{i'j} - \sum_{j' < j} t_{ij'} \geq 1, \quad i + j = n - 1 \quad (13)$$

$$u_i + v_j \geq \frac{1}{c+1}, \quad i + j = n - 1 \quad (14)$$

$$u_i + v_j - w_{i,n-1-i} + t_{ij} \geq 0, \quad i + j < n - 1 \quad (15)$$

$$u_i + v_j - w_{n-1-j,j} + s_{ij} \geq 0, \quad i + j < n - 1 \quad (16)$$

$$u_i, v_j, s_{ij}, t_{ij} \geq 0, \quad \forall i, j \quad (17)$$

Fig. 7 The dual linear program for analyzing unicast c -SNB multilog networks

to B_j explains constraint (6). Constraint (7) expresses the fact that, for every node-blocking request of type 1 in $A_i \times B_j$ ($i + j = n - 1$), there must be c accompanying secondary requests (left or right). Lastly, for each node-blocking request of type 1 in $A_i \times B_j$, ($i + j = n - 1$) constraint (8) says that for each $j' < j$, there is at most one left secondary request in $A_i \times B_{j'}$, and constraint (9) says that for each $i' < i$, there is at most one right secondary request in $A_{i'} \times B_j$. \square

Remark 3 It should be noted that while the number of blocking planes is the objective value of some feasible solution to the primal LP, the converse may not hold; namely, a feasible solution to the primal LP (even if integral) may not give rise to a blocking configuration with the number of blocking planes equal to the objective value.

The dual linear program is given in Fig. 7. The key idea is the following: due to weak duality, every dual-feasible solution induces an objective value which is at least the number of blocking planes.

As an illustration of the power of the LP-based method, let us first reproduce a few known results. The examples should give the reader the correct insight without delving into too much technicality.

Example 3 ($c = 0$, the node-blocking case) When $c = 0$, the problem becomes the strictly nonblocking problem in the node-blocking sense. It has been shown in [54] (which addressed the node-blocking problem in f -cast switches) that $m \geq d^{\lceil(n-1)/2\rceil} + d^{n-\lceil(n-1)/2\rceil} - 1$ is necessary and sufficient for $\log_d(N, 0, m)$ to be strictly nonblocking.

Let $i_0 = n - \lceil(n-1)/2\rceil$ and $j_0 = \lceil(n-1)/2\rceil$. Assign $u_i = 1$ for all i such that $i_0 \leq i \leq n-1$ and $v_j = 1$ for all j such that $j_0 \leq j \leq n-1$. All other variables are zeros. It is easy to check that this is a dual-feasible solution with objective value precisely $d^{\lceil(n-1)/2\rceil} + d^{n-\lceil(n-1)/2\rceil} - 2$. Thus, at most 1 more BY⁻¹(n) is needed for a total of $d^{\lceil(n-1)/2\rceil} + d^{n-\lceil(n-1)/2\rceil} - 1$ planes in the worst case, matching the known necessary and sufficient condition.

Example 4 ($c = n$, the link-blocking case) When $c = n$, the problem becomes the strictly nonblocking problem in the link-blocking sense. It has been shown in [14, 54] that $m \geq d^{\lceil n/2 \rceil - 1} + d^{\lfloor n/2 \rfloor} - 1$ is necessary and sufficient for $\log_d(N, 0, m)$ to be strictly nonblocking.

Since there can be no path with $n+1$ distinct SEs, the variables y_{ij} and z_{ij} are all zeros, so are the l_{ij} and r_{ij} . The primal LP becomes much simpler:

$$\begin{aligned} \max \quad & \sum_{i+j \geq n} x_{ij} \\ \text{s.t. } & \sum_{j: i+j \geq n} x_{ij} \leq d^{n-i} - d^{n-1-i} \quad \forall i \\ & \sum_{i: i+j \geq n} x_{ij} \leq d^{n-j} - d^{n-1-j} \quad \forall j \\ & x_{ij} \geq 0 \quad \forall i, j \end{aligned}$$

The dual LP is

$$\begin{aligned} \min \quad & \sum_i (d^{n-i} - d^{n-1-i}) u_i + \sum_j (d^{n-j} - d^{n-1-j}) v_j \\ \text{s.t.} \quad & u_i + v_j \geq 1 \quad i + j \geq n \\ & u_i, v_j \geq 0 \quad \forall i, j \end{aligned}$$

Let $i_0 = \lfloor n/2 \rfloor + 1$ and $j_0 = \lceil n/2 \rceil$. Assign $u_i = 1$ for all i such that $i_0 \leq i \leq n-1$ and, $v_j = 1$ for all j such that $j_0 \leq j \leq n-1$. All other variables are zeros. This solution is dual feasible with objective value $d^{\lceil n/2 \rceil - 1} + d^{\lfloor n/2 \rfloor} - 2$. Hence, $m \geq d^{\lceil n/2 \rceil - 1} + d^{\lfloor n/2 \rfloor} - 1$ is sufficient for $\log_d(N, 0, m)$ to be strictly nonblocking in this case. Again, the sufficient condition obtained with the LP-based method matches the known necessary and sufficient condition.

In light of the above examples, in the rest of this section, $1 \leq c \leq n-1$ is assumed.

4.2 The Case When n Is Odd

Lemma 1 Suppose $n = 2k + 1$ and $1 \leq c \leq 2k$. For any integer p where $0 \leq p \leq k$, there exists a feasible solution to the dual LP with objective value

$$2d^k - 2 + \frac{2d^k(d-1)}{c+2} \cdot f(c, d, p),$$

where

$$\begin{aligned} f(c, d, p) &= \sum_{i=0}^p \left(\frac{d}{c+1} \right)^i - \frac{1}{d} \sum_{i=0}^{p-1} \frac{1}{[d(c+1)]^i} \\ &= \begin{cases} p+1 - \frac{1-d^{-2p}}{d-\frac{1}{d}} & c+1 = d \\ \frac{1-\left(\frac{d}{c+1}\right)^{p+1}}{1-\frac{d}{c+1}} - \frac{1-\left(\frac{1}{d(c+1)}\right)^p}{d-\frac{1}{c+1}} & c+1 \neq d. \end{cases} \end{aligned} \quad (18)$$

Proof Consider the following assignment to the dual variables:

$$u_i = v_i = \begin{cases} 1 & k+p+1 \leq i \leq n-1 \\ 1 - \frac{1}{(c+2)(c+1)^{i-k-1}} & k+1 \leq i \leq k+p \\ \frac{1}{(c+2)(c+1)^{k-i}} & k-p \leq i \leq k \\ 0 & 0 \leq i < k-p \end{cases}$$

$$w_{ij} = \min\{u_i, u_j\}, \text{ for all } i+j = n-1$$

$$s_{ij} = t_{ij} = 0, \text{ for all } i+j < n-1.$$

First, it needs to be verified that this assignment is indeed a dual-feasible solution. Note that for $i \leq j$, it holds that $u_i = v_i \leq u_j = v_j$.

Consider the dual constraint (12). When $i+j \geq n = 2k+1$, either $i \geq k+1$ or $j \geq k+1$. Due to symmetry, only $i \geq k+1$ needs to be considered. Note that $v_j \geq v_{n-i}$ because $j \geq n-i$. If $i \geq k+1+p$, then $u_i \geq 1$ and thus $u_i + v_j \geq 1$. If $k+1 \leq i \leq k+p$, then $k-p \leq n-i \leq k$. Thus,

$$u_i + v_j \geq u_i + v_{n-i} = \left(1 - \frac{1}{(c+2)(c+1)^{i-k-1}} \right) + \frac{1}{(c+2)(c+1)^{k-(n-i)}} = 1.$$

Consider the dual constraint (13). When $i+j = n-1 = 2k$, either $i \geq k$ or $j \geq k$. Due to symmetry, only $i \geq k$ needs to be considered. If $i \geq k+p+1$, then $j = 2k-i \leq k-p-1$. Hence, $u_i = 1, v_j = 0, w_{ij} = 0$, and thus the constraint is satisfied. When $k+1 \leq i \leq k+p$, it follows that $k-p \leq j \leq k-1$. Thus,

$$u_i + v_j + cw_{ij} = \left(1 - \frac{1}{(c+2)(c+1)^{i-k-1}} \right) + (c+1) \frac{1}{(c+2)(c+1)^{k-j}} = 1.$$

When $i = j = k$, it follows that $u_i = v_j = w_{ij} = 1/(c+2)$. The constraint is straightforwardly verified.

Constraint (14) is verified similarly. Constraint (15) follows from the fact that $u_i \geq w_{i,n-1-i}$ (and all other variables are nonnegative). Similarly, constraint (16) follows from the fact that $v_j \geq w_{n-1-j,j}$.

Second, it is routine to verify that the objective value of the dual-feasible solution above is precisely $2d^k - 2 + \frac{2d^k(d-1)}{c+2} \cdot f(c, d, p)$. \square

Lemma 2 Suppose $n = 2k + 1$, $k \geq 2$, and $c = k$. There exists a feasible solution to the dual LP with objective value

$$2d^k - 2 + \frac{2(d-1)d^{k-2}}{c+2} \left(d^2 - d + \frac{d^3 - 1}{c+2} \right)$$

Proof The following solution is dual feasible with the desired objective value:

$$\begin{aligned} u_i &= v_i = 1 & k+3 \leq i \leq n-1 \\ u_{k+2} &= v_{k+2} = 1 - \frac{1}{(c+2)^2} \\ u_{k+1} &= v_{k+1} = 1 - \frac{1}{(c+2)} \\ u_k &= v_k = \frac{1}{(c+2)} \\ u_{k-1} &= v_{k-1} = \frac{1}{(c+2)^2} \\ u_i &= v_i = 0 & 0 \leq i \leq k-2 \\ w_{kk} &= \frac{1}{c+2} \\ w_{k+1,k-1} &= w_{k-1,k+1} = \frac{2}{(c+2)^2} \\ w_{k+2,k-2} &= w_{k-2,k+2} = \frac{1}{(c+2)^2} \\ w_{ij} &= 0 & \text{all other } i, j \\ s_{i,k-2} &= t_{k-2,i} = \frac{1}{(c+2)^2} & 0 \leq i \leq k-2 \\ s_{i,k-1} &= t_{k-1,i} = \frac{1}{(c+2)^2} & 0 \leq i \leq k-2 \\ s_{ij} &= t_{ij} = 0 & \text{all other } i, j \end{aligned}$$

Lemma 3 Suppose $n = 2k + 1$ and $k + 1 \leq c \leq 2k$. There exists a feasible solution to the dual LP with objective value

$$2d^k - 2 + \frac{2(d-1)^2 d^{k-1}}{c+2}.$$

Proof The following solution is dual feasible with the desired objective value:

$$u_i = v_i = 1 \quad k+2 \leq i \leq n-1$$

$$\begin{aligned}
u_{k+1} = v_{k+1} &= 1 - \frac{1}{(c+2)} \\
u_k = v_k &= \frac{1}{(c+2)} \\
u_i = v_i &= 0 \quad 0 \leq i \leq k-1 \\
w_{kk} &= \frac{1}{c+2} \\
w_{k+1,k-1} = w_{k-1,k+1} &= \frac{1}{(c+2)} \\
w_{ij} &= 0 \quad \text{all other } i, j \\
s_{i,k-1} = t_{k-1,i} &= \frac{1}{(c+2)} \quad 0 \leq i \leq k-1 \\
s_{ij} = t_{ij} &= 0 \quad \text{all other } i, j
\end{aligned}$$

□

The sufficiency conditions for the odd- n case is now ready to be shown.

Theorem 3 Consider the case when $n = 2k + 1$. Recall that $f(c, d, p)$ is defined in (18).

(a) If $1 \leq c \leq k-1$, then

$$m \geq 2d^k - 1 + \left\lfloor \frac{2d^k(d-1)}{c+2} \cdot f(c, d, \lfloor \log_d(c+1)/2 \rfloor) \right\rfloor$$

is sufficient for $\log_d(N, 0, m)$ to be c -SNB. In particular, when $c \leq \min\{k-1, d^2-1\}$,

$$m \geq 2d^k - 1 + \left\lfloor \frac{2d^k(d-1)}{c+2} \right\rfloor$$

is sufficient.

(b) If $c = k$, then

$$m \geq 2d^k - 1 + \left\lfloor \frac{2(d-1)d^{k-2}}{c+2} \left(d^2 - d + \frac{d^3 - 1}{c+2} \right) \right\rfloor$$

is sufficient for $\log_d(N, 0, m)$ to be c -SNB.

(c) If $k+1 \leq c \leq 2k$, then

$$m \geq 2d^k - 1 + \left\lfloor \frac{2(d-1)^2 d^{k-1}}{c+2} \right\rfloor$$

is sufficient for $\log_d(N, 0, m)$ to be c -SNB.

Proof By Theorem 2, the objective value of any dual-feasible solution is an upperbound on the number of $\text{BY}^{-1}(n)$ -planes blocking a new request. Hence, one plane more than the objective value is sufficient for the network to be c -SNB.

To see (a), [Lemma 1](#) is applied. The best dual-feasible solution is the one whose objective value is minimized. In this case, $f(d - 1, d, p)$ is minimized at $p_0 = \lfloor \log_d(c + 1)/2 \rfloor$. (To see this, notice that $f(c, d, p) \geq f(c, d, p + 1)$ if and only if $p \leq \frac{1}{2} \log_d(c + 1) - 1$.) Thus, one can make use of the dual-feasible solution in [Lemma 1](#) corresponding to $p = p_0$. In particular, when $c \leq d^2 - 1$, the function's minimum value is $f(c, d, \lfloor \log_d(c + 1)/2 \rfloor) = 1$.

Similarly, (b) follows from [Lemma 2](#), and (c) from [Lemma 3](#). □

Remark 4 Compared to the results of [53] (which was only for the binary network case, i.e., $d = 2$), the above sufficient conditions are better when $3 \leq c \leq k - 1$ and are at least as good for other ranges of c .

4.3 The Case When n Is Even

Lemma 4 Suppose $n = 2k$ and $1 \leq c \leq d - 1$. There exists a feasible solution to the dual LP with objective value $2d^k - 2$.

Proof The following solution is dual feasible with the desired objective value: $u_i = v_i = 1$ for all $i \geq k$, all other variables are set to 0. □

Lemma 5 Suppose $n = 2k$ and $d \leq c \leq n - 1$. For any integer p , $1 \leq p \leq k$, there exists a feasible solution to the dual LP with objective value

$$d^k + d^{k-1} - 2 + \frac{(d - 1)d^{k-1}}{c + 2} \cdot (d + g(c, d, p)),$$

where

$$\begin{aligned} g(c, d, p) &= \sum_{i=0}^p \left(\frac{d}{c+1} \right)^i - \sum_{i=0}^{p-1} \frac{1}{[d(c+1)]^i} \\ &= \begin{cases} p + 1 - \frac{1-d^{-2p}}{1-d^{-2}} & c+1=d \\ \frac{1-\left(\frac{d}{c+1}\right)^{p+1}}{1-\frac{d}{c+1}} - \frac{1-\left(\frac{1}{d(c+1)}\right)^p}{1-\frac{1}{d(c+1)}} & c+1 \neq d. \end{cases} \end{aligned} \quad (19)$$

Proof The following solution is dual feasible with the desired objective value:

$$u_i = 1, \text{ for } k + 1 \leq i \leq n - 1,$$

$$u_i = \frac{1}{(c+2)(c+1)^{k-i}}, \text{ for } k - p \leq i \leq k,$$

$$u_i = 0, \text{ for all other } i,$$

$$v_j = 1, \text{ for } k + p \leq j \leq n - 1,$$

$$v_j = 1 - u_{n-j} = 1 - u_{2k-j}, \text{ for } k \leq j \leq k + p - 1,$$

$$v_{k-1} = \frac{1}{c+2},$$

$$v_j = 0, \text{ for all other } j,$$

$$w_{ij} = \min\{u_i, u_j\}, \text{ when } i + j = n - 1.$$

$$s_{ij} = t_{ij} = 0, \text{ for all } i + j < n - 1.$$

□

Remark 5 The previous two Lemmas hold for any c between 1; and $n - 1$ however they were stated for the range of c where they are meant to be applied.

Lemma 6 Suppose $n = 2k$ and $k < c$. There exists a feasible solution to the dual LP with objective value $d^k + d^{k-1} - 2$.

Proof The following solution is dual feasible with the desired objective value:

$$u_i = v_i = 1 \quad k + 1 \leq i \leq n - 1$$

$$u_k = v_k = \frac{1}{2}$$

$$u_i = v_i = 0 \quad 0 \leq i \leq k - 1$$

$$w_{k,k-1} = w_{k-1,k} = 1/2$$

$$w_{ij} = 0 \quad \text{all other } i, j$$

$$s_{i,k-1} = t_{k-1,i} = 1/2 \quad 0 \leq i \leq k - 1$$

$$s_{ij} = t_{ij} = 0 \quad \text{all other } i, j$$

□

With the help of [Lemmas 4–6](#), the following theorem straightforwardly follows:

Theorem 4 Consider the case when $n = 2k$. Recall that $g(c, d, p)$ is defined in [\(19\)](#).

- (a) If $1 \leq c \leq d - 1$, then $m \geq 2d^k - 1$ is sufficient for $\log_d(N, 0, m)$ to be c -SNB.
- (b) If $d \leq c \leq k$, then

$$m \geq d^k + d^{k-1} - 1 + \left\lfloor \frac{(d-1)d^{k-1}}{c+2} \cdot \left(d + g \left(c, d, \left\lfloor \frac{\log_d(c+1)+1}{2} \right\rfloor \right) \right) \right\rfloor$$

is sufficient for $\log_d(N, 0, m)$ to be c -SNB. In particular, when $d \leq c \leq \min\{k, d^3 - 2\}$

$$m \geq d^k + d^{k-1} - 1 + \left\lfloor \frac{(d-1)d^k}{c+1} \right\rfloor$$

is sufficient.

- (c) If $k < c \leq 2k$, then $m \geq d^k + d^{k-1} - 1$ is sufficient for $\log_d(N, 0, m)$ to be c -SNB.

Remark 6 Compared to the results of [53] (which was only for the binary network case, i.e., $d = 2$), the above sufficient conditions are better when $7 \leq c \leq k$ and are at least as good for other ranges of c .

5 Strictly Nonblocking f -Casting Multilog Networks Under the General Crosstalk Constraint

This section derives conditions under which a $\log_d(N, 0, m)$ network is f -cast c -SNB. These results are from [42]. The problem setting is very general and thus challenging in many ways: (1) f -cast covers both unicast ($f = 1$) and multicast/broadcast ($f = N$), (2) d -ary networks are much more general than the commonly considered binary multilog networks, and (3) the crosstalk constraint covers both the usual link-blocking case ($c = n$) and the crosstalk-free case ($c = 0$). Prior to [42], sufficient conditions for the multilog networks to be unicast c -SNB were derived in [40, 53]. In the f -cast strictly nonblocking case, necessary and sufficient conditions are known for both link-blocking and crosstalk-free constraints [14, 54]. As shall be seen, most of these known results are corollaries of the results in this section. The same LP-duality analysis presented in Sect. 4 is deployed. The f -cast case is considerably more complex, however.

The main idea is similar to that in the previous section. Consider a multicast request (\mathbf{a}, B) , where \mathbf{a} is an input and B is a subset of outputs. For each output $\mathbf{b} \in B$, the branch (\mathbf{a}, \mathbf{b}) can be routed independently from all other branches of the request. The number α of $\text{BY}^{-1}(n)$ -planes through which (\mathbf{a}, \mathbf{b}) cannot be routed through (those are called “blocking planes”) can be expressed as the objective value of a feasible solution to a linear program. Thus, by weak duality, α is upperbounded by the objective value β of *any* feasible solution to the dual LP. Then, $m \geq \beta + 1$ planes will be sufficient to route (\mathbf{a}, \mathbf{b}) . Again, this approach makes sufficiency proofs very easy to verify since they only involve checking if a solution is indeed dual feasible. Of course, there is considerable effort “behind the scene” to discover good dual-feasible solutions because the linear programs are very general with many parameters (n, c, f , and d) and variables.

Let (\mathbf{a}, \mathbf{b}) be an arbitrary branch of a multicast request. For each $i \in \{0, \dots, n-1\}$, let A_i be the set of inputs \mathbf{x} other than \mathbf{a} , where $\mathbf{a}_{1..n-1}$ shares a *suffix* of length exactly i with $\mathbf{x}_{1..n-1}$. Formally, define

$$A_i := \{\mathbf{x} \in \mathbb{Z}_d^n - \{\mathbf{a}\} \mid \text{SUF}(\mathbf{x}_{1..n-1}, \mathbf{a}_{1..n-1}) = i\}.$$

Similarly, for each $j \in \{0, \dots, n - 1\}$, let B_j be the set of outputs other than \mathbf{b} which share a prefix of length exactly j with \mathbf{b} , namely,

$$B_j := \{\mathbf{y} \in \mathbb{Z}_d^n - \{\mathbf{b}\} \mid \text{PRE}(\mathbf{y}_{1..n-1}, \mathbf{b}_{1..n-1}) = j\}.$$

Note that $|A_i| = |B_i| = d^{n-i} - d^{n-1-i}$, for all $0 \leq i \leq n - 1$. Suppose the network $\log_d(N, 0, m)$ already had some (f -cast) routes established. Consider a $\text{BY}^{-1}(n)$ plane which blocks (\mathbf{a}, \mathbf{b}) . There can only be three cases for which this happens:

Case 1: There is a branch (\mathbf{x}, \mathbf{y}) of some f -cast request routed in the plane for which $R(\mathbf{x}, \mathbf{y})$ and $R(\mathbf{a}, \mathbf{b})$ share a link. By [Proposition 2](#), $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j \geq n$. Such a branch (\mathbf{x}, \mathbf{y}) shall be referred to as a *link-blocking branch*.

Case 2: There is a branch (\mathbf{x}, \mathbf{y}) of some f -cast request whose route $R(\mathbf{x}, \mathbf{y})$ already intersects c other routes at c distinct SEs on the same plane, and adding (\mathbf{a}, \mathbf{b}) would introduce an additional intersecting SE to the route $R(\mathbf{x}, \mathbf{y})$. These c other branches are called *secondary branches* accompanying (\mathbf{x}, \mathbf{y}) , and the branch (\mathbf{x}, \mathbf{y}) is called a *node-blocking branch of type 1*.

By [Proposition 1](#), $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j = n - 1$. Moreover, the common SE between $R(\mathbf{a}, \mathbf{b})$ and $R(\mathbf{x}, \mathbf{y})$ is at stage $j + 1$. The routes for secondary branches must thus intersect $R(\mathbf{x}, \mathbf{y})$ at stages strictly less than $j + 1$ or strictly greater than $j + 1$.

If a secondary branch (\mathbf{u}, \mathbf{v}) has its route intersects $R(\mathbf{x}, \mathbf{y})$ at stage $1 \leq s < j + 1$, it follows that $\text{PRE}(\mathbf{y}, \mathbf{v}) = s - 1 < j$ and $\text{SUF}(\mathbf{x}, \mathbf{u}) = n - 1 - (s - 1) = n - s > n - 1 - j = i$. Hence, $(\mathbf{u}, \mathbf{v}) \in A_i \times B_{s-1}$. Such branch (\mathbf{u}, \mathbf{v}) is called a *left secondary branch*.

If a secondary branch (\mathbf{u}, \mathbf{v}) has its route intersects $R(\mathbf{x}, \mathbf{y})$ at stage $j + 1 < s \leq n$, it follows that $\text{PRE}(\mathbf{y}, \mathbf{v}) = s - 1 > j$ and $\text{SUF}(\mathbf{x}, \mathbf{u}) = n - 1 - (s - 1) = n - s < n - 1 - j = i$. Hence, $(\mathbf{u}, \mathbf{v}) \in A_{n-s} \times B_j$. Such branch (\mathbf{u}, \mathbf{v}) is called a *right secondary branch*.

To summarize, there are two types of secondary branches accompanying (\mathbf{x}, \mathbf{y}) : the *left secondary branches* are the branches $(\mathbf{u}, \mathbf{v}) \in A_i \times B_{j'}$ for some $j' < j$, and the *right secondary branches* are the branches $(\mathbf{u}, \mathbf{v}) \in A_{i'} \times B_j$ for some $i' < i$. For each $i' < i$, there is at most one right secondary branch in $A_{i'} \times B_j$. Similarly, for each $j' < j$, there is at most one left secondary branch in $A_i \times B_{j'}$.

Case 3: There are $c + 1$ branches (of some f -cast requests) in the plane each of whose routes intersects (\mathbf{a}, \mathbf{b}) at exactly one SE at a distinct stage. These will be called *node-blocking requests of type 2*. If (\mathbf{x}, \mathbf{y}) is such a request, then $(\mathbf{x}, \mathbf{y}) \in A_i \times B_j$ for some $i + j = n - 1$. Moreover, two node-blocking branches of type 2 belonging to the same $A_i \times B_j$ must belong to different blocking planes.

Theorem 5 *The number of blocking planes for (\mathbf{a}, \mathbf{b}) is the objective value of a feasible solution to the primal linear program as shown in [Fig. 8](#).*

The Primal LP

Maximize

$$\sum_{i+j \geq n} x_{ij} + \sum_{i+j=n-1} y_{ij} + \frac{1}{c+1} \sum_{i+j=n-1} z_{ij} \quad (20)$$

Subject to

$$\sum_{j: i+j \geq n} x_{ij} + \sum_{j: i+j=n-1} (y_{ij} + z_{ij}) + \sum_{j: i+j < n-1} (l_{ij} + r_{ij}) \leq f(d^{n-i} - d^{n-1-i}) \quad \forall i \quad (21)$$

$$\sum_{i: i+j \geq n} x_{ij} + \sum_{i: i+j=n-1} (y_{ij} + z_{ij}) + \sum_{i: i+j < n-1} (l_{ij} + r_{ij}) \leq d^{n-j} - d^{n-1-j} \quad \forall j \quad (22)$$

$$z_{ij} \leq \frac{1}{c+1} \sum_{i'+j'=n-1} z_{i'j'} \quad i+j = n-1 \quad (23)$$

$$cy_{ij} = \left(\sum_{i' < i} r_{i'j} + \sum_{j' < j} l_{ij'} \right) \quad i+j = n-1 \quad (24)$$

$$l_{ij'} \leq y_{ij} \quad i+j = n-1, j' < j \quad (25)$$

$$r_{i'j} \leq y_{ij} \quad i+j = n-1, i' < i \quad (26)$$

$$x_{ij}, y_{ij}, z_{ij}, l_{ij}, r_{ij}, p_i \geq 0 \quad \forall i, j \quad (27)$$

Fig. 8 The dual linear programs for f -cast c -SNB

Proof For each pair $i + j \geq n$, let x_{ij} be the number of link-blocking branches in $A_i \times B_j$. For each $i + j = n - 1$, let y_{ij} be the number of node-blocking branches of type 1 and z_{ij} be the number of node-blocking branches of type 2 in $A_i \times B_j$. For each pair $i + j < n - 1$, let l_{ij} and r_{ij} be the number of left and right secondary branches in $A_i \times B_j$. The number of blocking planes is thus expressed in the objective function (20).

The next step is to verify that the variables satisfy all the constraints. Recall that $|A_i| = d^{n-i} - d^{n-1-i}$, $\forall i$. Thus, the number of branches of f -cast requests out of A_i is at most $f(d^{n-i} - d^{n-1-i})$, justifying constraint (21). Similarly, bounding the number of branches of requests into B_j explains constraint (22). Since all node-blocking branches of type 2 belonging to the same $A_i \times B_j$ must belong to different blocking planes, z_{ij} is at most the number of blocking planes involving these type-2 branches. This is constraint (23). Constraint (24) expresses the fact that, for every node-blocking request of type 1 in $A_i \times B_j$ ($i + j = n - 1$), there must be c accompanying secondary requests (left or right). Lastly, for each node-blocking request of type 1 in $A_i \times B_j$ ($i + j = n - 1$), constraint (24) says that for each $j' < j$, there is at most one left secondary request in $A_i \times B_{j'}$, and constraint (26) says that for each $i' < i$, there is at most one right secondary request in $A_{i'} \times B_j$. \square

The dual of the primal LP is shown in Fig. 9. To get a sense of how to select a good dual-feasible solution, let us reproduce a few known results. The following

The Dual LP

Minimize

$$\sum_{i=0}^{n-1} f(d^{n-i} - d^{n-1-i})u_i + \sum_{j=0}^{n-1} (d^{n-j} - d^{n-1-j})v_j \quad (28)$$

Subject to

$$u_i + v_j \geq 1, \quad i + j \geq n \quad (29)$$

$$u_i + v_j + cw_{ij} - \sum_{i' < i} s_{i'j} - \sum_{j' < j} t_{ij'} \geq 1, \quad i + j = n - 1 \quad (30)$$

$$u_i + v_j + \frac{c}{c+1} p_i - \frac{1}{c+1} \sum_{i' \neq i} p_{i'} \geq \frac{1}{c+1}, \quad i + j = n - 1 \quad (31)$$

$$u_i + v_j - w_{i,n-1-i} + t_{ij} \geq 0, \quad i + j < n - 1 \quad (32)$$

$$u_i + v_j - w_{n-1-j,j} + s_{ij} \geq 0, \quad i + j < n - 1 \quad (33)$$

$$u_i, v_j, s_{ij}, t_{ij} \geq 0, \quad \forall i, j \quad (34)$$

Fig. 9 The dual linear programs for f -cast c -SNB

corollaries should give the reader the correct insight without delving into too much technicality.

The following condition has been shown in [54] to be necessary and sufficient, with a fairly involved combinatorial argument.

Corollary 1 (The crosstalk-free case) *For $\log_d(N, 0, m)$ to be f -cast crosstalk-free strictly nonblocking, it is sufficient that*

$$m \geq f(d^{\lceil \frac{n-r-1}{2} \rceil} - 1) + d^{n-\lceil \frac{n-r-1}{2} \rceil}$$

where $r = \lfloor \log_d f \rfloor$.

Proof Being crosstalk-free, strictly nonblocking is the same as being 0-SNB (i.e., $c = 0$). Let $b = \lceil \frac{n-r-1}{2} \rceil$ and $a = n - b$. Consider the following assignment to the dual variables: $u_i = 1, \forall i \geq a, v_j = 1, \forall j \geq b$, and all other variables are 0. This solution is dual feasible with objective value

$$f(d^{\lceil \frac{n-r-1}{2} \rceil} - 1) + d^{n-\lceil \frac{n-r-1}{2} \rceil} - 1,$$

which is an upperbound on the number of blocking planes. One more plane is needed for the network to be strictly nonblocking, which completes the proof. \square

The following condition has been shown in [14, 54] to be necessary and sufficient for the link-blocking case using a complex combinatorial argument. The proof presented here is much simpler to verify.

Corollary 2 (The link-blocking case) *Define*

$$m_{lb}(n, f, d) := f(d^{\lceil \frac{n-r-2}{2} \rceil} - 1) + d^{n-\lceil \frac{n-r}{2} \rceil}.$$

For $\log_d(N, 0, m)$ to be f -cast strictly nonblocking in the link-blocking sense, it is sufficient that

$$m \geq \begin{cases} m_{lb}(n, f, d) & f < d^{n-2} \\ m_{lb}(n, d^{n-2}, d) & f \geq d^{n-2} \end{cases}$$

Proof First, suppose $f < d^{n-2}$. Let $b = \lceil \frac{n-r}{2} \rceil$ and $a = n + 1 - b$. Consider the following assignment to the dual variables: $u_i = 1, \forall i \geq a, v_j = 1, \forall j \geq b, w_{a-1,n-a} = 1, p_{a-1} = 1/c, s_{i,b-1} = t_{a-1,j} = 1$ when $i < a-1, j < b-1$, and all other variables are 0. This solution is dual feasible with objective value precisely $m_{lb}(n, f, d) - 1$. Second, when $f \geq d^{n-2}$, assign all $v_j = 1, j > 0, w_{n-1,0} = 1, p_{n-1} = 1/c, s_{i,0} = 1, \forall i < n-1$, and all other variables to be 0. \square

Henceforth, consider the case when $1 \leq c \leq n-1$. In the next three lemmas, several general dual-feasible solutions are derived. Then, depending on the relative relationships between f, n, c , and d , one of the lemmas is applied to get the best sufficient condition.

Lemma 7 *Suppose $1 \leq c \leq n-1$. Let a, b, p, q be any integers satisfying $0 \leq a \leq n-1, b = n-1-a, 0 \leq p \leq b, 0 \leq q \leq a$. Define $\delta = \frac{1}{d(c+1)}$ and $\lambda = \frac{d}{c+1}$. Then, there exists a dual-feasible solution with objective value*

$$f(d^b - 1) + (d^a - 1) - \frac{(d-1)[fd^b(1-\delta^p) + d^a(1-\delta^q)]}{d(c+2)(1-\delta)} + \frac{(d-1)}{c+2} \left[fd^b \frac{1-\lambda^{q+1}}{1-\lambda} + d^a \frac{1-\lambda^{p+1}}{1-\lambda} \right].$$

The following convention is made: for any integer k , $\frac{(1-\lambda^k)}{(1-\lambda)} = k$ when $\lambda = 1$.

Proof Consider the following assignment to the dual variables:

$$u_i = \begin{cases} 0 & 0 \leq i \leq a-q-1 \\ \frac{1}{(c+2)(c+1)^{a-i}} & a-q \leq i \leq a \\ 1 - \frac{1}{(c+2)(c+1)^{i-(a+1)}} & a+1 \leq i \leq a+p \\ 1 & a+p+1 \leq i \leq n-1 \end{cases}$$

$$v_j = \begin{cases} 0 & 0 \leq i \leq b-p-1 \\ \frac{1}{(c+2)(c+1)^{b-j}} & b-p \leq j \leq b \\ 1 - \frac{1}{(c+2)(c+1)^{j-(b+1)}} & b+1 \leq j \leq b+q \\ 1 & b+q+1 \leq j \leq n-1 \end{cases}$$

$$w_{ij} = \begin{cases} \min\{u_i, u_j\} & i+j = n-1 \\ 0 & \text{otherwise} \end{cases}$$

All other variables are 0.

The next step is to verify that this solution is indeed dual feasible. Constraints (29) and (31) can be verified straightforwardly. Constraints (32) and (33) follow from the fact that $u_i \geq w_{i,n-1-i}$, and $v_j \geq w_{n-1-j,j}$.

Next, consider the dual constraint (30). When $i \leq a-q-1$ or $i \geq a+p+1$, it holds that $v_j = 1$ or $u_i = 1$ (recall $i+j = n-1$). The constraint is satisfied because in these cases $w_{ij} = 0$. When $i = a$, $u_i + v_j + cw_{ij} = 1$. When $a+1 \leq i = a+k \leq a+p$, it follows that

$$\begin{aligned} u_i + v_j + cw_{ij} &= 1 - \frac{1}{(c+2)(c+1)^{k-1}} + \frac{1}{(c+2)(c+1)^k} + \frac{c}{(c+2)(c+1)^k} \\ &= 1 \end{aligned}$$

When $a-q \leq i = a-k \leq a-1$, the constraint is verified similarly. \square

Proposition 3 Given $a, b = n-1-a, q$, and a feasible solution as assigned as in Lemma 7, the dual objective value is minimized at

$$p_0 = \max \left\{ 0, \min \left\{ \left\lfloor \frac{1}{2} \log_d(f(c+1)) + \frac{n-1}{2} - a \right\rfloor, b \right\} \right\}$$

Proof To see this, it will be shown that $g(a, b, p, q) \geq g(a, b, p+1, q)$ iff $p \leq \frac{1}{2} \log_d(f(c+1)) + \frac{n-3}{2} - a$ where $g(a, b, q, p)$ is the dual-objective value of feasible solution as assigned as in Lemma 7. First,

$$\begin{aligned} g(a, b, p, q) &= \sum_{i=a-q}^a f(d^{n-i} - d^{n-1-i}) \frac{1}{(c+2)(c+1)^{a-i}} \\ &\quad + \sum_{i=a+1}^{a+p} f(d^{n-i} - d^{n-1-i}) \left(1 - \frac{1}{(c+2)(c+1)^{i-(a+1)}} \right) \\ &\quad + \sum_{j=b-p}^b (d^{n-j} - d^{n-1-j}) \frac{1}{(c+2)(c+1)^{b-j}} \end{aligned}$$

$$\begin{aligned}
& + \sum_{j=b+1}^{b+q} (d^{n-j} - d^{n-1-j})(1 - \frac{1}{(c+2)(c+1)^{j-(b+1)}}) \\
& + \sum_{i=a+p+1}^{n-1} f(d^{n-i} - d^{n-1-i}) + \sum_{j=b+q+1}^{n-1} (d^{n-j} - d^{n-1-j}) \\
g(a, b, p+1, q) = & \sum_{i=a-q}^a f(d^{n-i} - d^{n-1-i}) \frac{1}{(c+2)(c+1)^{a-i}} \\
& + \sum_{i=a+1}^{a+p+1} f(d^{n-i} - d^{n-1-i})(1 - \frac{1}{(c+2)(c+1)^{i-(a+1)}}) \\
& + \sum_{j=b-p-1}^b (d^{n-j} - d^{n-1-j}) \frac{1}{(c+2)(c+1)^{b-j}} \\
& + \sum_{j=b+1}^{b+q} (d^{n-j} - d^{n-1-j})(1 - \frac{1}{(c+2)(c+1)^{j-(b+1)}}) \\
& + \sum_{i=a+p+2}^{n-1} f(d^{n-i} - d^{n-1-i}) + \sum_{j=b+q+1}^{n-1} (d^{n-j} - d^{n-1-j})
\end{aligned}$$

Second,

$$\begin{aligned}
g(a, b, p+1, q) - g(a, b, p, q) = & f(d^{n-(a+p+1)} - d^{n-1-(a+p+1)}) \\
& \times (1 - \frac{1}{(c+2)(c+1)^p}) - f(d^{n-(a+p+1)} \\
& - d^{n-1-(a+p+1)}) \\
& + (d^{n-(b-p-1)} - d^{n-1-(b-p-1)}) \frac{1}{(c+2)(c+1)^{p+1}}
\end{aligned}$$

It is easy to see that $g(a, b, p, q) \geq g(a, b+1, p, q)$ if and only if

$$\begin{aligned}
& (d^{n-(b-p-1)} - d^{n-1-(b-p-1)}) \frac{1}{(c+2)(c+1)^{p+1}} \leq f(d^{n-(a+p+1)} \\
& - d^{n-1-(a+p+1)}) \frac{1}{(c+2)(c+1)^p}
\end{aligned}$$

By dividing both sides by $(d-1)$ and some simple cancelations, the following inequality is obtained:

$$d^{n-1-(b-p-1)} \leq f(c+1)d^{n-1-(a+p+1)}.$$

Now, taking logarithms of both sides, it is easy to obtain

$$p \leq \frac{1}{2} \log_d(f(c+1)) + \frac{n-1}{2} - a - 1.$$

As p is an integer and p must satisfy $0 \leq p \leq b$, the function $g(a, b, p, q)$ is minimized at

$$p_0 = \max \left\{ 0, \min \left\{ \left\lfloor \frac{1}{2} \log_d(f(c+1)) + \frac{n-1}{2} - a \right\rfloor, b \right\} \right\}.$$

□

Proposition 4 Given $a, b = n - 1 - a$, p , and a feasible solution as assigned as in [Lemma 7](#), the dual-objective value is minimized at

$$q_0 = \max \left\{ 0, \min \left\{ \left\lfloor \frac{1}{2} \log_d \frac{c+1}{f} + \bar{a} - \frac{n-1}{2} \right\rfloor, a \right\} \right\}$$

Proof Similar to the proof for [Proposition 3](#), one can prove that $g(a, b, p, q) \geq g(a, b, p, q+1)$ if and only if

$$q \leq \frac{1}{2} \log_d \frac{c+1}{f} + \bar{a} - \frac{n-1}{2} - 1.$$

□

Lemma 8 Suppose $c \geq \frac{n}{2} + 1$. Let a be any integer where $n - c + 1 \leq a \leq c - 1$ and x be any real number in the interval $\left[\frac{1}{c-a+1}, \frac{c+a-n}{c+a-n+1} \right]$. Then, there exists a feasible solution to the dual LP with objective value

$$f(d^{n-a-2} - 1) + (d^{a-1} - 1) + (d - 1)d^{n-a+1}x(f - d^{2a-n}).$$

Proof Consider the following assignment to the dual variables:

$$u_a = x,$$

$$v_{n-a} = 1 - x,$$

$$u_i = v_j = 1, \text{ when } i > a + 1 \text{ and } j > n - a + 1,$$

$$w_{a,n-1-a} = s_{i,n-1-a} = \frac{1}{c-a+1}, \text{ when } 0 \leq i \leq a-1,$$

$$w_{a-1,n-1} = t_{a-1,j} = \frac{1}{c+a-n+1}, \text{ when } 0 \leq j \leq n-a-1.$$

All other variables are 0. \square

Lemma 9 Suppose $c \geq \frac{n}{2} + 1$. Let a be any integer where $a > c - 1$. There exists a dual-feasible solution with objective value

$$f(d^{n-a-1} - 1) + (d^a - 1) + \frac{1}{c+2} d^{a-1} (d-1)^2 (fd^{n-2a-1} + 1).$$

Proof Consider the following assignment to the dual variables:

$$\begin{aligned} u_a &= v_{n-1-a} = \frac{1}{c+2}, \\ u_{a+1} &= v_{n-a} = \frac{c+1}{c+2}, \\ u_i &= v_j = 1, \text{ when } i > a, j > n-a, \\ w_{a-1,n-a} = t_{a_1,j} &= \frac{1}{(c+2)(c+a+1-n)} \text{ where } j \leq n-a-2, \\ w_a &= \frac{1}{c+2}, \\ w_{a+1} = s_{i,n-a-2} &= \frac{1}{(c+2)(c-a)}, \text{ when } i \leq a-1. \end{aligned}$$

All other variables are 0. \square

Theorem 6 Consider the case when $c \leq \frac{n}{2}$. Let $\bar{a} = \lfloor \frac{n+r}{2} \rfloor$, $\bar{b} = n-1-\bar{a}$, $\bar{p} = p_0$, $\bar{q} = q_0$, $\delta = \frac{1}{d(c+1)}$, and $\lambda = \frac{d}{c+1}$. Then,

$$\begin{aligned} m \geq f(d^{\bar{b}} - 1) + d^{\bar{a}} - &\frac{(d-1) \left[fd^{\bar{b}}(1-\delta^{\bar{p}}) + d^{\bar{a}}(1-\delta^{\bar{q}}) \right]}{d(c+2)(1-\delta)} \\ &+ \frac{(d-1)}{c+2} \left[fd^{\bar{b}} \frac{1-\lambda^{\bar{q}+1}}{1-\lambda} + d^{\bar{a}} \frac{1-\lambda^{\bar{p}+1}}{1-\lambda} \right] \end{aligned}$$

is sufficient for $\log_d(N, 0, m)$ to be f -cast c -SNB. (Again, set $\frac{(1-\lambda^k)}{(1-\lambda)} = k$ when $\lambda = 1$.)

Remark 7 The results for $c \leq n/2$ in [40, 53] are direct consequences of this theorem for $f = 1$.

Applying [Lemmas 8](#) and [9](#) appropriately, the following theorem is obtained. Again, the results for $c > n/2$ in [\[40, 53\]](#) are consequences of this theorem for $f = 1$.

Theorem 7 Consider the case when $c \geq \frac{n}{2} + 1$. Set $\bar{a} = \lceil (n+r)/2 \rceil$. Note that in this case, $n - c + 1 \leq \bar{a}$.

(i) If $\bar{a} \leq c - 1$ and $n + r$ is even, then

$$m \geq f(d^{n-\bar{a}-1} - 1) + d^{\bar{a}} + \frac{(d-1)d^{n-\bar{a}+1}(f - d^{2\bar{a}-n})}{c - \bar{a} + 1}$$

is sufficient for the network to be f -cast c -SNB.

(ii) If $\bar{a} \leq c - 1$ and $n + r$ is odd, then

$$m \geq f(d^{n-\bar{a}-1} - 1) + d^{\bar{a}} + \frac{(d-1)d^{n-\bar{a}+1}(f - d^{2\bar{a}-n})(c + \bar{a} - n)}{c + \bar{a} - n + 1}$$

is sufficient for the network to be f -cast c -SNB.

(iii) If $\bar{a} > c - 1$, then

$$m \geq f(d^{n-\bar{a}} - 1) + d^{\bar{a}-1} + \frac{d^{\bar{a}-2}(d-1)^2(fd^{n-2\bar{a}+1} + 1)}{c + 2}$$

is sufficient for the network to be f -cast c -SNB.

Proof (i) Apply [Lemma 8](#) with $x = \frac{1}{c-\bar{a}+1}$.

(ii) Apply [Lemma 8](#) with $x = \frac{c+\bar{a}-n}{c+\bar{a}-n+1}$.

(ii) Apply [Lemma 9](#) with $a = \bar{a}$.

□

6 Analyzing f -Cast Wide-Sense Nonblocking Multilog Networks

Let f, t be given integers with $0 \leq t \leq n$ and $1 \leq f \leq N = d^n$. This section analyzes f -cast wide-sense nonblocking $\log_d(N, 0, m)$ networks under the *window algorithm* with window size d^t . The algorithm was proposed and analyzed for one window size $d^{\lfloor n/2 \rfloor}$ in [\[49\]](#) and later analyzed more carefully for varying window sizes in [\[10\]](#). Both papers considered the multicast case with no fanout restriction. A more general theorem for the f -cast case will be derived in this section:

- *The window algorithm with window size d^t :* Given any integer t , $0 \leq t \leq n$, divide the outputs into “windows” of size d^t each. Each window consists of all outputs sharing a prefix of length $n - t$ for a total of d^{n-t} windows. Denote the windows by W_w , $0 \leq w \leq d^{n-t} - 1$. Given a new multicast request (\mathbf{a}, B) , where \mathbf{a} is an input and B is a subset of outputs, the routing rule is, for every $0 \leq w \leq d^{n-t} - 1$, the subrequest $(\mathbf{a}, B \cap W_w)$ is routed entirely on one

single $\text{BY}^{-1}(n)$ -plane. (Different subrequests can be routed through the same or different $\text{BY}^{-1}(n)$ -planes.)

Remark 8 There is a subtle point about the window algorithm due to which the original authors in [49] thought their multilog network was strictly nonblocking instead of wide-sense nonblocking. Basically, for some specific values of the parameters, the algorithm is *no* algorithm at all. In those cases, any sufficient condition for the network to be nonblocking under the window algorithm is in fact a strictly nonblocking condition, not a wide-sense nonblocking condition.

For example, in the unicast case, it holds that $f = 1$, which means the window algorithm does not specify any routing strategy; consequently, any nonblocking condition is actually a strictly nonblocking condition. Another example is when $t = 0$. In this case, the routing rule says that each branch of a (multicast) request should be routed on some plane, independent of other branches. Because there is no restriction on how to route the branches, any nonblocking condition is a strictly nonblocking one.

Yet another example is when $t = n$. Here, the routing rule is for each request to be routed entirely on some plane. If the $1 \times m$ -SE stage of the multilog network has fanout capability, then the rule *does* restrict how requests are routed, and thus the setting is indeed a wide-sense nonblocking setting. However, if the $1 \times m$ -SE stage is implemented with $1 \times m$ -unicast crossbars or $1 \times m$ -demultiplexers, then an algorithm following the rule *has to* route each request entirely on some plane. Thus, any sufficient condition is a strictly nonblocking condition.

6.1 Setting Up the Linear Program and Its Dual

Let (\mathbf{a}, B) be an arbitrary f -cast request to be routed using the window algorithm with window size d^t . Following the window algorithm, due to symmetry without loss of generality, it can be assumed that $B = \{\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(k)}\}$ where all the outputs $\mathbf{b}^{(l)}$ ($l \in [k]$) belong to the same window W_0 and $k \leq \min\{f, d^t\}$. The $\mathbf{b}^{(l)}$ thus share a common prefix of length $n - t$. (This is because subrequests to the same window are routed through the same plane and different subrequests of the same request are routed independently from each other and they do not block one another.)

For each $i \in \{0, \dots, n - 1\}$, let A_i be the set of inputs \mathbf{u} other than \mathbf{a} , where $\mathbf{u}_{1..n-1}$ shares a *suffix* of length exactly i with $\mathbf{a}_{1..n-1}$. Formally, define

$$A_i := \{\mathbf{u} \in \mathbb{Z}_d^n - \{\mathbf{a}\} \mid \text{SUF}(\mathbf{u}_{1..n-1}, \mathbf{a}_{1..n-1}) = i\}.$$

For each $j \in \{0, \dots, n - 1\}$, let B_j be the set of outputs other than those in B which share a *prefix* of length exactly j with *some* member of B , namely,

$$B_j := \left\{ \mathbf{v} \in \mathbb{Z}_d^n - B \mid \exists l \in [k], \text{PRE}(\mathbf{v}_{1..n-1}, \mathbf{b}_{1..n-1}^{(l)}) = j \right\}.$$

Note that

$$\begin{aligned}|A_i| &= d^{n-i} - d^{n-1-i}, 0 \leq i \leq n-1, \\ |B_j| &= d^{n-j} - d^{n-1-j}, 0 \leq j \leq n-t-1.\end{aligned}$$

Define $\mathcal{A} = \bigcup_{i=0}^{n-1} A_i$. It is easy to see that

$$\begin{aligned}\bigcup_{j=0}^{n-t-1} B_j &= \bigcup_{w=1}^{d^{n-t}-1} W_w \\ \bigcup_{j=n-t}^{n-1} B_j &= W_0 - B.\end{aligned}$$

Furthermore, for each $j \leq n-t-1$, B_j is the disjoint union of precisely $d^{n-j-t} - d^{n-1-j-t}$ windows each of size d^t .

Note that the sets B_j for $0 \leq j \leq n-t-1$ are mutually disjoint. On the other hand, the sets B_j for $n-t \leq j \leq n-1$ are not necessarily disjoint, because for the same output $\mathbf{v} \in W_0 - B$, it might be the case that $\text{PRE}(\mathbf{v}_{1..n-1}, \mathbf{b}_{1..n-1}^{(l)}) = j$ and $\text{PRE}(\mathbf{v}_{1..n-1}, \mathbf{b}_{1..n-1}^{(l')}) = j'$ for $j \neq j'$, $l \neq l'$. The following simple observation turns out to be an important analytical detail in many of the proofs.

Proposition 5 *Let q be an integer such that $n-t \leq q \leq n-1$. Then,*

$$\left| \bigcup_{j=q}^{n-1} B_j \right| \leq \min\{d^t - k, k(d^{n-q} - 1)\}$$

and

$$\left| \bigcup_{j=n-t}^{n-1} B_j \right| = d^t - k.$$

Proof To see the inequality, note that $\left| \bigcup_{j=q}^{n-1} B_j \right|$ counts the number of strings \mathbf{v} in $W_0 - B$ for which $\text{PRE}(\mathbf{v}_{1..n-1}, \mathbf{b}_{1..n-1}^{(l)}) \geq q$ for some $\mathbf{b}^{(l)}$, $l \in [k]$. As $|W_0| = d^t$, the upperbound $d^t - k$ for the number of such strings is trivial. On the other hand, the number of strings \mathbf{v} where $\text{PRE}(\mathbf{v}_{1..n-1}, \mathbf{b}_{1..n-1}^{(l)}) \geq q$ for a fixed string $\mathbf{b}^{(l)}$ is at most $d^{n-q} - 1$ (discounting $\mathbf{b}^{(l)}$ itself). Hence, the upperbound $k(d^{n-q} - 1)$ is obtained via a simple application of the union bound. The equality trivially holds because $\bigcup_{j=n-t}^{n-1} B_j = W_0 - B$. \square

For every input $\mathbf{u} \in \mathcal{A}$, let $i(\mathbf{u})$ denote the index i such that $\mathbf{u} \in A_i$. For every $w \in [d^{n-t} - 1] = \{1, \dots, d^{n-t} - 1\}$, let $j(w)$ be the index j such that $W_w \subseteq B_j$. For every $\mathbf{v} \in W_0 - B$, let $j(\mathbf{v})$ denote the largest j for which $\mathbf{v} \in B_j$. Note that $j(\mathbf{v}) \geq n - t$ for such output \mathbf{v} because $W_0 - B = \bigcup_{j=n-t}^{n-1} B_j$.

Lemma 10 *For each input $\mathbf{u} \in \mathcal{A}$ and each $w \in [d^{n-t} - 1]$ such that $i(\mathbf{u}) + j(w) \geq n$, define a variable $x_{\mathbf{u},w}$. Also, for each input $\mathbf{u} \in \mathcal{A}$ and each output $\mathbf{v} \in W_0 - B$ such that $i(\mathbf{u}) + j(\mathbf{v}) \geq n$, define a variable $x_{\mathbf{u},\mathbf{v}}$. Then, the number of Banyan planes blocking the new multicast request (\mathbf{a}, B) is upperbounded by the optimal value of the following linear program:*

$$\begin{aligned} \max \quad & \sum_{\mathbf{u}, w} x_{\mathbf{u}, w} + \sum_{\mathbf{u}, \mathbf{v}} x_{\mathbf{u}, \mathbf{v}} \\ \text{s.t.} \quad & \sum_{\mathbf{u}} x_{\mathbf{u}, w} \leq d^t \quad w \in [d^{n-t} - 1] \\ & x_{\mathbf{u}, w} \leq 1 \quad \forall \mathbf{u}, w \\ & \sum_{\mathbf{v}} x_{\mathbf{u}, \mathbf{v}} \leq 1 \quad \forall \mathbf{u} \in \mathcal{A} \\ & \sum_{\mathbf{u}} x_{\mathbf{u}, \mathbf{v}} \leq 1 \quad \forall \mathbf{v} \in W_0 - B \\ & \sum_w x_{\mathbf{u}, w} + \sum_{\mathbf{v}} x_{\mathbf{u}, \mathbf{v}} \leq f \quad \forall \mathbf{u} \in \mathcal{A} \\ & x_{\mathbf{u}, w}, x_{\mathbf{u}, \mathbf{v}} \geq \mathbf{0} \quad \forall \mathbf{u}, w, \mathbf{v} \end{aligned} \tag{35}$$

Obviously, the sums and the constraints only range over values for which the variables are defined.

Proof Suppose the network $\log_d(N, 0, m)$ already had some routes established. Consider a $\text{BY}^{-1}(n)$ -plane which blocks the new request (\mathbf{a}, B) . There must be one route $R(\mathbf{u}, \mathbf{v})$ on this plane for which $R(\mathbf{u}, \mathbf{v})$ and $R(\mathbf{a}, \mathbf{b}^{(l)})$ share a link for some $l \in [k]$. Note that the branch $R(\mathbf{u}, \mathbf{v})$ could be part of a multicast tree from input \mathbf{u} , but only an arbitrary blocking branch (\mathbf{u}, \mathbf{v}) of this tree is needed. Note also that $\mathbf{u} \neq \mathbf{a}$ because subrequests from the same input are parts of the same request and thus their routes do not block one another. Let S be the set constructed by arbitrarily taking exactly one blocking branch (\mathbf{u}, \mathbf{v}) per blocking plane. Then, the number of blocking planes is $|S|$.

Fact 1 If (\mathbf{u}, \mathbf{v}) and $(\mathbf{u}, \mathbf{v}')$ are both in S , then \mathbf{v} and \mathbf{v}' must belong to different windows because, if they belong to the same window, the window algorithm would have routed them through the same plane and S only contains one branch per blocking plane.

Fact 2 Each output \mathbf{v} can only appear once in S because each output can only be part of at most one existing request.

Fact 3 If $(\mathbf{u}, \mathbf{v}) \in S$, then $(\mathbf{u}, \mathbf{v}) \in A_i \times B_j$ for some $i + j \geq n$, thanks to Proposition 2.

Straightforwardly, it will be shown that S defines a feasible solution to the linear program with objective value precisely $|S|$. Set $x_{\mathbf{u},w} = 1$ if there is some $(\mathbf{u}, w) \in S$ such that $w \in W_w$ and $x_{\mathbf{u},v} = 1$ if there is some $(\mathbf{u}, v) \in S$ such that $v \in W_0 - B$. All other variables are set to 0. Due to Fact 3, the procedure does not set value for an undefined variable. Certainly $|S|$ is equal to the objective value of this solution.

The next step is to verify that the solution satisfies all the constraints. The first constraint expresses the fact that each output in a window W_w of size d^t only appears at most once in S (Fact 2). The second and third constraints are a restatement of Fact 1. Note that the sum in the third constraint is only over $v \in W_0 - B$. The fourth constraint says that each output $v \in W_0 - B$ appears at most once in S (Fact 2 again). The fifth constraint says that each input can only be part of at most f members of S due to the f -cast nature of the network. \square

The dual linear program can be written as follows:

$$\begin{aligned} \min & \sum_w d^t \alpha_w + \sum_{\mathbf{u},w} \beta_{\mathbf{u},w} + \sum_{\mathbf{u}} \gamma_{\mathbf{u}} + \sum_v \delta_v + \sum_{\mathbf{u}} f \epsilon_{\mathbf{u}} \\ \text{s.t. } & \alpha_w + \beta_{\mathbf{u},w} + \epsilon_{\mathbf{u}} \geq 1, \quad x_{\mathbf{u},w} \text{ defined (DC-1)} \\ & \gamma_{\mathbf{u}} + \delta_v + \epsilon_{\mathbf{u}} \geq 1, \quad x_{\mathbf{u},v} \text{ defined (DC-2)} \\ & \alpha_w, \beta_{\mathbf{u},w}, \gamma_{\mathbf{u}}, \delta_v, \epsilon_{\mathbf{u}} \geq 0 \quad \forall \mathbf{u}, \mathbf{v}, w \end{aligned} \quad (36)$$

Note that the dual constraints only exist over all $\mathbf{u}, \mathbf{v}, w$ for which $x_{\mathbf{u},w}$ and $x_{\mathbf{u},v}$ are defined; in particular, they exist for pairs (\mathbf{u}, w) such that $i(\mathbf{u}) + j(w) \geq n$ and pairs (\mathbf{u}, \mathbf{v}) such that $i(\mathbf{u}) + j(\mathbf{v}) \geq n$.

6.2 Specifying a Family of Dual-Feasible Solutions

To illustrate the technique, let us first derive a couple of known results “for free.” The first is Theorem III.2 in [54].

Corollary 3 (Theorem III.2 in [54]) *Let $r = \lfloor \log_d f \rfloor$. Suppose the $1 \times m$ -SE stage of the $\log_d(N, 0, m)$ network does not have fanout capability, then when $f \leq d^{n-2}$, the network is f -cast strictly nonblocking if*

$$m \geq d^{\lfloor \frac{n+r}{2} \rfloor} + f \left(d^{\lceil \frac{n-r-2}{2} \rceil} - 1 \right).$$

When $f > d^{n-2}$, the network is f -cast strictly nonblocking if $m \geq d^{n-1}$.

Proof Recall Remark 8: routing using the window algorithm with window size $t = n$ is the same as routing arbitrarily in the network when the $1 \times m$ -SE stage cannot fanout. Thus, any sufficient condition for the window algorithm to work is a strictly nonblocking condition. Note that when $t = n$, the dual constraints (DC-1) do not exist! A feasible solution to the dual linear program is constructed as follows:

When $f > d^{n-2}$, set $\gamma_{\mathbf{u}} = 1$ for all $\mathbf{u} \in \bigcup_{i=1}^{n-1} A_i$ and all other variables to be 0. The dual-objective value in this case is

$$\sum_{\mathbf{u} \in \bigcup_{i=1}^{n-1} A_i} \gamma_{\mathbf{u}} = \sum_{i=1}^{n-1} |A_i| = \sum_{i=1}^{n-1} (d^{n-i} - d^{n-i-1}) = d^{n-1} - 1,$$

and hence one more plane (i.e., $m \geq d^{n-1}$) is sufficient. Note that this solution is dual feasible, because for $\mathbf{u} \in \mathbf{A}_0$, there is no \mathbf{v} for which $i(\mathbf{u}) + j(\mathbf{v}) \geq n$. In other words, there is no dual constraint for which $\mathbf{u} \in A_0$.

Next, suppose $f \leq d^{n-2}$. Define $q = \lfloor \frac{n+r}{2} \rfloor + 1$. Note that $r + 1 \leq q \leq n - 1$ in this case; in particular,

$$kd^{n-q} < d^{r+1}d^{n-q} \leq d^n,$$

which implies

$$\min\{d^n - k, k(d^{n-q} - 1)\} = kd^{n-q}.$$

Set

$$\gamma_{\mathbf{u}} = \begin{cases} 1 & \text{if } i(\mathbf{u}) \geq n - q + 1 \\ 0 & \text{otherwise} \end{cases} \quad (37)$$

and

$$\delta_{\mathbf{v}} = \begin{cases} 1 & \text{if } \mathbf{v} \in \bigcup_{j=q}^{n-1} B_j \\ 0 & \text{otherwise.} \end{cases} \quad (38)$$

All other dual variables are 0. The solution is dual feasible because, for any pair (\mathbf{u}, \mathbf{v}) for which $i(\mathbf{u}) + j(\mathbf{v}) \geq n$, it must either be the case that $i(\mathbf{u}) \geq n - q + 1$ or that $j(\mathbf{v}) \geq q$ (which is the same as saying $\mathbf{v} \in \bigcup_{j=q}^{n-1} B_j$). Recalling [Proposition 5](#), the dual-objective value is

$$\begin{aligned} \sum_{\mathbf{u} : i(\mathbf{u}) \geq n - q + 1} \gamma_{\mathbf{u}} + \sum_{\mathbf{v} \in \bigcup_{j=q}^{n-1} B_j} \delta_{\mathbf{v}} &= \sum_{i=n-q+1}^{n-1} |A_i| + \left| \bigcup_{j=q}^{n-1} B_j \right| \\ &\leq d^{q-1} - 1 + \min\{d^n - k, k(d^{n-q} - 1)\} \\ &= d^{q-1} - 1 + k(d^{n-q} - 1) \\ &\leq d^{q-1} + f(d^{n-q} - 1) - 1. \end{aligned}$$

This is an upperbound on the number of blocking planes. Hence, one more plane is sufficient to route the new (arbitrary) request. \square

Because unicast is 1-cast, by setting $r = 0$ in the previous corollary, the following corollary is obtained, whose proof was about 5 pages long in [20]. Recall

Remark 8 which ensures that our result is a strictly nonblocking condition rather than a wide-sense nonblocking one.

Corollary 4 (Theorem 1 in [20]) *For $\log_d(N, 0, m)$ to be unicast strictly non-blocking, it is sufficient that $m \geq d^{\lceil n/2 \rceil - 1} + d^{\lfloor n/2 \rfloor} - 1$.*

Corollary 3 solves the $t = n$ case. Henceforth, $0 \leq t < n$ can be assumed. The next step is to specify a family of dual-feasible solutions to the dual LP (36). The main remaining task will be simple calculus as the best dual-feasible solution is selected, depending on the parameters f, n, d, t of the problem.

The family of dual-feasible solution is specified with two integral parameters where $0 \leq p \leq n - t - 1$ and $n - t \leq q \leq n$. The parameter p is used to set the variables $\epsilon_{\mathbf{u}}$, α_w and $\beta_{\mathbf{u},w}$, and the parameter q is used to set the variables $\gamma_{\mathbf{u}}$ and $\delta_{\mathbf{v}}$. As the variables are set, the feasibility of the constraints (DC-1) and (DC-2) are also verified, and the contributions of those variables to the final objective value are computed:

- *Specifying the $\epsilon_{\mathbf{u}}$ variables.* The $\epsilon_{\mathbf{u}}$ are defined as follows:

$$\epsilon_{\mathbf{u}} = \begin{cases} 1 & i(\mathbf{u}) \geq n - p \\ 0 & \text{otherwise} \end{cases} \quad (39)$$

The contribution of the $\epsilon_{\mathbf{u}}$ to the objective is

$$\sum_{\mathbf{u}} f \epsilon_{\mathbf{u}} = \sum_{i=n-p}^{n-1} f \sum_{\mathbf{u}:i(\mathbf{u})=i} 1 = \sum_{i=n-p}^{n-1} f |A_i| = f(d^p - 1).$$

- *Specifying the α_w and $\beta_{\mathbf{u},w}$ variables.* Next, define the α_w and $\beta_{\mathbf{u},w}$ as follows. The constraints (DC-1) with $i(\mathbf{u}) \geq n - p$ are already satisfied by the $\epsilon_{\mathbf{u}}$; hence, the α_w and $\beta_{\mathbf{u},w}$ only need to be set to satisfy (DC-1) when $j(w) \geq p + 1$. (If $j(w) \leq p$, then for the constraint to exist, it must hold that $i(\mathbf{u}) \geq n - j(w) \geq n - p$.) The variables α_w and $\beta_{\mathbf{u},w}$ are set differently based on three cases as follows:

Case 1. If $t \geq \lfloor \frac{n}{2} \rfloor$, then set

$$\beta_{\mathbf{u},w} = \begin{cases} 1 & p + 1 \leq j(w) \leq n - t - 1, \text{ and } n - j(w) \leq i(\mathbf{u}) \leq n - p - 1, \\ 0 & \text{otherwise} \end{cases} \quad (40)$$

and set all the α_w to 0. It can be verified straightforwardly that all constraints (DC-1) are satisfied. Recall that the number of windows W_w for which $j(w) = j$ is precisely $d^{n-j-t} - d^{n-j-t-1}$. Thus, the contributions of the α_w and $\beta_{\mathbf{u},w}$ to the dual-objective value are

$$\begin{aligned}
\sum_{\substack{\mathbf{u}, w \\ p+1 \leq j(w) \leq n-t-1 \\ n-j(w) \leq i(\mathbf{u}) \leq n-p-1}} \beta_{\mathbf{u}, w} &= \sum_{j=p+1}^{n-t-1} |\{w : j(w) = j\}| \sum_{i=n-j}^{n-p-1} |\{\mathbf{u} \in \mathcal{A} : i(\mathbf{u}) = i\}| \\
&= \sum_{j=p+1}^{n-t-1} (d^{n-j-t} - d^{n-j-t-1}) \sum_{i=n-j}^{n-p-1} |A_i| \\
&= \sum_{j=p+1}^{n-t-1} (d^{n-j-t} - d^{n-j-t-1})(d^j - d^p) \\
&= (n-t-1-p)(d^{n-t} - d^{n-t-1}) - d^{n-t-1} + d^p.
\end{aligned}$$

Case 2. When $p+1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$, set

$$\beta_{\mathbf{u}, w} = \begin{cases} 1 & p < j(w) \leq t \text{ and } n - j(w) \leq i < n - p \\ 0 & \text{otherwise} \end{cases}$$

and

$$\alpha_w = \begin{cases} 1 & t < j < n-t \\ 0 & \text{otherwise} \end{cases}$$

All constraints (DC-1) are thus satisfied. The α_w 's and $\beta_{\mathbf{u}, w}$'s contributions to the objective is

$$\begin{aligned}
&\sum_{\substack{w \\ t < j(w) < n-t}} d^t \alpha_w + \sum_{\substack{\mathbf{u}, w \\ p+1 \leq j(w) \leq t \\ n-j(w) \leq i(\mathbf{u}) \leq n-p-1}} \beta_{\mathbf{u}, w} \\
&= \sum_{j=t+1}^{n-t-1} d^t \cdot |\{w : j(w) = j\}| + \sum_{j=p+1}^t |\{w : j(w) = j\}| \\
&\quad \sum_{i=n-j}^{n-p-1} |\{\mathbf{u} \in \mathcal{A} : i(\mathbf{u}) = i\}| \\
&= \sum_{j=t+1}^{n-t-1} d^t (d^{n-j-t} - d^{n-j-t-1}) + \sum_{j=p+1}^t (d^j - d^p)(d^{n-j-t} - d^{n-j-t-1}) \\
&= (t-p)(d^{n-t} - d^{n-t-1}) + d^{n+p-2t-1} - d^t.
\end{aligned}$$

Case 3. When $t \leq p$ (which is $\leq n - t - 1$), set

$$\alpha_w = \begin{cases} 1 & p + 1 \leq j(w) \leq n - t - 1 \\ 0 & \text{otherwise} \end{cases} \quad (41)$$

and all the $\beta_{\mathbf{u}, w}$ to be zero. Again, the feasibility of the constraints (DC-1) is easy to verify. The contribution to the objective value is

$$\begin{aligned} \sum_{\substack{w \\ p < j(w) < n-t}} d^t \alpha_w &= \sum_{j=p+1}^{n-t-1} d^t \cdot |\{w : j(w) = j\}| \\ &= \sum_{j=p+1}^{n-t-1} d^t (d^{n-j-t} - d^{n-j-t-1}) \\ &= d^{n-p-1} - d^t. \end{aligned}$$

- *Specifying the $\gamma_{\mathbf{u}}$ and $\delta_{\mathbf{v}}$ variables.* Here, there are two cases.

When $q = n - t$, set $\delta_{\mathbf{v}} = 1$ for all $\mathbf{v} \in \bigcup_{j=n-t}^{n-1} B_j$ and all $\gamma_{\mathbf{u}} = 0$. The dual-objective contribution in this case is

$$\sum_{\mathbf{v} \in \bigcup_{j=n-t}^{n-1} B_j} \delta_{\mathbf{v}} = \left| \bigcup_{j=n-t}^{n-1} B_j \right| = d^t - k.$$

When $n - t + 1 \leq q \leq n$, define

$$\delta_{\mathbf{v}} = \begin{cases} 1 & \mathbf{v} \in \bigcup_{j=q}^{n-1} B_j \\ 0 & \text{otherwise} \end{cases} \quad (42)$$

and

$$\gamma_{\mathbf{u}} = \begin{cases} 1 & n - q + 1 \leq i(\mathbf{u}) \leq n - p - 1 \\ 0 & \text{otherwise} \end{cases} \quad (43)$$

From [Proposition 5](#), the total contribution of the $\gamma_{\mathbf{u}}$ and $\delta_{\mathbf{v}}$ to the dual objective is at most

$$\begin{aligned} \sum_{\substack{\mathbf{u} \\ n-q+1 \leq i(\mathbf{u}) \leq n-p-1}} \gamma_{\mathbf{u}} + \sum_{\mathbf{v} \in \bigcup_{j=q}^{n-1} B_j} \delta_{\mathbf{v}} &= \sum_{i=n-q+1}^{n-p-1} |\{\mathbf{u} : i(\mathbf{u}) = i\}| + \left| \bigcup_{j=q}^{n-1} B_j \right| \\ &\leq \sum_{i=n-q+1}^{n-p-1} |A_i| + \min\{d^t - k, k(d^{n-q} - 1)\} \\ &= d^{q-1} - d^p + \min\{d^t - k, k(d^{n-q} - 1)\}. \end{aligned}$$

The objective value $c(k, p, q)$
--

For $t \geq \lfloor \frac{n}{2} \rfloor$ and $q = n - t$,

$$c(k, p, q) = f(d^p - 1) + (n - t - 1 - p)(d^{n-t} - d^{n-t-1}) - d^{n-t-1} + d^p + d^t - k.$$

For $t \geq \lfloor \frac{n}{2} \rfloor$ and $q > n - t$,

$$\begin{aligned} c(k, p, q) = & f(d^p - 1) + (n - t - 1 - p)(d^{n-t} - d^{n-t-1}) - d^{n-t-1} + d^{q-1} \\ & + \min\{d^t - k, k(d^{n-q} - 1)\}. \end{aligned}$$

For $p + 1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$ and $q = n - t$

$$c(k, p, q) = f(d^p - 1) + (t - p)(d^{n-t} - d^{n-t-1}) + d^{n+p-2t-1} - k.$$

For $p + 1 \leq t \leq \lfloor \frac{n}{2} \rfloor - 1$ and $q > n - t$

$$\begin{aligned} c(k, p, q) = & f(d^p - 1) + (t - p)(d^{n-t} - d^{n-t-1}) + d^{n+p-2t-1} - d^t + d^{q-1} - d^p \\ & + \min\{d^t - k, k(d^{n-q} - 1)\}. \end{aligned}$$

For $t \leq p$ and $q = n - t$,

$$c(k, p, q) = f(d^p - 1) + d^{n-p-1} - k.$$

For $t \leq p$ and $q > n - t$,

$$c(k, p, q) = f(d^p - 1) + d^{n-p-1} - d^t + d^{q-1} - d^p + \min\{d^t - k, k(d^{n-q} - 1)\}.$$

Fig. 10 The dual-objective value of the family of dual-feasible solutions

The feasibility of all the constraints (DC-2) is easy to verify.

Define the “cost” $c(k, p, q)$ to be the total contribution of all variables to the dual-objective value. The values of $c(k, p, q)$ are summarized in Fig. 10. The following has just been proved:

Theorem 8 *The above family of solutions is feasible for the dual linear program (36) with objective value equal to $c(k, p, q)$. Consequently, for the network $\log_d(N, 0, m)$ to be wide-sense nonblocking under the window algorithm with window size d^t , it is sufficient that*

$$m \geq 1 + \max_{1 \leq k \leq \min(f, d^t)} \min_{p,q} c(k, p, q). \quad (44)$$

6.3 Selecting the Best Dual-Feasible Solution

It is a very straightforward though somewhat analytically tedious task to derive the best possible sufficient condition using Theorem 8. The idea is as follows. For a given $k \leq \min(f, d^t)$, p and q are set to be $p = p_k, q = q_k$ so that $c(k, p_k, q_k)$ is

as small as possible. Then, an upperbound $C(t, f) \geq \max_k c(k, p_k, q_k)$ is derived. Finally, the sufficient condition is $m \geq C(t, f) + 1$.

A technical lemma is in order.

Lemma 11 *Let d, n, k be positive integers and $x = \lfloor \log_d k \rfloor$. Then, the following function*

$$h(k) = d^{\lfloor \frac{n+x}{2} \rfloor} + k \left(d^{n-\lfloor \frac{n+x}{2} \rfloor-1} - 1 \right) \quad (45)$$

is nondecreasing in k .

Proof The lemma is proved by induction on k . The inequality trivially holds when $k = 1$. Consider $k > 2$. First, suppose k is not an exact power of d , i.e., $k > d^x$. In this case, it follows that

$$\begin{aligned} h(k-1) &= d^{\lfloor \frac{n+x}{2} \rfloor} + (k-1) \left(d^{n-\lfloor \frac{n+x}{2} \rfloor-1} - 1 \right) \leq d^{\lfloor \frac{n+x}{2} \rfloor} \\ &\quad + k \left(d^{n-\lfloor \frac{n+x}{2} \rfloor-1} - 1 \right) = h(k). \end{aligned}$$

Second, consider the case when $k = d^x$. It can be verified that, no matter what the parities of n and x are, the multiset $\{\lfloor \frac{n+x-1}{2} \rfloor, \lceil \frac{n+x-1}{2} \rceil\}$ is exactly equal to the multiset $\{\lfloor \frac{n+x}{2} \rfloor, \lceil \frac{n+x}{2} \rceil - 1\}$. Thus, noting that $\lfloor \log_d (k-1) \rfloor = x-1$, it follows that

$$\begin{aligned} h(k-1) &= d^{\lfloor \frac{n+x-1}{2} \rfloor} + (k-1) \left(d^{n-\lfloor \frac{n+x-1}{2} \rfloor-1} - 1 \right) \\ &= d^{\lfloor \frac{n+x-1}{2} \rfloor} + (d^x - 1) \left(d^{\lceil \frac{n-x-1}{2} \rceil} - 1 \right) \\ &= d^{\lfloor \frac{n+x-1}{2} \rfloor} + d^{\lceil \frac{n+x-1}{2} \rceil} - d^{\lceil \frac{n-x-1}{2} \rceil} - d^x + 1 \\ &= d^{\lfloor \frac{n+x}{2} \rfloor} + d^{\lceil \frac{n+x}{2} \rceil-1} - d^{\lceil \frac{n-x-1}{2} \rceil} - d^x + 1 \\ &\leq d^{\lfloor \frac{n+x}{2} \rfloor} + d^{\lceil \frac{n+x}{2} \rceil-1} - d^x \\ &= d^{\lfloor \frac{n+x}{2} \rfloor} + d^x \left(d^{n-\lfloor \frac{n+x}{2} \rfloor-1} - 1 \right) \\ &= h(k). \end{aligned}$$

□

Theorem 9 *The $\log_d(N, 0, m)$ network is nonblocking under the window algorithm with window size d^t if $m \geq 1 + C(t, f)$ where $C(t, f)$ is defined in Fig. 11.*

Proof Consider 5 cases in the definition of $C(t, f)$. For each k , the values p_k and q_k will be specified, and then, $c(k, p_k, q_k) \leq C(t, f)$ is verified.

- *Case 1:* $t < \lfloor \frac{n}{2} \rfloor, r \leq n - 2t - 1$. For any k , choose $p_k = \lceil \frac{n-r}{2} - 1 \rceil$ and $q_k = n - t$. Noting that $p_k \geq t$ and $k \geq 1$, the following holds

The upperbound $C(t, f)$

To shorten the notations, let $r = \lfloor \log_d f \rfloor$.

$$C(t, f) = \begin{cases} f \left(d^{\lceil \frac{n-r}{2} \rceil - 1} - 1 \right) + d^{n-\lceil \frac{n-r}{2} \rceil} - 1 & t < \lfloor \frac{n}{2} \rfloor, r \leq n - 2t - 1 \\ t(d-1)d^{n-t-1} + d^{n-2t-1} - 1 & t < \lfloor \frac{n}{2} \rfloor, r \geq n - 2t \\ [(n-t-1)(d-1)-1]d^{n-t-1} + d^t \\ -(d-1)d^{2t-n-1} & t \geq \lfloor \frac{n}{2} \rfloor, r \geq n - t \\ f(d^{n-t-r-1}-1) + [r(d-1)-1]d^{n-t-1} \\ + d^{n-t-r-1} + d^t - (d-1)d^{2t-n-1} & t \geq \lfloor \frac{n}{2} \rfloor, \text{ and} \\ 2t-n-2 < r \leq n-t-1 \\ f(d^{n-t-r-1}-1) + [r(d-1)-1]d^{n-t-1} \\ + d^{\lfloor \frac{n+r}{2} \rfloor} + f\left(d^{n-\lfloor \frac{n+r}{2} \rfloor-1}-1\right) & t \geq \lfloor \frac{n}{2} \rfloor, \text{ and} \\ r \leq \min(2t-n-2, n-t-1) \end{cases}$$

Fig. 11 Theorem 9 shows that $C(t, f) \geq \max_k \min_{p,q} c(k, p, q)$

$$c(k, p_k, q_k) = f \left(d^{\lceil \frac{n-r}{2} \rceil - 1} - 1 \right) + d^{n-\lceil \frac{n-r}{2} \rceil} - k \leq C(t, f).$$

- Case 2: $t < \lfloor \frac{n}{2} \rfloor, r \geq n - 2t$. For any k , set $p_k = 0$ and $q_k = n - t$. Then,

$$\begin{aligned} c(k, p_k, q_k) &= t(d^{n-t} - d^{n-t-1}) + d^{n-2t-1} - k \leq t(d^{n-t} - d^{n-t-1}) \\ &\quad + d^{n-2t-1} - 1 = C(t, f). \end{aligned}$$

- Case 3: $t \geq \lfloor \frac{n}{2} \rfloor, r \geq n - t$. This case is a little trickier analytically. Define $x = \lfloor \log_d k \rfloor$. The values p_k and q_k are set differently depending on how large x is so that the inequality $c(k, p_k, q_k) \leq C(t, f)$ always holds.

If $0 \leq x \leq 2t - n - 2$, which can only hold when $t \geq \frac{n+1}{2}$, then set $q_k = \lfloor \frac{n+x}{2} \rfloor + 1$ and $p_k = 0$. Note that $q_k > n - t$ and $x + 1 + n - q_k < t$. Thus, $kd^{n-q_k} < d^t$. Recall from Lemma 11 that function $h(k)$ defined in (45) is nonincreasing, and the fact that in this case $k \leq d^{x+1} - 1 \leq d^{2t-n-1} - 1$, it follows that

$$\begin{aligned} c(k, p_k, q_k) &= [(n-t-1)(d-1)-1]d^{n-t-1} + d^{q_k-1} \\ &\quad + \min\{d^t - k, k(d^{n-q_k} - 1)\} \\ &= [(n-t-1)(d-1)-1]d^{n-t-1} + d^{q_k-1} + k(d^{n-q_k} - 1) \\ &= [(n-t-1)(d-1)-1]d^{n-t-1} + h(k) \\ &\leq [(n-t-1)(d-1)-1]d^{n-t-1} + h(d^{2t-n-1} - 1) \end{aligned}$$

$$\begin{aligned}
&= [(n-t-1)(d-1)-1]d^{n-t-1} + d^{t-1} \\
&\quad + (d^{2t-n-1}-1)(d^{n-t}-1) \\
&< [(n-t-1)(d-1)-1]d^{n-t-1} + d^{t-1} \\
&\quad + d^{2t-n-1}(d-1)(d^{n-t}-1) \\
&= [(n-t-1)(d-1)-1]d^{n-t-1} + d^t - (d-1)d^{2t-n-1} \\
&= C(t, f).
\end{aligned}$$

If $x = 2t - n - 1$ and $k \leq d^{x+1} - d^x$, then set $q_k = \lfloor \frac{n+x}{2} \rfloor + 1 = t$ and $p_k = 0$. Note that $q_k = t > n - t$ because $x \geq 0$. In this case,

$$\begin{aligned}
c(k, p_k, q_k) &= [(n-t-1)(d-1)-1]d^{n-t-1} + h(k) \\
&\leq [(n-t-1)(d-1)-1]d^{n-t-1} + h(d^{2t-n} - d^{2t-n-1}) \\
&= [(n-t-1)(d-1)-1]d^{n-t-1} + d^{t-1} \\
&\quad + (d^{2t-n} - d^{2t-n-1})(d^{n-t}-1) \\
&= [(n-t-1)(d-1)-1]d^{n-t-1} + d^t - (d-1)d^{2t-n-1} \\
&= C(t, f).
\end{aligned}$$

If $x = 2t - n - 1$ and $k \geq d^{x+1} - d^x + 1$, then set $q_k = n - t$ and $p_k = 0$. Then,

$$\begin{aligned}
c(k, p_k, q_k) &= [(n-t-1)(d-1)-1]d^{n-t-1} + 1 + d^t - k \\
&\leq [(n-t-1)(d-1)-1]d^{n-t-1} + 1 + d^t - d^{x+1} + d^x - 1 \\
&= [(n-t-1)(d-1)-1]d^{n-t-1} + d^t - (d-1)d^{2t-n-1} \\
&= C(t, f).
\end{aligned}$$

Finally, when $x \geq 2t - n$, again set $q_k = n - t$ and $p_k = 0$. Note that $k \geq d^x \geq d^{2t-n} > (d-1)d^{2t-n-1}$. Thus, $d^t - k < d^t - (d-1)d^{2t-n-1} - 1$ and $c(k, p_k, q_k) \leq C(t, f)$ follow straightforwardly.

- *Case 4:* $t \geq \lfloor \frac{n}{2} \rfloor$, $2t - n - 2 < r \leq n - t - 1$. Note that this case can only happen when $t \leq 2n/3$. In particular, if $t > 2n/3$ and $r \leq n - t - 1$, then case 5 applies. Set $p_k = n - t - r - 1$ and $q_k = \lfloor \frac{n+x}{2} \rfloor + 1$. Proving $c(k, p_k, q_k) \leq C(t, f)$ is almost identical to Case 3 where one considers different ranges of $x = \lfloor \log_d k \rfloor$.
- *Case 5:* $t \geq \lfloor \frac{n}{2} \rfloor$, $r \leq \min(2t - n - 2, n - t - 1)$. Set $p_k = n - t - r - 1$ and $q_k = \lfloor \frac{n+x}{2} \rfloor + 1$. Showing $c(k, p_k, q_k) \leq C(t, f)$ is similar to Case 3. The only slight variation is, instead of bounding $k \leq d^{x+1} - 1$, the relation $k \leq f$ is applied directly. The function $h(k)$ is then bounded by $h(f)$. Furthermore, it is not necessary to consider the cases when $x \geq 2t - n - 1$ because $x \leq r \leq 2t - n - 2$. \square

6.4 Some Quick Consequences of Theorem 9

By plugging in the parameters t and f and computing $1 + C(t, f)$, the following results follow straightforwardly.

Corollary 5 (Theorem 4 in [14]) *Let $r = \lfloor \log_d f \rfloor$. The network $\log_d(N, 0, m)$ is f -cast strictly non-blocking if*

$$m \geq f \left(d^{\lceil \frac{n-r}{2} \rceil - 1} - 1 \right) + d^{n - \lceil \frac{n-r}{2} \rceil}.$$

Proof This corresponds to the $t = 0$ case of the window algorithm, which becomes a strictly nonblocking condition as noted earlier:

$$C(0, f) = f \left(d^{\lceil \frac{n-r}{2} \rceil - 1} - 1 \right) + d^{n - \lceil \frac{n-r}{2} \rceil} - 1. \quad \square$$

The following result took about 6 pages in [10] to be proved (in two theorems) with combinatorial reasoning. The result is on the general multicast case, without the fanout restriction f . In the current setting, one can simply set $f = N = d^n$. In fact, even though the corollary states exactly the same results as in [10], the statement is simpler.

Corollary 6 (Theorems 1 and 2 in [10]) *The d -ary multilog network $\log_d(N, 0, m)$ is wide-sense nonblocking with respect to the window algorithm with window size d^t if*

$$m \geq \begin{cases} d^{n-2t-1} + t d^{n-t-1}(d-1) & \text{when } t \leq \lfloor \frac{n}{2} \rfloor - 1, \\ d^{n-t-1}[(d-1)(n-t-1)-1] + d^t - d^{2t-n-1} & \\ (d-1) + 1 & \text{when } t \geq \lfloor \frac{n}{2} \rfloor. \end{cases}$$

Proof Note that $r = n$. It follows that

$$C(t, d^n) = d^{n-2t-1} + t d^{n-t-1}(d-1) - 1, \text{ when } t \leq \lfloor \frac{n}{2} \rfloor - 1$$

and

$$C(t, d^n) = d^{n-t-1}[(d-1)(n-t-1)-1] + d^t - d^{2t-n-1}(d-1), \text{ otherwise.}$$

\square

7 Analyzing Crosstalk-Free f -Cast Wide-Sense Nonblocking Multilog Networks

Thanks to [Proposition 1](#), to analyze crosstalk-free f -cast wide-sense nonblocking multilog networks under the window algorithm, basically all one has to do is to replace the constraint $i + j \geq n$ by the constraint $i + j \geq n - 1$. That was essentially the only difference between [Properties 1](#) and [2](#). Replacing n by $n - 1$ leads to changes in the final formula for the required number of Banyan planes. Deriving the formulas is relatively straightforward but also takes some calculus effort, and thus for completeness, they are done here. The overall outline of the analysis, however, is identical, and much of the analysis for the noncrosstalk-free case can be reused.

7.1 Setting Up the Linear Program and Its Dual

Identical notations as in the previous section are used. The following lemma is the crosstalk-free analog of [Lemma 10](#).

Lemma 12 *For each input $\mathbf{u} \in \mathcal{A}$ and each $w \in [d^{n-t} - 1]$ such that $i(\mathbf{u}) + j(w) \geq n - 1$, define a variable $x_{\mathbf{u},w}$. Also, for each input $\mathbf{u} \in \mathcal{A}$ and each output $\mathbf{v} \in W_0 - B$ such that $i(\mathbf{u}) + j(\mathbf{v}) \geq n - 1$, define a variable $x_{\mathbf{u},\mathbf{v}}$. Then, the number of Banyan planes blocking (\mathbf{a}, B) is upperbounded by the optimal value of the linear program (35), whose dual is (36).*

Some quick consequences of the formulation are derived next.

Corollary 7 (Theorem III.1 in [54]) *Let $r = \lfloor \log_d f \rfloor$. Suppose the $1 \times m$ -SE stage of the $\log_d(N, 0, m)$ network does not have fanout capability, then when $f \leq d^{n-2}(d - 1)$, the network is crosstalk-free f -cast strictly nonblocking if*

$$m \geq d^{\lfloor \frac{n+r+1}{2} \rfloor} + f \left(d^{\lceil \frac{n-r-1}{2} \rceil} - 1 \right).$$

When $f > d^{n-2}(d - 1)$, the network is f -cast strictly nonblocking if $m \geq d^n - d^{n-2}(d - 1)$.

Proof Routing using the window algorithm with window size, $t = n$ is the same as routing arbitrarily in the network when the $1 \times m$ -SE stage cannot fanout. Thus, any sufficient condition for the window algorithm to work is a strictly nonblocking condition. Note that when $t = n$, the dual constraints (DC-1) do not exist. Consider a solution to the dual LP as follows:

When $f > d^{n-2}(d - 1)$, consider two cases. If $k > d^{n-2}(d - 1)$, set $\delta_v = 1$ for all $\mathbf{v} \in \bigcup_{j=0}^{n-1} B_j$ and all other variables to be 0. Then, the dual-objective value is

$$\sum_{\mathbf{v} \in \bigcup_{j=0}^{n-1} B_j} \delta_{\mathbf{v}} = \left| \bigcup_{j=0}^{n-1} B_j \right| = d^n - k \leq d^n - d^{n-2}(d-1) - 1.$$

Thus, in this case $d^n - d^{n-2}(d-1)$, Banyan planes are sufficient. Next, suppose $k \leq d^{n-2}(d-1)$, in which case $kd < d^n$. Set $\gamma_{\mathbf{u}} = 1$ for all \mathbf{u} with $i(\mathbf{u}) \geq 1$, $\delta_{\mathbf{v}} = 1$ for all $\mathbf{v} \in B_{n-1}$, and all other variables to be 0. The solution is dual feasible with dual-objective value

$$\begin{aligned} \sum_{\mathbf{u}: i(\mathbf{u}) \geq 1} \gamma_{\mathbf{u}} + \sum_{\mathbf{v} \in B_{n-1}} \delta_{\mathbf{v}} &= \sum_{i=1}^{n-1} |A_i| + |B_{n-1}| \\ &\leq \sum_{i=1}^{n-1} (d^{n-i} - d^{n-i-1}) + \min\{d^n - k, k(d-1)\} \\ &= d^{n-1} - 1 + k(d-1) \\ &\leq d^{n-1} + d^{n-2}(d-1)^2 - 1 \\ &= d^n - d^{n-2}(d-1) - 1 \end{aligned}$$

and thus again $d^n - d^{n-2}(d-1)$, Banyan planes are sufficient.

Next, consider the case when $f \leq d^{n-2}(d-1)$. In this case, $r \leq n-2$. Let $p = \lceil \frac{n-r-1}{2} \rceil$. Then, $1 \leq p \leq \lceil \frac{n-1}{2} \rceil$. Furthermore,

$$kd^p \leq fd^p < d^{r+1}d^{\lceil \frac{n-r-1}{2} \rceil} = d^{\lceil \frac{n+r+1}{2} \rceil} \leq d^n.$$

Set $\gamma_{\mathbf{u}} = 1$ for all \mathbf{u} with $i(\mathbf{u}) \geq p$, $\delta_{\mathbf{v}} = 1$ for all $\mathbf{v} \in \bigcup_{j=n-p}^{n-1} B_j$, and all other variables to be 0. The solution is dual feasible because, for any pair (\mathbf{u}, \mathbf{v}) for which $i(\mathbf{u}) + j(\mathbf{v}) \geq n-1$, it must either be the case that $i(\mathbf{u}) \geq p$ or that $j(\mathbf{v}) \geq n-p$ (which is the same as saying $\mathbf{v} \in \bigcup_{j=n-p}^{n-1} B_j$). Recalling [Proposition 5](#) and the fact that $kd^p < d^n$ shown above, the dual-objective value is

$$\begin{aligned} \sum_{\mathbf{u}: i(\mathbf{u}) \geq p} \gamma_{\mathbf{u}} + \sum_{\mathbf{v} \in \bigcup_{j=n-p}^{n-1} B_j} \delta_{\mathbf{v}} &= \sum_{i=p}^{n-1} |A_i| + \left| \bigcup_{j=n-p}^{n-1} B_j \right| \\ &\leq \sum_{i=p}^{n-1} (d^{n-i} - d^{n-i-1}) + \min\{d^n - k, k(d^p - 1)\} \\ &= d^{n-p} - 1 + k(d^p - 1) \\ &\leq d^{n-p} + f(d^p - 1) - 1. \end{aligned}$$

Hence, in this case, $d^{n-p} + f(d^p - 1)$ is a sufficient number of Banyan planes. \square

7.2 Specifying a Family of Dual-Feasible Solutions

The family of dual-feasible solution is specified with two integral parameters where $0 \leq p \leq n - t - 1$ and $n - t \leq q \leq n$. The parameter p is used to set the variables ϵ_u , α_w , and $\beta_{u,w}$, and the parameter q is used to set the variables γ_u and δ_v . As the variables are set, the feasibility of the constraints (DC-1) and (DC-2) is also verified, and the contributions of those variables to the final objective value are computed:

- *Specifying the ϵ_u variables.* The ϵ_u are defined as follows:

$$\epsilon_u = \begin{cases} 1 & i(\mathbf{u}) \geq n - p \\ 0 & \text{otherwise} \end{cases} \quad (46)$$

The contribution of the ϵ_u to the objective is

$$\sum_{\mathbf{u}} f \epsilon_u = \sum_{i=n-p}^{n-1} f \sum_{\mathbf{u}:i(\mathbf{u})=i} 1 = \sum_{i=n-p}^{n-1} f |A_i| = f(d^p - 1).$$

- *Specifying the α_w and $\beta_{u,w}$ variables.* Next, the α_w and $\beta_{u,w}$ are defined. The constraints (DC-1) with $i(\mathbf{u}) \geq n - p$ are already satisfied by the ϵ_u ; hence, the α_w and $\beta_{u,w}$ only need to be set to satisfy (DC-1) when $j(w) \geq p$. (If $j(w) \leq p - 1$, then for the constraint to exist, it must hold that $i(\mathbf{u}) \geq n - 1 - j(w) \geq n - p$.) The variables α_w and $\beta_{u,w}$ are set differently based on three cases as follows:

Case 1. If $t \geq \lceil \frac{n}{2} \rceil$, then set

$$\beta_{u,w} = \begin{cases} 1 & p \leq j(w) \leq n - t - 1, \text{ and } n - 1 - j(w) \leq i(\mathbf{u}) \leq n - p - 1, \\ 0 & \text{otherwise} \end{cases} \quad (47)$$

and set all the α_w to 0. It can be verified straightforwardly that all constraints (DC-1) are satisfied. Thus, the contributions of the α_w and $\beta_{u,w}$ to the dual-objective value are

$$\begin{aligned} \sum_{\substack{\mathbf{u}, w \\ p \leq j(w) \leq n - t - 1 \\ n - 1 - j(w) \leq i(\mathbf{u}) \leq n - p - 1}} \beta_{u,w} &= \sum_{j=p}^{n-t-1} |\{w : j(w) = j\}| \\ &= \sum_{i=n-1-j}^{n-p-1} |\{\mathbf{u} \in \mathcal{A} : i(\mathbf{u}) = i\}| \\ &= \sum_{j=p}^{n-t-1} (d^{n-j-t} - d^{n-j-t-1}) \sum_{i=n-1-j}^{n-p-1} |A_i| \end{aligned}$$

$$\begin{aligned}
&= \sum_{j=p}^{n-t-1} (d^{n-j-t} - d^{n-j-t-1})(d^{j+1} - d^p) \\
&= (n-t-p)(d^{n-t+1} - d^{n-t}) - d^{n-t} + d^p.
\end{aligned}$$

The second equality follows from the fact that the number of windows W_w for which $j(w) = j$ is precisely $d^{n-j-t} - d^{n-j-t-1}$.

Case 2. When $p+1 \leq t \leq \lceil \frac{n}{2} \rceil - 1$, set

$$\beta_{\mathbf{u},w} = \begin{cases} 1 & p \leq j(w) \leq t-1 \text{ and } n-1-j(w) \leq i < n-p \\ 0 & \text{otherwise} \end{cases}$$

and

$$\alpha_w = \begin{cases} 1 & t \leq j < n-t \\ 0 & \text{otherwise} \end{cases}$$

All constraints (DC-1) are thus satisfied. The α_w 's and $\beta_{\mathbf{u},w}$'s contributions to the objective are

$$\begin{aligned}
&\sum_{\substack{w \\ t \leq j(w) < n-t}} d^t \alpha_w + \sum_{\substack{\mathbf{u},w \\ p \leq j(w) \leq t-1 \\ n-1-j(w) \leq i(\mathbf{u}) \leq n-p-1}} \beta_{\mathbf{u},w} \\
&= \sum_{j=t}^{n-t-1} d^t \cdot |\{w : j(w) = j\}| + \sum_{j=p}^{t-1} |\{w : j(w) = j\}| \\
&= \sum_{i=n-1-j}^{n-p-1} |\{\mathbf{u} \in \mathcal{A} : i(\mathbf{u}) = i\}| \\
&= \sum_{j=t}^{n-t-1} d^t (d^{n-j-t} - d^{n-j-t-1}) + \sum_{j=p}^{t-1} (d^{n-j-t} - d^{n-j-t-1})(d^{j+1} - d^p) \\
&= (t-p)(d^{n-t+1} - d^{n-t}) + d^{n-2t+p} - d^t.
\end{aligned}$$

Case 3. When $t \leq p$ (which is $\leq n-t-1$), set

$$\alpha_w = \begin{cases} 1 & p \leq j(w) \leq n-t-1 \\ 0 & \text{otherwise} \end{cases} \quad (48)$$

and all the $\beta_{\mathbf{u},w}$ to be zero. Again, the feasibility of the constraints (DC-1) is easy to verify. The contribution to the objective value is

$$\begin{aligned}
\sum_{\substack{w \\ p \leq j(w) < n-t}} d^t \alpha_w &= \sum_{j=p}^{n-t-1} d^t \cdot |\{w : j(w) = j\}| \\
&= \sum_{j=p}^{n-t-1} d^t (d^{n-j-t} - d^{n-j-t-1}) \\
&= d^{n-p} - d^t.
\end{aligned}$$

Specifying the γ_u and δ_v variables. When $q = n - t$, set $\delta_v = 1$ for all $v \in \bigcup_{j=n-t}^{n-1} B_j$ and all $\gamma_u = 0$. The dual-objective contribution in this case is

$$\sum_{v \in \bigcup_{j=n-t}^{n-1} B_j} \delta_v = \left| \bigcup_{j=n-t}^{n-1} B_j \right| = d^t - k.$$

When $n - t + 1 \leq q \leq n$, define

$$\delta_v = \begin{cases} 1 & v \in \bigcup_{j=q}^{n-1} B_j \\ 0 & \text{otherwise} \end{cases} \quad (49)$$

and

$$\gamma_u = \begin{cases} 1 & n - q \leq i(u) \leq n - p - 1 \\ 0 & \text{otherwise} \end{cases} \quad (50)$$

From [Proposition 5](#), the total contribution of the γ_u and δ_v to the dual objective is

$$\begin{aligned}
\sum_{\substack{u \\ n-q \leq i(u) \leq n-p-1}} \gamma_u + \sum_{v \in \bigcup_{j=q}^{n-1} B_j} \delta_v &= \sum_{i=n-q}^{n-p-1} |\{u : i(u) = i\}| + \left| \bigcup_{j=q}^{n-1} B_j \right| \\
&\leq \sum_{i=n-q}^{n-p-1} |A_i| + \min\{d^t - k, k(d^{n-q} - 1)\} \\
&= d^q - d^p + \min\{d^t - k, k(d^{n-q} - 1)\}.
\end{aligned}$$

The feasibility of all the constraints (DC-2) is easy to verify.

Define the “cost” $g(k, p, q)$ to be the total contribution of all variables to the dual-objective value. The values of $g(k, p, q)$ are summarized in [Fig. 12](#). The following has just been proved:

Theorem 10 *The above family of solutions is feasible for the dual LP (36) with objective value equal to $g(k, p, q)$. (Recall that, in this problem, the dual constraints*

The objective value $g(k, p, q)$
--

For $t \geq \lceil \frac{n}{2} \rceil$ and $q = n - t$,

$$g(k, p, q) = f(d^p - 1) + (n - t - p)(d^{n-t+1} - d^{n-t}) - d^{n-t} + d^p + d^t - k.$$

For $t \geq \lceil \frac{n}{2} \rceil$ and $q > n - t$,

$$g(k, p, q) = f(d^p - 1) + (n - t - p)(d^{n-t+1} - d^{n-t}) - d^{n-t} + d^q + \min\{d^t - k, k(d^{n-q} - 1)\}.$$

For $p + 1 \leq t \leq \lceil \frac{n}{2} \rceil - 1$ and $q = n - t$

$$g(k, p, q) = f(d^p - 1) + (t - p)(d^{n-t+1} - d^{n-t}) + d^{n+p-2t} - k.$$

For $p + 1 \leq t \leq \lceil \frac{n}{2} \rceil - 1$ and $q > n - t$

$$\begin{aligned} g(k, p, q) = & f(d^p - 1) + (t - p)(d^{n-t+1} - d^{n-t}) + d^{n+p-2t} - d^t + d^q - d^p \\ & + \min\{d^t - k, k(d^{n-q} - 1)\}. \end{aligned}$$

For $t \leq p$ and $q = n - t$,

$$g(k, p, q) = f(d^p - 1) + d^{n-p} - k.$$

For $t \leq p$ and $q > n - t$,

$$g(k, p, q) = f(d^p - 1) + d^{n-p} - d^t + d^q - d^p + \min\{d^t - k, k(d^{n-q} - 1)\}.$$

Fig. 12 The dual-objective value of the family of dual-feasible solutions

for which $i(\mathbf{u}) + j(\mathbf{v}) \geq n - 1$ and $i(\mathbf{u}) + j(\mathbf{w}) \geq n - 1$ are being considered.) Consequently, for the network $\log_d(N, 0, m)$ to be crosstalk-free f -cast wide-sense nonblocking under the window algorithm with window size d^t , it is sufficient that

$$m \geq 1 + \max_{1 \leq k \leq \min(f, d^t)} \min_{p,q} g(k, p, q). \quad (51)$$

7.3 Selecting the Best Dual-Feasible Solution

The proof of the following technical lemma is similar to that of [Lemma 11](#), and thus the proof is omitted.

Lemma 13 Let d, n, k be positive integers and $x = \lfloor \log_d k \rfloor$. Then, the following function

$$\bar{h}(k) = d^{\lfloor \frac{x+n+1}{2} \rfloor} + k \left(d^{n-\lfloor \frac{x+n+1}{2} \rfloor} - 1 \right) \quad (52)$$

is nondecreasing in k .

The upperbound $G(t, f)$

To shorten the notations, let $r = \lfloor \log_d f \rfloor$.

$$G(t, f) = \begin{cases} d^{n-t}[(n-t)(d-1)-1] \\ + d^t - d^{2t-n+1}(d-1) & t > n/2, r \geq \max\{2t-n-2, \\ & n-t+1\} \\ f(d^{n-t-r}-1) + rd^{n-t}(d-1) - d^{n-t} \\ + d^{\lfloor \frac{r+n+1}{2} \rfloor} + f(d^{n-\lfloor \frac{r+n+1}{2} \rfloor}-1) & t > n/2, r \leq \min\{2t-n-3, \\ & n-t\} \\ d^{n-t}[(n-t)(d-1)-1] + d^{\lfloor \frac{r+n+1}{2} \rfloor} \\ + f(d^{n-\lfloor \frac{r+n+1}{2} \rfloor}-1) & t > n/2, n-t+1 \leq r \leq 2t \\ & -n-3 \\ f(d^{n-t-r}-1) + d^{n-t}[r(d-1)-1] & t > n/2, 2t-n-2 \leq r \leq \\ + d^t - d^{2t-n-2}(d-1) & n-t \\ d^{n-t}[(n-t)(t-1)-1] + d^t \\ f(d^{\lceil \frac{n-r-1}{2} \rceil}-1) + d^{n-\lceil \frac{n-r-1}{2} \rceil}-1 & t = n/2 \\ f(d^{t-1}-1) + d^{n-t-1}(d^2-d+1)-1 & t < n/2, r \leq n-2t \text{ and } f \leq \\ & d^{n-2t}(d-1) \\ f(d^{n-t-r}-1) + (2t-n-r) & t < n/2, r \leq n-2t, f > \\ (d-1)d^{n-t} + d^{2n-3t-r}-1 & d^{n-2t}(d-1) \\ t(d-1)d^{n-t} + d^{n-2t}-1 & t < n/2, n-t < r \end{cases}$$

Fig. 13 Theorem 11 shows that $G(t, f) \geq \max_k \min_{p,q} g(k, p, q)$

Theorem 11 *The $\log_d(N, 0, m)$ network is crosstalk-free nonblocking under the window algorithm with window size d^t if $m \geq 1 + G(t, f)$ where $G(t, f)$ is defined in Fig. 13.*

Proof Suppose $t > n/2$, i.e., $2t \geq n+1$. Four cases are considered as follows. In all cases, define an integral variable $x = \lfloor \log_d k \rfloor$:

- *Case 1.* $r \geq \max\{2t-n-2, n-t+1\}$.

If $k \geq d^{2t-n-2}(d-1)+1$, then pick $p_k=0$ and $q_k=n-t$. Then,

$$\begin{aligned} g(k, p_k, q_k) &= (n-t)(d^{n-t+1}-d^{n-t}) - d^{n-t} + 1 + d^t - k \\ &\leq (n-t)(d^{n-t+1}-d^{n-t}) - d^{n-t} + 1 + d^t - d^{2t-n+1}(d-1)-1 \\ &= d^{n-t}[(n-t)(d-1)-1] + d^t - d^{2t-n+1}(d-1). \end{aligned}$$

On the other hand, if $k \leq d^{2t-n-2}(d-1)$, then set $p_k = 0$ and $q_k = \lfloor \frac{x+n+1}{2} \rfloor > n-t$. Note that $k d^{n-q_k} \leq d^t$. Thus, recall from Lemma 13 that the function $\bar{h}(k)$ defined in (52) is nondecreasing in k , it follows that

$$\begin{aligned} g(k, p_k, q_k) &= (n-t)(d^{n-t+1} - d^{n-t}) - d^{n-t} + d^{\lfloor \frac{x+n+1}{2} \rfloor} \\ &\quad + \min \left\{ d^t - k, k \left(d^{n-\lfloor \frac{x+n+1}{2} \rfloor} - 1 \right) \right\} \\ &= (n-t)(d^{n-t+1} - d^{n-t}) - d^{n-t} + d^{\lfloor \frac{x+n+1}{2} \rfloor} + k \left(d^{n-\lfloor \frac{x+n+1}{2} \rfloor} \right) \\ &= (n-t)(d^{n-t+1} - d^{n-t}) - d^{n-t} + \bar{h}(k) \\ &\leq (n-t)(d^{n-t+1} - d^{n-t}) - d^{n-t} + \bar{h}(d^{2t-n-2}(d-1)) \\ &= d^{n-t}[(n-t)(d-1)-1] + d^t - d^{2t-n+1}(d-1). \end{aligned}$$

- Case 2. $r \leq \min\{2t-n-3, n-t\}$.

In this case, set $p_k = n-t-r \leq t-1$ and $q_k = \lfloor \frac{x+n+1}{2} \rfloor > n-t$. Similar to the above analysis, the following is obtained:

$$\begin{aligned} g(k, p_k, q_k) &= f(d^{n-t-r}-1) + r d^{n-t}(d-1) - d^{n-t} + d^{\lfloor \frac{x+n+1}{2} \rfloor} \\ &\quad + k(d^{n-\lfloor \frac{x+n+1}{2} \rfloor} - 1) \\ &= f(d^{n-t-r}-1) + r d^{n-t}(d-1) - d^{n-t} + \bar{h}(k) \\ &\leq f(d^{n-t-r}-1) + r d^{n-t}(d-1) - d^{n-t} + \bar{h}(f) \\ &= f(d^{n-t-r}-1) + r d^{n-t}(d-1) - d^{n-t} + d^{\lfloor \frac{r+n+1}{2} \rfloor} \\ &\quad + f(d^{n-\lfloor \frac{r+n+1}{2} \rfloor} - 1). \end{aligned}$$

For the inequality, note that the function $\bar{h}(k)$ is nondecreasing in k and $k \leq f$.

- Case 3. $n-t+1 \leq r \leq 2t-n-3$. In this case, set $p_k = 0$ and $q_k = \lfloor \frac{x+n+1}{2} \rfloor > n-t$. Then,

$$g(k, p_k, q_k) \leq d^{n-t}[(n-t)(d-1)-1] + d^{\lfloor \frac{r+n+1}{2} \rfloor} + f(d^{n-\lfloor \frac{r+n+1}{2} \rfloor} - 1).$$

- Case 4. $2t-n-2 \leq r \leq n-t$. In this case, set $p_k = n-t-r$ and consider two sub-cases as in case 1: $k \geq d^{2t-n-2}(d-1)+1$ or $k \leq d^{2t-n-2}(d-1)$. The eventual bound is

$$g(k, p_k, q_k) \leq f(d^{n-t-r}-1) + d^{n-t}[r(d-1)-1] + d^t - d^{2t-n-2}(d-1).$$

When $t = n/2$, simply set $q_k = n-t$ and $p_k = 0$. Then,

$$g(k, p_k, q_k) = d^{n-t}[(n-t)(d-1)-1] + d^t.$$

Finally, suppose $t < n/2$. The value $p_k = q - t$ will always be set in this situation. Also consider four cases:

Case 1. $r \leq n - 2t$ and $f \leq d^{n-2t}(d - 1)$, then set $p_k = \lceil \frac{n-r-1}{2} \rceil \geq t$. In this case,

$$g(k, p_k, q_k) = f(d^{\lceil \frac{n-r-1}{2} \rceil} - 1) + d^{n-\lceil \frac{n-r-1}{2} \rceil} - 1.$$

Case 2. $r \leq n - 2t$ and $f > d^{n-2t}(d - 1)$, then set $p_k = t - 1$. In this case,

$$g(k, p_k, q_k) = f(d^{t-1} - 1) + d^{n-t-1}(d^2 - d + 1) - 1.$$

Case 3. $n - 2t + 1 \leq r \leq n - t$, then set $p_k = n - t - r$. In this case,

$$g(k, p_k, q_k) = f(d^{n-t-r} - 1) + (2t - n - r)(d - 1)d^{n-t} + d^{2n-3t-r} - 1.$$

Case 4. $n - t < r$, then set $p_k = 0$. In this case,

$$g(k, p_k, q_k) = t(d - 1)d^{n-t} + d^{n-2t} - 1. \quad \square$$

Corollary 8 (Theorems 1 in [39]) *The d -ary multilog network $\log_d(N, 0, m)$ is crosstalk-free wide-sense nonblocking with respect to the window algorithm with window size d^t if*

$$m \geq \begin{cases} d^{n-2t} + td^{n-t}(d - 1) & t < n/2 \\ d^{n-t}[(n - t)(d - 1) - 1] + d^t + 1 & t = n/2 \\ d^{n-t}[(n - t)(d - 1) - 1] + d^t - d^{2t-n-2}(d - 1) + 1 & t > n/2. \end{cases}$$

8 Conclusion

As illustrated with many examples in this chapter, the linear programming technique seems to be effective in deriving sufficient conditions for a switching network to be nonblocking. In many of the examples discussed above, the sufficient conditions turn out to be necessary also (Examples 1, 2, and many special cases of later theorems on the multilog networks). However, it is not clear how one can “quickly” prove corresponding necessary conditions. The main reason is that a dual-feasible solution may not naturally lead to an integral primal feasible solution, which is needed to combinatorially design a network configuration showing the necessary condition.

Another line of open research questions is on multilog networks with extra stages [22]. For example, adding $\log_d N$ extra stages to a Banyan plane, one obtains the

Beneš network [1], a very important building block for many useful switching network architectures such as the Cantor network [2].

Cross-References

- ▶ [Bin Packing Approximation Algorithms: Survey and Classification](#)
- ▶ [Combinatorial Optimization in Transportation and Logistics Networks](#)
- ▶ [Dual Integrality in Combinatorial Optimization](#)
- ▶ [Fractional Combinatorial Optimization](#)
- ▶ [Greedy Approximation Algorithms](#)
- ▶ [Network Optimization](#)
- ▶ [Optimization in Multi-Channel Wireless Networks](#)

Recommended Reading

1. V.E. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic*. Mathematics in Science and Engineering, vol. 17 (Academic, New York, 1965)
2. D.G. Cantor, On non-blocking switching networks. *Networks* **1**, 367–377 (1971/1972)
3. G.J. Chang, F.K. Hwang, L.-D. Tong, Characterizing bit permutation networks [MR1630366 (99h:94088)]. *Networks* **33**, 261–267 (1999). Selected papers from DIMACS Workshop on Switching Networks, Princeton, NJ, 1997
4. H.-B. Chen, F.K. Hwang, On multicast rearrangeable 3-stage clos networks without first-stage fan-out. *SIAM J. Discret. Math.* **20**, 287–290 (2006)
5. V. Chinni, T. Huang, P.-K. Wai, C. Menyuk, G. Simonis, Crosstalk in a lossy directional coupler switch. *J. Lightwave Technol.* **13**, 1530–1535 (1995)
6. S.-P. Chung, K.W. Ross, On nonblocking multirate interconnection networks. *SIAM J. Comput.* **20**, 726–736 (1991)
7. C. Clos, Bell Syst. Tech. J. **32**, 406–424 (1953)
8. P. Coppo, M. D'Ambrosio, R. Melen, Optimal cost/performance design of ATM switches. *IEEE/ACM Trans. Netw.* **1**, 566–575 (1993)
9. J.G. Dai, B. Prabhakar, The throughput of data switches with and without speedup, in *INFOCOM*, Tel-Aviv, 2000, pp. 556–564
10. G. Danilewicz, Wide-sense nonblocking $\log_d(n, 0, p)$ multicast switching networks. *IEEE Trans. Commun.* **55**, 2193–2200 (2007)
11. D. Li, Elimination of crosstalk in directional coupler switches. *Opt. Quantum Electron.* **25**, 255–260 (1993)
12. J. Duato, S. Yalamanchili, L. Ni, *Interconnection Networks: An Engineering Approach*, 1st edn. (IEEE Computer Society, Los Alamitos, 1997)
13. F.K. Hwang, B.-C. Lin, Wide-sense nonblocking multicast $\log_2(n, m, p)$ networks. *IEEE Trans. Commun.* **51**, 1730–1735 (2003)
14. F.K. Hwang, Y. Wang, J. Tan, Strictly nonblocking f-cast $\log_d(n, m, p)$ networks. *IEEE Trans. Commun.* **55**, 981–986 (2007)
15. B. Gao, F.K. Hwang, Wide-sense nonblocking for multirate 3-stage Clos networks. *Theor. Comput. Sci.* **182**, 171–182 (1997)
16. R.R. Goke, G.J. Lipovski, Banyan networks for partitioning multiprocessor systems, in *Proceedings of the 1st Annual Symposium on Computer Architecture (ISCA'73)*, Barcelona, Spain, 1973, pp. 21–28

17. P.E. Haxell, A. Rasala, G.T. Wilfong, P. Winkler, Wide-sense nonblocking WDM cross-connects, in *Algorithms—ESA '03*, Rome, Italy. Lecture Notes in Computer Science, vol. 2461 (Springer, Berlin, 2002), pp. 538–549
18. H.S. Hinton, *An Introduction to Photonic Switching Fabrics* (Plenum, New York, 1993)
19. J. H. Hui, *Switching and Traffic Theory for Integrated Broadband Networks* (Kluwer Academic, Boston/Dordrecht/London, 1990)
20. F. Hwang, Choosing the best $\log_2(n, m, p)$ strictly nonblocking networks. *IEEE Trans. Commun.* **46**, 454–455 (1998)
21. F.K. Hwang, *The Mathematical Theory of Nonblocking Switching Networks* (World Scientific, River Edge, 1998)
22. F.K. Hwang, W.-D. Lin, Necessary and sufficient conditions for rearrangeable log/sub d/(n, m, p). *IEEE Trans. Commun.* **53**, 2020–2023 (2005)
23. X. Jiang, A. Pattavina, S. Horiguchi, Rearrangeable f -cast multi- $\log_2 n$ networks. *IEEE Trans. Commun.* **56**, 1929–1938 (2008)
24. X. Jiang, A. Pattavina, S. Horiguchi, Strictly nonblocking f-cast photonic networks. *IEEE/ACM Trans. Netw.* **16**, 732–745 (2008)
25. X. Jiang, H. Shen, M.M. ur Rashid Khandker, S. Horiguchi, Blocking behaviors of crosstalk-free optical banyan networks on vertical stacking. *IEEE/ACM Trans. Netw.* **11**, 982–993 (2003)
26. W. Kabacinski, G. Danilewicz, Wide-sense and strict-sense nonblocking operation of multicast multi- $\log_2 n$ switching networks. *IEEE Trans. Commun.* **50**, 1025–1036 (2002)
27. C.-T. Lea, Muti- $\log_2 n$ networks and their applications in high speed electronic and photonic switching systems. *IEEE Trans. Commun.* **38**, 1740–1749 (1990)
28. C.-T. Lea, D.-J. Shyy, Tradeoff of horizontal decomposition versus vertical stacking in rearrangeable nonblocking networks. *IEEE Trans. Commun.* **39**, 899–904 (1991)
29. S.C. Liew, M.-H. Ng, C.W. Chan, Blocking and nonblocking multirate clos switching networks. *IEEE/ACM Trans. Netw.* **6**, 307–318 (1998)
30. Lucent Technologies Press Release, Lucent Technologies unveils ultra-high-capacity optical system; Time Warner Telecom first to announce it will deploy the system (2001), <http://www.lucent.com/press/0101/010117.nsa.html>
31. Lucent Technologies Press Release, Lucent Technologies engineer and scientists set new fiber optic transmission record (2002), <http://www.lucent.com/press/0302/020322.bla.html>
32. Lucent Technologies Website, What is dense wave division multiplexing (DWDM) (2002), <http://www.bell-labs.com/technology/lightwave/dwdm.html>
33. G. Maier, A. Pattavina, Design of photonic rearrangeable networks with zero first-order switching-element-crosstalk. *IEEE Trans. Comm.* **49**, 1248–1279 (2001)
34. B. Mukherjee, *Optical Communication Networks* (McGraw-Hill, New York, 1997)
35. H.Q. Ngo, A new routing algorithm for multirate rearrangeable Clos networks. *Theoret. Comput. Sci.* **290**, 2157–2167 (2003)
36. H.Q. Ngo, D.-Z. Du, Notes on the complexity of switching networks, in *Advances in Switching Networks*, ed. by D.-Z. Du, H.Q. Ngo. Network Theory and Applications, vol. 5 (Kluwer Academic, Boston, 2001) pp. 307–367
37. H.Q. Ngo, V.H. Vu, Multirate rearrangeable Clos networks and a generalized bipartite graph edge coloring problem. *SIAM J. Comput.* **32**, (2003), pp. 1040–1049
38. H.Q. Ngo, D. Pan, C. Qiao, Constructions and analyses of nonblocking wdm switches based on arrayed waveguide grating and limited wavelength conversion. *IEEE/ACM Trans. Netw.* **14**, 205–217 (2006)
39. H.Q. Ngo, T.-N. Nguyen, D.T. Ha, Crosstalk-free wide sense nonblocking multicast photonic switching networks, in *Proceedings of the 2008 IEEE Global Communications Conference (GLOBECOM)*, New Orleans, LA, USA (IEEE, Piscataway, 2008), pp. 2643–2647
40. H.Q. Ngo, Y. Wang, A. Le, A linear programming duality approach to analyzing strictly nonblocking d -ary multilog networks under general crosstalk constraints, in *Proceedings of the 14th Annual International Computing and Combinatorics Conference (COCOON)*, Beijing, China, LNCS (Springer, 2008), pp. 509–519

41. H.Q. Ngo, A. Rudra, A.N. Le, T.-N. Nguyen, Analyzing nonblocking switching networks using linear programming (duality), in *INFOCOM*, San Diego, 2010, pp. 2696–2704
42. T.-N. Nguyen, H.Q. Ngo, Y. Wang, Strictly nonblocking f -cast photonic switching networks under general crosstalk constraints, in *Proceedings of the 2008 IEEE Global Communications Conference (GLOBECOM)*, New Orleans, LA, USA (IEEE, 2008) pp. 2807–2811
43. A. Pattavina, G. Tesei, Non-blocking conditions of multicast three-stage interconnection networks. *IEEE Trans. Commun.* **46**, (2005), pp. 163–170
44. R. Ramaswami, K. Sivarajan, *Optical Networks: A Practical Perspective*, 2nd edn. (Morgan Kaufmann, San Francisco, 2001)
45. A. Rasala, G. Wilfong, Strictly non-blocking WDM cross-connects, in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms SODA'2000*, San Francisco, CA (ACM, New York, 2000), pp. 606–615
46. A. Rasala, G. Wilfong, Strictly non-blocking WDM cross-connects for heterogeneous networks, in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing STOC'2000*, Portland, OR (ACM, New York, 2000), pp. 513–524
47. D.-J. Shyy, C.-T. Lea, $\log_2(n, m, p)$ strictly nonblocking networks. *IEEE Trans. Commun.* **39**, 1502–1510 (1991)
48. T.E. Stern, K. Bala, *Multiwavelength Optical Networks: A Layered Approach* (Prentice Hall, Upper Saddle River, 1999)
49. Y. Tscha, K.-H. Lee, Yet another result on multi- $\log_2 n$ networks. *IEEE Trans. Commun.* **47**, 1425–1431 (1999)
50. J.S. Turner, R. Melen, Multirate Clos networks. *IEEE Commun. Mag.* **41**, 38–44 (2003)
51. J.S. Turner, N. Yamanaka, Architectural choices in large scale ATM switches. *IEICE Trans. Commun.* **E81-B**, 120–137 (1998)
52. M.M. Vaez, C.-T. Lea, Wide-sense nonblocking Banyan-type switching systems based on directional couplers. *IEEE J. Select. Areas Commun.* **16**, 1327–1332 (1998)
53. M.M. Vaez, C.-T. Lea, Strictly nonblocking directional-coupler-based switching networks under crosstalk constraint. *IEEE Trans. Commun.* **48**, 316–323 (2000)
54. Y. Wang, H.Q. Ngo, X. Jiang, Strictly nonblocking f -cast d -ary multilog networks under fanout and crosstalk constraints, in *Proceedings of the 2008 International Conference on Communications (ICC)*, Beijing, China (IEEE, 2008)
55. G. Wilfong, B. Mikkelsen, C. Doerr, M. Zirngibl, WDM cross-connect architectures with reduced complexity. *J. Lightwave Technol.* **17**, 1732–1741 (1999)
56. J.-C. Wu, T.-L. Tsai, Low complexity design of a wavelength-selective switch using raman amplifiers and directional couplers, in *GLOBECOM*, San Francisco, California, 2006
57. Y. Yang, G.M. Masson, Nonblocking broadcast switching networks. *IEEE Trans. Comput.* **40**, 1005–1015 (1981)
58. Y. Yang, J. Wang, Wide-sense nonblocking clos networks under packing strategy. *IEEE Trans. Comput.* **48**, 265–284 (1999)
59. Y. Yang, J. Wang, Designing WDM optical interconnects with full connectivity by using limited wavelength conversion. *IEEE Trans. Comput.* **53**, 1547–1556 (2004)
60. Y. Yang, J. Wang, C. Qiao, Nonblocking WDM multicast switching networks. *IEEE Trans. Parallel Distrib. Syst.* **11**, 1274–1287 (2000)

Map of Geometric Minimal Cuts with Applications*

Evanthia Papadopoulou, Jinhui Xu and Lei Xu

Contents

1	Introduction	1816
2	On Geometric Minimal Cuts and Their Map	1819
2.1	Geometric Cuts	1819
2.2	Identifying Geometric Minimal Cuts	1821
2.3	Generating Map of Geometric Minimal Cuts	1829
3	The MGMC Problem via Higher-Order Voronoi Diagrams	1844
3.1	The MGMC Problem in a VLSI Layout	1844
3.2	Review of Concepts on L_∞ Voronoi Diagrams	1846
3.3	Definitions and Problem Formulation	1848
3.4	A Higher-Order Voronoi Diagram Modeling Open Faults and the MGMC Problem	1851
3.5	Computing the Opens Voronoi Diagram	1856
4	The L_∞ Hausdorff Voronoi Diagram	1859
4.1	Definitions and Structural Complexity	1860
4.2	A Refined Plane Sweep Construction	1862
5	Conclusion	1867
	Recommended Reading	1867

*The work of the first author was supported in part by the Swiss National Science Foundation SNSF project 200021-127137. The work of the last two authors was supported in part by NSF through a CAREER Award CCF-0546509 and two grants IIS-0713489 and IIS-1115220.

E. Papadopoulou (✉)

Faculty of Informatics, Università della Svizzera italiana, Lugano, Switzerland

J. Xu • L. Xu

Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA

Abstract

This chapter considers the following problem of computing a map of geometric minimal cuts (called the MGMC problem): Given a graph $G = (V, E)$ and a planar embedding of a subgraph $H = (V_H, E_H)$ of G , compute the map of geometric minimal cuts induced by axis-aligned rectangles in the embedding plane. The MGMC problem is motivated by the *critical area* extraction problem in VLSI designs and finds applications in several other fields. This chapter surveys two different approaches for the MGMC problem based on a mix of geometric and graph algorithm techniques that can be regarded complementary. It is first shown that unlike the classic min-cut problem on graphs, the number of all rectilinear geometric minimal cuts is bounded by a low polynomial, $O(n^3)$. Based on this observation, the first approach enumerates all rectilinear geometric minimal cuts and computes their L_∞ Hausdorff Voronoi diagram, which is equivalent to the L_∞ Hausdorff Voronoi diagram of axis-aligned rectangles. The second approach is based on higher-order Voronoi diagrams and identifies necessary geometric minimal cuts and their Hausdorff Voronoi diagram in an iterative manner. The embedding in the latter approach includes arbitrary polygons. This chapter also presents the structural properties of the L_∞ Hausdorff Voronoi diagram of rectangles that provides the map of the MGMC problem and plane sweep algorithms for its construction.

1 Introduction

This chapter considers the following problem called *map of geometric minimal cuts (MGMC)*: Given a graph $G = (V, E)$ and a planar embedding of a subgraph $H = (V_H, E_H)$ of G , compute a map¹ \mathcal{M} of the embedding plane P of H so that for every point $p \in P$, the cell in \mathcal{M} containing p is associated with the “closest” *geometric cut* (in G) to p , where distance between a point p and a cut C is defined as the maximum distance between p and any individual element of C . A geometric cut C of G induced by a given geometric shape S is a set of edges and vertices in H that overlaps S in P and whose removal from G disconnects G . This chapter considers the case where geometric cuts are induced by axis-aligned rectangles and distances are measured in the L_∞ metric. The main objective of the MGMC problem is to compute the map \mathcal{M} of all geometric minimal (or canonical) cuts (the exact definition of geometric minimal cuts will be given in the next section) of the planar embedding of H .

¹A map means a partition of the embedding plane (as in a trapezoidal map) into cells so that all points in the same cell share the same “closest” geometric minimal cut.

The MGMC problem was formulated in [47]. It was introduced, as the *geometric min-cut problem*, in [34], in order to model the *critical area* computation problem for *open faults* in a VLSI design. Critical area is a measure reflecting the sensitivity of a VLSI design to random defects occurring during the manufacturing process. For a survey on VLSI yield analysis and optimization, see, for example, [16]. The critical area computation problem in a VLSI design for various types of fault mechanisms has been reduced to variants of generalized Voronoi diagrams; see, for example, [5, 32, 34–36, 46]. The Voronoi framework for critical area extraction asks for a subdivision of a layer A into regions that reveal, for every point p , the size of the smallest defect centered at p causing a circuit failure. For *open faults*, this results in the MGMC problem. In more detail, a VLSI net N can be modeled as a graph $G = (V, E)$ having a subgraph $H = (V_H, E_H)$ embedded on any conducting layer A ; see, for example, [34]. The embedded subgraph $H = (V_H, E_H)$ is vulnerable to random manufacturing defects that are associated with the layer of the embedding and may create open faults. The MGMC problem computes, for every point p on layer A , the disk of minimum radius, which is centered at p and induces a cut on graph G resulting in an open fault. The subdivision of layer A that solves the MGMC problem corresponds to the *Hausdorff Voronoi diagram* of the geometric minimal cuts (see, e.g., [34, 47]). The MGMC problem finds applications in other networks, such as transportation networks, where critical area may need to be extracted for the purpose of flow control and disaster avoidance.

This chapter surveys the literature on the MGMC problem and presents two complimentary approaches. These approaches are based on a mix of geometric and graph algorithm techniques, and they produce, by different means, the L_∞ Hausdorff Voronoi diagram of geometric minimal cuts on the embedding plane. In addition, this chapter presents structural properties and algorithms for the L_∞ Hausdorff Voronoi diagram, which provides a solution to the MGMC problem, and it is a geometric structure of independent interest.

The first approach to the MGMC problem presented in this chapter is based on the results of [47] for a rectilinear embedding of the embedded subgraph H . This approach first classifies geometric cuts into two classes – 1-D cuts and 2-D cuts – and shows that the number of all possible geometric 1-D and 2-D minimal cuts, given a rectilinear embedding of H , is $O(n^2)$ and $O(n^3)$, respectively, where n is the size of H . The $O(n^3)$ bound on the number of geometric minimal cuts makes the enumeration of geometric cuts feasible. In contrast, the corresponding bound of the classic min-cut problem in graphs is exponential. Based on interesting observations and dynamic connectivity data structures, the approach in [47] directly computes all geometric minimal cuts in $O(n^3 \log n (\log \log n)^3)$ time. A special case in which the inducing rectangle of a cut has a constantly bounded edge length is also considered, in which case the time complexity of the algorithm can be significantly improved to $O(n \log n (\log \log n)^3)$. Once all geometric minimal cuts are identified, the solution to the MGMC problem is reduced to computing their Hausdorff Voronoi diagram. The method in [47] revisits the plane sweep construction of the L_∞ Hausdorff Voronoi diagram of rectangles and presents the first output-sensitive algorithm for

its construction, which runs in $O((N + K) \log^2 N \log \log N)$ time and $O(N \log^2 N)$ space, where N is the number of rectangles and K is the complexity (or size) of the Hausdorff Voronoi diagram. The algorithm is based on a plane sweep construction in 3-D and uses several interesting data structures that can achieve an output-sensitive result.

The second approach is based on results in [34], and it is based on higher-order Voronoi diagrams. The embedded subgraph needs not be rectilinear, it can be arbitrary; moreover, it corresponds to polygonal shapes. Given a geometric graph with a subgraph embedded in the plane, the method in [34] iteratively identifies geometric minimal cuts and their *generators*,² based on an iterative construction of order- k Voronoi diagrams in increasing order of k . In the worst case, the method requires time $O(n^3 \log n)$ to compute higher-order Voronoi diagrams, plus time $O(n^3 \log n (\log \log n)^3)$ to answer connectivity queries on the graph G , assuming that the dynamic connectivity data structures of [20, 45] are used for this purpose. The resulting map provides a solution to the MGMC problem and reveals the Hausdorff Voronoi diagram of all geometric minimal cuts. In practice, typically, it is not necessary to guarantee the identification of all possible geometric cuts, in which case the iterative construction of [34] can be considerably sped up. Once a sufficient set of cut generators are identified, their weighted Voronoi diagram can be computed in a simple manner, providing a map that reliably approximates the solution to the MGMC problem. In the case of rectilinear embeddings, generators are simple axis-parallel line segments of constant weights. For arbitrary embeddings, generators correspond to portions of farthest Voronoi diagrams of cuts in the final map, and their weights correspond to farthest distance functions. Assuming that the number of iterations is kept to a small constant, as experimental results in [34] suggest, this results in an $O(n \log n (\log \log n)^3)$ algorithm for an approximate map solving the MGMC problem.

The Hausdorff Voronoi diagram of a set S of point clusters in the plane is a subdivision of the plane into regions, such that the Hausdorff Voronoi region of a cluster P is the locus of points *closer* to P than to any other cluster in S , where distance between a point t and a cluster P is measured as the *farthest* distance between t and any point in P . The Hausdorff Voronoi diagram first appeared in [11] as the *cluster Voronoi diagram*, where several combinatorial bounds were derived by means of a powerful geometric transformation in three dimensions and an $O(n^2)$ -time algorithm for its construction. A tight combinatorial bound on its structural complexity in the Euclidean plane, as well as a plane sweep construction were given in [33]. The L_∞ version of the problem was introduced in [32] for the VLSI *critical area extraction* problem for a defect mechanism called a *via-block*. In [32], the problem was reduced to an L_∞ Voronoi diagram of additively weighted line segments, and it was computed by a plane sweep procedure. The plane sweep construction was revisited in [47] and [39]. Additional work on Hausdorff Voronoi

²The generator of a cut is a portion of the farthest Voronoi diagram of the elements constituting the cut.

diagrams is included in [1, 9, 38]. In this chapter, we first present an output-sensitive version of the plane sweep in three dimensions for the L_∞ Hausdorff Voronoi diagram of rectangles as given in [47]. We also present the structural properties of this diagram and a simple two-dimensional plane sweep as given in [32, 39], which is not output sensitive; nevertheless, it achieves optimal $O(n \log n)$ -time and $O(n)$ -space in the case of non-crossing rectangles. In case of a large number of crossings, the size of the Hausdorff Voronoi diagram may be $\Omega(n^2)$ [11], and the $O(n^2)$ algorithm of [11] may result in a better approach.

This chapter is organized as follows. [Section 2](#) refers to [47], [Section 3](#) refers to [34], and [Section 4](#) to [39]. The sections are presented independently and they can be read in any order. In more detail, [Sect. 2.1](#) introduces the concepts of 1-D and 2-D geometric minimal cuts; [Sect. 2.2](#) presents the algorithms of [47] for computing the geometric minimal cuts; and [Sect. 2.3](#) presents the output-sensitive plane sweep algorithm in 3-dimensions of [47]. [Section 3](#) presents an iterative approach to the MGMC problem via higher-order Voronoi diagrams based on [34]. [Section 4](#) analyzes the structural complexity of the L_∞ Hausdorff Voronoi diagram and summarizes the simple two-dimensional plane sweep algorithm for its construction presented in [32, 39]. In [Section 5](#) we provide concluding remarks.

2 On Geometric Minimal Cuts and Their Map

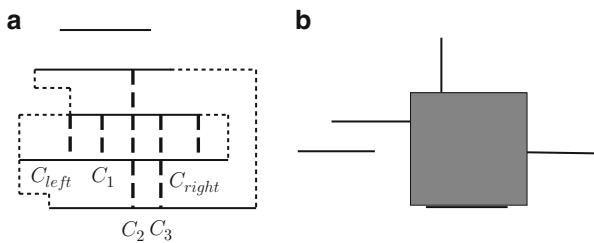
2.1 Geometric Cuts

Let $G = (V, E)$ be the undirected graph in an MGMC problem and $H = (V_H, E_H)$ be its planar subgraph embedded in the plane P with $|V| = N$, $|E| = M$, $|V_H| = n$, and $|E_H| = m$. Due to the planarity of H , $m = O(n)$. In this chapter, unless explicitly mentioned otherwise, all edges in H are either horizontal or vertical straight-line segments. A pair of vertices u and v in a graph are connected if there is a path in this graph from u to v . Otherwise, they are disconnected. A graph is connected if every pair of its distinct vertices is connected. Without loss of generality, G is assumed to be connected in this chapter. A cut C of G is a subset of edges in G whose removal disconnects G . A cut C is minimal if removing any edge from C no longer forms a cut.

Definition 1 Let R be a connected region in P , and $C = R \cap H$ be the set of edges in H intersected by R . The cut C is called a geometric cut induced by R if the removal of C from G disconnects G .

When there is no ambiguity of the region R , the cut induced by R is called a “geometric cut” for simplicity. For a given cut C , its *minimum inducing region* $R(C)$ is the minimum axis-aligned rectangle which intersects every edge of C (i.e., if we shrink the width or length of $R(C)$ by any arbitrarily small value ϵ , some edge in C will no longer be intersected by $R(C)$). For some geometric cut C , its minimum inducing region $R(C)$ could be degenerated into a horizontal or vertical

Fig. 1 (a) 1-D cuts C_1 , C_2 , and C_3 with C_3 being the minimal cut. (b) A 2-D cut bounded by 4 edges



line segment, or even a single point. It is easy to see that if $R(C)$ is not degenerate, it is unique.

Definition 2 A geometric cut C is called a 1-D geometric cut (or a 1-D cut) if $R(C)$ is a line segment. If $R(C)$ is an axis-aligned rectangle, then C is called a 2-D geometric cut (or a 2-D cut).

Definition 3 A geometric cut C is a geometric minimal cut if the set of edges intersected by any region obtained by shrinking $R(C)$ no longer forms a geometric cut.

The following lemma easily follows from the above definitions and the problem setting.

Lemma 1 Let C be any geometric minimal cut. If C is a 1-D cut, then each endpoint u of $R(C)$ is incident to either an endpoint of an edge e in H with the same orientation as $R(C)$ and $e \cap R(C) = u$ or an edge in H with different orientation as $R(C)$ (see Fig. 1a). If C is a 2-D cut, each bounding edge s of $R(C)$ is incident to either an endpoint v of an edge e in H of different orientation with s and $R(C) \cap e = v$, or an edge in H with the same orientation as s (see Fig. 1b; note that s could be incident to multiple edges).

From the above lemma, it is clear that each 1-D geometric minimal cut is bounded by up to two edges in H and each 2-D geometric minimal cut is bounded by up to 4 edges in H . For a cut C , let $B(C)$ denote the set of edges bounding C . Due to the minimality nature of C , removing any edge in $B(C)$ will lead to a non-cut. This means that any edge in $B(C)$ is necessary for forming the cut. However, this is not necessarily true for edges in $C \setminus B(C)$. Thus, a geometric minimal cut may not be a minimal cut. Note that for a given graph, the number of minimal cuts could be exponential. However, as shown later, the number of geometric minimal cuts is much less (i.e., bounded by a low degree polynomial).

For a 1-D geometric minimal cut C , it is possible that all edges in C have different orientation with $R(C)$. In this case, there may exist an infinite number of 1-D geometric minimal cuts, all cutting the same set of edges C (in other words, the minimum inducing region for such a 1-D geometric C is not unique). For example, sliding $R(C)$ along $B(C)$ will obtain an infinite number of 1-D minimal cuts.

Among these cuts, there are two extreme cuts, C_{left} and C_{right} (or C_{top} and C_{bottom} if $R(C)$ is horizontal), incident with the left and right (or up and bottom) endpoints of some edges in C , respectively. Since all these geometric cuts cut the same set of edges, they are counted as one cut.

2.2 Identifying Geometric Minimal Cuts

To solve the MGMC problem, the main idea in [47] is to first identify all possible geometric minimal cuts and then construct a Hausdorff Voronoi diagram of these cuts as the map \mathcal{M} . Thus, three major problems need to be solved:

1. Finding all 1-D minimal cuts;
2. Finding all 2-D minimal cuts; and
3. Efficiently constructing the Hausdorff Voronoi diagram for the geometric cuts.

This section discusses the main ideas for problems 1 and 2. The approach for problem 3 will be discussed in the next section.

2.2.1 Computing 1-D Geometric Minimal Cuts

As discussed in the previous section, a 1-D geometric minimal cut can be induced by either horizontal or vertical segments. The following lemma bounds from above the total number of 1-D geometric minimal cuts.

Lemma 2 *There are $O(n^2)$ 1-D geometric minimal cuts in H .*

Proof Only the 1-D cuts induced by vertical segments are considered. Cuts induced by horizontal segments can be proved similarly. If we place a vertical line through each vertex (or endpoint), then the plane P is partitioned into $O(n)$ vertical slabs, with each slab containing no endpoint in its interior. For a particular slab S containing k edges, say e_1, e_2, \dots, e_k , it could be shown that there are at most $O(k)$ 1-D geometric minimal cuts. To see this, all 1-D minimal cuts formed by the k edges are considered. A cut C is owned by an edge e_i if $e_i \in B(C)$. Clearly, each 1-D minimal cut has at least one owner. Now consider each edge e_i , it can only be the owner of up to two 1-D minimal cuts, one bounded by e_i from the bottom and one bounded by e_i from the top. Note that due to the fact that the cuts are all minimal, it is impossible to have two cuts bounded by e_i from the top (or bottom) simultaneously. Thus, each edge in S owns at most two 1-D geometric minimal cuts. Hence, the total number of 1-D geometric minimal cuts in S is $O(k)$. Summing over all slabs, the total number of 1-D geometric minimal cuts is $O(n^2)$. Thus, the lemma follows. \square

To compute the $O(n^2)$ 1-D geometric minimal cuts, since each cut is a set of edges $C \in H$ which appear consecutively in some slab and whose removal disconnects G , it is necessary to first (a) Identify all possible cuts in H , and then (b) For each possible cut, determine whether it is indeed a cut (such a test is called a *cut query*). To overcome the two difficulties, a straightforward way is to enumerate

all possible cuts, and for each possible cut C , use graph search algorithms (such as depth first search (DFS) or breadth first search (BFS)) to check the connectivity of $G \setminus C$. As shown in the proof of Lemma 2, the embedding plane P can be partitioned into $O(n)$ slabs, and for each slab with k edges, there are $O(k^2)$ subsets of consecutive edges. Thus, a total of $O(n^3)$ possible subsets need to be checked, and the connectivity checking for each subset takes $O(N + M)$ time. Therefore, this will lead to a total of $O(n^3(N + M))$ time.

To speed up the computation, the idea is to first simplify the graph G . Since the cuts involve only the edges in H , the connectivity in $G \setminus E_H$ will not be affected no matter which subset of edges in H is removed. To make use of this invariant, the connected components of $G \setminus E_H$ are firstly computed. For each connected component CC , it is contracted into a supernode v_{CC} . For each vertex $u \in H \cap CC$, an edge (u, v_{CC}) is added. Let the resulting graph be $G' = (V', E')$ (called contracted graph). The following lemma gives some properties of this graph.

Lemma 3 *The number of vertices in G' is $|V'| = O(n)$ and the number of edges in G' is $|E'| = O(n)$. Furthermore, a subset of edges in H is a cut of G if and only if it is a cut of G' .*

Proof Follows from the construction of G' . □

The above lemma shows that the size of G' could be much smaller than G . Thus, the time needed for answering a cut query is significantly reduced from $O(N + M)$ to $O(n)$.

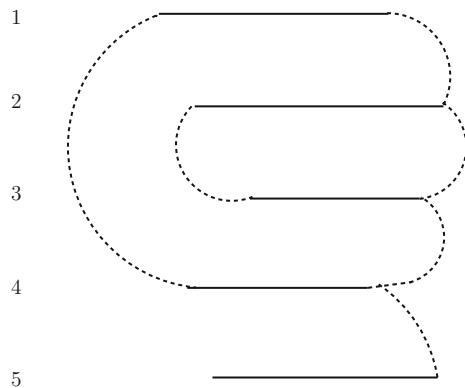
Converting Cut Queries to Connectivity Queries As discussed previously, to compute all 1-D geometric minimal cuts, it is necessary to check $O(n^3)$ possible subsets of edges in H . Many of them are quite similar (i.e., differ only by one or a small number of edges). Thus, it would be highly inefficient to handle them independently and answer each cut query by searching the graph G' . To yield a better solution, it is better to consider all these cut queries simultaneously and share the connectivity information among similar cut queries.

To share information among slightly different cut queries, one possible way is to use persistent data structure [10]. However, due to the fact that inserting (or removing) an edge could cause a linear, instead of a constant, number of 1-D geometric minimal cuts to be destroyed or generated, persistent data structures require substantial amount of updating after each insertion or deletion, thus they do not lead to a highly efficient solution.

To overcome this difficulty, the main idea is to further decompose each cut query into a set of connectivity queries with each connectivity query involving only one edge. The reduced granularity in query enables us to better share information among related cut queries.

Connectivity Query Before continuing the discussion on connectivity queries, some existing algorithms and data structures for the dynamic connectivity problem

Fig. 2 An example for 1DSlab, where the dotted lines are the edges in $G' \setminus H$. $\{2, 3\}$ and $\{5\}$ are the 1-D geometric minimal cuts



are briefly reviewed. In the fully dynamic connectivity problem, the input is a graph G whose vertex set is fixed and whose edges can be inserted and deleted. The objective is to construct a data structure for G so that the connectivity of any pair of vertices can be efficiently determined. Most fully dynamic connectivity data structures support the following three operations: (1) $Insert(e)$, (2) $Delete(e)$, and (3) $Connectivity(u, v)$, where the $Insert(e)$ operation inserts edge e into G , the $Delete(e)$ operation removes edge e from G , and the $Connectivity(u, v)$ operation determines whether u and v are connected in the current graph G . Extensive research has been done on this problem, and a number of results were obtained [12, 13, 17, 18, 20, 45]. In [20], Thorup et al. gave a simple and interesting solution for this problem which answers each connectivity query in $O(\log n)$ time and takes $O(\log^2 n)$ time for each update. Later Thorup gave a near-optimal solution for this problem [45] which answers each connectivity query in $O(\log n / \log \log \log n)$ time and completes each insertion or deletion operation in $O(\log n (\log \log n)^3)$ expected amortized time.

The approach described in this chapter uses the data structure in [45] for the MGMC problem. In practice (e.g., critical area computation), the simpler algorithm in [20] may be more practical. When the choice of the connectivity data structure is unclear, $MaxQU$ is used to represent the maximum of the connectivity query time and the update operation time.

Enumerating 1-D Geometric Minimal Cuts in a Slab The problem of identifying 1-D geometric minimal cuts in a vertical slab S with k edges (see Fig. 2) is to make use of the connectivity data structure. Clearly, there are $O(k^2)$ possible 1-D geometric cuts and $O(k)$ 1-D geometric minimal cuts. To find out the $O(k)$ minimal cuts from $O(k^2)$ possible locations, edges are sorted based on the y coordinates, and let $e_1 = (u_1, v_1), e_2 = (u_2, v_2), \dots, e_k = (u_k, v_k)$ be the k edges. A fully dynamic connectivity data structure $FDC(G')$ for the contracted graph G' is built, and then the algorithm for a slab (called 1DSlab) is run on it. The main steps of 1DSlab are the follows.

1. Initialize a variable $r = 1$ to represent the index of the next to-be-deleted edge.
2. Starting from e_r , repeatedly delete edges of S from G' according to their sorted order and store them in a queue Q . For each deleted edge e_i , query $FDC(G')$ about the connectivity of u_i and v_i . Stop the deletion until encountering the first edge e_j whose two endpoints u_j and v_j are disconnected or the last edge. In the latter case, insert all deleted edges back and stop.
3. Insert the deleted edges in Q back in the same order as they are deleted and updating $FDC(G')$. After inserting each edge e_i , query the connectivity of the two endpoints of e_j, u_j , and v_j .
4. If u_j and v_j are disconnected, add a forward pointer from e_i to e_j and insert edges in Q back to G' .
5. If u_j and v_j are connected, add a forward pointer from e_i to e_j , set $r = j + 1$, and repeat Steps 2–5 until encountering the last edge e_k . In this case, insert all remaining edges in Q back to G' and $FDC(G')$.
6. Reverse the order of the k edges and repeat the above procedure. In this step the added pointers are backward pointers.
7. For each edge e_j , find the nearest edge e_i which has a forward pointer to e_j and the nearest edge e'_i with a backward pointer to e_j . Output the set of edges between e_i and e_j (including e_i and e_j) as a 1-D geometric minimal cut and edges between e_j and e'_i (including e_j and e'_i) as another 1-D geometric minimal cut.

The correctness of the above algorithm is shown below.

Lemma 4 *Assume G' is originally a connected graph. In Step 2 of the 1DSlab algorithm, if u_i and v_i (the endpoints of the last deleted edge e_i) are connected, the set of deleted edges (i.e., e_r to e_i) does not form a cut in G' . On the other hand, if u_i and v_i are disconnected, then the set of deleted edges does form a cut in G .*

Proof Firstly, if u_i and v_i are disconnected, obviously the set of deleted edges forms a cut. Thus, only the case that u_i and v_i are connected is considered. In this case, the set of deleted edges does not form a cut.

This could be proved by induction on the number i of deleted edges. The base case is $i = 1$. In this case, since G' is originally a connected graph and u_i and v_i are still connected in $G' \setminus \{e_i\}$, obviously there exists no cut. Assume that after deleting $i - 1$ edges, the set of edges $S_{i-1} = \{e_1, e_2, \dots, e_{i-1}\}$ does not form a cut. Then since $G' \setminus S_{i-1}$ is a connected graph, after deleting edge e_i , the case becomes the same as the base case. Thus, the lemma follows. \square

Lemma 5 *In the 1DSlab algorithm, the disconnectivity of u_j and v_j in Step 4 or the connectivity of u_j and v_j in Step 5 implies that the set $S_{i,j} = \{e_i, e_{i+1}, \dots, e_j\}$ forms a cut in G' .*

Proof The first case (i.e., u_j and v_j are disconnected in Step 4) is obvious. For the second case (i.e., u_j and v_j are connected in Step 5), since u_j and v_j are disconnected before the insertion of e_i , it follows that $S_{i,j}$ is a cut. \square

The above lemmas indicate that the cut query can be answered by a sequence of connectivity queries.

Theorem 1 *The 1DSlab algorithm generates all 1-D geometric minimal cuts in the slab S in $O(k \text{Max}QU)$ time, where $\text{Max}QU$ is the maximum of the query time and the updating time of the $\text{FDC}(G')$ data structure.*

Proof The reason that the 1DSlab algorithm generates all 1-D geometric minimal cuts is shown below. By Lemmas 4 and 5, it is known that the forward pointers to e_j indicate all cuts whose edge set appears consecutively and ends at edge e_j . Similarly, the backward pointers to e_j indicate all cuts whose edge set appears consecutively and starts from e_j . By finding the nearest edges with forward and/or backward pointers to e_j , the algorithm identifies the (up to) two 1-D geometric minimal cuts starting from and ending at e_j . Since the computation is for every edge in S , it generates all 1-D geometric minimal cuts.

For the running time, it is clear that the sorting takes $O(k \log k)$ time. After sorting, in each of the forward and backward computations, every edge in S is deleted and inserted once. As for the connectivity queries, the endpoints of each edge can be queried multiple times (one in Step 2 and others in Step 3). To bound the total query time, the query on u_j and v_j in Step 3 is charged to edge e_i . Since e_i is inserted only once, it will be charged only once. Thus, each edge in S will be responsible for at most two queries. Therefore, the total number of connectivity queries is $O(k)$. Since the insertion, deletion, and query operations take no more than $O(\text{Max}QU)$ time, the theorem follows. \square

From the above theorem, it is known that even though there are $O(k^2)$ consecutively appeared subsets of edges that could be 1-D geometric minimal cuts, the connectivity data structure can reduce the time to near linear.

Generating 1-D Geometric Minimal Cuts The algorithm 1DSlab is combined with a plane sweep algorithm to generate all possible 1-D geometric minimal cuts in H .

A straightforward way of using 1DSlab is to partition the plane P into $O(n)$ slabs and apply the 1DSlab algorithm in each slab. This will lead to a $O(n^2 \text{Max}QU)$ -time solution. A more output-sensitive solution is to use the following plane sweep algorithm.

In the plane sweep algorithm, a vertical sweeping line L is used to sweep through all edges in H to generate those 1-D geometric minimal cuts induced by vertical segments. Similarly, those 1-D geometric minimal cuts induced by horizontal segments can be generated by sweeping a horizontal line. For the vertical sweeping line algorithm, the event points are the set V_H of endpoints in H . At each event point, the set of edges intersecting the sweeping line L forms a slab. However, instead of applying the 1DSlab algorithm to the whole set of intersecting edges, only a subset of edges are considered.

Let u be the event point. If u is the left endpoint of an edge e , then e is the new edge just encountered by L . Thus, it is only needed to identify all 1-D geometric minimal cuts which contain e . This means that the 1DSlab algorithm needs only to consider the minimal cut C_1 bounded by e from the bottom, the minimal cut C_2 bounded by e from the top, and all minimal cuts between the top edge e_t of C_1 and the bottom edge e_b of C_2 . To deal with this case, the 1DSlab algorithm can be easily modified. Particularly, the modified algorithm can start from e , have a backward computation, and stop at the first edge e_t , whose removal disconnects its two endpoints, to generate C_1 . Similarly, the algorithm can start from e and have a forward computation to generate C_2 . In this way, dealing with possibly many edges beyond e_t and e_b is avoided.

If u is the right endpoint of e , it is necessary to check those cuts containing e to see whether they are still geometric minimal cuts. It is also needed to check whether new cuts can be generated. Clearly, it is only needed to consider the edges between e'_t and e'_b , where e'_t is the top edge of the 1-D geometric minimal cut bounded by e from the bottom and e'_b is the bottom edge of the 1-D geometric minimal cut bounded by e from the top. Thus, such information from the previous step (i.e., the step before encountering u) are first obtained and then the 1DSlab algorithm on the subset of edges between e'_t and e'_b is applied directly.

Theorem 2 All 1-D geometric minimal cuts of H can be found in $O(n \times \text{Max } C \times \text{Max } QU)$ time, where $\text{Max } C$ is the maximum size of a 1-D geometric minimal cut.

Proof Follows from the above discussion. □

2.2.2 Computing 2-D Geometric Minimal Cuts

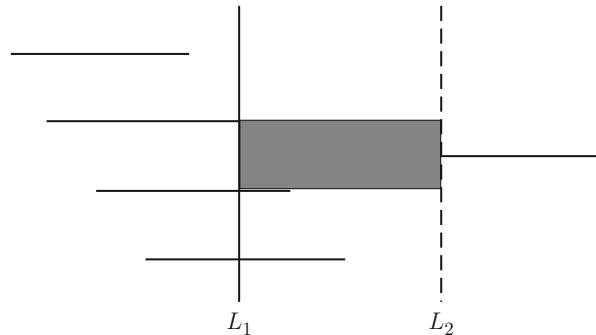
This section extends the algorithm in Sect. 2.2.1 to compute 2-D geometric minimal cuts.

To compute 2-D geometric minimal cuts, it is easy to see that a 1-D geometric minimal cut is a degenerate version of a 2-D geometric minimal cut. The only difference is that the minimum inducing region of a 1-D cut has two opposite sides degenerated to points. Thus, if two opposite sides of the minimum inducing region $R(C)$ of a 2-D cut C are conceptually “contracted” into “points,” the plane sweep algorithm for 1-D cuts can be applied to generate 2-D cuts.

To implement this idea, two parallel sweeping lines L_1 and L_2 (called primary and secondary sweeping lines) are used to bound the “contracted” sides of $R(C)$. By Lemma 1, it is known that each 2-D geometric minimal cut is bounded by the endpoints (or edges) of up to four edges. This suggests that the possible locations of L_1 and L_2 are the endpoints of the input edges. Similarly to the plane sweep algorithm for 1-D cuts, the edges in H are swept twice, one time vertically and the other time horizontally. The discussion below is focused on horizontal sweeping (i.e., using vertical sweeping lines).

The double plane sweep algorithm first sorts all edges in H based on the x coordinates of their left endpoints (for vertical segments, their upper endpoints are viewed as the left endpoints and their lower endpoints as the right endpoints).

Fig. 3 An example for illustrating the double plane sweep algorithm for 2-D cuts



Let $S_1 = \{e_1, e_2, \dots, e_n\}$ be the sorted order. The primary sweeping line L_1 stops at the left endpoint of each edge e in the order of S_1 . If the right endpoint of e is to the left of the left endpoint of the next edge $e' \in S_1$, the sweep line L_1 is moved to the right endpoint of e (see Fig. 3). If e is a cut by itself, L_1 is moved to the next edge in S_1 . If e is not a cut, L_1 is fixed and the secondary sweeping line L_2 is swept from the left endpoint of the next edge e' in S_1 to the last edge in S_1 . When L_2 stops, the plane sweep algorithm for 1-D cuts (in Sect. 2.2) is used to compute all 2-D geometric minimal cuts formed by the set S of edges intersecting the vertical region VR bounded by L_1 and L_2 .

For edges in S , there are two sorted orders S_L and S_U . S_L is sorted based on the y coordinates of the lower endpoints, and S_U is based on the y coordinates of the upper endpoints. (For horizontal edges, their left endpoints are viewed as upper endpoints and right endpoints as the lower endpoints. It is not difficult to see that the two sorted lists can be dynamically maintained in the double plane sweep algorithm.) S_L is used to generate cuts for edges above e and S_U is used to generate cuts for edges below e .

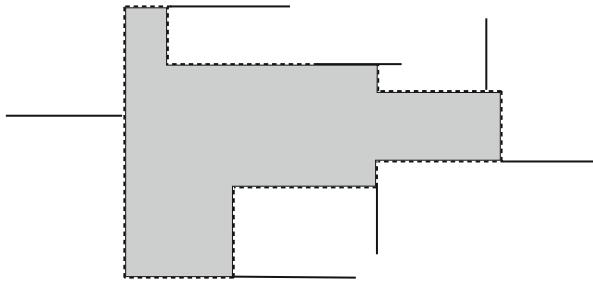
Each 2-D geometric minimal cut C in the vertical region VR is bounded by L_1 from left, L_2 from right, an edge e_u from above, and an edge e_d from below. Since all edges in S are in sorted order and $R(C)$ intersects edges in the same order in either S_L or S_U (depending on their relative locations to e), the 2-D geometric minimal cuts in VR can be viewed as 1-D geometric minimal cuts induced by vertical “segments” with a segment width equal to the horizontal distance of L_1 and L_2 . Thus, they can be generated by using the plane sweep algorithm for 1-D cuts. Furthermore, similar to the plane sweep algorithm for 1-D cuts, in VR it is only needed to consider those 2-D geometric minimal cuts containing edge e (see Fig. 4).

Once all the cuts in VR are identified, L_2 is moved to the next edge in S_1 . After sweeping L_2 , L_1 is moved to the next edge in S_1 and the above procedure is repeated until L_1 sweeps every edge.

The following lemma is used to analyze the double plane sweep algorithm.

Lemma 6 *There are at most $O(n^3)$ 2-D geometric minimal cuts in H .*

Fig. 4 Example of the double plane sweep algorithm for 2-D cuts. The region bounded by the dotted lines is the actual region where the algorithm searches for all 2-D geometric minimal cuts bounded by the leftmost segment from the *left*.



Proof By Lemma 1, it is known that the inducing region of each 2-D geometric minimal cut is bounded by an edge on each side. There are $O(n^2)$ pairs of edges to bound the left and right sides of the inducing region. For each pair, the vertical region is similar to a slab (according to the above discussion). By a similar argument in the proof of Lemma 2, it is easy to see that in each vertical region, there at most $O(n)$ 2-D geometric minimal cuts. Thus, the lemma follows. \square

For the running time of the double plane sweep algorithm, the primary sweeping line L_1 stops at $O(n)$ location. For each fixed location of L_1 , L_2 sweeps all edges not yet encountered by L_1 , whose number can be $O(n)$. For each vertical region bounded by L_1 and L_2 , it takes $O(\text{MaxC} \times \text{MaxQU})$ time (by Theorem 2). Thus, the total time is $O(n^2 \times \text{MaxC} \times \text{MaxQU})$.

Theorem 3 All 2-D geometric minimal cuts in H can be found in $O(n^2 \times \text{MaxC} \times \text{MaxQU})$ time.

Proof Follows from the above discussion. \square

Corollary 1 All geometric minimal cuts in the MGMC problem can be found in $O(n^3 \log n (\log \log n)^3)$ time in the worst case.

Proof Follows from Theorems 2 and 3, and article [45]. \square

Corollary 2 If the maximum size of a cut is bounded by a constant, then all geometric minimal cuts in H can be found in $O(n \log n (\log \log n)^3)$ -time.

Proof If the maximum size of a defect is bounded by some constant, the size of an inducing rectangle is bounded by a constant. Furthermore, since edges in VLSI design are separated by some constant distance, thus the total number of edges in an inducing rectangle is also a constant. In this case, the secondary sweeping line L_2 needs only to sweep a constant number of edges. Thus, the running time of both plane sweep algorithms is $O(n \times \text{MaxQU})$. Also from Theorems 2 and 3, article [45], the lemma follows. \square

2.3 Generating Map of Geometric Minimal Cuts

This section shows that Problem 3 can be solved by using the Hausdorff Voronoi diagram.

2.3.1 From Geometric Minimal Cuts to Hausdorff Voronoi Diagram

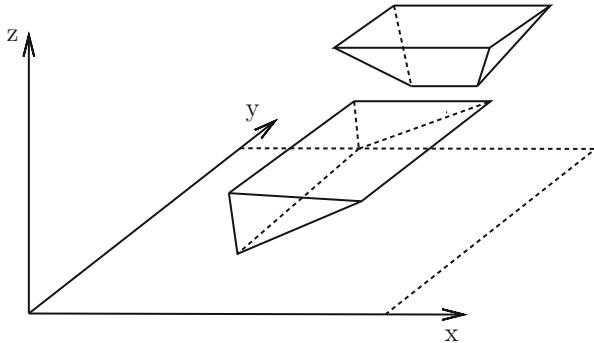
Given two sets A and B , the directed Hausdorff distance from A to B is $h(A, B) = \{\max_{a \in A} \min_{b \in B} d(a, b)\}$, and the undirected Hausdorff distance between A and B is $d_h(A, B) = \max\{h(A, B), h(B, A)\}$, where $d(a, b)$ is the distance between the points a and b . In this chapter, the L_∞ metric is used to measure the distance.

Given a set \mathcal{C} of geometric minimal cuts of H , the Hausdorff Voronoi diagram of \mathcal{C} is a partition of the embedding plane P of H into regions (or cells) so that the Hausdorff Voronoi cell of a cut $C \in \mathcal{C}$ is the union of all points whose Hausdorff distance to C is closer than to any other cut in \mathcal{C} . This means that for any point $p \in P$, if an L_∞ ball is grown from p (i.e., an axis-aligned square centered at p), the ball entirely contains $R(C(p))$ earlier than the minimum inducing region of any other cut, where $C(p)$ is the cut owning the Hausdorff Voronoi cell containing p . Thus, if such a map \mathcal{M} exists for the critical area extraction problem, then for each point p , one can easily determine the minimum size of a defect centered at p and disconnecting the circuit.

In the MGMC problem, two types of objects exist, the minimum inducing regions of 1-D geometric minimal cuts and the minimum inducing regions of 2-D geometric minimal cuts. For every 2-D cut C , the rectangle $R(C)$ is fixed. However, for a 1-D cut C , the location of $R(C)$ is not fixed, since there may be an infinite number of 1-D cuts cutting the same set of edges. In this case, the union of the infinite number of inducing segments $R(C)$ is used to represent the cut, which is a rectangle (denoted by $UR(C)$) bounded by the two extreme 1-D cuts C_{left} and C_{right} (or C_{top} and C_{bottom}) and the two bounding edges $B(C)$ of C . Thus, from thereafter, the objects of the Hausdorff Voronoi diagram are a set of axis-aligned rectangles.

As it is well known, the Hausdorff Voronoi diagram construction can be viewed as propagating a wave from each rectangle with unit speed. The Hausdorff distance from an arbitrary point p to an axis-aligned rectangle $R(C)$, corresponding to the minimum inducing region of a 2-D cut C , is considered to better illustrate the wave propagation. The Hausdorff distance of $d_h(p, R(C))$ is achieved at one of the four corner points, $v_1(C), v_2(C), v_3(C)$, and $v_4(C)$, of $R(C)$. Thus, $d_h(p, R(C)) = \max\{d_\infty(p, v_1(C)), d_\infty(p, v_2(C)), d_\infty(p, v_3(C)), d_\infty(p, v_4(C))\}$. To propagate a wave $W(C)$ from $R(C)$, the initial wavefront $\partial W(C)$ is the set of points whose Hausdorff distance to $R(C)$ is the minimum. Notice that when $R(C)$ has positive size (i.e., $R(C)$ is not a point), the minimum Hausdorff distance is positive (i.e., $\max\{a, b\}/2$, where a, b are the length and width of $R(C)$ respectively) and it is achieved when $d_\infty(p, v_1(C)), d_\infty(p, v_2(C)), d_\infty(p, v_3(C))$ and $d_\infty(p, v_4(C))$ are equal. The wavefront then expands to points having larger Hausdorff distance to $R(C)$. Since the Hausdorff distance to $R(C)$ is determined by the four corner points, an equivalent view is to propagate four distinct waves from the four corner points of $R(C)$ with each being an L_∞ ball. Let $B_1(C), B_2(C), B_3(C)$, and $B_4(C)$ be the

Fig. 5 Wavefronts of geometric minimal cuts



four L_∞ balls. The common intersections of the four balls constitute the wave of $R(C)$ (i.e., $W(C) = \cap_{i=1}^4 B_i(C)$). Initially, $\partial W(C)$ is empty. Once the size of the four balls $B_i(C)$ reaches the minimum Hausdorff distance to $R(C)$, their common intersection forms a segment s_C located at the center of $R(C)$ and parallel to the shorter side of $R(C)$. As $B_i(C)$ grows, $\partial W(C)$ becomes a rectangle.

To visualize the whole growing process, the waves can be lifted to 3-D with time being the third dimension. Thus, the wavefront of $R(C)$ becomes a 4-sided facet cone in the 3-D space and apedex at s_C (i.e., the apex is not in the xy plane due to its positive minimum Hausdorff distance; see Fig. 5). Each facet of $\partial W(C)$ forms a 45° angle with the xy plane.

For a 1-D cut C , its wavefront is slightly different. Let $UR(C)$ be the rectangle of C . Since $UR(C)$ is the union of an infinite number of inducing segments $R(C)$, the Hausdorff distance to an arbitrary point p is calculated differently. For a fixed inducing segment $R(C) \in UR(C)$, let $u_1(R(C))$ and $u_2(R(C))$ be its two endpoints. The Hausdorff distance from $R(C)$ to p is achieved at one of the two endpoints (i.e., $d_h(p, R(C)) = \max\{d(p, u_1(R(C))), d(p, u_2(R(C)))\}$). Thus, the wavefront of $R(C)$ is the common intersection of two L_∞ balls centered at the two endpoints, respectively, which is also a facet cone in 3-D space.

The Hausdorff distance from p to $UR(C)$ is the minimum distance from p to one of the inducing segments in $UR(C)$, that is,

$$d_h(p, UR(C)) = \min_{R(C) \in UR(C)} d_h(p, R(C)).$$

Thus, the wavefront of $UR(C)$ is the union of an infinite number of wavefronts, which is still a facet cone with similar shape to the wavefront of a 2-D cut. The difference between the wavefront of $UR(C)$ and that of a 2-D cut with exactly same-shaped $R(C)$ is that their respective s_C may orient differently and locate at different heights.

Lemma 7 *Let C be a 1-D or 2-D geometric minimal cut. At any moment, the wavefront of C is either empty or an axis-aligned rectangle. Furthermore, the*

wavefront in 3-D is a facet cone apexed at a segment and with each facet forming a 45° angle with the xy plane.

Proof Follows from the above discussion. \square

With the 3-D wavefronts of all cuts, the Hausdorff Voronoi diagram can be constructed by computing the vertical projection of the lower envelope of the set of 3-D wavefronts.

Lemma 8 *The Hausdorff Voronoi diagram can be obtained by projecting the lower envelope of the 3-D facet cones to the xy plane.*

Proof Follows from the definitions of the 3-D wavefronts and the Hausdorff Voronoi diagram. \square

2.3.2 Plane Sweep Approach and Properties of 3-D Cones and Hausdorff Voronoi Diagram

To efficiently construct the Hausdorff Voronoi diagram $HVD(C)$, the approach follows the spirit of Fortune's plane sweep algorithm for points [15] and sweeps along the x axis direction a tilted plane Q in 3-D which is parallel to the y axis and forms a 45° with the xy plane (see Fig. 6a). Q intersects the xy plane at a sweep line L parallel to the y axis.

Since every facet of a 3-D facet cone forms a 45° angle with the xy plane and apexed at either a horizontal or vertical segment, the intersection of Q and a cone $\partial W(C)$ is either a V -shape curve (i.e., consisting of a 45° ray and a 135° ray on Q) or a U -shape curve (i.e., consisting of a 45° ray, a segment parallel to L , and a 135° ray; see Fig. 6b). When the cone is first encountered, it introduces either a V -shape curve or a U -shape curve to Q . When L (or Q) moves, the curve grows and its shape may change from a V -shape to a U -shape. Accordingly, the lower envelope (or beach line) of all those V - or U -shape curves forms a monotone polygonal curve, instead of parabolic arcs as in Fortune's algorithm.

With the beach line, one might think of directly applying Fortune's algorithm to sweep the objects. However, due to the special properties of this problem, quite a few significant differences exist between the MGMC problem and the ordinary Voronoi diagram problem, which fail the Fortune's algorithm. Below is a list of major differences.

1. When a cone is first encountered by Q , its corresponding initial V - or U -shape curve may not necessarily be part of the beach line.
2. The initial V - or U -shape curve may affect a number of curves in the beach line.
3. Once a V shape moves away from the beach line, it may re-appear in the beach line in a later time.

The differences indicate that it is not sufficient to maintain only the beach line in order to extract all possible event points and produce the Voronoi diagram. More information of the arrangement of the V - and U -shape curves is needed. This seemingly suggests that an algorithm with running time of the order of the

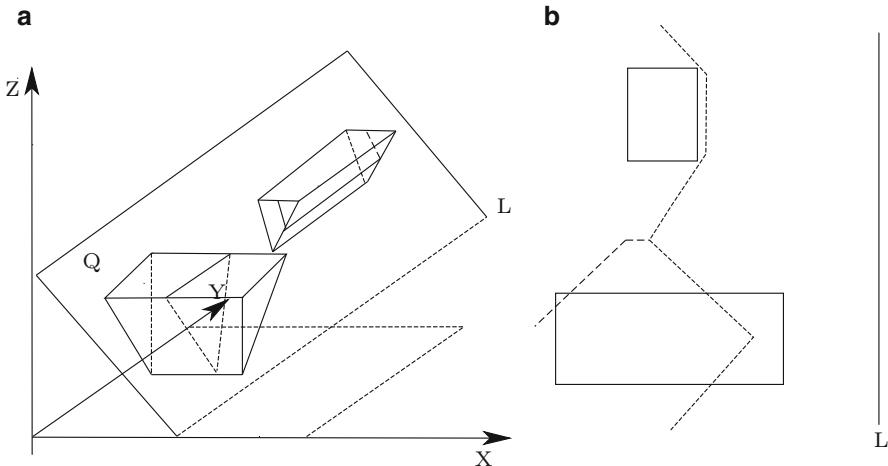


Fig. 6 (a) The intersections of 3-D plane Q and cones. (b) The Voronoi diagram with sweep line and beach line

complexity of the arrangement might be unavoidable. With several interesting observations and ideas, an output-sensitive plane sweep algorithm could be achieved. First, some basic properties of the Hausdorff Voronoi diagram of \mathcal{C} are presented below.

Definition 4 A 3-D facet cone $\partial W(C)$ is a U cone (or V cone) if its apex segment s_C is parallel to the y (or x) axis.

For U cones, let $\partial W(C)$ be any U cone with apex segment s_C , and v_1 and v_2 be the two endpoints of s_C . When the sweep plane Q first encounters $\partial W(C)$, it introduces a U -shape curve C_u to Q . Let r_l , r_r , and s_m be the left and right rays and the middle segment of C_u , respectively. Initially, s_m is the apex segment s_C , and r_l and r_r are the two edges of facet cone. When Q (or L) moves, C_u grows and always maintains its U -shape.

Lemma 9 Let $\partial W(C)$, C_u , r_l , r_r , and s_m be defined as above. When Q moves in the direction of the x axis, C_u is always a U -shape curve. The two supporting lines of r_l and r_r remain the same on Q , and the two endpoints of s_m (also the fixed points of r_l and r_r) move upwards in unit speed along the two supporting lines.

Proof Follows from the shape and orientation of a U cone. \square

For an arbitrary V cone $\partial W(C')$, let $s_{C'}$ be its apex segment and v'_1 and v'_2 be its two endpoints (or left and right endpoints). When Q first touches $\partial W(C')$ at v'_1 ,

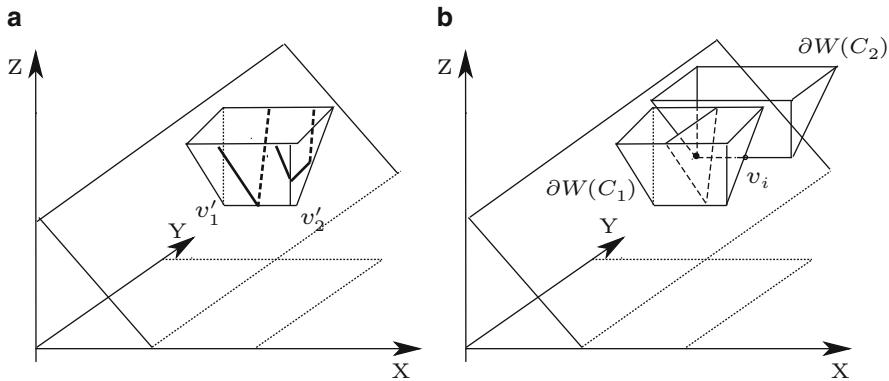


Fig. 7 (a) The V -shape curve changes to U -shape curve. (b) The V cone is hidden by another V cone

it generates a V -shape curve C'_v . C'_v remains a V -shape curve before encountering v'_2 . After that, C'_v becomes a U -shape curve (see Fig. 7a).

Lemma 10 Let r_l and r_r be the two rays of C'_v , and s_m be the middle segment of the U -shape curve C'_v after Q visits v'_2 . During the whole sweeping process, the supporting lines of r_l and r_r are fixed lines on Q . C'_v remains the same V -shape curve on Q before encountering v'_2 . s_m moves upwards in unit speed along the supporting lines of r_l and r_r after Q encounters v'_2 .

Proof Follows from the shape and orientation of a V cone. \square

As mentioned earlier, the apex segment of each 3-D cone is located at different height (i.e., its minimum Hausdorff distance). The heights and shapes of the 3-D cones are the main reasons which cause the three differences in the MGMC problem. For example, due to the existence of height in a 3-D cone, the initial curve created by a newly encountered cone may be above the beach line (i.e., Difference (1)). In addition due to the different size of the initial curve (unlike the ordinary Voronoi diagram in which the initial curve is a vertical ray), it may intersect a number of segments or rays of the beach line (i.e., Difference (2)). More importantly, due to the existence of U - and V -shape curves, a V -shape curve which is not part of the beach line could become part of the beach line in a later time (i.e., Difference (3)). It is shown in the following lemma.

Lemma 11 Let $\partial W(C_1)$ be either a U or V cone and $\partial W(C_2)$ be a V cone with its left endpoint v_1 of s_{C_2} being inside of $\partial W(C_1)$ and its right endpoint v_2 being outside of $\partial W(C_1)$. If $\partial W(C_2)$ is not entirely contained by the union $\cup_{C_i \in \mathcal{C}; C_i \neq C_2} \partial W(C_i)$, the V -shape curve C_2 introduced by $\partial W(C_2)$ will be hidden by the beach line at the

beginning and will become part of the beach line later. This is the only case in which a hidden U- or V-shape curve can appear in the beach line.

Proof To simplify the proof, $\partial W(C_1)$ and $\partial W(C_2)$ are assumed to be the only two cones in \mathcal{C} (see Fig. 7b). The multiple cones case can be proved similarly by induction.

By the definition of the two 3-D cones, it is known that Q first encounters $\partial W(C_1)$ and generates a curve C_1 on Q . C_1 is the only curve in the beach line. When v_1 of $\partial W(C_2)$ is first encountered by Q , it introduces a V-shape curve C_2 on Q . Since v_1 is inside $\partial W(C_1)$ and every facet of the two cones forms a 45° angle with the xy plane, the two rays of C_2 will be inside the region defined by the two rays of C_1 . This means C_2 will not be part of the beach line (or lower envelope). Since $\partial W(C_2)$ is not entirely inside $\partial W(C_1)$, v_2 must be outside of $\partial W(C_1)$. Let v_i be the intersection point of s_{C_2} and the wall of $\partial W(C_1)$. When Q sweeps through v_i , the apex point of the V-shape curve C_2 will intersect the U-shape curve C_1 (note that at this moment C_1 can only be a U-shape curve even if $\partial W(C_1)$ is a V cone), since C_1 is the intersection of Q and $\partial W(C_1)$ and v_i is the intersection of $\partial W(C_1)$ and s_{C_2} . Thus, C_2 becomes part of the beach line.

To show that this is the only case where a hidden curve could appear in the beach line, first it is noted that the apex segment s_{C_2} of $\partial W(C_2)$ cannot be completely outside of $\partial W(C_1)$, otherwise the initial curve of C_2 will be part of the beach line. Second, $\partial W(C_1)$ cannot be a U cone. If this is the case, then s_{C_2} is either partially or entirely inside $\partial W(C_1)$. For the first case, the middle segment s_m of the initial U-shape curve C_2 will intersect one of the two rays of C_1 , which makes C_2 be part of the beach line. For the second case, the initial U-shape curve C_2 will be hidden by C_1 . When Q moves, only the middle segment s_m of C_2 moves upwards in unit speed along the two rays of C_2 (by Lemma 9). If $\partial W(C_1)$ is a U cone, then by Lemma 9, the middle segment of C_1 will also move upwards in unit speed. Thus, it will never catch up s_m . Therefore C_2 will never be part of the beach line. Similarly the same for the case $\partial W(C_1)$ is a V cone can be proved. Thus, $\partial W(C_2)$ has to be a V cone. In this case, if s_{C_2} is completely inside of $\partial W(C_1)$, then by the same argument, it can be shown that C_2 will never be part of the beach line. Thus, the lemma follows. \square

In the above lemma, the point v_i indicates that when Q sweeps it, the beach line is having a topological structure change. Thus, v_i needs to be an event point for the plane sweep algorithm. However, since v_i is the intersection point of an apex segment and a 3-D cone, it has to be computed on the fly. This indicates that in the MGMC problem there is a new type of event points.

The ideas for constructing the Hausdorff Voronoi diagram $HVD(\mathcal{C})$ are discussed below. First, the bisector of two rectangles (or cuts) is considered. Let C_1 and C_2 be two axis-aligned rectangles in \mathcal{C} . The bisector of C_1 and C_2 is a segment with two rays as shown in Fig. 8. Each bisector contributes two vertices to the Hausdorff Voronoi diagram. Hence, the Hausdorff Voronoi diagram consists of two types of vertices: (a) The intersection points of the bisectors; and (b) The vertices of the bisectors.

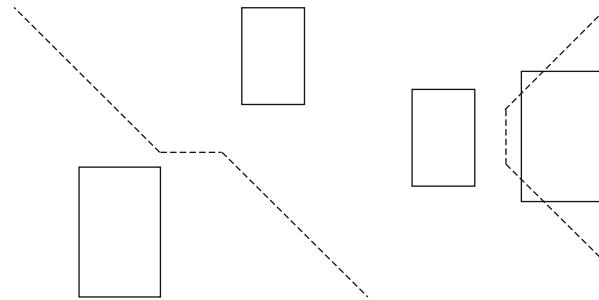


Fig. 8 Bisector is composed by a segment with two half-lines

Lemma 12 Let \mathcal{C} be a set of N rectangles. The edges of $HVD(\mathcal{C})$ are either segments or rays, and the vertices of the $HVD(\mathcal{C})$ are either the vertices of bisectors or the intersections of bisectors.

Proof Follows from the above discussion. \square

To obtain a plane sweep algorithm, it is needed to design data structures to maintain the beach line and the event points. In the MGMC problem, the beach line is the lower envelope of the set of V - and U -shape curves and is a y -monotone polygonal curve. For non-disjoint 3-D cones, the complexity of the beach line may not be linear in the number of the rectangles in \mathcal{C} . Figure 9b shows a newly generated U -shape curve intersecting the beach line a number of times and contributing multiple edges to it. Consequently, the complexity of $HVD(\mathcal{C})$ is not linear. The following lemma is a straightforward adaptation of Theorem 1 in [33] for the L_∞ metric.

Lemma 13 The size of the L_∞ Hausdorff Voronoi diagram of N rectangles is $O(N + M')$, where M' is the number of intersecting pairs of rectangles. In the worst case, the bound is tight.

2.3.3 Events

For event points, it is needed to detect all events that cause the beach line to have topological structure changes. More specifically, it is necessary to identify all the moments when a U - or V -shape curve is inserted or deleted from the beach line. There are two ways by which a curve could appear in the beach line:

- (A) A newly generated U - or V -shape curve becomes part of the beach line.
- (B) A hidden V -shape curve appears in the beach line.

There are also two ways for a curve or a portion of a curve to disappear from the beach line:

- (C) A curve (or part of the curve) becomes hidden by a newly generated curve.
- (D) A U -shape curve (or part of the U -shape curve) moves out of the beach line.

Fig. 9 The beach line L is intersected by a new U cone (dashed line) or V cone (dotted line)

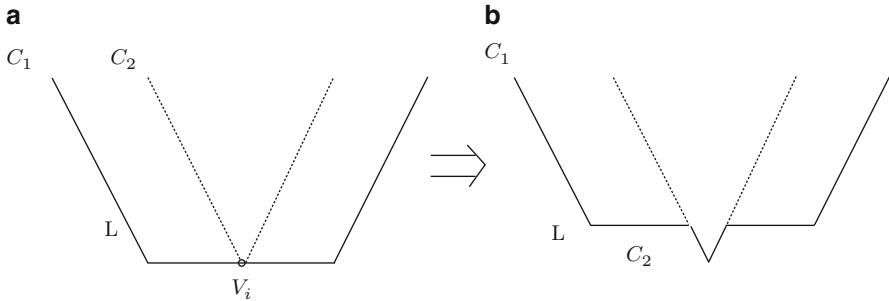
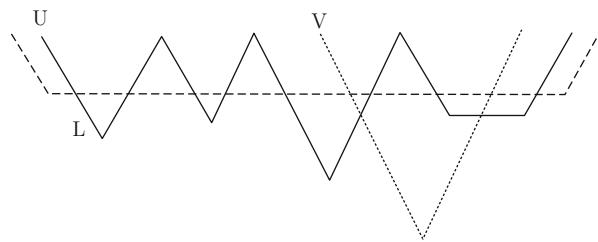


Fig. 10 (a) The *bottom* segment of U -shape curve C_1 reached V_i . (b) The hidden V -shape curve C_2 appeared on the beach line L

For (A), it is known that this is caused by the sweep plane Q encountering a new 3-D cone $\partial W(C)$. Such events can be detected in advance and are called *site events*. A site event, however, does not necessarily lead to a topological structure change to the beach line, since the new curve C can be hidden by the beach line. If $\partial W(C)$ is a U cone, the two endpoints of s_C are encountered by Q at the same time and either of them can be treated as a site event. If the corresponding U -shape curve C is not hidden, it may intersect the beach line multiple times as shown in Fig. 9. In this case, it is necessary to update the beach line for each breakpoint introduced by C . Consequently, the U -shape curve C will be partitioned into multiple pieces. Each piece is either a part of the beach line (called unhidden portion of C) or hidden by other U - or V -shape curves in the beach line (called a hidden portion of C). If $\partial W(C)$ is a V cone, then the left endpoint v_1 of s_C is encountered first and can be viewed as a site event. When Q sweeps the right endpoint v_2 of s_C , the C changes from a V -shape curve to a U -shape curve. To distinguish from the site event, v_2 is called as a U event. For unhidden V -shape curve C , it intersects the beach line at most twice (see Fig. 9).

For (B), it occurs when an unhidden portion of the bottom segment s_m of a U -shape curve C_1 moves upwards and encounters the apex point of a hidden V -shape curve C_2 . This kind of events is called V events (see Figs. 10 and 7b). The V events are not known in advance and need to be computed by using the saved information in the data structures. Note that when s_m moves upwards, its hidden portions may also encounter the apex point of some hidden V -shape curve. In this case, it is not viewed as an event since the beach line has no topological structure

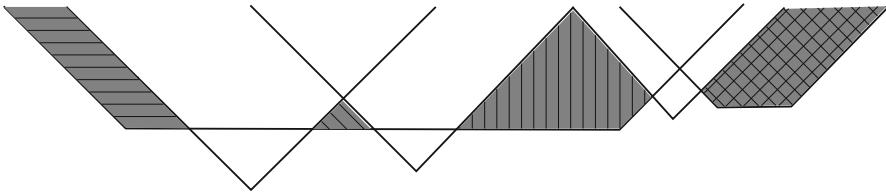


Fig. 11 Dominating regions of unhidden portions of U -shape curves

change, thus generating no new Voronoi vertex or edge. To distinguish the two cases, each unhidden portion of a U -shape curve is associated with a region, called *dominating region* (see Fig. 11), which is the region swept by the unhidden portion when it moves upwards. The dominating regions could have a few different shapes (see the shaded regions in Fig. 11), with each of them bounded by zero, one, or two 45° rays, zero, one, or two 135° rays, and an unhidden portion of the bottom segment s_m of some U -shape curve. Clearly, for a hidden V -shape curve to cause a V event, its apex point has to fall in a dominating region of an unhidden portion of a U -shape curve. Thus, to capture all V events, it is only needed to focus on those hidden V -shape curves whose apex points fall in the dominating regions.

For (C), it is obviously caused by a site event and thus can be detected in advance. For (D), the disappearing U -shape curve C (or its unhidden portion) is caused by the upward movement of the bottom segment of C . It involves three curves, C and its immediate left and right neighbors C' and C'' in the beach line. Since this event is similar to the circle event in the computation of the standard Voronoi diagram, it is also called here a circle event. The circle events cannot be detected in advance and have to be computed on the fly.

Thus, there are in total four types of events, site events, circle events, U events, and V events. The ideas on how to handle these events are discussed in the next section.

2.3.4 Data Structures and Events Handling

To construct $HVD(C)$, doubly connected edge lists are used to store $HVD(C)$. Two data structures for the plane sweep algorithm are also needed: an event queue and a sweep plane status structure representing the beach line.

The status structure for the beach line consists of three balanced binary search trees \mathcal{T} , $\mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$. \mathcal{T} stores the y -monotone polygonal curve of the beach line. Each part of the curve corresponds to a V -shape curve or an unhidden portion of a U -shape curve. The leaves of \mathcal{T} correspond to the V -shape curves and the unhidden portions of the U -shape curves on the beach line sorted by their y coordinates. Each leaf also stores location information of the corresponding 3-D cone. The internal nodes of \mathcal{T} adjacent to the leaves represent the breakpoints (i.e., the intersection of a pair of U or V curves) on the beach line. A breakpoint is stored at an internal node by an ordered tuple of curves $\langle C_i, C_j \rangle$, where C_i is the left curve of the breakpoint and C_j is the right curve of the breakpoint. $\mathcal{T}_{\pi/4}$ is used to maintain the

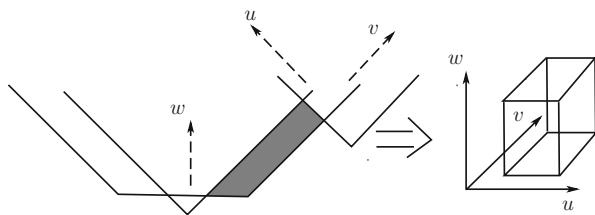
orders (along the norm direction) of the 45° rays of all U - or V -shape curves which appear in the beach line. Similarly, $\mathcal{T}_{3\pi/4}$ maintains the orders of the 135° rays of U - or V -shape curves which appear in the beach line. Each leaf node in $\mathcal{T}_{\pi/4}$ or $\mathcal{T}_{3\pi/4}$ represents a 45° or 135° ray, and each ray corresponds to a U - or V -shape curve (represented by a leaf node in \mathcal{T}) in the beach line. For each leaf node of $\mathcal{T}_{\pi/4}$ and $\mathcal{T}_{3\pi/4}$, a pointer pointing to the corresponding leaf node in \mathcal{T} is maintained. In this case, by doing binary search on $\mathcal{T}_{\pi/4}$ and $\mathcal{T}_{3\pi/4}$ and the pointers between the trees, the positions in the beach line for the apex points of each newly encountered cone at a site event could be located in $O(\log N)$ time, and the beach line is updated in $O(k \log N)$ time, where k is the number of breakpoints destroyed and created after inserting the newly encountered U - or V -shape curve into the beach line.

The event queue \mathcal{Q} is implemented by a priority queue, where the priority of an event is the x coordinate of the corresponding event point. If two points have the same x -coordinate, the one with larger y coordinate has the priority. All the site events and U -events are known in advance and are stored in \mathcal{Q} . The main challenge is to detect the V events.

For a V event, it is known that it occurs when the apex point of a hidden V -shape curve C_2 appears in the beach line. To detect such events, it is necessary to store in the data structure the information of all hidden V -shape curves. This could potentially require us to maintain the whole arrangement of all curves on \mathcal{Q} and therefore results in unnecessarily high computational cost. To efficiently detect all possible V events, the main idea is not to maintain the arrangement, but rather to use the properties of V events to convert the problem into a query problem in 3-D. To achieve this, first it is easy to see that a V event is always caused by the upward movement of an unhidden portion of the bottom segment s_m of some U -shape curve C_1 and occurs when s_m coincides with the apex point v of a hidden V -shape curve C_2 (by Lemma 11). Thus, in order to detect all possible V events, it is needed to solve the following two difficulties: (1) Identify the next V event associated with each unhidden portion of a U -shape curve, and (2) for each newly encountered hidden V -shape curve (in a site event), find the unhidden portion of a U -shape curve which will later cause a V event involving this V -shape curve. There is another one (Difficulty (3)) related to the two difficulties: How to find the exact boundary of the dominating region for a given unhidden portion c of a U -shape curve.

First Difficulty (1) is considered. For this difficulty, it is known that the next V event associated with an unhidden portion c of a U -shape curve C_1 is the hidden V -shape curve C_2 whose apex point v lies inside the dominating region of c and is the closest to the bottom segment s_m of C_1 . However, as it is noticed in the last section, the dominating region could have various shapes which seemingly suggest that it is costly to find the next V event even if the dominating region is known. To overcome this difficulty, first it is observed that the dominating region is bounded by 45° and 135° rays and the bottom segment. From Lemmas 9 and 10, it is known that the rays of any U or V curve have fixed directions (e.g., forming 45° and 135° angles with the y axis) and their supporting lines remain the same during the whole sweeping process. This suggests that the dominating regions can be orthogonalized in 3-D space. The three dimensions of the new

Fig. 12 A dominating region is converted to a 3-D box in the orthogonalized 3-D space for MD



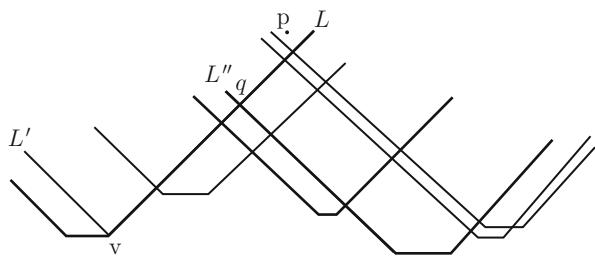
space are the orthogonal directions of the 45° and 135° lines in the sweep plane Q and the orthogonal direction of the bottom segment (or the y axis). In this way, each dominating region is converted into a (possibly unbounded) box in 3-D (see Fig. 12).

To efficiently find the next V event in the orthogonalized dominating region, the apex points of all V -shape curves are processed into a 3-D dynamic range search tree data structure MD [29, 31]. In MD, the apex point of each hidden V -shape curve is mapped into a 3-D point. Thus, for a particularly dominating region R , its orthogonalized box can be used as the query range to find the closest apex point (among all hidden V -shape curves whose apex points fall in R) to the unhidden portion of the bottom segment of a U -shape curve in $O(\log^2 N \log \log N)$ time [29].

To efficiently maintain the MD, it can be built in advance for all possible V -shape curves. In each node p of the MD tree, a mark is stored to indicate whether there is any active V -shape curve in the subtree rooted at p . A V -shape curve C is active if its corresponding 3-D cone has already been encountered by the sweep plane Q , and C has not yet changed to a U -shape curve due to a U event. In this way, it is only needed to change the marks when the status of a V -shape curve changes and therefore avoid complicate updating (such as tree rotation) to the MD.

To make use of MD, it is necessary to identify all scenarios in which it is needed to either update MD or query MD for detecting potential V events. First, it is noticed that a site event or a U event could introduce (a) a new U -shape curve C to the beach line and generate a set of unhidden portions of C and (b) a hidden V -shape curve C' . Thus, for (a), in each such event, a 3-D range query in MD is performed for each unhidden portion c_i to find the closest hidden V -shape curve to c_i in its dominating region and insert a V event into the event queue \mathcal{Q} . For (b), it is necessary to find out the exact dominating region R which contains the apex point of the newly encountered hidden V -shape curve C' (i.e., Difficulty (2)) and then insert the hidden V event into MD, since the V event of C' might be the new next V event of the unhidden portion c of R . (The idea for this challenging problem will be discussed later.) Second, after a V event, it is also necessary to perform a 3-D range query in MD to find the closest hidden V -shape curve to the new U -shape curve converted from the V -shape curve of the V event. Third, if an unhidden portion c of a U -shape curve C disappears from the beach line (e.g., a circle event), its associated V event is deleted from \mathcal{Q} , since c will never appear in the beach line again by Lemma 11.

Fig. 13 Finding the dominating region of p



The ideas for Difficulty (2) (i.e., finding the unhidden portion of the dominating region containing the apex point of a hidden V -shape curve in a site event) are discussed below. Let p be the apex point of the hidden V -shape curve. As shown in Fig. 13, finding the four rays (two 45° rays and two 135° rays) bounding p does not necessarily give us the correct dominating region. This is because the rays bounding p could be stopped by the rays bounding the actual dominating region. For example, L is stopped by L'' at point q . Thus, a ray inside a dominating region may not be a ray bounding that dominating region.

Definition 5 A ray (or a portion of a ray) is active if it bounds some dominating region and inactive otherwise.

Let l be any ray in the beach line. If starting from the bottom (i.e., the endpoint) of l and walking along it, l is active until it is stopped by some other ray and thus becomes inactive. It is easy to see that once a ray becomes inactive, it will never be active again. In Fig. 13, L is active until stopped by L'' and will no longer be active. A ray has two sides and it may not be active on both sides simultaneously. Also one side of a ray may bound more than one dominating region (L'' bounds the dominating regions generated by U -shape curves with bottom segments a and c). It is easy to see that for any ray, there is always one side bounding at most one dominating region. Otherwise, there will be an intersection between two dominating regions, thus contradicting the definition of dominating regions.

With these observations, to find out the actual dominating region of p , first the four rays bounding it are found by searching the $\mathcal{T}_{\pi/4}$ and $\mathcal{T}_{3\pi/4}$ trees. Further, it is needed to find out whether or not these rays are active on the sides containing p . To determine this, it is needed to know whether each of the four rays has been stopped by other rays. If the ray L which stops these rays can be found out, then it means that p belongs to the dominating region bounded by L . This means that for a given ray, it is needed to have a way to efficiently determine which ray stops it.

To achieve this, the $\mathcal{T}_{\pi/4}$ and $\mathcal{T}_{3\pi/4}$ trees are augmented. For each node of the trees, the z -coordinate of the lowest bottom (segment) of all rays in the subtree rooted at that node is stored. The z -coordinate of the bottom segment of a V -shape curve is the z -coordinate of its apex point minus the length of bottom segment of the

3-D V cone. With the augmented information, the ray L'' which stops a ray L can be searched in the two trees in $O(\log N)$ time.

To better understand the idea, consider the example in Fig. 13. Let L be a 45° ray with its endpoint v . A ray L'' which stops L is a 135° ray if it exists. To find L'' , another 135° ray L' with endpoint v is created. First the position of L' is searched in $\mathcal{T}_{3\pi/4}$. Then it is only needed to find the ray between L' and p (in the direction of L) with (1) lower z -coordinate than v and (2) the closest z -coordinate. This can be done by following the searching path of L' in $\mathcal{T}_{3\pi/4}$ upwards until finding a node satisfying the two conditions and then moving downwards to locate the ray L'' . In Fig. 13, the dominating region with bottom segment b dominates p .

Lemma 14 *It takes $O(\log N)$ time to find out the exact dominating region of an unhidden portion c for the apex point p of the hidden V -shape curve C' generated by the site event.*

Proof Follows from the above discussion. \square

It is not difficult to see that the augmented information can be maintained during the whole sweep process within the same time bound. With the augmented information, the exact boundary of a dominating region R for an unhidden portion c of a U -shape curve (i.e., Difficulty (3)) can also be found in $O(\log N)$ time following the same idea (i.e., finding the rays stopping the bounding rays of R).

Circle events can be handled in a way similar to the standard Voronoi diagram. More specifically, every new triple of consecutive U - or V -shape curves that appear in the beach line is checked. If such a new triple has converging breakpoints, the event is inserted into the event queue \mathcal{Q} . Furthermore, for all disappearing triples, the corresponding event is de-queued from \mathcal{Q} if it has been inserted.

2.3.5 Algorithm and Analysis

The entire plane sweep algorithm is described below. The main steps of the algorithm are as follows.

Algorithm HAUSDORFF-VORONOI-DIAGRAM(\mathcal{C})

Input. A set \mathcal{C} of axis-aligned rectangles (or geometric minimal cuts) in the plane.

Output. The Hausdorff Voronoi diagram in a doubly connected edge list \mathcal{D} .

1. Initialize the event queue \mathcal{Q} with all site events and U events; initialize empty sweep plane status structures \mathcal{T} , $\mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$, and an empty doubly-connected edge list \mathcal{D} ; initialize a 3-D range search tree MD for all possible V -shape curves with all nodes marked as inactive.
2. **while** \mathcal{Q} is not empty
3. **do** Remove an event with the smallest x -coordinate from \mathcal{Q} .
4. **if** the event is a site event, **then** HANDLE-SITE-EVENT.
5. **if** the event is a circle event, **then** HANDLE-CIRCLE-EVENT.
6. **if** the event is a U -event, **then** HANDLE-U-EVENT.
7. **else** the event is a V event; HANDLE-V-EVENT.

HANDLE-SITE-EVENT

1. Let C be the new curve. If the status structure is empty, insert C into it and mark the apex point in MD as active if it is a V cone. Otherwise, continue with steps 2–8.
2. If C is a V -shape curve, mark its apex point in MD as active and continue with steps 3–5.
3. Locate the position of the apex point of C in the beach line by searching $\mathcal{T}, \mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$.
4. If C is not hidden, insert C to the beach line by updating $\mathcal{T}, \mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$. This includes inserting C into the beach line, computing new breakpoints, inserting possible circle events into \mathcal{Q} , and removing all curves hidden by C from the beach line. If an unhidden portion of a U -shape curve is removed, delete its corresponding V event from \mathcal{Q} . If an unhidden portion of a U -shape curve is partially blocked by C , search for its closest hidden V -shape curve in the reduced dominating region if necessary.
5. Else if (the apex of) C is inside the dominating region an unhidden portion c of a U -shape curve C' , update the associated V event of c if needed.
6. Else if C is a U -shape curve, continue with steps 7–8.
7. Locate the position of C in the beach line by searching $\mathcal{T}, \mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$.
8. If C is not fully hidden, insert C into the beach line by updating $\mathcal{T}, \mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$. This includes computing possibly multiple breakpoints and the corresponding circle events for all its unhidden portions, removing blocked curves (similarly to Step 4), and finding the possible V event for each unhidden portion of C and partially blocked unhidden portion.

HANDLE-CIRCLE-EVENT

1. Update the beach line by updating $\mathcal{T}, \mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$. Delete unnecessary circle events from \mathcal{Q} involving the part disappearing from the beach line. If an unhidden portion of a U -shape curve moves out of the beach line, delete its associated V event.
2. Add the vertex to \mathcal{D} . Two new breakpoints of the beach line will be traced out.
3. Check the new triples involving the left or right neighbor of the disappearing part and insert the corresponding circle event into \mathcal{Q} , if necessary.

HANDLE-U-EVENT

1. If the V -shape curve C introduced by the V cone appeared on the beach line, the corresponding part of the beach line is changed from a V -shape curve to a U -shape curve. Update $\mathcal{T}, \mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$, and add vertex to \mathcal{D} . Also find the possible V event for C by querying MD, and insert it into \mathcal{Q} .
2. Mark the node in MD containing C as inactive, and update its ancestors if needed.
3. Detect the circle events and insert them into \mathcal{Q} if necessary.

HANDLE-V-EVENT

1. Update $\mathcal{T}, \mathcal{T}_{\pi/4}$, and $\mathcal{T}_{3\pi/4}$ by inserting the V -shape curve C into the beach line. Create new breakpoints, detect the possible circle event, and insert them into \mathcal{Q} .
2. For the corresponding unhidden portion c of a U -shape curve C' , break it into two unhidden portions, c' and c'' , and find possible new V event for c' and c'' , respectively by querying MD.

Theorem 4 *The L_∞ Hausdorff Voronoi diagram $HVD(\mathcal{C})$ of a set \mathcal{C} of geometric minimal cuts (or axis-aligned rectangles) can be constructed by a plane sweep algorithm in $O((N + K) \log^2 N \log \log N)$ time, where $N = |\mathcal{C}|$ and K is the complexity of the Hausdorff Voronoi diagram.*

Proof The discussion of the algorithm shows that $HVD(\mathcal{C})$ can be constructed by a plane sweep algorithm. Thus, the proof focuses on the time complexity.

First, site events are considered. Let C be a newly encountered U - or V -shape curve. If C is a V -shape curve, it is either part of the wavefront or a hidden V -shape curve. For the former case, the computation cost includes inserting C into the beach line and updating MD and \mathcal{D} . The cost for these is $O(\log^2 N \log \log N + k \log N)$, where k is the total number of breakpoints hidden by C . Since each breakpoint corresponds to a Voronoi vertex in $HVD(\mathcal{C})$, the cost of $O(k \log N)$ can be charged to the breakpoints (and their corresponding Voronoi edges). Thus, each will be charged a cost of $O(\log N)$. The cost of $(\log^2 N \log \log N)$ can be charged to each V -shape curve. For the latter case, if C is not in the dominating region of any unhidden portion of a U -shape curve, the only cost is $O(\log^2 N \log \log N)$, which can be charged to C . If C is inside the dominating region of some unhidden portion c of a U -shape curve which can be checked in $O(\log N)$ time, it is also needed to check whether C is the closest V -shape curve to c , and this can be done in $O(1)$ time. Thus, in the latter case, C is charged a cost of $O(\log^2 N \log \log N)$. If C is a U -shape curve, it could be fully hidden by the beach line, and thus, will never appear in the beach line in a later time. In this case, the total cost is $O(\log N)$, which can be charged to C . If C appears in the beach line and contributes some unhidden portions, then it is needed to update the sweep plane status structure, which takes $O(k \log N)$ time, and find the closest V -shape curve for each unhidden portion c . The cost of $O(k \log N)$ can be evenly charged to all hidden and newly created breakpoints. Thus, each of them will be charged a cost of $O(\log N)$. The closest V -shape curve to each unhidden portion c can be found by a query to MD, which takes $O(\log^2 N \log \log N)$ time and can be charged to the breakpoint bounding c . In a site event, up to two unhidden portions can be partially hidden by the newly encountered U - or V -shape curve C . In this case, each of them may need to find a new closest hidden V -shape curve in its reduced dominating region. For this, the cost of $O(\log^2 N \log \log N)$ is charged to the new breakpoint created by the two rays of C and the two unhidden portions. Thus, after processing all site events, each V -shape curve will be charged a cost of $(\log^2 N \log \log N)$, and each U -shape curve will be charged a cost of $O(\log N)$. Some breakpoints will be charged a cost of $O(\log^2 N \log \log N)$.

Clearly, each circle event can be handled in $O(\log N)$ -time, and the total number of such events is bounded by $O(K)$. Thus, the total cost for all circle events is $O(K \log N)$.

For U events, there are only $O(N)$ such events. Each event takes $O(\log^2 N \log \log N)$ time to update MD, and find the closest hidden V -shape curve to the new U -shape curve. Again, all the cost is charged to the V -shape curve.

For V events, it is clear from the algorithm, that each such event takes $O(\log^2 N \log \log N)$ time. To bound the total cost of all V events, it is needed to bound their total number. For this, it is noticed that at any moment, (1) Each unhidden portion of a U -shape curve is associated with only one hidden V -shape curve; and (2) The association of a hidden V -shape curve C and an unhidden portion c of a U -shape curve can change for only two reasons: (a) The dominating region of c changes (in a site event), and (b) a new hidden V -shape curve C' is activated and C' is closer to c than C . For (a), it means that part of c is hidden by other U - or V -shape curve C'' and the cost of de-association can be charged to the edges or vertices introduced by C'' . For (b), the cost of de-association can be charged to C' . Each V -shape will be charged no more than once. This means that computation cost of each V event can be charged to the Voronoi edge or vertex corresponding to the two breakpoints bounding the unhidden portion. Each Voronoi vertex and edge is charged a constant times with a total cost of $O(\log^2 N \log \log N)$. Thus, the total cost for all V events is $O(K \log^2 N \log \log N)$.

Putting all together, the total cost is $O((N + K) \log^2 N \log \log N)$. Thus, the theorem follows. \square

3 The MGMC Problem via Higher-Order Voronoi Diagrams

In this section, we present the iterative approach of [34] to the MGMC problem via higher-order Voronoi diagrams. A more general version of the MGMC problem is addressed in this section, where the embedded subgraph H corresponds to arbitrary polygonal shapes. The polygonal shapes are not assumed rectilinear, but they may consist of edges in arbitrary orientation. A polygonal shape is reduced to a collection of (additively) weighted line segments by means of its L_∞ medial axis. The approach is complementary to the one presented in Sect. 2 and it does not precompute any minimal cuts. On the contrary, it discovers those geometric minimal cuts that are guaranteed to participate in the final map, on the fly, by iteratively constructing variants of higher-order Voronoi diagrams. At the end of the iteration, the derived subdivision is the Hausdorff Voronoi diagram of all geometric cuts, that is, the solution to the MGMC problem. Definitions, results, and most figures in this section are reproduced from [34].

3.1 The MGMC Problem in a VLSI Layout

In a VLSI setting, the *geometric min-cut* problem is as follows [34]: We are given a collection of geometric graphs that have portions embedded on a plane (a VLSI layer) A . The embedded portions on plane A are vulnerable to random defects that may form *cuts* on the given graphs. A defect of size r is a disk of radius r . The size of a geometric cut C at a given point t is the size of the smallest defect centered at t that overlaps all elements in C , as opposed to the number of edges in C in the classic min-cut problem. Compute, for every point t on the vulnerable plane A , the

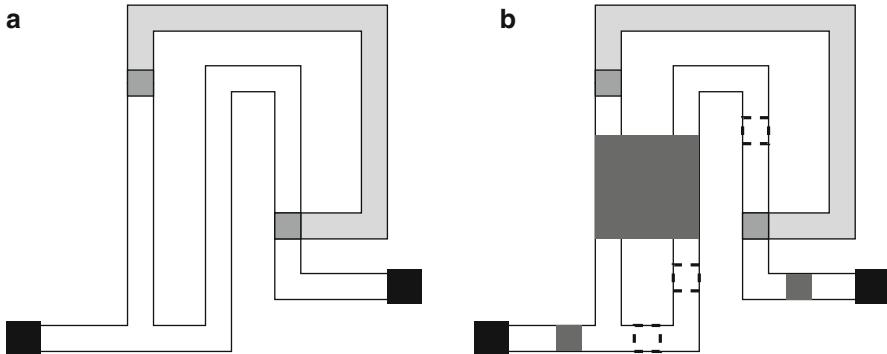


Fig. 14 (a) A net N spanning over two layers. (b) Dark defects create opens while transparent defects cause no faults

size of the minimum defect causing a geometric cut at t . The size of the minimum geometric cut at a point t is the *critical radius* for open faults at t . The resulting subdivision is referred to as the *opens Voronoi diagram*, and it corresponds to the map of the MGMC problem.

Figure 14a, reproduced from [34], illustrates an example of a simple VLSI net N spanning over two metal layers, say M1 and M2, where M2 is illustrated shaded. The two contacts illustrated as black squares are designated as *terminal shapes*. A VLSI net remains functional as long as terminal shapes remain interconnected. In Fig. 14b, defects that create open faults are illustrated as dark squares, and defects that cause no fault are illustrated hollow in dashed lines. Hollow defects do break wires of layer M1; however, they do not create opens as no terminals get disconnected.

A compact graph representation for a VLSI net N , denoted $G(N)$, can be obtained as follows: Each graph node of $G(N)$ represents a connected component of a net N on a conducting layer, and two nodes are connected by an edge if there exists at least one contact or via connecting the respective components of N . Some of the shapes constituting net N are designated as *terminal* representing the entities that the net must interconnect. A node containing terminal shapes is designated as a *terminal node*.

Given a layer A of interest, the *extended graph* of N on A , $G(N, A)$, is derived from $G(N)$ by expanding the components of N on layer A by their medial axis. Contact points between neighboring layers are approximated as points on the medial axis of the corresponding shape, called *via points*. Any via point or any portion of the medial axis corresponding to terminal shapes is identified as terminal. Figure 15a illustrates $G(N, A)$, where $A = M1$, for the net of Fig. 14; terminal points are indicated by hollow circles; dashed lines represent the original M1 polygon and they are not part of $G(N, A)$. $G(N, A)$ can be cleaned up from any trivial parts [34] by computing biconnected components, bridges, and articulation points. In the following, we assume that $G(N, A)$ is free from any trivial parts, as shown in Fig. 15b. The collection of $G(N, A)$ for all nets N involved on a layer A is the graph

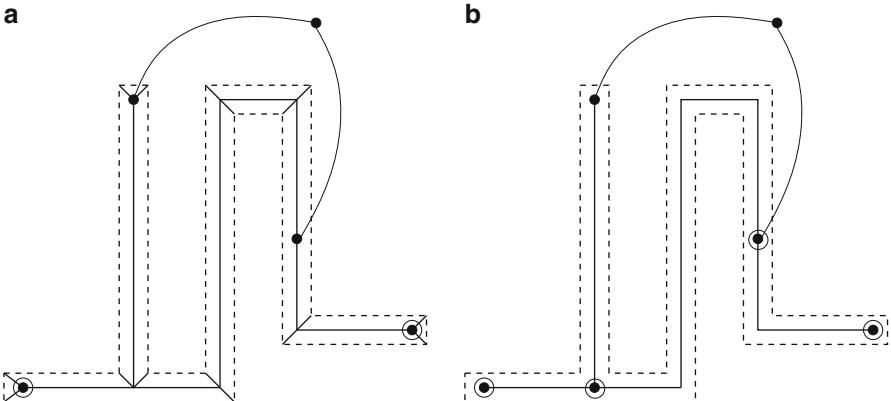


Fig. 15 The net graph of Fig. 14 before (a) and after (b) clean-up of trivial parts

of the MGMC problem, and its portion on layer A is the planar embedded subgraph H . For more details on the derivation of $G(N, A)$, see [34].

3.2 Review of Concepts on L_∞ Voronoi Diagrams

The Voronoi diagram of a set of polygonal sites in the plane is a partitioning of the plane into regions, one for each site, called *Voronoi regions*, such that the Voronoi region of a site s is the locus of points closer to s than to any other site. The Voronoi region of s is denoted as $\text{reg}(s)$ and s is called the *owner* of $\text{reg}(s)$. In the interior of a simple polygon, the Voronoi diagram is known as the *medial axis* of the polygon (a minor difference in the definition is ignored (see [26])). The boundary between two Voronoi regions is called a *Voronoi edge* and it consists of portions of *bisectors* between the owners of the neighboring regions. The bisector of two polygonal objects (such as points, segments, polygons) is the locus of points equidistant from the two objects. A point where three or more Voronoi edges meet is called a *Voronoi vertex*.

The L_∞ distance between two points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ is $d(p, q) = \max\{|x_p - x_q|, |y_p - y_q|\}$. In the presence of additive weights, the (weighted) distance between p and q is $d_w(p, q) = d(p, q) + w(p) + w(q)$, where $w(p)$ and $w(q)$ denote the weights of points p and q , respectively. In case of a weighted line l , the (weighted) distance between a point t and l is $d_w(t, l) = \min\{d(t, q) + w(q), \forall q \in l\}$. The (weighted) bisector between two polygonal elements s_i and s_j is $b(s_i, s_j) = \{y \mid d_w(s_i, y) = d_w(s_j, y)\}$.

In L_∞ , any Voronoi edge or vertex can be treated as an additively weighted line segment. For brevity and in order to differentiate from ordinary line segments, the term *core segment*, more generally *core element*, is used to denote any portion of interest along an L_∞ Voronoi edge or vertex. The term *standard-45°s* is used

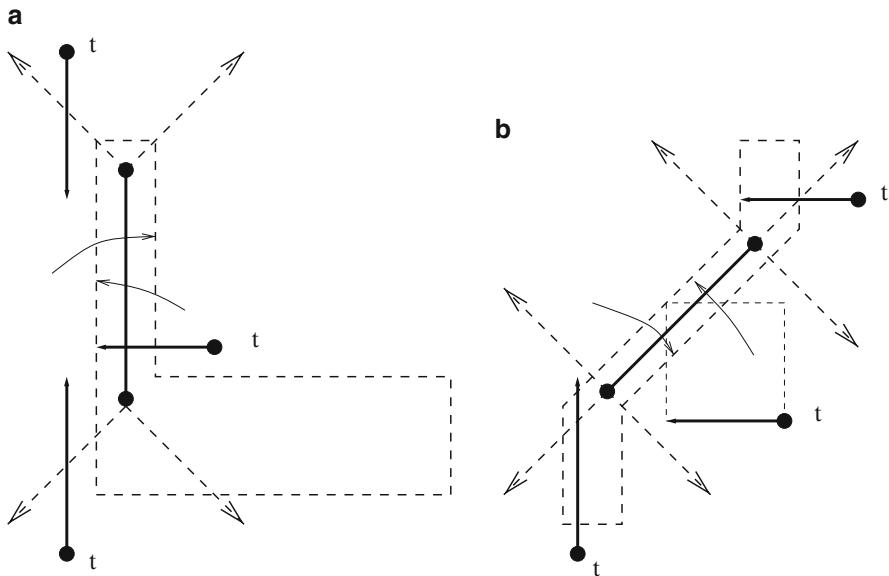


Fig. 16 The regions of influence of the core elements of a core segment: two endpoints and an open line segment. (a) an axis-parallel core segment, (b) a non-axis-parallel core segment

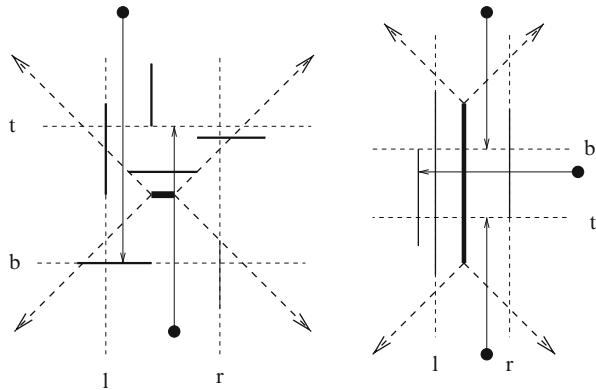
to refer to Voronoi edges of slope ± 1 that correspond to bisectors of axis-parallel lines. Figure 16 illustrates examples of core segments.

Let s be a core segment induced by the polygonal elements e_l, e_r . Every point p along a core segment s is weighted with $w(p) = d(p, e_l) = d(p, e_r)$, where e_l, e_r are the polygonal elements inducing s . The 45° rays³ emanating from the endpoints of s partition the plane into the regions of influence of either the open core segment portion or the core endpoints. In Fig. 16, the L_∞ distance is indicated by straight-line arrows emanating from various points t . In the north and south (resp. east and west) regions, the L_∞ distance simplifies to a vertical (resp. horizontal) distance. In the region of influence of a non-axis-parallel core segment, it is measured by the side of a square touching e_i as shown in Fig. 16b. In the region of influence of a core point p , distance is measured in the ordinary weighted sense, that is, for any point t , $d_w(t, p) = d(t, p) + w(p)$. In the region of influence of an open core segment s , distance, in essence, is measured according to the farthest polygonal element defining s , that is, $d_w(t, s) = d(t, e_i)$, where e_i is the polygonal element at the opposite side of s than t ; see, for example, the small arrows in Fig. 16. In L_∞ , this is equivalent to the ordinary weighted distance between t and s . For more details see [34].

The (weighted) bisector between two core elements is defined in the ordinary way, always taking the weights of the core elements into consideration. Similarly, the (weighted) Voronoi diagram of a set of core elements is defined as above,

³A 45° ray is a ray of slope ± 1 .

Fig. 17 The L_∞ farthest Voronoi diagram of axis parallel segments



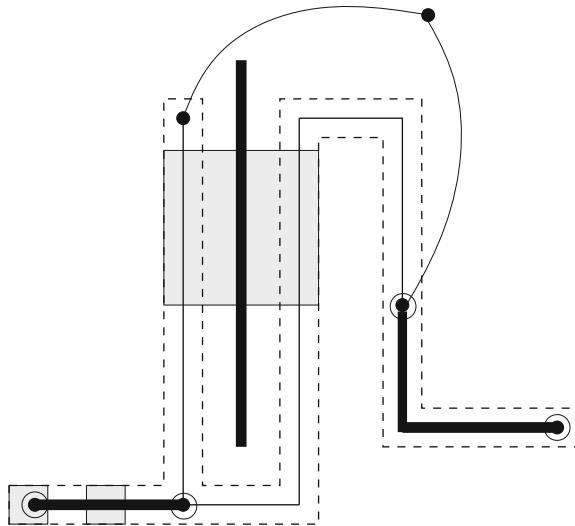
with the difference that distance between a point t and a core element s is always measured in an additive weighted sense, $d_w(t, s)$. The (weighted) Voronoi diagram of *core* medial axis segments was first introduced in [32] as a solution to the critical area computation problem for a simpler notion of an open, called *break*, that was based solely on geometric information. For Manhattan geometries, core segments are simple (additively weighted) axis-parallel line segments and points.

The *farthest Voronoi diagram* of a set of polygonal sites is a partitioning of the plane into regions, such that the *farthest Voronoi region* of a site s is the locus of points *farther away* from s than from any other site. For typical cases (e.g., points, line segments), it is a tree-like structure consisting only of unbounded regions (see, e.g., [4, 6]). In the L_∞ metric, when sites are points or axis-parallel segments, its structure is particularly simple, consisting of exactly four regions as shown in Fig. 17. In each region, the L_∞ distance to the farthest element is measured as the vertical or horizontal distance to an axis-parallel line marked by t, b, l, r , where t (resp. b) is the horizontal line through the topmost (resp. bottommost) lower (resp. upper) endpoint of all core segments and l (resp. r) is the vertical line through the leftmost (resp. rightmost) right (resp. left) endpoint of all core segments. In Fig. 17, the thin arrows indicate the farthest L_∞ distance of selected points.

3.3 Definitions and Problem Formulation

Let $\text{core}(N, A)$ denote the collection of all core elements of a net N on a layer A , that is, the collection of all medial axis vertices and edges of $G(N, A)$ on A that are of interest. For simplicity, all standard 45° edges are excluded from $\text{core}(N, A)$. The union of $\text{core}(N, A)$ for all nets N on layer A is denoted as $\text{core}(A)$. Core segments and points in $\text{core}(A)$ represent all wire segments vulnerable to defects on layer A and correspond to weighted line segments. Core segments are assumed to consist of three distinct core elements: two endpoints and an open line segment. Open faults are determined based on the connectivity information of $G(N, A)$ and the geometry information of $\text{core}(N, A)$.

Fig. 18 Generators for strictly minimal opens



Definition 6 A *minimal open* is a defect D of minimal size that breaks a net N , that is, if D is shrunk by $\epsilon > 0$, then D no longer breaks N . An *open* is any defect that entirely overlaps a minimal open. A minimal open is called *strictly minimal* if it contains no other open in its interior.

Definition 7 The center point of an open D , is called a *generator point* for D and it is weighted with the radius of D . The generator of a strictly minimal open is called *critical*. A segment formed as the union of generator points is called a *generator segment* or simply a *generator*.

In Fig. 14, strictly minimal opens are illustrated by dark shaded disks, other than the original via and contact shapes. Figure 18 illustrates the generators for strictly minimal opens for the net of Fig. 14, thickened; the shaded squares indicate strictly minimal opens. For brevity, we say that a defect D *overlaps* a core element $c \in \text{core}(A)$, but we mean that D overlaps the entire width of the wire segment induced by c .

Definition 8 A *cut* for a net N is a collection C of core elements, $C \subset \text{core}(N, A)$, such that $G(N, A) \setminus C$ is disconnected leaving nontrivial articulation or terminal points in at least two different sides. A cut C is called minimal if $C \setminus \{c\}$ is not a cut for any element $c \in C$. A defect of minimal size that overlaps all elements of cut C is called a *cut-inducing* defect. The centerpoint p of a cut-inducing defect that encloses no other defect in its interior is called a *generator point* for cut C . If, in addition, the cut-inducing defect is a strictly minimal open, then p is called *critical*. The collection of all generator points of a cut C is referred to as the *generator(s)* of C .

The generator of a cut C can consist of *critical* and *noncritical* portions. Critical portions correspond to generators of strictly minimal opens. Noncritical portions correspond to centers of cut-inducing disks that, in addition to overlapping C , may also overlap some additional cut on a layer A , and thus, although they break C , they do not correspond to strictly minimal opens.

Definition 9 Generators of minimal cuts that consist of a single core element are called *first-order generators*. Generators of minimal cuts that consist of more than one core element are called *higher-order generators*. The set of all critical generators on layer A is denoted as $G(A)$.

In Fig. 18 first-order generators are illustrated as the thickened core segments in the interior of polygons; the vertical thick segment in the exterior of polygons is a higher-order generator that involves a pair of core elements. Given $G(N, A)$, we can detect *biconnected components*,⁴ *bridges*, and *articulation points*⁵ using *depth-first search* (DFS) as described in [21, 42]. By definition, we have the following property.

Lemma 15 *The set of first-order generators on a layer A , denoted as $G_1(A)$, consists of all the bridges, terminal edges, articulation points, and terminal points of $G(N, A) \cap \text{core}(N, A)$, for all nets N . All first-order generators are critical.*

The generator of a minimal cut C that consists of more than one core element must be a subset of the L_∞ farthest Voronoi diagram of C , derived by ignoring the standard-45° edges of the diagram. For Manhattan geometries, the generator of any cut is always a single axis-parallel segment (that can degenerate to a point); see, for example, Fig. 17. Any generator point p of a cut C is weighted with $w(p) = \max\{d_w(p, c), \forall c \in C\}$. The disk D centered at p of radius $w(p)$ is clearly an open. If in addition D is strictly minimal, then p is a critical generator.

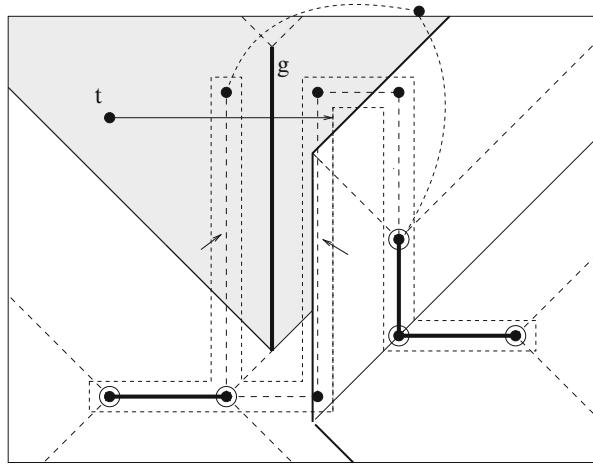
Definition 10 The *Voronoi diagram for opens* on a layer A is a subdivision of A into regions, such that the *critical radius* of any point t in a Voronoi region is determined by the owner of the region. The critical radius of a point t , $r_c(t)$, is the size (radius) of the smallest defect centered at t and causing an open.

The following theorem is easy to see by properties of Voronoi diagrams. For a proof, see [34].

⁴A biconnected component of a graph G is a maximal set of edges, such that any two edges in the set lie on a common simple cycle.

⁵An articulation point (resp. bridge) of a graph G is a vertex (resp. edge) whose removal disconnects G .

Fig. 19 The Voronoi diagram for open faults on layer A . The shaded region illustrates the Voronoi region of the higher-order generator g



Theorem 5 *The Voronoi diagram for open faults on a layer A corresponds to the (weighted) Voronoi diagram of the set $G(A)$ of all critical generators for strictly minimal opens on A , denoted as $\mathcal{V}(G(A))$.*

Figure 19 illustrates the opens Voronoi diagram for the net of Fig. 14. The shaded region illustrates the Voronoi region of a higher-order generator g , $reg(g)$. Generator g is the critical generator of a cut consisting of two core segments as indicated by two small arrows. The critical radius of a sample point t in $reg(g)$ is indicated by the arrow emanating from t .

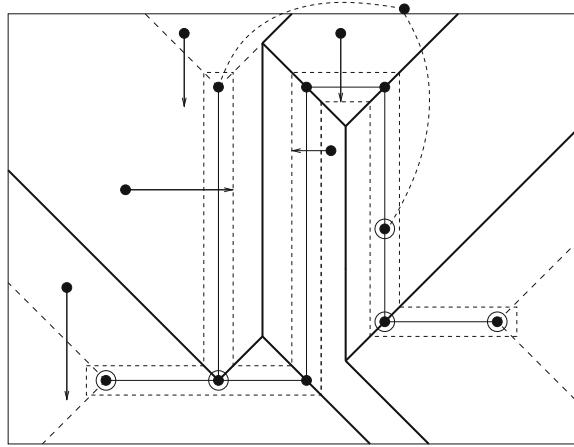
Corollary 3 *Given the opens Voronoi diagram, the critical radius of any point t in the region of a generator g is $r_c(t) = d_w(t, g)$. If g is a higher-order generator of cut C , then $r_c(t) = d_w(t, g) = \max\{d_w(t, c), \forall c \in C\}$.*

In the following section, the Voronoi diagram for opens is formulated as a special higher-order Voronoi diagram of elements in $core(A)$.

3.4 A Higher-Order Voronoi Diagram Modeling Open Faults and the MGMC Problem

Let $\mathcal{V}(A)$ denote the (weighted) Voronoi diagram of the set $core(A)$ of all core elements on a plane (layer) A . If there were no loops associated with A , then $\mathcal{V}(A)$ would provide the opens Voronoi diagram on A , and $core(A)$ would be the set of all critical generators. $\mathcal{V}(A)$ for Manhattan layouts has been defined in detail in [32]. Figure 20 illustrates $\mathcal{V}(A)$ for the net graph of Fig. 14. The arrows in Fig. 20 illustrate several minimal radii of defects that break a wire segment. Given a point t in the region of generator s , $d_w(t, s)$ gives the radius of the smallest defect centered at t

Fig. 20 The L_∞ Voronoi diagram of $\text{core}(A)$ on layer A , $\mathcal{V}(A)$



that overlaps the wire segment induced by s . Assuming no loops, $d_w(t, s)$ would be the critical radius of t .

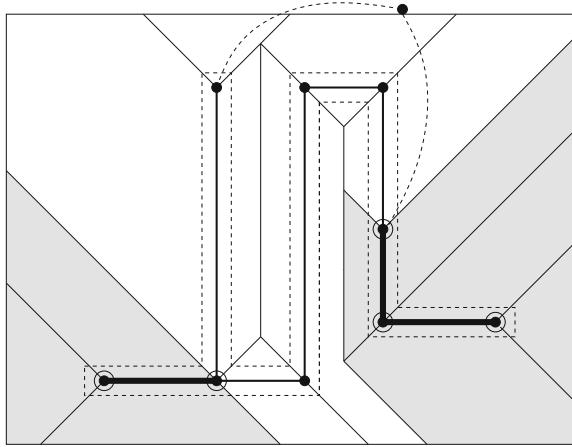
Once loops are taken into consideration, only bridges, articulation, and terminal points, among the elements of $\text{core}(A)$, correspond to critical generators. Let us augment $\mathcal{V}(A)$ with information reflecting critical generators. In particular, the regions of first-order generators get colored red reflecting the regions of critical generators. The critical radius of point t in a red region of owner s is $r_c(t) = d_w(t, s)$. In Fig. 21, red regions are shown shaded and critical generators are shown thickened.

Let us now define the order- k Voronoi diagram on the plane A , denoted as $\mathcal{V}^k(A)$. For $k = 1$, $\mathcal{V}^k(A) = \mathcal{V}(A)$. Following the standard definition of higher-order Voronoi diagrams, a region of $\mathcal{V}^k(A)$ corresponds to a maximal locus of points with the same k nearest neighbors among the core elements in $\text{core}(A)$. The open portion of a core segment and its two endpoints count as different entities. A k th-order Voronoi region, $k > 1$, belongs to a k -tuple C representing the k nearest neighbors of any point in the region of C . The region of C is denoted $\text{reg}(C)$, and it is further subdivided into finer subregions by the farthest Voronoi diagram of C . For any point $t \in \text{reg}(C)$, $d(t, C) = \max\{d(t, c), \forall c \in C\}$. If C constitutes a cut of a net N , then the region of C is colored red.

In order to appropriately model open faults, the above standard definition is slightly modified, and in certain cases fewer than k elements are allowed to own a Voronoi region of order k . In the following, the term *k-th order Voronoi diagram* implies the modified version of the diagram as follows:

- A red region corresponds to a maximal locus of points with the same r , $1 \leq r \leq k$, nearest neighbors, C , among the core elements in $\text{core}(A)$, such that C constitutes a minimal cut for some net N .
- Any time a core segment s and one of its endpoints p participate in the same set C of nearest neighbors, s is discarded from C ; this is because $d(t, p) \geq d(t, s)$

Fig. 21 The first-order opens Voronoi diagram on layer A , $\mathcal{V}^1(A)$. Shaded regions belong to first-order generators and the critical radius of any point within is determined by the region owner



$\forall t \in \text{reg}(C)$. Intuitively, a defect that destroys a core endpoint automatically also destroys all incident core segments, but not vice versa.

Figures 22 and 23 illustrate $\mathcal{V}^2(A)$ and $\mathcal{V}^3(A)$, respectively, for the net of our example. k th-order Voronoi regions are illustrated in solid lines; red regions are illustrated shaded. The darker shaded region in $\mathcal{V}^2(A)$ shows the 2nd-order red region of a pair of core segments that constitute a cut. The thick dashed lines indicate the farthest Voronoi diagram subdividing a k th-order region. In a red region, the thick dashed lines (excluding standard 45°s) correspond to critical generators. All critical generators are indicated thickened; solid ones are first-order generators and dashed ones in red regions are higher-order generators. All thin dashed lines in Figs. 21–23 can be ignored. Due to our conventions, the Voronoi region of any core endpoint p in $\mathcal{V}^1(A)$ remains present in $\mathcal{V}^2(A)$ and expands into the regions of the core segments incident to p .

Theorem 6 *The Voronoi diagram for opens on a layer A is the minimum-order m Voronoi diagram of $\text{core}(A)$, $\mathcal{V}^m(A)$, $m \geq 1$, such that all regions of $\mathcal{V}^m(A)$ are colored red. Any region $\text{reg}(H)$, where $|H| > 1$, is subdivided into finer regions by the farthest Voronoi diagram of H . The critical radius for any point t in $\text{reg}(H)$ is $r_c(t) = d_w(t, H) = \max\{d_w(t, h), h \in H\}$.*

For a proof of Theorem 6, see [34]. Figure 24 illustrates the opens Voronoi diagram, for our example; arrows indicate the critical radius of several points; all critical generators are indicated in thick solid lines. Note that in L_∞ , the Voronoi subdivision is not unique but depends on the conventions used on how to distribute regions that are equidistant from collinear elements on axis-parallel lines. Critical area calculations are immune to such differences, however, they may have an effect on the number of iterations needed to compute the opens Voronoi diagram. The adapted convention is that critical generators have priority over non-critical ones and regions equidistant from a critical and a noncritical generator are assigned to the critical one and get colored red.

Fig. 22 The second-order opens Voronoi diagram, $\mathcal{V}^2(A)$. The darker shaded region belongs to a pair of core segments forming a cut

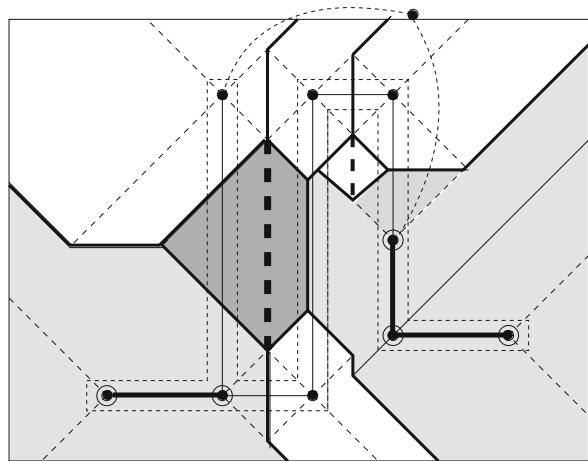
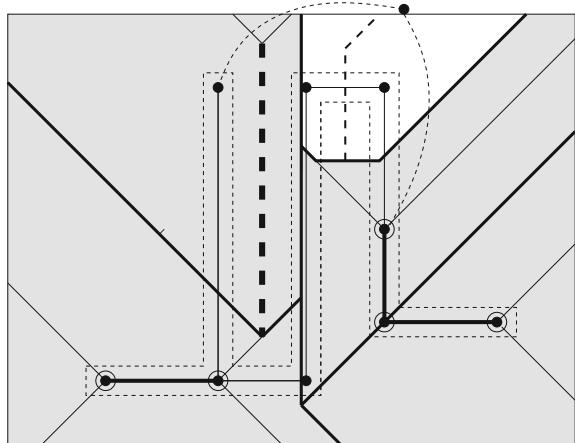


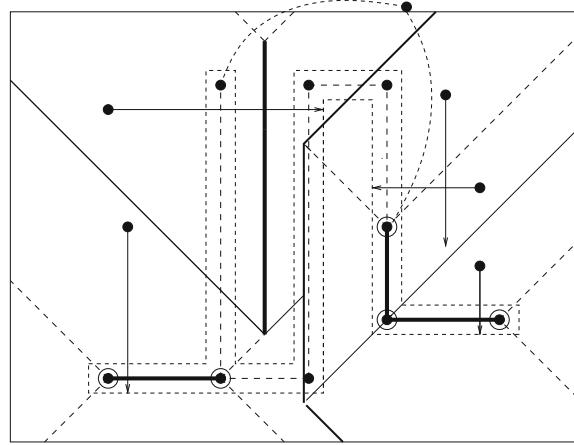
Fig. 23 The third-order opens Voronoi diagram, $\mathcal{V}^3(A)$



Corollary 4 *The higher-order critical generators on a layer A are exactly the farthest Voronoi edges and vertices, excluding the standard- 45° Voronoi edges, constituting the farthest Voronoi subdivisions in the interior of each region in $\mathcal{V}^m(A)$. All higher-order critical generators are encoded in the graph structure of $\mathcal{V}^k(A)$, for some k , $1 \leq k < m$.*

Let $G(A)$ denote the set of all critical generators on layer A , including first-order and higher-order generators. Let us classify higher-order critical generators according to the minimum-order- k Voronoi diagram they first appear in. In particular,

Fig. 24 The Voronoi diagram for open faults on a layer A . Arrows illustrate the critical radius of several points



higher-order generators encoded in $\mathcal{V}^k(A)$ are classified as $(k + 1)$ -order generators and they are denoted as $G_{k+1}(A)$, $1 \leq k < m$. Let $G(A) = \cup_{1 \leq i \leq m} G_i(A)$. By [Theorems 5 and 6](#), $\mathcal{V}(G(A))$ and $\mathcal{V}^m(A)$ are identical.

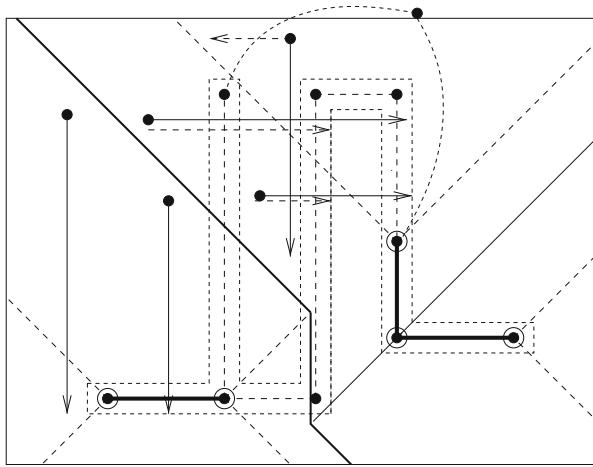
3.4.1 An Approximate Opens Voronoi Diagram

Given any subset $G'(A)$ of the set of critical generators $G(A)$, the (weighted) Voronoi diagram of $G'(A)$ can be used as an approximation to $\mathcal{V}(G(A))$. Clearly, the more critical generators are included in $G'(A)$, the more accurate the result is. In practice, we can derive $G'(A)$ as $\cup_{1 \leq i \leq k} G_i(A)$, including all i th-order generators up to a small constant k . Since the significance of critical generators reduces drastically with the increase in their order, $\mathcal{V}(G'(A))$ should be sufficient for critical area computation for all practical purposes.

Corollary 5 Let $G'(A) = \cup_{1 \leq i \leq k} G_i(A)$ be a subset of critical generators including all generators up to order k for a given constant k . The (weighted) Voronoi diagram of $G'(A)$, $\mathcal{V}(G'(A))$, can serve as an approximation to the opens Voronoi diagram. If $G'(A) = G(A)$, then the two diagrams are equivalent.

[Figure 25](#) illustrates the (weighted) Voronoi diagram of $G_1(A)$. $\mathcal{V}(G_1(A))$ reveals critical radii for opens under the (false) assumption that all loops are immune to open faults. In [Fig. 25](#), solid arrows indicate selected critical radii as derived by $\mathcal{V}(G_1(A))$, while dashed arrows indicate true critical radii. Several critical radii can be overestimated in $\mathcal{V}(G_1(A))$, resulting in underestimating the total critical area for open faults. As k increases, however, $\mathcal{V}(\cup_{1 \leq i \leq k} G_i(A))$ converges fast to $\mathcal{V}(G(A))$ (see, e.g., Sect. 8 of [34] for experimental results).

Fig. 25 $\mathcal{V}(G_1(A))$ as an approximate opens Voronoi diagram



3.5 Computing the Opens Voronoi Diagram

In this section, we give algorithmic details on how to compute $G_{k+1}(A)$ and $\mathcal{V}^{k+1}(A)$, given $\mathcal{V}^k(A)$, for $1 \leq k < m$. We also discuss how to compute $\mathcal{V}(\cup_{1 \leq i \leq m} G_i(A))$ and $\mathcal{V}(G(A))$.

The iterative process to compute higher-order generators and higher-order opens Voronoi diagrams. Let's first discuss how to identify the set $G_{k+1}(A)$ of $(k+1)$ -order generators, given $\mathcal{V}^k(A)$, for $k \geq 1$. The following property is shown in [34].

Lemma 16 *A Voronoi edge g that bounds two non-red Voronoi regions $reg(H)$ and $reg(J)$ in $\mathcal{V}^k(A)$ corresponds to a critical generator if and only if both the core elements $h \in H$ and $j \in J$ that induce g ($g \in b(h, j)$) are part of the same biconnected component B , and in addition, $H \cup J$ corresponds to a cut of B , that is, removing $H \cup J$ from B disconnects B , leaving articulation points in at least two sides. No Voronoi edge bounding a red region can be a critical generator.*

To determine if Voronoi edge g is a critical generator, we need to pose a connectivity query to biconnected component B after removing $H \cup J$. To perform connectivity queries efficiently, we can use the fully dynamic connectivity data structures of [20], which support edge insertion and deletions in $O(\log^2 n)$ -time, while they can answer connectivity queries fast. For simplicity in the implementation, instead of employing dynamic connectivity data structures, reference [34] gives a simple (almost brute force) algorithm as follows: Remove the elements of H from B and determine new nontrivial bridges, articulation points, and biconnected components of $B \setminus H$. For any Voronoi edge g bounding $reg(H)$, where g is portion of $b(h, j)$, $h \in H$, $j \in J$, g is a critical generator if and only if j is a new non-trivial

bridge or articulation point of $B \setminus H$. Generator g gets associated with the tuple of core elements $H \cup J$, simplified, in case j or h are core endpoints, by removing any core segment incident to j, h .

The above process can be considerably simplified in the special case where the biconnected component B is a simple cycle. In this case, a simple coloring scheme in the DFS tree of B can efficiently identify all cuts of B that may be associated with a second-order generator. The time complexity of determining $G_{k+1}(A)$, given $\mathcal{V}^k(A)$, is summarized in the following lemma. Note that the size of $\mathcal{V}^k(A)$ is $O(k(n - k))$ (see [26]).

Lemma 17 *The $(k + 1)$ -order generators can be determined from $\mathcal{V}^k(A)$ in time $O(kn \log^2 n)$ using the dynamic connectivity data structures of [20] or in time $O(kn^2)$ using the simple algorithm presented above. In case of biconnected components forming simple cycles, second-order generators can be determined from $\mathcal{V}(A)$ in $O(n)$ time.*

Let us now discuss how to obtain $\mathcal{V}^{k+1}(A)$ from $\mathcal{V}^k(A)$, $k \geq 1$. The following is an adaptation of the iterative process to compute higher-order Voronoi diagrams of points [26], to the case of (weighted) segments. Let $reg(H)$ be a non-red region of $\mathcal{V}^k(A)$. Let $N(H)$ denote the set of all core elements that induce a Voronoi edge bounding $reg(H)$ in $\mathcal{V}^k(A)$.

1. Compute the (weighted) L_∞ Voronoi diagram of $N(H)$ and truncate it within the interior of $reg(H)$; this gives the $(k + 1)$ -order subdivision within $reg(H)$. Each $(k + 1)$ -order subregion of $reg(H)$ is attributed to a tuple $J = H \cup \{c\}$, $c \in N(H)$. In case c is a core endpoint incident to a core segment s in H , J simplifies to $J = H \setminus \{s\} \cup \{c\}$. In case c is part of a cut C owning a neighboring red region of $\mathcal{V}^k(A)$, the subregion of J gets colored red and gets as owner the cut C .
2. Once the $(k + 1)$ -order subdivision within all non-red regions neighboring $reg(H)$ has been performed, merge any incident $(k + 1)$ -order subregions that belong to the same tuple of owners J into a maximal $(k + 1)$ -order region, $reg(J)$. The edges of $\mathcal{V}^k(A)$ included within $reg(J)$ constitute the finer subdivision of $reg(J)$ by its farthest Voronoi diagram. All $(k + 1)$ -order red subregions are merged into the neighboring red regions of $\mathcal{V}^k(A)$ forming the maximal red regions of $\mathcal{V}^{k+1}(A)$.

Using established bounds for higher-order Voronoi diagrams of points (see, e.g., [26]) we conclude the following.

Lemma 18 *$\mathcal{V}^{k+1}(A)$ can be computed from $\mathcal{V}^k(A)$ in time $O(k(n - k) \log n)$, plus the time $T(k, n)$ to determine the $(k + 1)$ -order generators, where $T(k, n)$ is as given in Lemma 17.*

3.5.1 Computing the Opens Voronoi Diagram from Critical Generators

The iterative process of Sect. 3.5 can continue until all regions are colored red and the complete opens Voronoi diagram (i.e., the map of the MGMC problem) is guaranteed to be available. By Lemmas 17 and 18, this results in

an $O(n^3 \log n (\log \log n)^3)$ -time algorithm. In practice, however, this would be unnecessarily inefficient. Note that the iterative process may continue for several rounds without any new critical generators being identified, only the regions of existing critical generators keep enlarging into neighboring non-red regions. Note also that as the number of iterations k increases, the weight of order- k critical generators (if any) increases as well, and their contribution to the total critical area drastically reduces. In practice, we can restrict the number of iterations to a small predetermined constant k , or to a small number determined adaptively, and compute only a sufficient set of critical generators $G'(A) = \cup_{1 \leq i \leq k} G_i(A)$. We can then use **Theorem 5** to report $\mathcal{V}(G'(A))$ as an approximate opens Voronoi diagram. The overall algorithm can be broken into two independent parts:

- Part I: Compute the set of critical generators $G'(A) = \cup_{1 \leq i \leq k} G_i(A)$, up to a given (or adaptively determined) order k .
- Part II: Compute the (weighted) Voronoi diagram of $G'(A)$, $\mathcal{V}(G'(A))$, as the opens Voronoi diagram.

Part I can be performed using the iterative process of [Sect. 3.5](#). Experimental results in [34] suggest that $k = 2$ is often adequate and no $k > 4$ is ever needed. Alternatively, k can be determined adaptively, for example, to the first round such that no new critical generators are determined. Part II can be performed using a plane sweep algorithm for computing $\mathcal{V}(A)$. Critical generators have similar properties to the core elements of $\text{core}(A)$, and the same plane sweep algorithm can be used to compute either (see [32, 36]). The computations of Parts I and II can be synchronized: once a generator is discovered in Part I, it can be immediately scheduled for processing in Part II. For more details on the plane sweep construction and the synchronization of Parts I and II that is important in maintaining the locality of the computation see [34]. We thus conclude:

Theorem 7 *Assuming that $G(N)$ is available for all nets under consideration and given a small constant k , the approximate opens Voronoi diagram $\mathcal{V}(G'(A))$, $G'(A) = \cup_{1 \leq i \leq k} G_i(A)$ can be computed in time $O(n \log n)$ plus the time needed to answer connectivity queries. The latter can be done in time $O(n \log^2 n)$.*

The original implementation of this method, whose experimental results are reported in [34], used a slightly different approach in order to guarantee accuracy while the locality property was preserved. Namely, the iterative process of [Sect. 3.5](#) was applied to each biconnected component independently. The advantages of considering each biconnected component independently were locality as well as the ability to run the process on each individual component to completion and guarantee the accuracy. The disadvantage was that generators produced in this manner, $G''(A)$, did not need all be critical. Including noncritical generators in the set $G''(A)$ complicates the algorithm of Part II because it amounts to computing the *Hausdorff Voronoi diagram* of corresponding geometric minimal cuts on layer A . Note that critical generators can be treated as simple additively weighted segments, having special weights, such that Voronoi regions remain connected and the entire generator is always enclosed in its Voronoi cell. This property considerably simplifies the construction of their (weighted) Voronoi diagram, which is, the Hausdorff Voronoi

diagram, which remains similar to the construction of ordinary Voronoi diagrams of line segments. Once non-critical generators are present, however, this property no longer holds and the algorithms is equivalent to constructing the Hausdorff Voronoi diagram of the corresponding cuts as presented in Sects. 2.3 and 4.

For information on critical area computation once the solution to the MGMC problem is available, see, for example, [32, 34, 36].

4 The L_∞ Hausdorff Voronoi Diagram

In this section, we review structural properties of the L_∞ Hausdorff Voronoi diagram of clusters of points, or equivalently, the L_∞ Hausdorff Voronoi diagram of rectangles, presented in [39]. The L_∞ Hausdorff Voronoi diagram of rectangles provides a solution to the MGMC problem after geometric minimal cuts have been identified. Results in this section are reproduced from [39].

Given a set S of point clusters in the plane, the *Hausdorff Voronoi diagram* of S , denoted $HVD(S)$, is a subdivision of the plane into regions, such that the *Hausdorff Voronoi region* of a cluster P , denoted $hreg(P)$, is the locus of points *closer* to P than to any other cluster in S , where distance between a point t and a cluster P is measured as the *farthest distance* between t and any point in P , $d_f(t, P) = \max\{d(t, p), p \in P\}$, $hreg(P) = \{x \mid d_f(x, P) < d_f(x, Q), \forall Q \in S\}$. It is subdivided into finer regions by the *farthest Voronoi diagram* of P , $FVD(P)$. The farthest distance $d_f(t, P)$ is equivalent to the *Hausdorff distance*⁶ $d_h(t, P)$ between t and P . In the L_∞ metric, $d_f(t, P)$ (equiv. $d_h(t, P)$) is equivalent to $d_f(t, P')$ (equiv. $d_h(t, P')$), where P' is the minimum enclosing axis-aligned rectangle of P . Thus, the L_∞ Hausdorff Voronoi diagram of S is equivalent to the L_∞ Hausdorff Voronoi diagram of the set S' of the minimum enclosing rectangles of all clusters in S . In the following, the terms cluster and rectangle are used interchangeably.

In this section, we review the tight bound on the structural complexity of the L_∞ Hausdorff Voronoi diagram given in [39]. It is shown that the structural complexity of the L_∞ Hausdorff Voronoi diagram is $\Theta(n + m)$, where n is the number of input clusters (equiv. rectangles) and m is the number of *essential pairs of crossing* clusters (see Definition 11). We also review a simple plane sweep construction in two dimensions given initially in [32] and improved in [39]. The improved algorithm consists of an $O((n + M) \log n)$ -time preprocessing step, based on *point dominance* in \mathbb{R}^3 , followed by the main plain sweep algorithm that runs in $O((n + M) \log n)$ -time and $O(n + M)$ -space, where M reflects special crossings that are *potentially essential* (see Definition 12); m, M are $O(n^2)$, $m \leq M$, but $m = M$, in the worst case. In practice, typically, $m, M \ll n^2$. For non-crossing rectangles the algorithm simplifies to optimal $O(n \log n)$ -time and $O(n)$ -space.

⁶The (directed) Hausdorff distance from a set A to a set B is $h(A, B) = \max_{a \in A} \min_{b \in B} \{d(a, b)\}$. The (undirected) Hausdorff distance between A and B is $d_h(A, B) = \max\{h(A, B), h(B, A)\}$.

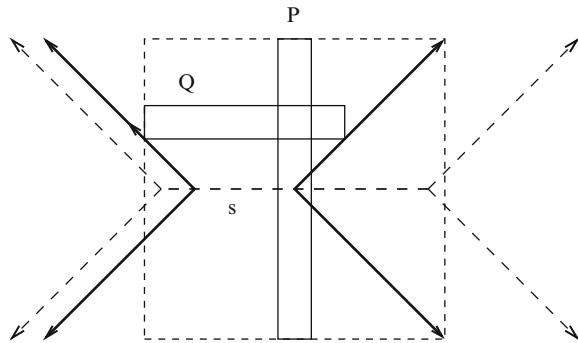
An $O(n \log n)$ -expected time algorithm can be derived in the case of non-crossing rectangles using the randomized incremental construction for abstract Voronoi diagrams [24] (see [1]). For arbitrary rectangles, however, the L_∞ Hausdorff Voronoi diagram does not fall under the umbrella of *abstract Voronoi diagrams* [23] (see, e.g., [38]). This algorithm is efficient for a relatively small number of crossing rectangles as motivated by the critical area application. For a large number of crossings, the $O(n^2)$ -time approach of [11] can be preferable. Section 2.3 presented an output-sensitive version of the plane sweep construction in three dimensions of increased space complexity.

4.1 Definitions and Structural Complexity

Let S be a set of n rectangles, or a set of n point clusters in the plane, where each cluster has been substituted by its minimum enclosing axis aligned rectangle. A pair of rectangles (P, Q) is called *crossing* if P and Q intersect in the shape of a cross. Given a crossing pair (P, Q) , P is assumed to be at least as long as Q . For a rectangle P , let P^n, P^s, P^e , and P^w denote the north, south, east, and west edges of P , respectively. P is called horizontal (resp. vertical) if P^n is longer (resp. shorter) than P^e . The axis-parallel line through edge P^i , $i = n, s, e, w$, is denoted as $l(P^i)$. The term P^i is also used to denote the main coordinate of edge P^i . The *core segment* of P is the locus of centers of all minimum enclosing squares of P , and it is given by the axis-parallel line segment of the L_∞ farthest Voronoi diagram of P . It can be viewed as an ordinary line segment s additively weighted with $w(s) = d_f(s, P)$. In Fig. 26, $FVD(P)$ is illustrated in dashed lines and the core segment is indicated by s . The L_∞ Hausdorff Voronoi diagram of S is equivalent to the (weighted) Voronoi diagram of the set of *core segments* of all clusters in S (see [32]).

The Hausdorff bisector between two clusters P, Q is $b_h(P, Q) = \{y \mid d_f(y, P) = d_f(y, Q)\}$. As shown in [38], $b_h(P, Q)$ is a subgraph of $FVD(P \cup Q)$. For a rectangle Q strictly enclosed in the interior of a minimum enclosing square of P , $b_h(P, Q)$ consists of either one (if P and Q are non-crossing) or two (if P and Q are crossing) chains, each one forming a V-shape out of the ± 1 -slope rays of $FVD(P \cup Q)$; the apex of each chain is called a *V-vertex*. A V-vertex v is incident to the core segment of P and its 90° -angle faces the portion of the plane closer to P . It is characterized as *up*, *down*, *right*, or *left*, depending on whether its 90° -angle is facing north, south, east, or west, respectively. In addition, it is characterized as *crossing*, if Q is crossing P , and *non-crossing*, otherwise. The minimum enclosing square of P centered at V-vertex v is also enclosing Q and it is denoted as $\text{square}(P, v)$. It is also denoted as $\text{square}(P, Q^i)$, where Q^i is the non-crossing edge of Q that delimits one of its edges. Figure 26 illustrates $b_h(P, Q)$ consisting of two crossing V-vertices, one right and one left; $\text{square}(P, Q^w)$ is illustrated dashed. $\text{square}(P, Q^i)$ is referred to as an *extremal minimum enclosing square* of P and Q . The V-vertices of $HVD(S)$ are referred to as *Voronoi V-vertices*.

Fig. 26 The L_∞ Hausdorff bisector of crossing rectangles



Definition 11 A pair of crossing rectangles (P, Q) is called *essential* if there is an extremal minimum enclosing square of P and Q , $\text{square}(P, Q^i)$, that is empty of any other rectangle.

The following lemma is easy to see.

Lemma 19 A pair of crossing rectangles (P, Q) induces a Voronoi V-vertex v in $\text{HVD}(S)$ if and only if (P, Q) is an essential crossing. Assuming that P is a vertical rectangle, v is a right (resp. left) V-vertex if and only if $\text{square}(P, Q_i^w)$ (resp. $\text{square}(P, Q_i^e)$) is empty of other rectangles. Similarly for a horizontal rectangle.

Combining Lemma 19 with the structural complexity results of [38], the following bound can be derived (see [39] for details).

Theorem 8 The structural complexity of the L_∞ Hausdorff Voronoi diagram of a set S of n point clusters, equivalently n rectangles, is $\Theta(n + m)$, where m is the total number of essential crossings.

Definition 12 A collection of crossings for a vertical rectangle P , (P, Q_i) , $i = 1, \dots, k$, is called a *staircase*, if $Q_i^w < Q_{i+1}^w$ and $Q_i^e < Q_{i+1}^e$, $i = 1, \dots, k$. If in addition, $\text{square}(P, Q_i^w)$ is empty of $Q_j \neq Q_i$, the staircase and its crossings are called *potentially essential*. The maximum size of a potentially essential staircase for P is the *number of potentially essential crossings* for P . Let M denote the total number of potentially essential crossings for all vertical rectangles in S , plus the number of essential crossings for all horizontal rectangles in S .

Figure 27 shows a potentially essential staircase for a vertical rectangle P . In the absence of additional rectangles, all crossings are essential, that is, they all induce Voronoi V-vertices in the Hausdorff Voronoi diagram. The shaded regions in Fig. 27 belong to $\text{hreg}(P)$.

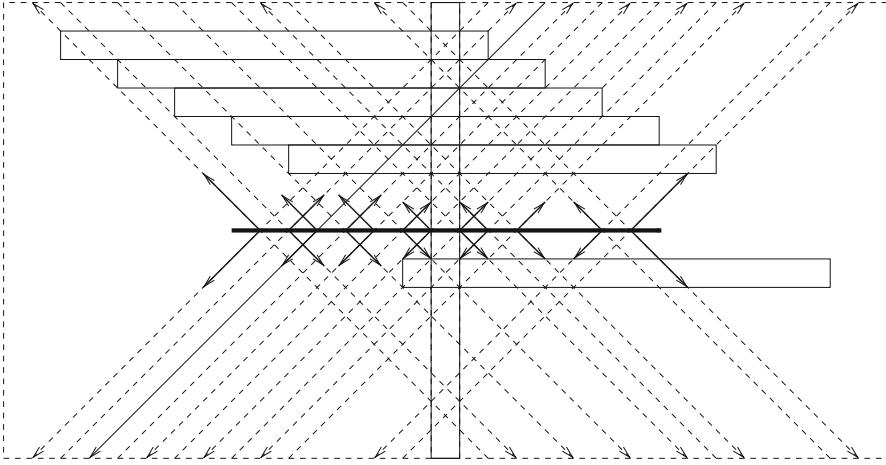


Fig. 27 Collection of essential crossings. Shaded regions belong to P

4.2 A Refined Plane Sweep Construction

In this section we present the plane sweep construction of [32, 39]. It is based on the standard plane sweep paradigm for Voronoi diagrams [8, 15], its adaptation for line segments in L_∞ [36], and the generalization to handle the special features of Hausdorff Voronoi diagrams introduced in [32, 33]. In the Hausdorff Voronoi diagram, sites need not be enclosed in their Voronoi regions and Voronoi regions may be disconnected, which are features not addressed by the standard plane sweep paradigm for Voronoi diagrams.

Assume a vertical sweep-line l_t sweeping the entire plane from left to right. At any instant t of the sweeping process, $HVD(S_t \cup l_t)$ is computed, for $S_t = \{P \in S \mid l(P^e) < t\}$. The boundary of the Voronoi region of l_t is the *wavefront* at time t . Voronoi edges and core segments incident to the wavefront are called *spike bisectors* and *spike core segments*, respectively. The combinatorial structure of the wavefront changes at specific *events* organized in a priority queue. We have four types of *site events*: *start-vertical-rectangle*, *end-vertical-rectangle*, *V-vertex* events (for brevity *V events*), and *horizontal-rectangle* events. Site events are partially similar to those for ordinary line segments [36] enhanced with additional functions to handle V-vertices and disconnected Voronoi regions. *Spike events* are caused by the intersection of incident spike bisectors, and they remain the same as in the ordinary plane sweep paradigm.

The *wave-curve* of a rectangle R is the bisector between R and the sweep line l_t , at time t , $b(R, l_t) = \{y \mid d_f(y, R) = d(y, l_t)\}$, where $d(y, l_t)$ is the ordinary distance between y and l_t . In L_∞ , it consists of two or three *waves*: a ray of slope -1 , corresponding to $b(R^s, l_t)$, a ray of slope $+1$, corresponding to $b(R^n, l_t)$, and possibly a vertical line segment corresponding to $b(R^w, l_t)$, if appropriate. The wave-curve of R can be seen equivalently as the (weighted) bisector between the

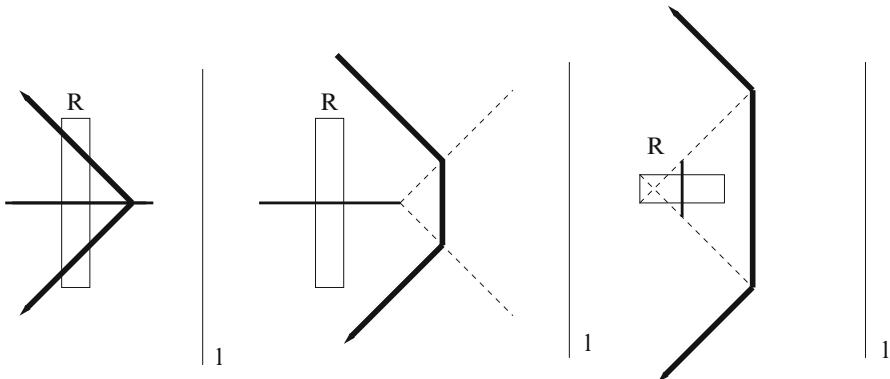


Fig. 28 The wave-curve of a rectangle R

core segment s of R and l_t , that is, $b(R, l_t) = \{y \mid d_w(y, s) = d(y, s) + w(s) = d(y, l_t)\}$. In Fig. 28, the wave-curve of several instances of rectangles is illustrated. The bold axis-aligned segment illustrates the core segment of R . The *wavefront* (equiv. the *beach line*) at time t is the lower envelope with respect to the sweep-line, of the *wave-curves* of all rectangles in S_t . Note that the term *beach line* of Sect. 2.3 is equivalent to the term *wavefront* of this section, which follows the terminology of [39]. In L_∞ , the wavefront is monotone with respect to any line of slope ± 1 . The wavefront is typically maintained as a height balanced binary tree, \mathcal{T} , ordered from bottom to top. Leaf nodes correspond to waves, while internal nodes correspond to spike bisectors and spike core segments revealing *breakpoints* between incident waves.

In [39], \mathcal{T} gets augmented with nonstandard additional information in order to efficiently answer queries regarding V-vertices. This augmentation is essential for the time complexity bound of the main plane sweep algorithm. Each node x is augmented with a *w-max* value representing the rightmost west edge of all rectangles contributing a wave to the portion of the wavefront rooted at x , and two *x-min* values (*x-min-I* and *x-min-II*) that in combination represent the minimum x -coordinate of the portion of the wavefront rooted at x . In particular, for a leaf node representing a wave of rectangle R , the *w-max* value is R^w and both *x-min* values are $+\infty$. For an internal node x , *w-max* is the maximum between the *w-max* values of its children. If node x corresponds to a horizontal (resp. ± 1 -slope) bisector, then *x-min-I* (resp. *x-min-II*) points to the breakpoint of minimum x -coordinate among its own breakpoint and the *x-min-I* (resp. *x-min-II*) values of its children; otherwise *x-min-I* (resp. *x-min-II*) points to the breakpoint of minimum x -coordinate among its children only. The minimum x -coordinate of the portion of the wavefront rooted at node x is $x\text{-min} = \min\{\text{x-min-I}, \text{x-min-II}\}$. The augmentation values *w-max*, *x-min-I*, *x-min-II* remain the same unless a combinatorial change in the wavefront (i.e., an event) takes place.

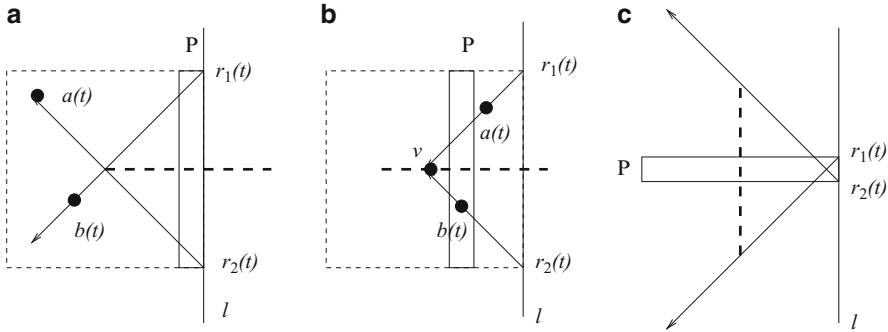


Fig. 29 (a) At $t = \text{priority}(P)$ the wavefront has not reached s . (b) An invalid event at time $t = \text{priority}(v)$. (c) A horizontal rectangle event

Any Voronoi point in $HVD(S)$ enters the wavefront at the time of its *priority*. The *priority* of a point v is the rightmost x -coordinate of the smallest square centered at v that is entirely enclosing the rectangle P that induces v . The priority of a rectangle P is denoted as $\text{priority}(P)$ and corresponds to the x -coordinate of P^e .

Let us now discuss the handling of various types of events of a rectangle P of core segment s (see Fig. 29). At time t , let $r_1(t)$ and $r_2(t)$ be the rays of slope $+1$ and -1 , respectively, emanating from $l(P^n) \cap l_t$, and $l(P^s) \cap l_t$, respectively, extending towards the left of l_t . Let $a(t)$ and $b(t)$ be the intersection points of $r_1(t)$ and $r_2(t)$ with the wavefront, respectively, at time t . Since the wavefront is ± 1 monotone, $a(t)$ and $b(t)$ can be determined by binary search in $O(\log n)$ time. In case of a wave collinear with $r_1(t)$ or $r_2(t)$, the rightmost endpoint is assigned to $a(t), b(t)$, adopting the convention that an equidistant region is assigned to the rectangle preceding P . Because the wavefront is monotone with respect to any line of slope ± 1 , in case of a vertical rectangle, the entire portion of the wavefront between $a(t)$ and $b(t)$ must be either to the left (Fig. 29a) or the right (Fig. 29b) of the intersection point of $r_1(t)$ and $r_2(t)$, and thus, it may intersect $r_1(t), r_2(t)$, or the core segment of s at most once. For a horizontal rectangle (Fig. 29c), the wavefront can intersect the vertical core segment of P a number of times.

Consider time $t = \text{priority}(P)$. There are three cases: (1) The wavefront has not reached core segment s yet (either at a start-vertical-rectangle or a horizontal-rectangle event); (2) The wavefront has already covered portion of s , where s is horizontal (start-vertical-rectangle event); and (3) The wavefront has already covered a portion of s but s is not horizontal (horizontal-rectangle event).

In case 1, the handling of the corresponding event (a start-vertical-rectangle or a horizontal-rectangle event) is similar to processing an ordinary line-segment event [32, 36]: The portion of the wavefront between $a(t)$ and $b(t)$ is finalized and gets substituted by the wave-curve of P . There is one new action to take: For any crossing V-vertex on the finalized portion of the wavefront, induced by a rectangle Q , generate a V event for the right V-vertex of $b_h(P, Q)$ and insert it to the event queue.

In case 2 (start-vertical rectangle event), a V event is generated to predict the first right V-vertex along s (if any). Given the wavefront, perform a binary search in the augmented wavefront to determine the wave between $a(t)$ and $b(t)$ with the rightmost w-max value as induced by a rectangle Q . If no other information is available, generate a V event for the right V-vertex of $b_h(P, Q)$. Note that at a start-rectangle event, Q must be non-crossing with P . Case 3 will be discussed later at a horizontal-rectangle event.

A V event v is processed similarly. If at time $t = \text{priority}(v)$ the event is *invalid*, i.e., the wavefront has already covered v , generate a new V event, given the wavefront, as described above. In particular, let Q be the rectangle inducing the wave with the rightmost w-max value among the waves between $a(t)$ and $b(t)$. If no additional information is available and assuming that Q is crossing P , generate a V event for the right V-vertex of $b_h(P, Q)$.

End-rectangle events are similar to right events of [32]. For a proof of correctness in handling vertical-rectangle events, see Lemma 2 in [39].

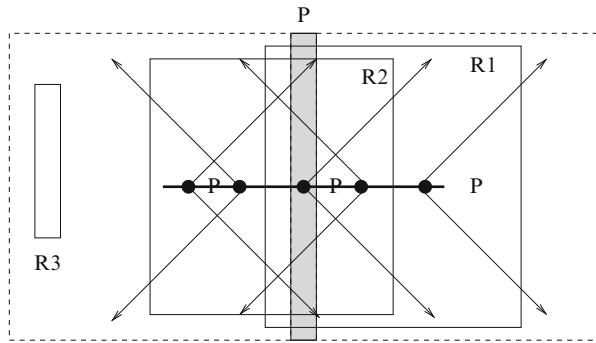
A horizontal-rectangle event is processed at time $t = \text{priority}(P)$. The problem is to identify the intersections (V-vertices) of the wavefront with the vertical core segment s (if any). Note that at time t , the wavefront may intersect s a number of times, each intersection corresponding to a V-vertex, where only the first and the last may be non-crossing V-vertices. If there are no intersections because the wavefront has not reached any portion of s yet, then we have Case 1 of $t = \text{priority}(P)$ discussed earlier. Otherwise, this is Case 3. Any portion of the wavefront to the left of s is finalized and gets substituted by the wave-curve of P as fragmented by the V-vertices and their incident spike bisectors.

To identify V-vertices efficiently, the x -min value of the augmentation is used. Let r be the breakpoint of minimum x -min value between $a(t)$ and $b(t)$. If r is to the right of s , then s must be entirely covered by the wavefront and there can be no intersections, that is, $hreg(P) = \emptyset$. Otherwise, trace the wavefront sequentially, starting at r , until the first intersections above and below r are determined. The intersection above (resp. below) corresponds to a *down* (resp. *up*) V-vertex v (resp. u). Repeat the process for the portions of the wavefront above v and below u until all intersections are determined. Any time the x -min value of a portion of the wavefront is to the right of s , this portion can be eliminated as it contains no intersections with s . The correctness of handling of a horizontal-rectangle event follows easily. A horizontal rectangle event covers also squares.

The time complexity of the plane sweep algorithm, as presented, is $O((n + m + E) \log n)$, where E is the number of invalid V events. There are two reasons for invalid V events: (1) Potentially essential staircases of vertical rectangles whose crossings are not all essential; and (2) Sequences of *strongly dominated* vertical rectangles, even in the case of non-crossing rectangles. Given a pair of vertical rectangles (P, Q) , P is said to *dominate* Q if $Q^w < P^w$ and $Q^n < P^n$, $Q^s > P^s$. If, in addition, there is a minimum enclosing square of Q that is crossing P , P is said to *strongly dominate* Q .

To eliminate source-2 of invalid V-vertex events, a preprocessing step can be added to the algorithm that precomputes *dominating sequences* of vertical

Fig. 30 The dominating sequence of rectangle P consisting of rectangles R_1, R_2, R_3



rectangles, which is described in [39]. Given the preprocessing information, the number of all V events generated is bounded by $O(n + M)$ as shown in [39].

Definition 13 The *dominating sequence* of a vertical rectangle P , denoted $ds(P)$, is a maximal collection of vertical rectangles $R_i, i = 1, \dots, k$, such that every R_i is entirely enclosed in a minimum enclosing square of P , R_1 is the rectangle with the rightmost west edge among all rectangles dominated by P , and if R_i is crossing P , $i \geq 1$, then R_{i+1} is the vertical rectangle with the rightmost west edge dominated by $\text{square}(P, R_{i-1}^e)$.

Every R_i in the *dominating sequence* of P , except possibly the last rectangle R_k , is crossing P . Rectangle R_1 in the dominating sequence of P is referred to as the *rightmost* rectangle dominated by P . In the case of non-crossing rectangles, the dominating sequence of P is R_1 (if not empty). Figure 30 shows the dominating sequence of a vertical rectangle P consisting of three rectangles R_1, R_2, R_3 . Considering only those four rectangles, the region of P is alternating with regions of R_i on the core segment of P , as shown in Fig. 30. Lemma 3 in [39] shows that the dominating sequence of P (except the last rectangle R_k , if R_k is non-crossing with P) forms a maximal staircase of vertical rectangles crossing P that is potentially essential. No vertical rectangle, other than the dominating sequence of P , may induce a right Voronoi V-vertex on the core segment of P , even when only $ds(P)$ is considered. Given the dominating sequences of vertical rectangles, every time a V event is generated, source-2 of invalid V events can be completely eliminated. The exact algorithmic details are given in [39].

Using the information of dominating sequences of vertical rectangles, the total number of V events that may be produced throughout the algorithm is shown in [39] to be $O(n + M)$. The problem of computing the dominating sequence of every vertical rectangle can be transformed into a *point dominance* problem in \mathbb{R}^3 which can be solved by plane sweep in $O((n + M) \log n)$ -time, as described in Sect. 4 of [39]. Combining with Lemma 3 of [39] we conclude.

Theorem 9 *HVD(S) can be computed by plane sweep in $O((n + M) \log n)$ -time and $O(n + M)$ -space. In the case of non-crossing rectangles, this results in optimal $O(n \log n)$ -time and $O(n)$ -space.*

For the plane-sweep algorithm to compute dominating sequences of vertical rectangles, see [39] (Sect. 4). It is based on a simple transformation to a variant of the standard *point dominance* problem in R^3 (see, e.g., [40]) and the use of an auxiliary standard *priority search tree* [28].

5 Conclusion

Given a graph $G = (V, E)$ with a subgraph H embedded in the plane, this chapter presented the problem of computing the *map of geometric minimal cuts* (MGMC) of H , as induced by axis-aligned rectangles in the embedding plane. It surveyed two different approaches for the solution of the MGMC problem [34, 47], that were based on a mix of geometric and graph-algorithm techniques. The chapter also surveyed results on the related Hausdorff Voronoi diagram, which provides a solution to the MGMC problem.

Recommended Reading

1. M. Abellanas, G. Hernandez, R. Klein, V. Neumann-Lara, J. Urrutia, A combinatorial property of convex sets. *Discret. Comput. Geom.* **17**, 307–318 (1997)
2. E. Aurenhammer, Voronoi diagrams – a survey of a fundamental data structure. *ACM Comput. Surv.* **23**(3), 345–405 (1991)
3. F. Aurenhammer, R. Klein, Voronoi diagrams, in *Handbook of Computational Geometry*, ed. by J. Sack, G. Urrutia (Elsevier, Amsterdam, 2000), pp. 201–290
4. F. Aurenhammer, R. Drysdale, H. Krasser, Farthest line segment Voronoi diagrams. *Inf. Process. Lett.* **100**, 220–225 (2006)
5. S.C. Braasch, J. Hibbeler, D. Maynard, M. Koshy, R. Ruehl, D. White, Model-based verification and analysis for 65/45nm physical design, *CDNLive!* September 2007
6. M. de Berg, O. Schwarzkopf, M. van Kreveld, M. Overmars, *Computational Geometry: Algorithms and Applications*, 2nd edn. (Springer, Berlin/New York, 2000)
7. J.P. de Gyvez, C. Di, IC defect sensitivity for footprint-type spot defects. *IEEE Trans. Comput. Aided Des.* **11**, 638–658 (1992)
8. F. Dehne, R. Klein, “The big sweep”: on the power of the wavefront approach to Voronoi diagrams. *Algorithmica* **17**, 19–32 (1997)
9. F. Dehne, A. Maheshwari, R. Taylor, A coarse grained parallel algorithm for Hausdorff Voronoi diagrams, in *Proceedings of the 2006 International Conference on Parallel Processing*, Columbus (2006), pp. 497–504
10. J.R. Driscoll, N. Sarnak, D. Sleator, R. Tarjan, Making data structures persistent, in *STOC*, Berkeley, CA (1986), pp. 109–121
11. H. Edelsbrunner, L.J. Guibas, M. Sharir, The upper envelope of piecewise linear functions: algorithms and applications. *Discret. Comput. Geom.* **4**, 311–336 (1989)
12. D. Eppstein, Z. Galil, G.F. Italiano, A. Nissenzweig, Sparsification – a technique for speeding up dynamic graph algorithms. *J. ACM* **44**(5), 669–696 (1997)
13. G.N. Frederickson, Data structures for on-line updating of minimum spanning trees, with applications. *SIAM J. Comput.* **14**(4), 781–798 (1985)

14. M. Fredman, M. Henzinger, Lower bounds for fully dynamic connectivity problems in graphs. *Algorithmica* **22**(3), 351–362 (1998)
15. S. Fortune, A sweepline algorithm for Voronoi diagrams. *Algorithmica* **2**, 153–174 (1987)
16. P. Gupta, E. Papadopoulou, Yield analysis and optimization, in *The Handbook of Algorithms for VLSI Physical Design Automation*, ed. by C.J. Alpert, D.P. Mehta, S.S. Sapatnekar (Taylor & Francis/CRC, London, 2008)
17. M.R. Henzinger, M. Thorup, Sampling to provide or to bound: with applications to fully dynamic graph algorithms. *Random Struct. Algorithm* **11**, 369–379 (1997)
18. M.R. Henzinger, V. King, Randomized dynamic graph algorithms with polylogarithmic time per operation, in *Proceedings of the 27th STOC*, Las Vegas (1995), pp. 519–527
19. J. Holm, K. de Lichtenberg, M. Thorup, Polylogarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity, in *Proceedings of the 30th STOC*, Dallas, TX (1998), pp. 79–89
20. J. Holm, K. Lichtenberg, M. Thorup, Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM* **48**(4), 723–760 (2001)
21. J. Hopcroft, R. Tarjan, Efficient algorithms for graph manipulation. *Commun. ACM* **16**(6), 372–378 (1973)
22. A.B. Kahng, B. Liu, I.I. Mandoiu, Non-tree routing for reliability and yield improvement. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **23**(1), 148–156 (2004)
23. R. Klein, *Concrete and Abstract Voronoi Diagrams*. Lecture Notes in Computer Science, vol. 400 (Springer, Berlin/New York, 1989)
24. R. Klein, K. Mehlhorn, S. Meiser, Randomized incremental construction of abstract Voronoi diagram. *Comput. Geom.* **3**, 157–184 (1993)
25. R. Klein, E. Langetepe, Z. Nilforoushan, Abstract Voronoi diagrams revisited. *Comput. Geom.* **42**(9), 885–902 (2009)
26. D.T. Lee, On k-nearest neighbor Voronoi diagrams in the plane. *IEEE Trans. Comput.* **C-31**, 478–487 (1982)
27. W. Maly, J. Deszczka, Yield estimation model for VLSI artwork evaluation. *Electron Lett.* **19**(6), 226–227 (1983)
28. E.M. McCreight, Priority search trees. *SIAM J. Comput.* **14**, 257–276 (1985)
29. K. Mehlhorn, S. Näher, Dynamic fractional cascading. *Algorithmica* **5**, 215–241 (1990)
30. P.B. Miltersen, S. Subramanian, J.S. Vitter, R. Tamassia, Complexity models for incremental computation. *Theor. Comput. Sci.* **130**(1), 203–236 (1994)
31. Y. Nakamura, S. ABE, Y. Ohsawa, M. Sakauchi, MD-tree: a balanced hierarchical data structure for multi-dimensional data with highly efficient dynamic characteristics. *IEEE Trans. Knowl. Data Eng.* **5**(4), 682–694 (1993)
32. E. Papadopoulou, Critical area computation for missing material defects in VLSI circuits. *IEEE Trans. Comput. Aided Des.* **20**(5), 583–597 (2001)
33. E. Papadopoulou, The Hausdorff Voronoi diagram of point clusters in the plane. *Algorithmica* **40**, 63–82 (2004)
34. E. Papadopoulou, Net-aware critical area extraction for opens in VLSI circuits via higher-order Voronoi diagrams. *IEEE Trans. Comput. Aided Des.* **30**(5), 704–716 (2011). Preliminary version in *ISAAC'07*. Lecture Notes in Computer Science, vol. 4835 (2007), pp. 716–727
35. E. Papadopoulou, D.T. Lee, Critical area computation via Voronoi diagrams. *IEEE Trans. Comput. Aided Des.* **18**(4), 463–474 (1999)
36. E. Papadopoulou, D.T. Lee, The L_∞ Voronoi diagram of segments and VLSI applications. *Int. J. Comput. Geom. Appl.* **11**, 503–528 (2001)
37. E. Papadopoulou, D.T. Lee, The min-max Voronoi diagram of polygonal objects and applications in VLSI manufacturing, in *Proceedings of the 13th International Symposium on Algorithms and Computation*, Vancouver, BC. Lecture Notes in Computer Science, vol. 2518, (2002), pp. 511–522
38. E. Papadopoulou, D.T. Lee, The Hausdorff Voronoi diagram of polygonal objects: a divide and conquer approach. *Int. J. Comput. Geom. Appl.* **14**(6), 421–452 (2004)

39. E. Papadopoulou, J. Xu, The L_∞ Hausdorff Voronoi diagram revisited, in *IEEE-CS Proceedings, ISVD 2011, International Symposium on Voronoi Diagrams in Science and Engineering*, Qingdao (2011)
40. F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction* (Springer, New York, 1985)
41. D. Sleator, R. Tarjan, A data structure for dynamic trees. *J. Comput. Syst. Sc.* **26**(3), 362–391 (1983)
42. R. Tarjan, Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1**, 146–160 (1972)
43. R.E. Tarjan, Efficiency of a good but not linear set union algorithms. *J. ACM* **22**, 215–225 (1975)
44. M. Thorup, Incremental dynamic connectivity, in *Proceedings of the 8th SODA*, New Orleans (1997), pp. 305–313
45. M. Thorup, Near-optimal fully-dynamic graph connectivity, in *STOC*, Portland (2000), pp. 343–350
46. “Voronoi CAA: Voronoi Critical Area Analysis”, IBM CAD Tool, Department of Electronic Design Automation, IBM Microelectronics, Burlington, VT. Initial patents: US6178539, US6317859. Distributed by Cadence
47. J. Xu, L. Xu, E. Papadopoulou, Computing the map of geometric minimal cuts, in *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC 2009)*, Hawaii, USA. Lecture Notes in Computer Science, vol. 5878, 16–18 Dec 2009, pp. 244–254

Max-Coloring

Julián Mestre and Rajiv Raman

Contents

1	Introduction.....	1872
1.1	Applications.....	1872
1.2	Notations and Definitions.....	1874
1.3	Summary of Known Results.....	1875
2	Tools.....	1881
3	Max-Coloring Paths.....	1886
4	Max-Coloring Trees.....	1888
5	Max-Coloring Bipartite and k -Colorable Graphs.....	1894
5.1	Planar Graphs.....	1897
5.2	Restricted Bipartite Graphs.....	1897
5.3	Split Graphs.....	1898
6	Max-Coloring Interval Graphs.....	1898
7	Max-Coloring on Hereditary Graphs.....	1900
7.1	The Online Case.....	1902
8	Max-Edge-Coloring.....	1903
9	Bounded Max-Coloring.....	1906
10	Conclusion.....	1909
	Cross-References.....	1909
	Recommended Reading.....	1910

J. Mestre (✉)

School of Information Technologies, The University of Sydney, Sydney, NSW, Australia
e-mail: julian.mestre@sydney.edu.au

R. Raman

Indraprastha Institute of Information Technology, Delhi, India
e-mail: rajiv@iiitd.ac.in

Abstract

The *max-coloring problem* is a natural generalization of the classical vertex coloring problem. The input is a vertex-weighted graph. The objective is to produce a proper coloring such that the overall weight of color classes is minimized, where the weight of each class is defined to be the *maximum weight* of vertices in that class.

Max-coloring has received significant attention over the last decade. Approximation algorithms and hardness results are now known for a number of graph classes in both the off-line and online setting. The objective of this chapter is to survey the algorithmic state of the art for the max-coloring problem.

1 Introduction

The *max-coloring problem* is a natural generalization of the classical vertex coloring problem. The input is a vertex-weighted graph. The objective is to produce a proper coloring such that the overall weight of color classes is minimized, where the weight of each class is defined to be the *maximum weight* of vertices in that class. More formally, let $G = (V, E)$ be an undirected graph and $w : V \rightarrow \mathbb{N}$ a weight function. The problem is to find a proper vertex coloring C_1, C_2, \dots, C_k of G that minimizes $\sum_{i=1}^k \max_{v \in C_i} \{w(v)\}$. Note that the special case of this problem in which $w(v) = 1$ for all $v \in V$ is simply the classical minimum coloring problem.

Max-coloring has received significant attention over the last decade. Approximation algorithms and hardness results are now known for a number of graph classes in both the off-line and online setting. The objective of this chapter is to survey the algorithmic state of the art for the max-coloring problem.

The rest of this section reviews some of the applications that motivated the study of max-coloring, introduces some notation, and describes briefly known results. Later sections discuss in more detail some key results.

1.1 Applications

The max-coloring problem arises naturally in the context of buffer management in operating systems and batch scheduling in certain manufacturing settings.

1.1.1 Buffer Minimization

Programs that run with stringent memory or timing constraints use a dedicated memory manager that provides better performance than the general purpose memory management of the operating system. An example of such programs is protocol stacks like GPRS and 3G that run on mobile devices. These protocol stacks have stringent memory requirements as well as soft real-time constraints, so a dedicated memory manager is a natural design choice. A dedicated memory manager for these

stacks must have deterministic response times and use as little memory as possible. The most commonly used memory manager design for this purpose is the *segregated buffer pool*. This consists of a fixed set of buffers of various sizes with buffers of the same size linked together in a linked list. As each memory request arrives, it is satisfied by a buffer whose size is large enough.

The assignment of buffers to memory requests can be viewed as an assignment of colors to the requests – all requests that are assigned a buffer are colored identically. Thus, the problem of determining whether a given segregated buffer pool suffices for a particular sequence of allocation requests can be formalized as follows: Let $G = (V, E)$ be a graph whose vertices are objects that need memory and whose edges connect pairs of objects that are alive at the same time. Requests which are live at the same time have to be satisfied by different buffers. Let $w : V \rightarrow \mathbb{N}$ be a weight function that assigns a natural number weight to each vertex in V . For any object v , $w(v)$ denotes the size of memory it needs. Suppose the segregated buffer pool contains k buffers with weights w_1, w_2, \dots, w_k . The problem is to determine if there is a k -coloring of G into color classes C_1, C_2, \dots, C_k such that for each i , $1 \leq i \leq k$, $\max_{v \in C_i} w(v) \leq w_i$. The optimization version of this problem is the max-coloring problem: Solving the max-coloring problem and selecting buffers according to the optimal solution lead to a segregated buffer pool that uses minimum amount of total memory. Note that this is an off-line problem. In the online version, buffers have to be allocated to requests as they arrive and without knowledge of future requests. The off-line version is also useful because designers of protocol stacks would like to estimate the size of the total memory block needed by extracting large traces of memory requests and running off-line algorithms on these traces.

When memory allocation requests are generated by *straight-line program*, (i.e., programs without loops or branching statements), each memory allocation request is made for a specific duration of time. Thus, these requests can be viewed as intervals on the real line. In this restriction to straight-line programs, the underlying graph is an interval graph. See [Sect. 6](#) for a description.

1.1.2 Batch Scheduling

In several scenarios involving the scheduling of a set of jobs on a set of machines, the jobs are partitioned into *batches*. Jobs in a batch are processed simultaneously, but batches are processed sequentially: Processing of the next batch of jobs starts only after all the jobs in the current batch are complete. Batched scheduling is common in manufacturing processes where a machine is capable of handling several types of jobs but requires a different setup for each job. In this case, all jobs of the same type can be scheduled on the machine, before changing the setup for a different type of job. For example, materials in a factory may have to be processed in a kiln, where all materials that need to be processed at the same temperature can be processed simultaneously, before starting a next batch with a different temperature. Batching thus leads to a higher utilization of the machine and hence more efficient schedules. Another application where batch scheduling is used is in scheduling jobs

on a cluster machine, where jobs that need to communicate to accomplish a task are scheduled together. A paper by Potts and Kovalyov [33] surveys recent results in batched scheduling.

Batch scheduling problems with conflicting jobs can be modeled as a max-coloring problem as follows: The input to a batch scheduling problem is a set of jobs $\mathcal{J} = \{J_1, \dots, J_n\}$ with processing times $w : \mathcal{J} \rightarrow \mathbb{N}$, and a graph G , whose vertex set is the set of jobs and edge between a pair of jobs, implies the two jobs are not compatible and hence cannot be scheduled simultaneously. A proper vertex coloring of this graph corresponds to a schedule where each color class constitutes a batch. The processing time for a batch is the maximum processing time of any job in this batch, and the processing time for the entire set of jobs is the sum of processing times of the batches. Thus, if S_1, \dots, S_k are the independent sets in a coloring, the *weight* of an independent set $w(S_i) = \max_{v \in S_i} w(v)$, and the cost of this coloring is the sum of the weights of the independent sets; That is, $\sum_{i=1}^k w(S_i)$. The problem of minimizing the makespan of this batched schedule is precisely the max-coloring problem.

In some settings, in addition to the time it takes to process the batches themselves, there is a setup time or delay d incurred by each individual batch. This is easily modeled by increasing the weight of every vertex by d .

1.2 Notations and Definitions

In this chapter, all graphs are assumed to be undirected and connected, and in all but for edge-coloring, the graphs are assumed to be simple. A graph $G = (V, E)$ is made up of a vertex set $V(G)$ and an edge set $E(G)$. The notation $n(G)$ and $m(G)$ is used to denote the cardinality of $V(G)$ and $E(G)$, respectively. The maximum degree of G will be denoted by $\Delta(G)$. (For the sake of succinctness, (G) is omitted from these quantities whenever the graph G is clear from the context.) A vertex weight function $w : V \rightarrow \mathbb{N}$ is simply an assignment of positive integer values to the vertices of the graph; similarly, an edge weight function assigns a positive integer value to every edge in the graph. Unless otherwise stated, the weight functions will be of the vertex type.

The notion of an induced subgraph is used in several places throughout the survey. For $S \subseteq V$, $G[S]$ is the vertex-induced subgraph or simply *induced subgraph*, of G that has S for vertex set and $\{(u, v) \in E : u, v \in S\}$ for edge set. Similarly, one can define an *edge-induced* subgraph, or simply *subgraph* as follows: Let $F \subseteq E$, then $G[F]$ is the edge-induced subgraph of G that has F for edge set and endpoints of edges in F for vertex set.

A proper k -coloring of the graph $G = (V, E)$ is a function $\chi : V \rightarrow \{1, \dots, k\}$ such that for every edge $(u, v) \in E$, it is the case that $\chi(u) \neq \chi(v)$. The color classes C_1, \dots, C_k of χ are defined as $C_i = \{v \in V : \chi(v) = i\}$ for each $i \in \{1, \dots, k\}$. Given a weight function w , the weight of color class C_i is defined to be

$$w(C_i) = \max_{v \in C_i} w(v)$$

and the weight of coloring χ to be

$$w(\chi) = \sum_{i=1}^k w(C_i).$$

The chromatic number of a graph G is the smallest k such that G admits a proper k -coloring; this quantity is denoted by $\chi(G)$. The *max-chromatic number*, denoted $\chi_{mc}(G)$, denotes the smallest number k such that there is an optimal max-coloring using k colors, and $\text{OPT}_{mc}(G)$ denotes the weight of an optimal max-coloring of G . Finally, $\omega(G)$ denotes the size of the largest clique in the graph; notice that $\omega(G) \leq \chi_{mc}(G)$.

Let \mathcal{A} be an algorithm for a given minimization problem, and let $\rho \geq 1$. The algorithm \mathcal{A} is said to be a ρ -approximation or that \mathcal{A} attains an approximation ratio of ρ if for all instances of the optimization problem, the solution returned by \mathcal{A} has cost at most ρ times the optimal solution. In the online context, where the instance is not revealed all at once but rather progressively, over time, here, algorithm \mathcal{A} is said to be ρ -competitive if it returns a solution whose cost is no more than ρ times the optimal solution that can be constructed after the whole instance has been revealed. A polynomial time approximation scheme (PTAS) is an algorithm such that for all $\epsilon > 0$, the algorithm returns a $1 + \epsilon$ approximate solution in $O(n^{f(\epsilon)})$, where f is an arbitrary function.

Let G be a graph, and let C_1, \dots, C_k be some proper coloring of G . The chapter always follows the convention of listing color classes in non-increasing order of their weights; that is, $w(C_1) \geq w(C_2) \geq \dots \geq w(C_k)$. Further, it is assumed without loss of generality that C_i is a *maximal independent set* in the graph induced by $\cup_{j=i}^k C_j$; indeed, otherwise, vertices can be migrated to lower indexed color classes without increasing the weight of the coloring.

1.3 Summary of Known Results

The majority of papers on max-coloring focus on solving the problem on different graph classes. These results are summarized in [Table 1](#) for the off-line setting and in [Table 2](#) for the online setting, where vertices are revealed one by one; in which case, a vertex must be assigned a color as soon as it arrives. Another line of work has studied a variant of max-coloring where the objective is to color the edges of the graph; these results are summarized in [Table 3](#). Yet another line of work has studied a variant of all these problems where there is an additional constraint on the size of the color classes used; these results are summarized in [Table 4](#).

The rest of this section briefly describes the results in [Tables 1–4](#). Later sections highlight and elaborate in more detail a selected subset of these results.

Table 1 Table for off-line max-coloring vertices of graphs. A lowerbound of $c > 1$ means that there is no $(c - \epsilon)$ -approximation unless $P = NP$. All algorithms run in polynomial time unless stated otherwise in the notes column

Graph class	Lowerbounds	Upperbounds	Notes	Section
Path	–	1 [14, 18, 19, 22]		§ Sect. 3
Tree	–	1 [30]	$n^{O(\log n)}$ time	§ Sect. 4
		$(1 + \epsilon)$ [18, 30]	$n^{O(1/\epsilon)}$ time	§ Sect. 4
Low treewidth	–	$(1 + \epsilon)$ [14]	$n^{O(1/\epsilon + tw)}$ time	–
Bipartite	8/7 [30]	8/7 [14, 30]		§ Sect. 5
k -Colorable	–	$\frac{k^3}{3k^2 - 3k + 1}$ [14]	Assumes k -coloring is computable	§ Sect. 5
Split	NP-hard [8]	$(1 + \epsilon)$ [7]	$n^{O(1/\epsilon)}$ time	–
Interval	NP-hard [31]	$(1 + \epsilon)$ [28]		§ Sect. 6
Perfect	8/7 [30]	e [34]		–
Hereditary	–	$e \cdot c$ [12, 34]	Assumes c -approx. for regular coloring	§ Sect. 7

Table 2 Table for online max-coloring vertices of graphs. All results appear in the paper of Epstein and Levin [12]. Each row is subdivided into results for deterministic and randomized algorithms. A lowerbound of $c > 1$ means that there is no $(c - \epsilon)$ -competitive algorithm. For interval graphs, two models are considered: In the list model, intervals arrive at in arbitrary order; in the time model, intervals arrive sorted by their left endpoint. These results are covered in Sect. 7.1

Graph class	Lowerbounds	Upperbounds	Notes
Hereditary	–	$4 \cdot c$	Assumes c -comp. for regular online coloring
	–	$e \cdot c$	
Interval (list model)	4	12	Intervals arrive in arbitrary order
	4	$3e$	
Unit interval (list model)	2	8	Unit intervals arrive in arbitrary order
	11/6	$2e$	
Interval (time model)	$\phi \approx 1.618$	4	Intervals arrive sorted by left endpoint
	4/3	e	

Table 3 Table for max-edge-coloring graphs. A lowerbound of $c > 1$ means that there is no $(c - \epsilon)$ -approximation unless $P = NP$

Graph class	Lowerbounds	Upperbounds
Paths	–	1 [14, 18, 19, 22]
Trees	–	PTAS [26]
Bipartite	7/6 [7]	1.74 [26]
General simple graphs	$4/3$ [20]	$2 - \frac{2}{\Delta+2}$ [4]
General multigraphs	$4/3$ [20]	$2 - \frac{2}{1.5\Delta+1}$ [4]

1.3.1 Paths

The simplest graph class imaginable is that of simple paths. It should not be surprising then to learn that max-coloring paths can be solved optimally in polynomial time. Indeed, it is not hard to see: First, note that an optimal solution

Table 4 Table for bounded max-coloring and bounded max-edge-coloring. A lowerbound of $c > 1$ means that there is no $(c - \epsilon)$ -approximation unless $P = NP$

Coloring type	Graph class	Lowerbounds	Upperbounds	Notes
Vertex	Trees	—	PTAS [2]	$O(n^{2^{1/\epsilon}})$ time
	Bipartite	4/3 [3]	17/11 [2]	
	Hereditary	—	$c + 1$ [6]	Assumes c -approx. for max-coloring
Edge	Trees	NP-hard [2]	2 [2]	
	Bipartite	7/8 [7]	e [2]	
	General	4/3 [20]	3 [2,6]	

uses at most three colors; otherwise, one can recolor the vertices in the lightest color class and decrease the cost of the solution. The algorithm guesses the weights of these three color classes (there are $O(n^3)$ possibilities to consider) and for each guess uses dynamic programming to check in linear time whether a 3-coloring with these weights exists. This is the basis of the $O(n^4)$ time algorithm of Guan et al. [18]. The running time was subsequently improved to $O(n^2)$ by Escoffier et al. [14], then to $O(n \log n)$ by Halldorsson and Schachnai [19], and finally to $O(n + S(n))$ by Kavitha and Mestre [22], where $S(n)$ is the time it takes to sort the weights. The last paper also provided a matching lower-bound of $\Omega(n \log n)$ in the algebraic computation tree model on the running time required for max-coloring a path. The algorithm of Kavitha and Mestre is covered in Sect. 3.

1.3.2 Trees

While the classic graph coloring problem is trivial on trees (any tree can be colored with at most two colors), the max-coloring problem on trees has turned out to be a challenging problem. It can be shown that on trees, an optimal max-coloring may use as many as $\lceil \log_2 n \rceil$ colors, and the best algorithm to compute an optimal max-coloring runs in quasi-polynomial time, that is, in time $2^{\text{poly} \log n}$. This quasi-polynomial time algorithm can also be used as a basis for obtaining a PTAS for the problem, and these results extend directly to graphs of bounded treewidth, provided a tree decomposition is given. These results are presented in Sect. 4. No hardness result is known for max-coloring trees. Indeed, settling its computational complexity remains an intriguing open problem.

1.3.3 Bipartite Graphs

While the max-coloring problem on trees is yet to be fully resolved, the situation is different for the class of bipartite graphs that includes trees. On bipartite graphs, it can be shown that an optimal max-coloring may use as many as $O(n)$ colors on a graph with n vertices. Pemmaraju and Raman [30] and independently Escoffier et al. [14] gave tight results for the max-coloring problem on bipartite graphs. The authors show that the problem is hard to approximate beyond $\frac{8}{7} - \epsilon$ for any $\epsilon > 0$.

unless $P = NP$, and they give a combinatorial algorithm that achieves a ratio of $\frac{8}{7}$. The surprising aspect of this algorithm is that it uses at most four colors, while an optimal max-coloring may use as many as $O(n)$ colors.

Several authors have further tried to delineate the boundary between polynomial time solvable cases and the NP-hard case for bipartite graphs. DeWerra et al. [7] prove that on P_5 -free bipartite graphs (i.e., bipartite graphs that do not have a path on five vertices as an induced subgraph), the max-coloring problem admits a polynomial time solution, while on P_8 -free bipartite graphs, the authors show that the problem becomes NP-hard.

1.3.4 Split Graphs

A Split graph is a graph $G = (K \cup S, E)$, where the vertex set V can be partitioned into a complete graph K and an independent set S . Between K and S is an arbitrary bipartite graph. Split graphs are a subclass of perfect graphs, and hence, the classic coloring problem admits a polynomial time algorithm. DeWerra et al. [7] study the max-coloring problem on split graphs. They prove the problem to be NP-complete and give a PTAS.

1.3.5 k -Colorable Graphs

Even though coloring general graphs is a very difficult problem, certain graph classes can be optimally colored or good approximations exist. It makes sense then to study how well max-coloring can be approximated in these special classes. Max-coloring is APX-hard even for bipartite graph, where an optimal coloring can be found in linear time, so one should not expect an approximation preserving reduction between the two problems. Nevertheless, being able to compute a k -coloring of the input graph is useful for approximating max-coloring. Indeed, any proper k -coloring is a k -approximation for max-coloring. Can this trivial observation be improved? Bourgeois et al. [4] considered this question and showed how to get a $\frac{k^3}{3k^2 - 3k + 1}$ approximation for max-coloring provided a k -coloring for the input graph can be computed. This result is described in Sect. 5. Here, two applications to planar graphs and triangle-free planar graphs are discussed.

The celebrated four-color theorem states that every planar graph can be vertex-colored using four colors. Robertson et al. [35] provide an $O(n^2)$ time algorithm for 4-coloring any planar graph. Using the above result, one can get a $\frac{64}{37} < 1.73$ approximation for max-coloring planar graphs. On the negative side, the hardness of graph coloring planar graphs [16] means it is NP-hard to approximate max-coloring better than $4/3 > 1.33$.

Grötzsch theorem states that every triangle-free planar graph can be 3-colored. Dvorak et al. [10] give a linear time algorithm to compute such a coloring. Using the above result, one can get a $\frac{27}{19} < 1.43$ approximation for max-coloring triangle-tree planar graphs. On the negative side, De Werra et al. [7] showed that in this class of graphs, max-coloring cannot be approximated better than $\frac{7}{6} > 1.16$.

1.3.6 Interval Graphs

Interval graphs are intersection graphs of intervals on the real line. In other words, a graph is an interval graph if and only if each vertex can be associated with an interval on the real line (closed or open) such that two intervals intersect if and only if their corresponding vertices are adjacent. Interval graphs arise in several applications from scheduling to computational biology. They are a sub-class of perfect graphs, and hence, $\chi(H) = \omega(H)$ holds for any interval graph G and induced subgraph $H \subseteq G$. Another feature that will be useful is that given a graph, one can decide in polynomial time whether it is an interval graph and, if so, construct an interval representation [5]. Hence, for these algorithms, one can assume that the interval representation is given.

Interval graphs being perfect, minimum coloring can be solved in polynomial time. However, one can compute a coloring much easier than for perfect graphs by sorting the intervals in order of their left end-points and coloring them in a greedy manner with the smallest available color at each step. (See the book by Golumbic [17] for further details.) However, the max-coloring problem is NP-hard on interval graphs. A 2-approximation was presented by Pemmaraju et al. [31]; this result appears in Sect. 6. Very recently, a PTAS has been proposed by Nonner [28], which settles the computational complexity of max-coloring in interval graphs.

1.3.7 Hereditary Graph Classes

A class of graphs \mathcal{G} is called *hereditary* if it is closed under taking vertex-induced. Hence, for all graphs $G \in \mathcal{G}$, if $H \subseteq G$ is an induced subgraph of G , this implies $H \in \mathcal{G}$ as well. Most of the graph classes considered in this survey are hereditary graph classes. Clearly, the max-coloring problem is NP-hard on hereditary graphs. However, one can show that max-coloring can be approximated as well (up to constant factors) as regular coloring. More precisely, a c -approximation algorithm for classic coloring can be used to produce an $e \cdot c$ -approximation algorithm for max-coloring on this class of graphs; here, e is the base of the natural logarithm. This result is presented in Sect. 7.

1.3.8 Online

In some application domains, the instance is not given to us all at once, but rather, vertices are revealed one by one. For instance, this is the case in the buffer minimization application described in Sect. 1.1, where memory requests depend on the actual execution path of the program that is running.

Epstein and Levin [12] studied max-coloring in the online setting. Vertices arrive one by one. When a vertex u arrives, its neighborhood (restricted to the vertices that have already arrived) is revealed. Upon u 's arrival, the algorithm must assign a valid color to u that cannot be changed later on. They showed a reduction from online max-coloring to online coloring in hereditary graph classes. Given a c -competitive algorithm for online coloring, they design a deterministic $4 \cdot c$ -competitive algorithm for max-coloring and a randomized $e \cdot c$ -competitive algorithm. This result and some applications thereof are described in Sect. 7.1.

Epstein and Levin also considered the interesting class of interval graphs. Here, they assumed that the interval representation, rather than the graph itself, is revealed. They considered two different interval arrival models: In the *list model*, intervals arrive in some arbitrary order, while in the *time model*, intervals arrive sorted by their left endpoint. In the time model, a simple greedy algorithm solves the online coloring problem optimally. Thus, the reduction for general hereditary graph classes yields 4-competitive deterministic and ϵ -competitive randomized algorithms for max-coloring interval graphs in the time model. Similar results are obtained by using the 3-competitive algorithm of Kierstead and Trotter [24] for coloring interval graphs in the list model and the 2-competitive algorithm of Epstein and Levy [13] for coloring unit interval graphs in the list model.

On the negative side, the authors show a number of lower bounds for online max-coloring interval graphs under these two models. The results are summarized in [Table 2](#).

1.3.9 Max-Edge-Coloring

An important application of max-coloring arises in the context of scheduling transmissions in optical switches. Optical switches can be modeled as a complete bipartite graph connecting n input ports and n output ports. In each time step, the switch transmits packets along a perfect matching between input and output ports. A scheduler procedure is responsible for deciding which matching to use at each time step in order to route a given traffic demand pattern. In the language of graphs, the problem is defined by an edge-weighted bipartite graph $G = (A, B, E)$, where vertices in A and B represent input and output ports, respectively; edges represent data transfers that must be carried out; and weights represent the time it takes to carry out the transfer. Transmissions occur in rounds or batches. The set of edges associated with the transmissions carried out during a single round must form a matching. The objective is to minimize the makespan of the schedule. In a sense, this is just another example of batch scheduling. However, the difference here is that colors are assigned to edges of the input graph instead of the vertices.

The problem was first studied by Kesselman and Kogan [23], who showed that a simple greedy algorithm is a 2-approximation for the problem. This was later slightly improved by Bourgeois et al. [4], who showed that taking the best between greedy and an arbitrary k -coloring is a $2 - \frac{2}{k+1}$ approximation. These algorithms are described in [Sect. 8](#). It is worth noting that for general graphs, max-edge-coloring inherits the $(\frac{4}{3} - \epsilon)$ -hardness of regular edge-coloring [20].

Max-edge-coloring remains NP-hard in planar bipartite of degree 3 with edges weights in $\{1, 2, 3\}$ in fact, this case is hard to approximate better than $\frac{7}{6}$ [7]; this is in stark contrast with regular edge-coloring, which is polynomially solvable for general bipartite graphs. Bourgeois et al. [4] give two approximation algorithms for bipartite graphs: The first one has an approximation ratio of $2 - \frac{2}{\Delta+1}$, while the second one has attained an approximation ratio of $\frac{2(\Delta+1)^3}{\Delta^3+5\Delta^2+5\Delta+3-(-1/\Delta)\Delta}$. Very recently, Lucarelli and Milis [26] gave a 1.74-approximation algorithm, the current best factor for the

problem. It is worth mentioning that this last algorithm can be adapted for general graph, however, yielding an asymptotic approximation factor of 1.74.

The complexity of max-edge-coloring trees is open, as in the case of max-(vertex)-coloring trees. As in this other case, there is PTAS for max-edge-coloring trees [26].

Max-edge-coloring can be reduced to max-(vertex)-coloring by creating a new graph, where every new vertex corresponds to an old edge and two new vertices are connected if the corresponding old edges share an endpoint; the new graph is called the *line graph* of the old graph. Since the line graph of a path is again a path, max-edge-coloring can be solved optimally in paths.

1.3.10 Bounded Max-Coloring

Consider the batch scheduling application, where each batch consists of jobs that are non-conflicting. In the language of max-coloring, each color class consists of a batch of jobs that can run simultaneously. In many applications, however, there is a fixed number of machines, m , and each job in a batch runs on a separate machine. In such applications, the number of machines is typically fixed, so the size of the batches should not exceed m ; in other words, each color class must have at most m vertices.

Coloring unweighted graphs with a bounded size on each color class is well studied, and there are results on several restricted graph classes. The problem was introduced by Baker and Coffmann [1] under the title “Mutual Exclusion Scheduling.” For permutation graphs that are a subclass of comparability graphs, Jansen [21] proved that the problem is NP-hard for $m \geq 6$. For interval graphs, Bodlaender and Jansen [3] have shown that the problem on interval graphs is NP-hard for $m \geq 4$. For $m = 2$, the problem can be solved easily in polynomial time via matching techniques, while Gardi [15] gave linear time algorithms for interval and circular-arc graphs for $m = 2$.

Clearly all the hardness results for the max-coloring problem carry over to the bounded case. The bounded case, however, seems to be harder: For some graph classes for which the max-coloring problem can be solved in polynomial time, the bounded max-coloring problem is NP-hard. On the positive side, there is an algorithm for hereditary graph classes such that given an α -approximation algorithm for the max-coloring problem, it is possible to compute an $\alpha + 1$ -approximation for the bounded max-coloring problem. Bampis et al. [2] obtained a number of results for different graphs classes, which are summarized in [Table 4](#). These results are presented in [Sect. 9](#).

2 Tools

This section develops some tools and techniques that have proved useful for attacking the max-coloring problem on various graph classes. First, consider the simple bound on $\chi_{mc}(G)$ for any graph G . Assume C_1, C_2, \dots, C_k is an optimal max-coloring of a given graph G .

Lemma 1 ([18]) *For any graph G and any weight function $w : V \rightarrow \mathbb{N}$, $\chi_{mc}(G) \leq \Delta + 1$*

Proof Let $k = \chi_{mc}(G)$. If $k > \Delta + 1$, then for each vertex $v \in C_k$, there is a color class C_i , $i < k$, such that v has no neighbors in C_i . Note that since $w(C_i) \geq w(C_k)$, when v is moved into C_i , the weight of C_i does not increase. Furthermore, when C_k becomes empty, the weight of the coloring decreases, contradicting the optimality of C_1, C_2, \dots, C_k . \square

Consider an optimal max-coloring of a graph G such that $\chi(G) = l$. The reason max-coloring uses more colors than l to color G is that coloring heavy weight vertices with the same color offsets the increase in weight due to the use of many more color classes. This relation is captured in the following lemma, which is a key property of any optimal max-coloring.

Lemma 2 *Let G be an l -colorable graph, and let C_1, \dots, C_k be an optimal max-coloring with $k > l$ colors. Then, for each $i = 1, \dots, k$, $(l - 1)w(C_i) > \sum_{j=i+1}^k w(C_j)$.*

Proof The vertices in $\cup_{j=i}^k C_j$ can be l -colored for a cost of at most $l \cdot w(C_i)$. Since k colors are used, it must be that $\sum_{j=i}^k w(C_j) < l \cdot w(C_i)$, and the result follows. \square

In the max-coloring problem, it makes sense to color the large weight vertices with the same color to avoid paying for them several times in each color class. A natural approach is to partition the vertices into groups such that within each group either the number of distinct weights is small or the variance between the largest and smallest weights is small (or both). Then one can develop algorithms that work well in this restricted setting, apply the algorithm separately to each set of the partition, and obtain a max-coloring for the entire graph. The next lemma shows that such a partition can be always found at the expense of only a small increase in the approximation factor.

Lemma 3 ([28]) *Let $G = (V, E)$ be a vertex-weighted graph with weight function $w : V \rightarrow \mathbb{N}$, and let $\epsilon > 0$ be a constant. Then, G can be partitioned into induced subgraphs G_0, G_1, \dots such that within each induced subgraph, the ratio w_{max}/w_{min} is $O(\epsilon^{-2})$, the number of distinct weights is $2^{O(1/\epsilon)}$, and optimally max-coloring each subgraph independently leads to a $(1 + \epsilon)$ approximation for G .*

Proof Let $\alpha = \alpha(\epsilon) > 1$ be a constant that depends on ϵ . Define a new weight function as follows: Let $d(v) = \lceil \log_\alpha w(v) \rceil$, and let $w'(v) = \alpha^{d(v)}$ for each $v \in V$. Hence, each vertex weight is rounded up to the nearest power of α . The graph with the new weights is called the *rounded instance*. Let $\text{OPT}_{mc}^\alpha(G)$ denote the weight of an optimal coloring with the new weights w' . Then, the following is true:

$$\text{OPT}_{mc}(G) \leq \text{OPT}_{mc}^\alpha(G) \leq \alpha \cdot \text{OPT}_{mc}(G) \quad (1)$$

The first inequality clearly holds since the weight of each vertex increases. The second inequality holds since each vertex weight increases by at most a factor of α , so using the coloring with weight $\text{OPT}_{mc}(G)$ with the new weights has cost at most $\alpha \cdot \text{OPT}_{mc}(G)$.

Now, let $\delta = \delta(\epsilon) > 0$ be a constant that depends on ϵ . Let $m = \max_{v \in V} d(v)$. Hence, m is the number of distinct weights in the rounded instance. Partition the set of vertices into $l = \delta \cdot m$ groups, each group consisting of $\frac{1}{\delta}$ contiguous weights. Assume that $\frac{1}{\delta} \in \mathbb{N}$ and that $\delta \cdot m \geq 1$; otherwise, m is already constant. The partitioning will be randomized. To this end, pick a value z uniformly at random from the range $\{1, 2, \dots, \frac{1}{\delta}\}$. Using this value z define the partition of the vertex set as follows:

$$V_i = \left\{ v \in V : \frac{i-1}{\delta} + z < d(v) \leq \frac{i}{\delta} + z \right\}, \quad i = 0, 1, \dots, l-1.$$

The graph G_i is the graph induced by the vertices in V_i for $i = 0, 1, \dots, l-1$. The idea is to compute an optimal max-coloring for each group separately, using a fresh palette for each G_i , and return the union as a max-coloring of G . It can be shown that this solution has weight that is not much more than $\text{OPT}_{mc}(G)$.

Note that the colorings produced by partitioning and solving the max-coloring problem on each group separately has the feature that each color class consists only of vertices from the same set V_i . In order to show that not much is lost by partitioning, consider an optimal max-coloring of the rounded instance of weight $\text{OPT}_{mc}^\alpha(G)$ with color classes C_1, \dots, C_k . Let $d(C_i) = \lceil \log_\alpha w(C_i) \rceil$. Define $D_{ij} = C_i \cap V_j$, for $i = 1, \dots, k$ and $j = 0, \dots, l-1$. Then, the color classes D_{ij} consist only of vertices from the set V_j . Let $S^\alpha(G)$ denote the weight of the new coloring. Then, we have

$$\begin{aligned} \mathbb{E}[S^\alpha(G)] &= \mathbb{E} \left[\sum_{i=1}^k \sum_{j=0}^{l-1} w(D_{ij}) \right], \\ &= \sum_{i=1}^k \mathbb{E} \left[\alpha^{d(C_i)} + \sum_{j: \frac{j}{\delta} + z < d(C_i)} w(D_{ij}) \right], \\ &\leq \sum_{i=1}^k \alpha^{d(C_i)} + \mathbb{E} \left[\sum_{j: \frac{j}{\delta} + z < d(C_i)} \alpha^j \right], \\ &= \sum_{i=1}^k \alpha^{d(C_i)} + \sum_{x=1}^{\frac{1}{\delta}} \mathbb{P}[z = x] \cdot \sum_{j: \frac{j}{\delta} + z < d(C_i)} \alpha^j, \end{aligned}$$

$$\begin{aligned}
&= \sum_{i=1}^k \alpha^{d(C_i)} + \delta \cdot \sum_{j < d(C_i)} \alpha^j \text{ (each } j \text{ occurs exactly once per choice of } \delta), \\
&\leq \sum_{i=1}^k \alpha^{d(C_i)} + \delta \cdot \frac{\alpha^{d(C_i)+1}}{\alpha - 1}, \text{ (sum of geometric series)} \\
&= \left(1 + \frac{\delta}{\alpha - 1}\right) \sum_{i=1}^k \alpha^{d(C_i)} = \left(1 + \frac{\delta}{\alpha - 1}\right) \cdot \text{OPT}_{mc}^\alpha(G).
\end{aligned}$$

Now from inequality (1), it follows that

$$\begin{aligned}
\mathbb{E}[S^\alpha(G)] &\leq \left(1 + \frac{\delta}{\alpha - 1}\right) \text{OPT}_{mc}^\alpha(G), \\
&\leq \alpha \cdot \left(1 + \frac{\delta}{\alpha - 1}\right) \text{OPT}_{mc}(G), \\
&= \left(\alpha + \frac{\alpha}{\alpha - 1} \cdot \delta\right) \text{OPT}_{mc}(G),
\end{aligned}$$

so setting $\alpha = 1 + \epsilon/2$ and $\delta \leq \frac{\epsilon^2/4}{1+\epsilon/2}$,

$$\begin{aligned}
&\leq \left(1 + \epsilon/2 + \frac{1 + \epsilon/2}{\epsilon/2} \cdot \frac{\epsilon^2/4}{1 + \epsilon/2}\right) \text{OPT}_{mc}(G), \\
&= (1 + \epsilon) \text{OPT}_{mc}(G).
\end{aligned}$$

Such a partition can be found deterministically by trying all possible values of z and hence de-randomize the above construction.

Finally, note that within each subgraph G_i , the ratio w_{\max}/w_{\min} is at most $\alpha^{1/\delta}$. Thus, $w_{\max}/w_{\min} = 2^{O(1/\delta)}$. \square

With Lemma 3 in hand, one can focus on developing algorithms for the max-coloring problem, where the number of distinct weights is bounded by a constant and the ratio w_{\max}/w_{\min} is bounded by a constant. This is the approach used for several graph classes such as trees, interval graphs, and hereditary graph classes.

A second useful tool is a connection between the max-coloring problem and the problem of *online coloring* of unweighted graphs. Online graph coloring has been studied for several years. For many graph classes there are good bounds on the online chromatic number and algorithms that produce good colorings. Suppose there is an online algorithm for some hereditary graph class. Using this online algorithm as a “black-box,” one can design an algorithm, MCA, for off-line max-coloring.

The MCA algorithm simply sorts the vertices of G in non-increasing order of weights and feeds the vertices in this order to an online algorithm. For any hereditary

Algorithm 1 MCA(G, w)

-
1. Sort vertices of G in non-increasing order of weights
// Let (v_1, \dots, v_n) be this ordering
 2. Present the vertices in this order to A
 3. **return** Coloring produced by A
-

class of graphs, one can give a bound on the approximation ratio achieved by MCA in terms of the *competitive ratio* of the online algorithm for this class of graphs. The algorithm MCA is described in [Algorithm 1](#).

Lemma 4 ([31]) *Let \mathcal{C} be a hereditary class of graphs, and let A be an algorithm for online graph coloring such that for some $c > 0$ and for any graph $G \in \mathcal{C}$, algorithm A k -colors G for some $k \leq c \cdot \chi(G)$. Then, for any $G \in \mathcal{C}$ and for any weight function $w : V(G) \rightarrow \mathbb{N}$, MCA produces a coloring for G whose weight is at most $c \cdot \text{OPT}_{mc}(G)$.*

Proof Let C_1, C_2, \dots, C_k be a coloring of G that is optimal for the max-coloring problem. Note that $k \geq \chi(G)$, and $\text{OPT}_{mc}(G) = \sum_{i=1}^k w(C_i)$. Let A_1, A_2, \dots, A_t be the coloring of G produced by MCA. As per convention, assume that $w(A_1) \geq w(A_2) \geq \dots \geq w(A_t)$. From the hypothesis, it follows that $t \leq c \cdot \chi(G) \leq c \cdot k$. For notational convenience, define sets $A_{t+1} = A_{t+2} = \dots = A_{c \cdot k} = \emptyset$, and let $w(A_i) = 0$ for $i, t < i \leq c \cdot k$. Suppose that for each i , $1 \leq i \leq k$, and each j , $c(i-1) < j \leq c \cdot i$, $w(C_i) \geq w(A_j)$. This implies the result sought because the coloring produced by MCA has weight

$$\begin{aligned} \sum_{\ell=1}^{c \cdot k} w(A_\ell) &= \sum_{i=1}^k \sum_{j=c(i-1)+1}^{c \cdot i} w(A_j) \\ &\leq \sum_{i=1}^k c \cdot w(C_i) \\ &\leq c \cdot \text{OPT}_{mc}(G). \end{aligned}$$

Since $w(C_1)$ is the maximum weight of any vertex in G , the supposition is trivially true for $i = 1$. For any $i \geq 2$, let $V_i \subseteq V$ be defined as $V_i = \{v : w(v) > w(C_i)\}$. The coloring C_1, C_2, \dots, C_{i-1} is an $i-1$ coloring of $G[V_i]$.

Because of the order in which vertices are presented to A , all vertices in V_i are presented to A before any vertex with weight $w(C_i)$. Therefore, by the hypothesis, algorithm A colors $G[V_i]$ with no more than $c \cdot (i-1)$ colors. Therefore, the weight of the heaviest vertex in color classes A_j for $j, c(i-1) < j \leq c \cdot i - 1$ is at most $w(C_i)$. \square

A simple and natural heuristic for online coloring a graph is called *First fit*. The algorithm is as follows: Assume that the colors are natural numbers. When a vertex is presented, color it with the *smallest* available color. Let $\chi_{FF}(G)$ denote the maximum number of colors required by First fit to color the graph G over all possible presentations of the vertices. The quantities $\chi_{mc}(G)$ to $\chi_{FF}(G)$ can be related as follows.

Lemma 5 ([18]) *Let $G = (V, E)$ be a weighted graph. Then $\chi_{mc}(G) \leq \chi_{FF}(G)$.*

Proof Let χ_{mc} be an optimal max-coloring of G , and let C_1, \dots, C_k be the color classes in this coloring. Presenting the vertices in this order, that is, all the vertices of C_1 followed by all the vertices of C_2 and so on, to the first-fit algorithm, the algorithm uses at least χ_{mc} colors. Hence, $\chi_{mc}(G) \leq \chi_{FF}(G)$. \square

The first-fit heuristic, due to its simplicity and good performance in practice, has been studied for various graph classes. Lemma 5 along with Lemma 4 immediately gives good approximation factors for max-coloring on several hereditary graph classes. For example, the first-fit algorithm is known to be 8-competitive on interval graphs, and since interval graphs are a hereditary graph class, an 8-approximation algorithm for max-coloring interval graphs follows. Similarly, the result implies nontrivial approximation ratios for k -colorable graphs even when there are no algorithm to compute a k -coloring. In particular, for 3-colorable hereditary graphs, Lemma 4 implies an $O(\sqrt{n})$ -approximation for the max-coloring problem, since the class of 3-colorable graphs can be online colored with at most $O(\sqrt{n})$ colors [37].

3 Max-Coloring Paths

This section describes a fast algorithm for max-coloring a path P . Since paths have maximum degree two, it follows from Lemma 1 that any path admits an optimal max-coloring using at most three colors. Hence, from now on, just consider finding an optimal max-3-coloring of the given input path. For ease of exposition, assume that the set of vertices with maximum weight forms an independent set (this can be checked in linear time); otherwise, any 2-coloring is an optimal max-coloring.

The algorithm works in iterations. In each iteration, it records a signature $[w_1, w_2, w_3]$ and (implicitly) a coloring $\{A_1, A_2, A_3\}$ such that $w(A_i) \leq w_i$. To that end, keep two sets L and H . The idea is to color H with colors $\{1, 2\}$ and L with all three colors $\{1, 2, 3\}$. In each iteration, transfer the heaviest vertex from L to H and thus decrease the value of w_3 until it reaches zero. At the end, the best coloring found is returned. Call this algorithm MAX-COLORING-PATHS; its pseudo-code is given in Algorithm 2. The rest of this section deals with proving the following theorem.

Theorem 1 *Algorithm MAX-COLORING-PATHS finds an optimal max-coloring of a path in $O(n + S(n))$ time, where $S(n)$ is the time it takes to sort the vertex weights.*

Algorithm 2 MAX-COLORING-PATHS(P, w)

-
1. $w_2 \leftarrow w_3 \leftarrow \max_{(u,v) \in P} \min\{w(u), w(v)\}$
 2. $w_1 \leftarrow \max_{v \in P} w(v)$
 3. $H \leftarrow \{v \in P : w(v) > w_2\}$
 4. $L \leftarrow \{v \in P : w(v) \leq w_2\}$
 5. sort vertices in L in non-increasing order of weight
 6. **for** $z \in L$ in sorted order **do**
 7. transfer z from L to H
 8. let p be the subpath of H containing z
 9. $[\alpha, \beta] \leftarrow$ optimally 2-color p
 10. $w_2 \leftarrow \max\{\beta, w_2\}$
 11. $g_i \leftarrow \max_{v \in L} w(v)$
 12. record $[w_1, w_2, w_3]$ as a candidate signature
 13. **return** coloring associated with best signature recorded
-

Proof First, it has to be shown that the algorithm can be implemented to run in the stated time. The algorithm spends $S(n)$ time in Line 5; thus, it must be shown that the rest of the algorithm runs in linear time. The only challenging steps are Lines 8 and 9; these operations can be done in $O(1)$ time by keeping some information about the set H .

Each connected component of H is a subpath of P . Let $p = (u_0, u_1, \dots, u_\ell)$ be one such component. Associated with p are the values $e(p) = \max_{u_i \in p: i \text{ even}} w(v_i)$ and $o(p) = \max_{u_i \in p: i \text{ odd}} w(v_i)$. Given this information, it is trivial to carry out Line 9, since $\alpha = \max\{e(p), o(p)\}$ and $\beta = \min\{e(p), o(p)\}$. Notice that when transferring z from L to H , a new subpath in H could be started, or u could be added to an already existing subpath, or u could merge two adjacent subpaths. In each case, $e(p)$ and $o(p)$ should be computed $e(p)$ and $o(p)$ for the new subpath. This can easily be done if in addition to $e(p)$ and $o(p)$, the parity of the length of each component p in H is kept. The information for a particular subpath p is kept at the endpoints of p . With the aid of this basic data structure, the algorithm, excluding Line 5, runs in linear time.

The correctness of the algorithm relies on the following two lemmas.

Claim 1 *Let $[w_1, w_2, w_3]$ be a signature recorded by the algorithm. Then there exists a coloring $\{A_1, A_2, A_3\}$ such that $w(A_i) \leq w_i$ for $i = 1, 2, 3$.*

Proof Let H and L be the sets of vertices the algorithm had when the signature was recorded. Notice that H can be 2-coloring with a signature $[w_1, w_2]$. This is trivially true at the start since H forms an independent set by the assumption that the maximum weight vertices form an independent set. Think of each iteration as updating this coloring for the vertices in the subpath of H containing z , the latest vertex transfer into H . This is precisely why the value of w_2 is updated in Line 10.

This 2-coloring of H is then extended to P using all three colors. The connected components of $P \setminus H$ is a collection of subpaths of P . For each subpath p , color a single vertex with 3 and then extend the coloring of H using $\{1, 2\}$ by alternating colors to get a proper 3-coloring. Clearly, in this new coloring, the weight of

the first two color classes does not change, while the third one cannot exceed $w_3 = \max_{v \in L} w(v)$. \square

Claim 2 *Let $[\alpha, \beta, \gamma]$ be the signature of an arbitrary 3-coloring of P , where $\alpha \geq \beta \geq \gamma$. And let $[w_1, w_2, w_3]$ be the last signature recorded by the algorithm such that $w_2 \leq \beta$. Then $\gamma \geq w_3$ and $\alpha = w_1$.*

Proof The last part, $\alpha = w_1$, follows trivially since in any coloring the weight of the heaviest color class must equal the maximum vertex weight in the path.

For the first part, note that the algorithm must have recorded at least one signature $[w_1, w_2, w_3]$ with $w_2 \leq \beta$ because $\beta \geq \max_{(u,v) \in P} \min\{w(u), w(v)\}$, and w_2 is initialized to this value and only gets larger with time. If $[w_1, w_2, w_3]$ is the very last signature recorded, then $w_3 = 0$, and the statement is trivially true. Otherwise, let $[w'_1, w'_2, w'_3]$ be the signature recorded immediately after $[w_1, w_2, w_3]$; notice that by definition, $w'_2 > \beta \geq w_2$. Let z be the vertex transferred by the algorithm from L to H in going from $[w_1, w_2, w_3]$ to $[w'_1, w'_2, w'_3]$, and let p be the subpath in H that z belonged to after the transfer. Notice that every vertex in p has weight $w(z)$ or more, and $w_3 = w(z)$. If $\gamma < w(z)$, then all vertices in p such be colored using $\{1, 2\}$ in the coloring $[\alpha, \beta, \gamma]$, but then, this implies that $\beta \geq w'_2$, since optimally, 2-coloring p alone requires this. This contradicts the assumption; thus, $\gamma \geq w(z) = w_3$. \square

To finish the proof, let $[\alpha, \beta, \gamma]$ be an optimal max-coloring of the input path. By Lemma 2, the algorithm must have recorded a signature $[w'_1, w'_2, w'_3]$ such that $w'_1 + w'_2 + w'_3 \leq \alpha + \beta + \gamma$. The algorithm return the coloring $\{A_1, A_2, A_3\}$ associated with the best signature $[w_1, w_2, w_3]$; that is, $w_1 + w_2 + w_3 \leq w'_1 + w'_2 + w'_3$. Furthermore, by Lemma 1, $w(A_1) + w(A_2) + w(A_3) \leq w_1 + w_2 + w_3$. Therefore, the coloring returned by the algorithm is optimal. \square

It is natural to wonder whether the sorting time $S(n)$ is really necessary in the running time of MAX-COLORING-PATHS or whether perhaps there is a linear time algorithm. Kavitha and Mestre [22] showed a lowerbound of $\Omega(n \log n)$ time in the algebraic computation tree model; this is a model where the operations $\{+, -, \times, \div, \sqrt{} = 0, \geq 0\}$ on reals can be performed at unit cost.

Theorem 2 ([22]) *The complexity of max-coloring a path with real vertex weights is $\Theta(n \log n)$ in the algebraic computation tree model.*

4 Max-Coloring Trees

The max-coloring problem on trees has turned out to be surprisingly difficult. This section starts with some simple upper bounds on $\chi_{mc}(T)$ for any tree T and then gives a quasi-polynomial time algorithm for max-coloring trees exactly. Finally, it

is shown how this algorithm can become the basis for obtaining a PTAS for the problem.

From [Lemma 2](#) on the property of optimal max-colorings of k -colorable graphs, one can obtain the following corollary for 2-colorable graphs and hence trees.

Corollary 1 *Let G be a 2-colorable graph with $\{C_1, C_2, \dots, C_k\}$ an optimal max-coloring of G . Then, $w(C_i) \geq 2w(C_{i+2})$, for $i = 1, \dots, k-2$, and hence, $w(C_1) \geq 2^{\lfloor \frac{i-1}{2} \rfloor} \cdot w(C_i)$.*

[Corollary 1](#) states that the weights of the color classes decrease geometrically. Hence, one can expect that $\chi_{mc}(T)$ may not be too large for any tree T , and this is captured by the following lemma.

Lemma 6 *Let T be a vertex-weighted tree on n vertices. Then, $\chi_{mc}(T) \leq \lfloor \log_2 n \rfloor + 1$.*

Proof Let $\{C_1, \dots, C_k\}$ be an optimal max-coloring of T with $\chi_{mc}(T)$ colors. For each vertex $v \in C_1$, let $T(v)$ denote the rooted tree with one vertex, namely, v . For each $v \in C_i$, $i > 1$, define $T(v)$ as the tree rooted at v , such that:

1. The children of v in $T(v)$ are exactly the neighbors of v in T belonging to color classes C_1, C_2, \dots, C_{i-1} .
2. For each child u of v , the subtree of $T(v)$ rooted at u is simply $T(u)$.

For each i , $1 \leq i \leq k$, let $S_i = \min\{|T(v)| \mid v \in C_i\}$. In other words, S_i is the size of a smallest tree $T(v)$ rooted at a vertex v in C_i . Recall that each C_i is a maximal independent set in the graph induced by $\cup_{j=i}^k C_j$. This means that every vertex $u \in C_i$ has at least one neighbor in C_j for all $j < i$. Therefore,

$$S_1 = 1, \text{ and}$$

$$S_i \geq \sum_{j=1}^{i-1} S_j + 1, \text{ for each } i > 1.$$

This implies that $S_i \geq 2^{i-1}$ for $1 \leq i \leq k$. Using the fact that $S_k \leq n$, one gets $\chi_{mc}(T) \leq \lfloor \log_2 n \rfloor + 1$. \square

The number of colors required by an optimal max-coloring also depends on the number of distinct weights. Indeed, if all vertex weights are the same, then the optimal max-coloring uses at most two colors. Hence, the optimal max-coloring can be bounded in terms of the number of distinct weights. This relation is stated precisely below.

Lemma 7 *Let T be a vertex-weighted tree on n vertices, and let W be the ratio of the weight of heaviest vertex to the weight of the least heavy vertex. Then, $\chi_{mc}(T) \leq \lceil \log_2 W \rceil + 1$.*

Proof Let $k = \chi_{mc}(T)$, and let C_1, C_2, \dots, C_k be an optimal max-coloring of T ; as usual, assume $w(C_1) \geq w(C_2) \geq \dots \geq w(C_k)$. Notice that $w(C_1)$ equals the weight of the heaviest vertex in the tree. Let ℓ be the smallest integer such that $w(v) \geq w_1/2^\ell$ for all $v \in V$; in other words, $\ell = \lceil \log_2 W \rceil$.

Consider the following collection of disjoint intervals $\mathcal{I} = \{I_0, I_1, \dots, I_{\ell-1}\}$:

$$I_i = \left[\frac{w(C_1)}{2^{i+1}}, \frac{w(C_1)}{2^i} \right) \text{ for } i = 1, \dots, \ell - 1, \text{ and } I_0 = \left[\frac{w(C_1)}{2}, w(C_1) \right].$$

Because of the choice of ℓ , for each vertex $v \in V(T)$, its weight $w(v)$ belongs to exactly one interval I_j . Let $V_j = \{v \in V(T) \mid w(v) \in I_j\}$, $j = 0, 1, \dots, \ell - 1$. Vertex v is said to *contribute* to a color class C_i if $v \in C_i$, and $w(v) = w(C_i)$. The *contribution* of an interval I_j is the maximum number of vertices in V_j that contribute to distinct color classes.

[Corollary 1](#) says that $w(C_i) \geq 2 \cdot w(C_{i+2})$ for $i = 1, \dots, k-2$. Therefore, no interval I_j , $j = 1, 2, \dots, \ell-1$ has a contribution of more than two. If the contribution of I_0 is three or more, then it must be the case that one can construct a 2-coloring with the same or smaller weight, compared to $\{C_1, C_2, \dots, C_k\}$. This contradicts the fact that $k = \chi_{mc}(T)$, and C_1, C_2, \dots, C_k is an optimal max-coloring of T .

Now suppose that intervals $I_{i_1}, I_{i_2}, \dots, I_{i_t}$, $0 \leq i_1 < i_2 < \dots < i_t \leq \ell - 1$ are the sequence of all intervals in \mathcal{I} , each of whose contribution is two. The following claim can be shown.

Claim 3 *For any pair of consecutive intervals I_p , $p = i_j$ and I_q , $q = i_{j+1}$, where $j < t$; it is the case that there is an interval in $\{I_{p+1}, I_{p+2}, \dots, I_{q-1}\}$ with contribution zero.*

Given this claim, one can charge the “extra” contribution of each I_{i_j} to an interval between I_{i_j} and $I_{i_{j+1}}$, whose contribution is zero. This implies that the contributing vertices in all intervals except I_t can be reassigned to a distinct interval. Since there are ℓ intervals and since the contribution of I_t is at most two, there is a total contribution of at most $\ell + 1$, implying that there are at most $\ell + 1$ color classes.

The claim is proved by contradiction, assuming that the contribution of every interval in $\{I_{p+1}, I_{p+2}, \dots, I_{q-1}\}$ is one. Let $\{x_p, x_{p+1}, \dots, x_q\} \cup \{y_p, y_q\}$ be vertices such that:

- (1) For each $j = p, p+1, \dots, q$, $x_j \in V_j$ and x_j contribute to some color class.
- (2) For each $j \in \{p, q\}$, $y_j \in V_j$ and x_j and y_j contribute to distinct color classes.

Since $x_j \in V_j$, $w(x_j) \geq \frac{w(C_1)}{2^{j+1}}$, $j = p, p+1, \dots, q$. Also, since $y_q \in V_q$, $w(y_q) \geq \frac{w(C_1)}{2^{q+1}}$. Therefore,

$$\begin{aligned}
\sum_{j=p}^q w(x_j) + w(y_q) &\geq \sum_{j=p}^q \frac{w(C_1)}{2^{j+1}} + \frac{w(C_1)}{2^{q+1}} \\
&= w(C_1) \frac{2^{q-p+1} - 1}{2^{q+1}} + \frac{w(C_1)}{2^{q+1}} \\
&= \frac{w(C_1)}{2^p} > w(y_p)
\end{aligned}$$

This contradicts [Lemma 2](#) and proves the claim. \square

The upperbounds in the preceding lemmas are all tight, as the following example shows. Let T_0, T_1, \dots be a sequence of trees, where T_0 is a single vertex, with weight 1, and $T_i, i > 0$ is constructed from T_{i-1} as follows: Let $V(T_{i-1}) = \{u_1, u_2, \dots, u_k\}$. To construct T_i , start with T_{i-1} and add a set of new vertices $\{v_1, v_2, \dots, v_k\}$, each with weight 2^i , and edges $\{u_i, v_i\}$ for all $i = 1, 2, \dots, k$. Thus, the leaves of T_i are $\{v_1, v_2, \dots, v_k\}$, and every other vertex in T_i has a neighbor v_j for some j . Now consider a tree T_n in this sequence. Clearly, $|V(T_n)| = 2^n$ and the maximum vertex degree of T_n , $\Delta(T_n) = n-1$. See [Fig. 1](#) for T_0, T_1, T_2 , and T_3 . Now consider the coloring of T_n defined by $C_i = \{\text{leaves of } T_{n+1-i}\}$, $1 \leq i \leq n+1$. Note that $w(C_i) = 2^{n+1-i}$ and therefore the weight of the coloring is $2^{n+1} - 1$. It is not hard to see that any coloring using fewer than $n+1$ colors has weight at least 2^{n+1} . Therefore,

$$\chi_{mc}(T) = n+1 = \log_2 |V(T_n)| + 1 = \Delta + 1 = \log_2 \left(\frac{2^n}{1} \right) + 1.$$

Everything is in place to describe the algorithm to find an optimal max-coloring. From [Lemma 6](#), it follows that $\chi_{mc}(T) \leq \log_2 n + 1$. Hence, there are at most $n^{O(\log n)}$ candidate weight sequences corresponding to feasible max-colorings. A candidate weight sequence consists of a sequence (W_1, \dots, W_k) of weights, with $k \leq \lceil \log_2 n \rceil$, $W_1 \geq W_2 \geq \dots \geq W_k$, and each W_i is the weight of some vertex in T . For a given candidate weight sequence (W_1, \dots, W_k) , the algorithm needs to decide if there is a coloring of T that respects this weight sequence. A coloring of T respects the given weight sequence if the coloring is proper and uses at most k colors C_1, \dots, C_k , and $\max_{v \in C_i} w(v) \leq W_i$ for each $i = 1, \dots, k$. The *feasible- k -coloring* problem asks to find, given a weighted tree T and a sequence (W_1, W_2, \dots, W_k) , a coloring satisfying the sequence (W_1, \dots, W_k) , or if such a coloring does not exist, to report so.

A simple dynamic programming algorithm solves feasible- k -coloring on trees in $O(nk)$ time. Let T be rooted at an arbitrary vertex r . Let $\text{children}(v)$ denote the set of children of v , and let $\text{parent}(v)$ denote the parent of v . For a vertex v , let $T(v)$ denote the subtree of T rooted at v . Let $[k]$ denote the set of colors $\{1, \dots, k\}$. For a vertex v , let $F(v) = \{j \mid w(v) > W_j\}$ be the set of forbidden colors, and let

$$S(v) = \{i : \text{there exists a feasible coloring of } T(v) \text{ with } \chi(v) = i\}. \quad (2)$$

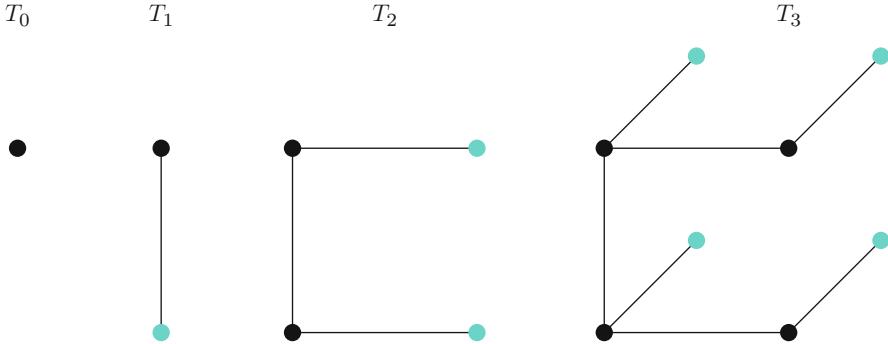


Fig. 1 Sequence of trees which show that the upperbounds of Lemmas 1, 6 and 7 are all tight. T_i is obtained by adding the *light-colored* vertices to T_{i-1} . The figure above shows the trees T_0, \dots, T_3

Algorithm 3 computes a feasible coloring, if one exists; we call this algorithm FKC. The correctness of the algorithm and the running time are easily established and are described in the lemma below.

Lemma 8 *Algorithm FKC solves the feasible- k -coloring problem in $O(nk)$ time.*

Proof Assume that the sets $S(v)$ computed by the algorithm FKC follow Eq.(2). Given this, it is trivial to see that the algorithm indeed solves the feasible- k -coloring problem. If there exists some vertex v such that $S(v) = \emptyset$, then $S(r) = \emptyset$, and clearly, there is no feasible coloring. Otherwise, for each $v \neq r$, one must argue that Line 14 can assign a valid color to v . Indeed, if $|S(v)| > 1$, this can always be done, and if $|S(v)| = 1$, then the single color present in $S(v)$ does not belong to $S(\text{parent}(v))$. Thus, Line 14 can always be carried out.

Algorithm 3 $\text{FKC}(T, w, W)$

1. Let $F(v) = \{j : w(v) > W_j\}, \forall v \in T$
 2. **for** each vertex v in a post-order traversal of T **do**
 3. Let $S(v) = [k] - F(v)$
 4. **for** each $u \in \text{children}(v)$ **do**
 5. **if** $S(u) = \emptyset$ **then**
 6. Set $S(v) = \emptyset$
 7. **if** $|S(u)| = 1$ **then**
 8. Set $S(v) = S(v) - S(u)$
 9. **if** $S(r) = \emptyset$ **then**
 10. **return** no feasible color exists
 11. **else**
 12. Pick an arbitrary $i \in S(r)$ and set $\chi(r) = i$
 13. **for** each $v \neq r$ in a pre-order traversal of T **do**
 14. Pick an arbitrary $j \in S(v) - \chi(\text{parent}(v))$
 15. Set $\chi(v) = j$
 16. **return** χ
-

Consider the following inductive argument on the height of $T(v)$ to show that the sets $S(v)$ obey (2): If the height of $T(v)$ is 0, then v is a leaf, and so, $S(v) = [k] - F(v)$, which is precisely what the algorithm returns.

Assume the claim is true for all subtrees of height $< m$. Let v be a vertex such that $T(v)$'s height is m . Suppose that v has a child u such that $S(u) = \emptyset$. By inductive hypothesis, $T(u)$ does not admit a feasible coloring and so neither does $T(v)$. Hence, the assignment $S(v) = \emptyset$ in Line 6 is correct. Now suppose that v has a child u such that $S(u) = \{c\}$. By inductive hypothesis, $S(u)$ obeys (2), and so c is the only color that can be assigned to u in any feasible coloring of $T(u)$. Hence, the c cannot be given to v , namely, $c \notin S(v)$, and so Line 8 correctly removes c from $S(v)$. For each color that remain in $S(v)$, v can be assign color and then extend the coloring to the remaining vertices in $T(v)$ by following for loop of Line 13. This finishes the proof that the $S(v)$ sets obey Eq. (2). \square

Theorem 3 *There is quasi-polynomial time algorithm for max-coloring trees that runs in $n^{O(\log n)}$ time.*

Proof Since $\chi_{mc}(T)$ is at most $\log_2 n + 1$ for any tree T , the algorithm only has to consider weight sequences (W_1, W_2, \dots, W_k) with $k \leq \log n + 1$. There are $n^{\log n + 2}$ possibilities. Each sequence can be checked for feasibility with FKC in $O(n \log n)$ time. An optimal max-coloring is simply a feasible sequence minimizing $W_1 + W_2 + \dots + W_k$. \square

The main idea underlying the PTAS is the reduction of the number of distinct weights of the vertices down to a constant. Candidates for the weights of the color classes are chosen; then for each such choice, using the algorithm for feasible- k -coloring, it can be tested if there is a legal coloring of the tree with the chosen weights for the color classes.

Let T be the input graph. Using Lemma 3, one can, with a loss of $1 + \epsilon$ in the objective, reduce the general problem to a number of smaller instances T_0, T_1, \dots such that the number of distinct to the special problem where the number of weights used in each T_i is at most $O(1/\epsilon^2)$. Together with the FKC algorithm, this means that a $(1 + \epsilon)$ approximate solution can be found in $O(n^{O(\epsilon^{-2})})$ time. Below is an alternative proof that improves the dependency on $1/\epsilon$ in the exponent.

Theorem 4 *There is a PTAS for max-coloring trees.*

Proof Let T be a tree, with weight function $w : V \rightarrow \mathbb{N}$ and an $\epsilon > 0$. Let $c > 0$ be an integer such that $(2 \log c + 3)/c \leq \epsilon$, and let $\alpha = (w_1 - 1)/c$. Let I_1, I_2, \dots, I_c be a partition of the range $[1, w_1]$, where $I_i = [1 + (i - 1)\alpha, 1 + i \cdot \alpha)$, $1 \leq i \leq c$. Let T' be a tree that is identical to T , except in its vertex weights. The tree T' has vertex weights $w' : V \rightarrow \mathbb{N}$ defined by the rule for any $v \in V$, if $w(v) \in I_j$, then $w'(v) = 1 + (j - 1) \cdot \alpha$, and if $w(v) = w_1$, then $w'(v) = w_1$. In other words, except for vertices with maximum weight w_1 , all other vertices have their weights “rounded” down. As a result, T' has $c + 1$ distinct vertex weights. Now let OPT'_{mc}

denote the weight of an optimal max-coloring of T' and let $\mathcal{C}' = C'_1, C'_2, \dots, C'_k$ be the color classes corresponding to OPT'_{mc} . Since the weights of vertices have fallen in going from T to T' , clearly, $\text{OPT}'_{mc} \leq \text{OPT}_{mc}$. The coloring \mathcal{C}' for T has weight, is at most $\text{OPT}'_{mc} + k\alpha$. Substituting $(w_1 - 1)/c$ for α and noting that $w_1 \leq \text{OPT}'_{mc}$, one can see that weight of \mathcal{C}' used as a coloring for T' is at most $(1 + \frac{k}{c})\text{OPT}'_{mc}$. It was shown that given the distribution of vertex weights of T' , $k = O(\log c)$. To see this, first observe that the weights of last three color classes C'_k, C'_{k-1} , and C'_{k-2} cannot all be identical, by Lemma 2. Also, observe that the possible vertex weights of T' are $1, 1 + \alpha, 1 + 2\alpha, \dots$. Therefore, $w(C'_{k-2}) \geq 1 + \alpha$. From Corollary 1,

$$1 + \alpha \leq w(C_{k-2}) \leq \frac{w_1}{2^{\lfloor (k-3)/2 \rfloor}}.$$

Solving this for k yields $k \leq 2 \log_2(c) + 3$. Therefore, by the choice of c ,

$$\frac{k}{c} \leq \frac{2 \log_2(c) + 3}{c} \leq \epsilon.$$

Thus, $(1 + \epsilon)\text{OPT}'_{mc}$ is an upperbound on the weight of \mathcal{C}' used as a coloring for T . Since $\text{OPT}'_{mc} \leq \text{OPT}_{mc}$, it follows that the weight of OPT'_{mc} used as a coloring for T is at most $(1 + \epsilon)\text{OPT}_{mc}$.

To construct OPT'_{mc} in polynomial time, for each $k = 1, \dots, 2 \log c + 3$, generate all $O(c^k)$ possible sequences of weights and call algorithm FKC for each subsequence and pick the coloring with the minimum weight. This gives OPT'_{mc} . Each call to FKC takes $O(nk)$ time, and we have $O(c^k)$ sequences, for $k = 1, \dots, 2 \log c + 3$. Using the fact that $(2 \log_2 c + 3)/c \leq \epsilon$, a little bit of algebra yields a running time that is linear in n and exponential in $1/\epsilon$. \square

The same algorithm can be extended to coloring graphs of bounded treewidth.

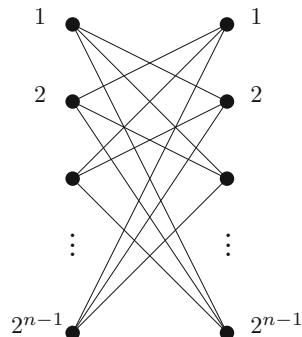
Theorem 5 ([14]) *There is a PTAS for max-coloring graphs of bounded treewidth, provided a tree-decomposition is given.*

5 Max-Coloring Bipartite and k -Colorable Graphs

The max-coloring problem on bipartite graphs, unlike the case with trees, is fully resolved. There is a $\frac{8}{7}$ -approximation algorithm and is inapproximable beyond $\frac{8}{7} - \epsilon$ for any $\epsilon > 0$ unless P = NP. Further, the approach used to max-color bipartite graphs can be extended to an approximation algorithm for coloring k -colorable graphs for which a k -coloring can be obtained in polynomial time.

Figure 2 shows that an optimal max-coloring may use as many as $O(n)$ colors. The example is a graph $G = (U \cup V, E)$, where $|U| = |V| = n$. The vertices in U

Fig. 2 An instance of max-coloring of a bipartite graph on $2n$ vertices that requires n colors in an optimal max-coloring. The weights of the vertices are powers of 2 and are shown next to the vertices. A k -coloring for any $k < n$ has weight at least 2^n , while a coloring with n colors has weight $2^n - 1$



are labeled $1, \dots, n$. and the vertices in V are labeled $1, \dots, n$. Vertex i has weight 2^{i-1} . Each vertex $i \in U$ is adjacent to all vertices in V except the vertex labeled i . A 2-coloring of G with all vertices in U colored 1 and all vertices in V colored 2 has a weight of $2^{n-1} + 2^{n-1} = 2^n$, while an optimal coloring consists of assigning color i to the vertices labeled i in both U and V , which has a weight of $2^n - 1$.

The PTAS for the max-coloring problem on trees relied on the fact that the feasible- k -coloring problem on trees can be solved in polynomial time for any k . However, feasible- k -coloring is NP-complete for bipartite graphs for $k \geq 3$ as shown by Kratochvil et al. [25]. Hence, a different approach for k -colorable graphs is needed. Consider the following algorithm that uses at most $k + 2$ colors for max-coloring a k -colorable graph.

Assume that a polynomial time algorithm is given to compute an arbitrary proper k -coloring of the input graph. The algorithm exploits the fact that an optimal max-2-coloring of a bipartite graph can be found efficiently. The basic idea is to optimally 2-color the i heaviest vertices in the graph (if possible) and then use the k -coloring algorithm to color the rest. We repeat this for each $i = 0, \dots, n$, and return the best coloring found. Call this algorithm MAX-COLORING-K-COLORABLE. Its pseudo-code is given in Algorithm 4.

Theorem 6 *Provided with a procedure for finding a proper k -coloring of the input graph, algorithm MAX-COLORING-K-COLORABLE is a $\frac{k^3}{3k^2-3k+1}$ -approximation for max-coloring.*

Proof First consider the issue of finding an optimal 2-coloring of $G[V_i]$. Notice that the graph to be colored is a collection of connected components. Each component is connected bipartite graph, which admits only two proper 2-colorings. Color each component so that the heaviest color class is 1; the resulting coloring of $G[V_i]$ is an optimal max-2-coloring. Thus, provided a k -coloring of the input graph can be computed, the algorithm MAX-COLORING-K-COLORABLE runs in polynomial time.

Algorithm 4 MAX-COLORING-K-COLORABLE(G, w)

-
1. Sort the vertices in non-increasing weight order v_1, v_2, \dots, v_n
 2. **for** $i \in \{0, n\}$ **do**
 3. $V_i \leftarrow \{v_1, \dots, v_i\}$
 4. **if** $G[V_i]$ is bipartite **then**
 5. optimally 2-color $G[V_i]$
 6. arbitrarily k -color $G[V \setminus V_i]$ using k new colors
 7. let χ_i be the resulting coloring
 8. **return** best coloring among the χ_i for $i = 1, \dots, n$
-

Second, consider the quality of the solution returned. Three different upper bounds on the value of the solution returned by the algorithm are derived. Later, it will be argued that the best of these bounds attains the desired approximation ratio of $\frac{k^3}{3k^2 - 3k + 1}$. The value of the solution found is denoted by MCKC.

Assume the vertices are sorted in non-increasing weight order v_1, v_2, \dots, v_n . Let $\{C_1, \dots, C_{k^*}\}$ be an optimal max-coloring for G . Let j_i be the smallest index such $v_{j_i} \in C_i$ for each $i = 1, \dots, n$.

Notice that when $i = 0$, the graph $G[V_i]$ is empty, so there is no cost incurred by the 2-coloring step. Hence,

$$\text{MCKC} \leq kw(C_1). \quad (3)$$

Let i be the largest index such that $w(v_i) > w(v_{j_2})$. Notice that the set V_i is independent in G . Therefore, the graph $G[V_i]$ can be colored with a single color class at the cost of $w(C_1)$. The rest of the graph $G[V \setminus V_i]$ can be colored with additional k colors at a cost of at most $kw(C_2)$. Hence,

$$\text{MCKC} \leq w(C_1) + kw(C_2). \quad (4)$$

Finally, let i be the largest index such that $w(v_i) > w(v_{j_3})$. In this case, the graph $G[V_i]$ is bipartite, and the coloring $\{C_1 \cap V_i, C_2 \cap V_i\}$ has a cost of at most $w(C_1) + w(C_2)$. The rest of the graph $G[V \setminus V_i]$ can be colored with additional k colors at a cost of at most $kw(C_3)$. Hence,

$$\text{MCKC} \leq w(C_1) + w(C_2) + kw(C_3). \quad (5)$$

Now combine these three bounds using the right multipliers: Take $\frac{(k-1)^2}{k^3}$ times (3), plus $\frac{k(k-1)}{k^3}$ times (4), plus $\frac{1}{k}$ times (5) to obtain the following equation:

$$\text{MCKC} \cdot \frac{3k^2 - 3k + 1}{k^3} \leq w(C_1) + w(C_2) + w(C_3).$$

Recall that $\{C_1, \dots, C_{k^*}\}$ was an optimal max-coloring. Thus, it was shown that MAX-COLORING-K-COLORABLE is a $\frac{k^3}{3k^2 - 3k + 1}$ approximation. \square

For bipartite graphs, the algorithm yields an $\frac{8}{7}$ -approximation that uses at most four colors. One might be tempted now to continue this and try to come up with an algorithm that uses five or more colors and obtains a better approximation ratio for bipartite graphs. Surprisingly, no such improvement is possible in polynomial time as there is a *gap introducing* reduction from the problem of *pre-coloring extension* on graphs which was shown to be NP-complete by [25].

Theorem 7 ([9], [30]) *For any $\epsilon > 0$, there is no $(8/7 - \epsilon)$ -approximation algorithm for max-coloring on bipartite graphs, unless $P = NP$.*

5.1 Planar Graphs

The four-color theorem states that every planar graph can be colored at most four colors. Robertson et al. [35] provide an $O(n^2)$ time algorithm for 4-coloring any planar graph. Using Theorem 6, one gets a $\frac{64}{37} < 1.73$ approximation for max-coloring planar graphs. On the negative side, the hardness of graph coloring planar graphs [16] means it is NP-hard to approximate max-coloring better than $4/3 > 1.33$.

Theorem 8 *Max-coloring in planar graphs can be approximated within $\frac{64}{37}$ but not within $\frac{4}{3} - \epsilon$ for any $\epsilon > 0$ unless $P = NP$.*

Grötzsch theorem states that every triangle-free planar graph can be 3-colored. Dvorak et al. [10] give a linear time algorithm to compute such a coloring. Using Theorem 6, one gets a $\frac{27}{19} < 1.43$ approximation for max-coloring triangle-free planar graphs. On the negative side, De Werra et al. [7] showed that in this class of graphs, max-coloring cannot be approximated better than $\frac{7}{6} > 1.16$.

Theorem 9 *Max-coloring in triangle-free planar graphs can be approximated within $\frac{22}{19}$ but not within $\frac{7}{6} - \epsilon$ for any $\epsilon > 0$ unless $P = NP$.*

5.2 Restricted Bipartite Graphs

DeWerra et al. [7] have attempted to further delineate the boundary between polynomial time solvable and NP-hard cases of the max-coloring problem. The authors consider bipartite graphs that do not contain an induced path on l vertices.

Theorem 10 ([7]) *Max-coloring on P_8 -free bipartite graphs is NP-complete.*

Further, they show that for P_5 -free bipartite graphs, the problem can be solved in polynomial time. There are several characterizations of P_5 -free bipartite graphs (See [7] and references therein). In particular, a bipartite graph G is P_5 -free if and

only if each connected component of G is $2K_2$ -free, that is, it does not contain edges whose vertices are nonadjacent. The authors then use this characterization to prove that any optimal max-coloring of a P_5 -free bipartite graph uses at most three colors.

Theorem 11 ([7]) *Max-coloring can be solved in polynomial time on P_5 -free bipartite graphs.*

Further, the authors observe that P_5 -free bipartite graphs have clique-width at most 5 and leave as an open question the problem of max-coloring on graphs of bounded clique-width.

5.3 Split Graphs

A graph class similar to bipartite graphs is that of split graphs. A graph $G = (K \cup S, E)$ is a split graph if its vertex set can be partitioned into two sets K and S such that K is a complete graph on n_1 vertices, and S is an independent set on n_2 vertices. The edge set of G includes an arbitrary bipartite graph between the vertices of K and S . Split graphs are a subclass of perfect graphs, and hence, classic coloring is easy. It is also easy to see that $n_1 \leq \chi(G) \leq n_1 + 1$ for any split graph. The max-coloring problem on split graphs, however, is NP-hard even with just two distinct weights as shown by Demange et al. [8].

However, a PTAS can easily be obtained as follows: Consider an optimal max-coloring of a split graph G , C_1, \dots, C_k where $k \in \{n_1, n_1 + 1\}$. There is some $i_0 \leq k + 1$ such that all color classes with index $i < i_0$ have at least one vertex from K , and the remaining color class i_0 has all the remaining vertices of S . For a fixed $\epsilon > 0$, trying all bijections between K and the first $\lceil \frac{1}{\epsilon} \rceil$ color classes, one obtains a solution which is at least as good as the first $\lceil \frac{1}{\epsilon} \rceil$ color classes.

Theorem 12 ([7]) *Max-Coloring on Split Graphs admits a PTAS.*

6 Max-Coloring Interval Graphs

An interval graph is an intersection graph of intervals on the real line. Recall that given an interval graph, an interval representation of it can be produced in polynomial time. Hence, assume that the graph is given as a set of intervals on the x -axis. Each interval has a weight w . If two intervals overlap, their corresponding vertices are adjacent. Interval graphs arise in several applications including resource allocation and scheduling and hence have been the subject of intensive study over several years. In particular, the online coloring problem on interval graphs is well studied, and it is known that the first-fit coloring algorithm is at most 8-competitive ([27, 31], G. Brightwell, H.A. Kierstead and W.T. Trotter, 2003, First-fit coloring of interval graphs, Personal communication). Applying Lemma 4 from

Algorithm 5 BETTER-MCA(G, w)

-
1. Sort the vertices of G in non-increasing order of weights
// Let (v_1, \dots, v_n) be this ordering of the vertices of G
 2. Let $k \leftarrow 1; S_k \leftarrow \emptyset$
 3. **for** $j \leftarrow 1$ **to** n **do**
 4. Insert v_j into set S_i
 5. where $i \leq k$ is the smallest value such that
 6. $S_1 \cup S_2 \cup \dots \cup (S_i \cup \{u\})$ does not contain an $(i + 1)$ -clique
 7. If no such i exists,
 8. set $k \leftarrow k + 1$ and insert v_j into S_k
 9. Use color 1 to color vertices in S_1
 10. **for** $i \leftarrow 2$ **to** k **do**
 11. Use colors $2i - 2$ and $2i - 1$ to 2-color the vertices in S_i
 12. **return** coloring
-

[Sect. 2](#) immediately gives an approximation algorithm for max-coloring with an approximation factor of 8, after noting that interval graphs form a hereditary graph class.

The best-known algorithm for online coloring interval graphs is a slight modification of the first-fit algorithm. This algorithm by Kierstead and Trotter [24] has a competitive ratio of 3, and now with [Lemma 4](#), one immediately gets a 3-approximation algorithm for max-coloring. What follows is a description of the Kierstead-Trotter algorithm, which is the basis for the 2-approximation algorithm.

The Kierstead-Trotter algorithm maintains sets S_1, S_2, \dots such that when a vertex u is presented, it finds the smallest i such that $S_1 \cup S_2 \cup \dots \cup (S_i \cup \{u\})$ does not contain an $(i + 1)$ -clique. (Recall that a maximum clique can be found in polynomial time on interval graphs.) The vertex u is then inserted into S_i . The fact that the Kierstead-Trotter algorithm is 3-competitive is a consequence of the following observations:

1. Let k be the largest index of a nonempty set S_i . Then, an interval I was placed in S_k since it induced a k -clique with intervals presented before. Hence, $k \leq \omega(G)$.
2. Let I_1, I_2, I_3 be a triple of intervals in S_i that pairwise intersect. Without loss of generality, assume that $I_3 \subseteq I_1 \cup I_2$. Then, it follows that I_3 induces an $i + 1$ -clique with intervals presented before it and hence cannot have been placed in S_i . Hence, each S_i is a disjoint union of paths.

Let k be the largest index of a nonempty set S_i . Then, $k \leq \omega(G) = \chi(G)$. Since each S_i is a disjoint union of paths, each S_i can be online 3-colored. Hence, the total number of colors used is at most $3 \cdot k \leq 3 \cdot \chi(G)$. However, since the max-coloring problem is an off-line problem, color each set S_i off-line using at most two colors. The algorithm is described in detail in [Algorithm 5](#).

Theorem 13 BETTER-MCA is a 2-approximation algorithm for the max-coloring problem on interval graphs.

Proof Let C_1, C_2, \dots, C_k be a coloring of G that is optimal for the max-coloring problem. Let $w_i = \max_{v \in C_i} w(v)$, and without loss of generality, assume that $w_1 \geq w_2 \geq \dots \geq w_k$. Now note that $k \geq \chi(G)$ and $\text{OPT}_{mc}(G) = \sum_{i=1}^k w_i$.

Suppose that at the end of Step (6) in BETTER-MCA, we have sets S_1, S_2, \dots, S_t . An element is inserted into S_t only because $S_1 \cup S_2 \cup \dots \cup (S_{t-1} \cup \{u\})$ has a t -clique. Therefore, $\chi(G) \geq t$, and it follows that $t \leq k$. Let $s_i = \max_{v \in S_i} w(v)$. The claim is that for each i , $1 \leq i \leq t$, $s_i \leq w_i$.

It is clear that $s_1 = w_1$. To obtain a contradiction, suppose that for some i , $s_i > w_i$, and let i be the smallest such value. This implies any vertex x with weight s_i or larger is in an earlier color class, C_j , for some $j < i$ in the optimal coloring. Therefore, vertices with weight s_i or larger are $(i-1)$ -colorable, so they induce a clique of size at most $i-1$, and therefore, in Step (6), no vertex with weight s_i will get inserted into S_i – a contradiction.

In Steps (9) and (10), the vertex partition S_1, S_2, \dots, S_t is converted into a coloring whose weight is at most $s_1 + 2 \cdot \sum_{i=2}^t s_i$. The following inequalities

$$s_1 + 2 \cdot \sum_{i=2}^t s_i \leq w_1 + 2 \cdot \sum_{i=2}^t w_i \leq 2 \cdot \text{OPT}_{mc}(G)$$

give the result sought. \square

Pemmaraju et al. [31] showed via a reduction from E4-set splitting that max-coloring on interval graphs is NP-hard. A simpler proof of NP-hardness was discovered by Escoffier et al. [14] and independently by Pemmaraju et al. [32] via a reduction from coloring circular-arc graphs.

Theorem 14 ([14, 32]) *Max-Coloring interval graphs is NP-hard.*

The results on NP-hardness of max-coloring interval graphs left open the question of whether there exists a PTAS for the problem. This has been answered in the affirmative recently by Nonner [28].

7 Max-Coloring on Hereditary Graphs

Recall that a class of graphs closed under taking vertex-induced subgraphs is called hereditary. Let \mathcal{G} be a hereditary class of graphs for which the classic vertex coloring admits a c -approximation algorithm. In this section, we describe approximation algorithms for the max-coloring problem on this class of graphs. The algorithm is quite natural and works by partitioning the vertices of the input graph G into *weight classes*, where each weight class consists of vertices of nearly equal weight, and solving the classic coloring problem in the subgraph induced by each weight class separately using a fresh palette for each. The section first describes how this approach leads to a 4-approximation algorithm for the max-coloring problem and

later shows how the approximation factor can be improved to e , the base of the natural logarithm.

Let $G = (V, E)$ be the input graph with weight function w . Let w_{\min} and w_{\max} denote the minimum and maximum weight vertices, respectively, and assume $w_{\min} = 1$ without loss of generality. The algorithm partitions the vertices of G into weight classes, $R_0, R_1, \dots, R_{\lceil \log_2 w_{\max} \rceil}$, where $R_i = \{v : w(v) \in [2^i, 2^{i+1})\}$, and applies the c -approximate coloring algorithm for each weight class ignoring the weights. This algorithm is called WP, for weighted partition.

Theorem 15 *Let \mathcal{G} be a hereditary class of graphs which admit a c -approximation algorithm for coloring. Then WP is a deterministic $4c$ -approximation algorithm for max-coloring for the class \mathcal{G} of graphs.*

Proof Let C_1, \dots, C_k be the color classes of OPT_{mc} . Consider a fixed color class C_i , and let $j_{\max} = \max\{j : R_j \cap C \neq \emptyset\}$. Partition C into at most $j_{\max} + 1$ classes $D_j^i = C \cap R_j$, $j = 0, \dots, j_{\max}$. Then, since this partition colors vertices in each R_i with the same color,

$$\begin{aligned} \text{WP} &\leq \sum_{i=1}^k \sum_{j=0}^{j_{\max}} c \cdot w(D_j^i) \leq c \cdot \sum_{i=1}^k \sum_{j=0}^{j_{\max}} 2^{j+1} \\ &\leq c \cdot \sum_{i=1}^k 2^{j_{\max}+2} \leq 2c \cdot \sum_{i=1}^k 2^{j_{\max}+1} \\ &\leq 4c \cdot \sum_{i=1}^k w(C_i) = 4c \cdot \text{OPT}_{mc} \end{aligned} \quad \square$$

To improve the approximation factor, the algorithm has to randomize the partitioning of the vertices into weight classes. Let q be a constant to be chosen later, and let α be chosen uniformly at random from $[0, 1]$. We now redefine the weight classes as

$$R_i = \{v : w(v) \in [q^{i+\alpha}, q^{i+1+\alpha})\}$$

Using this new partitioning and applying the c -approximation algorithm within each weight class, one obtains an ec -approximation algorithm. The algorithm, called RWP, is described in [Algorithm 6](#).

Theorem 16 *Let \mathcal{G} be a hereditary class of graphs which admit a c -approximation algorithm for coloring. Then RWP is an ec -approximation algorithm for max-coloring for the class \mathcal{G} of graphs.*

Proof Let C_1, \dots, C_k be the color classes of OPT_{mc} . For color class C_i , let $w(C_i) = q^{x_i}$. $R_i = \{i : w(v) \in [q^{i+\alpha}, q^{i+1+\alpha})\}$, for $i = 0, 1, \dots, \lceil \log_q w_{\max} \rceil$. Let $j_{\max} = \max\{j : R_j \cap C_i \neq \emptyset\}$. Then, partitioning C_i into at most $j_{\max} + 1$

Algorithm 6 RWP(G, w)

1. Choose α uniformly at random from $[0, 1)$
 2. **for** $i = 0, 1, \dots$ **do**
 3. Let $R_i = \{v \mid q^{i+\alpha} < w(v) \leq q^{i+1+\alpha}\}$
 4. Color each R_i using the c -approximation algorithm
 5. **return** Union of the individual colorings
-

color classes, D_j , where $w(v) \in R_j$, for all $v \in D_j$, there is a coloring such that each color class contains vertices from only one set R_i . However, since RWP colors each set R_i using a c -approximation algorithm,

$$\begin{aligned} \mathbb{E}[\text{RWP}] &\leq \mathbb{E}\left[c \cdot \sum_{i=1}^k \sum_{j=0}^{j_{\max}} w(D_j^i)\right] \leq c \sum_{i=1}^k \mathbb{E}\left[\sum_{j=0}^{j_{\max}} q^{j+1+\alpha}\right], \\ &\leq c \cdot \sum_{i=1}^k \mathbb{E}\left[\frac{q^{j_{\max}+1+\alpha+1}}{q-1}\right] = c \cdot \sum_{i=1}^k \frac{q^{x+1}}{q-1} \int_0^1 q^{1+\alpha-x} d\alpha, \\ &= c \cdot \sum_{i=1}^k \frac{q^{x+1}}{q-1} \frac{q-1}{\ln q} = \frac{qc}{\ln q} \sum_{i=1}^k w(C_i) \leq \frac{qc}{\ln q} \cdot \text{OPT}_{mc}. \end{aligned}$$

Since $\frac{q}{\ln q}$ is minimized at $q = e$, the base of the natural logarithm, the algorithm is an ec -approximation for max-coloring. \square

It is worth noting that RWP can be de-randomized by running the algorithm on some critical set of α values and then keeping the best solution produced. For each vertex v , there is a unique value α_v at which the vertex changes from R_i to R_{i+1} for some i . For values of α in between two consecutive critical values, the algorithm behaves the same way, so the minimum cost solution produced at these critical value is no more than the average case.

7.1 The Online Case

Even though the proof of [Theorem 15](#) provided above focused on the off-line setting, the same algorithms and the same arguments can be used to prove the following theorem about the online setting. The only difference is that one cannot de-randomize the second algorithm, so in this case, there is a gap between deterministic and randomized performance

Theorem 17 *Let \mathcal{G} be a hereditary class of graphs which admits a c -competitive algorithm for online coloring. Then WP is a deterministic $4c$ -competitive algorithm*

and RWP is a randomized ec -competitive algorithm for online max-coloring graphs in the class \mathcal{G} .

Two applications to max-coloring interval graphs are discussed to exemplify the usefulness of the above theorem. Assume that the interval representation, rather than the graph itself, is revealed. Two interval arrival models are considered: In the *list model*, intervals arrive in some arbitrary, while in the *time model*, intervals arrive sorted by their left endpoint.

First fit is a simple heuristic that assigns the lowest color available to an incoming interval. It is well known that first fit is optimal in the time model, namely, first fit is a deterministic 1-competitive algorithm in the time model. Plugging this result into [Theorem 17](#), one gets the following:

Corollary 2 *There is a 4-competitive deterministic algorithm and a e -competitive randomized algorithm for max-coloring interval graphs in the time model.*

In the list model, a different algorithm by Kierstead and Trotter [24] is 3-competitive for online coloring interval graphs. Plugging this result into [Theorem 17](#), one gets the following:

Corollary 3 *There is a 12-competitive deterministic algorithm and a $3e$ -competitive randomized algorithm for max-coloring interval graphs in the list model.*

Finally, it should be noted that better algorithms are known for restricted classes of interval graphs. Unit interval graphs are a special class of interval graph where every interval has unit length. Epstein and Levy [13] designed a 2-competitive algorithm for this special class. Thus, one gets the following result:

Corollary 4 *There is an 8-competitive deterministic algorithm and a $2e$ -competitive randomized algorithm for max-coloring unit interval graphs in the list model.*

8 Max-Edge-Coloring

This section first describes a simple approximation algorithm for max-edge-coloring a graph G . Later, it is shown how the approximation ratio can be improved slightly.

The algorithm processes the edges in non-increasing weight order. When an edge is considered, it assigns the lowest color possible so as not to create a conflict with adjacent previously colored edges. This algorithm is called GREEDY-MAX-EDGE-COLORING. Its pseudo-code is given in [Algorithm 7](#).

Every edge-coloring χ is just a collection of matchings $\{M_1, M_2, \dots, M_k\}$. As usual, these matchings are listed in non-increasing weight order; that is,

Algorithm 7 GREEDY-MAX-EDGE-COLORING(G, w)

-
1. $A(u) \leftarrow \{1, 2, \dots, 2\Delta - 1\}$, for all $u \in V$
 2. sort edge in non-increasing weight
 3. **for** $(u, v) \in E$ in sorted order **do**
 4. $\chi(u, v) = \min A(u) \cap A(v)$
 5. remove $\chi(u, v)$ from $A(u)$ and from $A(v)$
 6. **return** χ
-

$w(M_1) \geq w(M_2) \geq \dots \geq w(M_k)$. In the proofs, we will say that a coloring $\{M_1, \dots, M_k\}$ is *nice* if for all $(u, v) \in M_i$ there is no $j < i$ such that both u and v are unmatched in M_j .

For a given input instance (G, w) , denote by $\{C_1, \dots, C_{k^*}\}$ an optimal max-coloring of G ; assume $w(C_1) \geq \dots \geq w(C_{k^*})$. In addition, OPT_{mc} is used to denote the cost of this optimal solution, that is, $\text{OPT}_{mc} = \sum_{i=1}^{k^*} w(C_i)$.

Theorem 18 *The algorithm GREEDY-MAX-EDGE-COLORING is a $\min \left\{ 2 - \frac{1}{\Delta}, 2 - \frac{w_{\max}}{\text{OPT}_{mc}} \right\}$ approximation for general multigraphs, where $w_{\max} = \max_e w(e)$.*

Proof Start by showing that the algorithm indeed computes a coloring. The only subtlety that needs to be justified is that Line 4 indeed assigns a color to (u, v) . At the beginning of the algorithm, $A(u) \cap A(v) = \{1, \dots, 2\Delta - 1\}$, so initially, this is trivially feasible. By the time (u, v) is considered in the for loop, at most $\Delta - 1$ edges incident on u and another $\Delta - 1$ edges incident on v could have been colored already. Each edge can potentially disallow a distinct color; however, even after removing these $2(\Delta - 1)$ colors from $\{1, \dots, 2\Delta - 1\}$, there must remain at least one color that can be used for (u, v) .

Let $\{M_1, \dots, M_k\}$ be the coloring computed by the algorithm. For each $i = 1, \dots, k$, let e_i be the edge in M_i attaining the maximum weight in the matching (i.e., $w(e_i) = w(M_i)$) and let E_i be the edges that were colored before e_i and e_i itself. Since the coloring computed by the schedule is nice, it follows that the subgraph E_i has degree at least $\lceil \frac{i+1}{2} \rceil$ for each $i = 1, \dots, k$. All edges in E_i have weight $w(e_i)$ or larger. Hence, there must be at least $\lceil \frac{i+1}{2} \rceil$ matching in the optimal solution whose weight is at least $w(e_i)$. In other words, $w(C_{\lceil \frac{i+1}{2} \rceil}) \geq w(e_i)$. Therefore,

$$\begin{aligned} \sum_{i=1}^k w(M_i) &= \sum_{i=1}^k w(e_i) \leq \sum_{i=1}^{2\Delta-1} w(C_{\lceil \frac{i+1}{2} \rceil}) \\ &\leq w(C_1) + 2 \sum_{i=2}^{\Delta} w(C_i) \end{aligned}$$

$$\begin{aligned}
&= 2 \cdot \text{OPT}_{mc} - w(C_1) - 2 \sum_{i=\Delta+1}^{k^*} w(C_i) \\
&= \text{OPT}_{mc} \left(2 - \frac{w(C_1) + 2 \sum_{i=\Delta+1}^{k^*} w(C_i)}{\sum_{i=1}^{k^*} w(C_i)} \right) \\
&\leq \text{OPT}_{mc} \left(2 - \frac{w(C_1)}{\sum_{i=1}^{\Delta} w(C_i)} \right)
\end{aligned}$$

The proof is completed by noting that $\sum_{i=1}^{\Delta} w(C_i) \leq \Delta w(C_1)$ and $w(C_1) = w_{\max}$. \square

Recall that when edge weights are uniform, max-edge-coloring reduces to regular edge-coloring. In the uniform case, GREEDY-MAX-EDGE-COLORING returns an arbitrary nice edge-coloring. Simple examples exist showing that such a coloring can be $2 - \frac{1}{\Delta}$ away from the optimal max-coloring. However, for instances where the weights are uniform, or close to uniform, one can simply ignore the weights and approximate the underlying edge-coloring instance. The next lemma states that taking the best between these two strategies leads to an improved approximation algorithm.

Lemma 9 *Let χ be the edge-coloring returned by GREEDY-MAX-EDGE-COLORING and $\widehat{\chi}$ be an arbitrary edge-coloring that uses \widehat{k} colors. Then the best solution between these two is a $2 - \frac{2}{\widehat{k}+1}$ approximation for max-edge-coloring.*

Proof Recall that by [Theorem 18](#)

$$w(\chi) \leq \text{OPT}_{mc} \left(2 - \frac{w_{\max}}{\text{OPT}_{mc}} \right).$$

On the other hand,

$$w(\widehat{\chi}) \leq \widehat{k} \cdot w_{\max}.$$

Consider taking a convex combination of these two inequalities with coefficients $\frac{\widehat{k}}{\widehat{k}+1}$ and $\frac{1}{\widehat{k}+1}$, respectively. The result is

$$\frac{\widehat{k} \cdot w(\chi)}{\widehat{k}+1} + \frac{w(\widehat{\chi})}{\widehat{k}+1} \leq 2 \cdot \text{OPT}_{mc} \cdot \frac{\widehat{k}}{\widehat{k}+1} - \frac{\widehat{k} \cdot w_{\max}}{\widehat{k}+1} + \frac{\widehat{k} \cdot w_{\max}}{\widehat{k}+1} = \text{OPT}_{mc} \left(2 - \frac{2}{\widehat{k}+1} \right)$$

This proof is finished by noting that any convex combination of $w(\chi)$ and $w(\widehat{\chi})$ can only be larger than $\min\{w(\chi), w(\widehat{\chi})\}$. \square

Thus, if one can efficiently edge-color the input graph with few colors, then we can obtain a slightly better approximation than that offered by [Theorem 18](#). For bipartite multigraphs a Δ -edge-coloring can be computed in polynomial time [[36](#), Ch. 20]. For general simple graphs, Vizing's algorithm finds an $(\Delta + 1)$ -edge-coloring in polynomial time [[36](#), Ch. 28]. For general multigraphs, Shannon's algorithm finds a $\frac{3}{2}\Delta$ -edge-coloring in polynomial time [[36](#), Ch. 28]. From these facts, one obtains the following theorem.

Corollary 5 *There is a $2 - \frac{2}{\Delta+1}$ approximation for bipartite multigraphs, a $2 - \frac{2}{\Delta+2}$ approximation for general simple graphs, and a $2 - \frac{2}{1.5\Delta+1}$ approximation for general multigraphs.*

On the negative side, max-edge-coloring inherits the hardness of regular edge-coloring in general graphs. Unlike, regular edge-coloring, it remains hard even for bipartite graphs.

Theorem 19 ([[7, 20](#)]) *For any $\epsilon > 0$, max-edge-coloring cannot be approximated within $\frac{4}{3} - \epsilon$ in general graphs or within $\frac{7}{6} - \epsilon$ in bipartite graphs.*

9 Bounded Max-Coloring

This section studies the bounded max-coloring problem that is defined as follows. Given a graph $G = (V, E)$ with weight function $w : V \rightarrow \mathbb{N}$ on the vertices and a number $m \in \mathbb{N}$, the *bounded max-coloring* problem is to produce a proper vertex coloring of G such that if C_1, \dots, C_k are the color classes, then $|C_i| \leq m$ for $i = 1, \dots, k$, and $\sum_{i=1}^k \max_{v \in C_i} w(v)$ is minimized.

An algorithm for hereditary graph classes is presented that transforms an α -approximation algorithm for the max-coloring problem into an $\alpha + 1$ -approximation for the bounded max-coloring problem. The section first describes this generic reduction and then proceeds to discuss special graph classes.

Theorem 20 ([[6](#)]) *For a class of graphs \mathcal{G} if there is an α -approximation algorithm for the max-coloring problem, then for any $m \in \mathbb{N}$, there is an $\alpha + 1$ -approximation algorithm for the bounded max-coloring problem with a bound of m on the size of any color class.*

Proof The proof is algorithmic. Given an α -approximate solution to the max-coloring problem, an algorithm converts this solution into a feasible solution to the bounded max-coloring problem. The algorithm simply sorts the vertices in each color class in non-increasing order of weights and partitions each color class into sets of size at most m in this order. The algorithm is described in [Algorithm 8](#).

The goal is to bound the performance of Algorithm B-MAX-COLOR which we do as follows. Notice that

Algorithm 8 B-MAX-COLOR(G, w, m)

-
1. Let C_1, \dots, C_k be a max-coloring of G
 2. Let $c_i = |C_i|$
 3. **for** $i = 1, 2, \dots, k$ **do**
 4. Sort C_i in non-increasing weight order // Let v_1, \dots, v_{c_i} be this order
 5. **for** $j = 1, 2, \dots, \lceil c_i/m \rceil$ **do**
 6. Let $C_{ij} = \{v_k : \lfloor k/m \rfloor = j\}$
 7. **return** C_{ij} for all i, j as the coloring
-

$$\begin{aligned}
 \text{OPT}_{bmc}(G) &= \sum_{i,j} w(C_{ij}) \\
 &= \sum_{i=1}^k w(C_{i1}) + \sum_{i=1}^k \sum_{j=2}^{\lfloor \frac{c_i}{m} \rfloor} w(C_{ij}) \\
 &\leq \alpha \cdot \text{OPT}_{mc}(G) + \sum_{i=1}^k \sum_{j=2}^{\lfloor \frac{c_i}{m} \rfloor} w(C_{ij})
 \end{aligned} \tag{6}$$

Let $V' = V \setminus \cup_{i=1}^k C_{i1}$. The goal now is to bound the second term in the right-hand side of (6). Let $H = (V, \emptyset)$ be the weighted graph with no edges, where the vertices have the same weights as in the original graph G . An optimal bounded max-coloring of H follows by sorting the vertices in non-increasing weight order and greedily partitioning this order into sets of size m .

The color classes of an optimal bounded max-coloring of H induce a permutation π^* of V , defined as follows: The color classes of H are ordered in non-increasing weight order, and within each color class, we orders the vertices again in non-increasing weight order. This is equivalent to ordering all the vertices in non-increasing weight order by the discussion in the previous paragraph. Similarly, define a permutation π on the vertices in V' as follows. The color classes C_{ij} , $i = 1, \dots, k$, $j = 2, \dots, \lfloor c_i \rfloor$ are ordered in non-increasing weight order, where the weight of $C_{ij} = \max_{v \in C_{ij}} w(v)$, and within each color class, order the vertices in non-increasing weight order. Since a greedy partitioning is used in algorithm B-MAX-COLOR, all but at most one color class C_{ij} has size $< m$ for each i . Assume that all color classes have size m by adding dummy elements equal to the smallest element in the color class. Further, assume that ties are broken identically in π^* and π . Consider the position $\pi(u)$ of a vertex u . Next, it is shown that $\pi(u) \geq \pi^*(u)$ for all vertices $u \in V'$.

Suppose vertex $u \in V'$ is in color class C_i in the original coloring. Let $t(u)$ denote the number of color classes C_s , $s \neq i$ such that there is a vertex $v \in C_s$ that precedes u in the permutation π , that is, $\pi(v) < \pi(u)$. Let $S_{<}(u)$ and $S_{\geq}(u)$ denote the sets of vertices that have weight at most u and greater than u , respectively, in positions $1, 2, \dots, \pi(u)$. Then,

$$\pi(u) \leq |S_{<}(u)| + |S_{\geq}(u)| \quad (7)$$

Consider the number of elements in $S_{<}(u)$. Clearly, each color class C_s , $s \neq i$ contributes at most $m - 1$ elements to $S_{<}(u)$, and hence,

$$|S_{<}(u)| \leq t(u) \cdot (m - 1) \quad (8)$$

To count $|S_{\geq}(u)|$, all the vertices in π^* that appear before u are in $S_{\geq}(u)$. However, at least $t \cdot m$ vertices have been eliminated that appear in color classes C_{i1} for all i . Hence,

$$|S_{\geq}(u)| \leq \pi^*(u) - t \cdot m \quad (9)$$

Now, substituting the inequalities (8) and (9) in the inequality (7), one gets

$$\begin{aligned} \pi(u) &\leq t(u) \cdot (m - 1) + \pi^*(u) - t(u) \cdot m \\ &= \pi^*(u) - t(u) \\ &\leq \pi^*(u) \end{aligned}$$

Since $\pi(u) \leq \pi^*(u)$ for each element $u \in V'$, it follows that

$$\begin{aligned} \sum_{i=1}^k \sum_{j=2}^{\lceil \frac{C_i}{m} \rceil} w(C_{ij}) &\leq \text{OPT}_{bmc}(H) \\ &\leq \text{OPT}_{bmc}(G) \end{aligned} \quad (10)$$

Hence, plugging this into inequality (6), one gets

$$\text{OPT}_{bmc}(G) \leq (\alpha + 1)\text{OPT}_{bmc}(G) \quad \square$$

Theorem 20 gives a constant factor approximation algorithm for bounded max-coloring whenever max-coloring has a constant factor approximation. In particular, there is a constant approximations for max-coloring bipartite and interval graphs and for max-edge-coloring general graphs.

Bampis et al. [2] considered the bounded max-coloring and bounded max-edge-coloring problem in bipartite graphs and trees. In doing so, they were able to obtain better bounds than those provided by **Theorem 20**. In particular, for bipartite graphs, they get the following results.

Theorem 21 ([2]) *For bipartite graph, bounded max-coloring can be approximated within $\frac{17}{11}$, and bounded max-edge-coloring can be approximated within e .*

On the negative side, Gardi [15] showed that bounded coloring in bipartite graphs (even when all weights are uniform) cannot be approximated within $\frac{4}{3} - \epsilon$ for any $\epsilon > 0$ unless $P = NP$. For bounded max-edge-coloring, the best hardness is $\frac{7}{6} - \epsilon$ for any $\epsilon > 0$, and it is inherited from regular max-edge-coloring [7].

In the case of trees, they gave a PTAS for bounded max-coloring and a 2-approximation for bounded max-edge-coloring. Like the regular max-coloring, it is unknown whether bounded max-coloring trees is hard or not. Interestingly, bounded max-edge-coloring trees is hard.

Theorem 22 ([2]) *For trees, bounded max-coloring admits a PTAS, and bounded max-edge-coloring can be approximated within 2. The latter is NP-hard.*

10 Conclusion

This chapter has described some results on the max-coloring problem, a natural generalization of classic graph coloring. There have been several results on this problem for various graph classes, in particular, for perfect graphs and subclasses thereof. While this survey is not meant to be exhaustive, the presentation here was restricted to some representative graphs and simple algorithms. Besides the natural question of closing the gap between the known lower and upper bounds, there is one intriguing open question that deserves mentioning.

The max-coloring problem on trees admits an exact algorithm running in $2^{\text{poly log } n}$ time, and hence, there is no hope of showing NP-hardness. There are a few natural problems known that have such a property, for example, computing a minimum dominating set in a tournament or satisfiability with n clauses and $\log^2 n$ variables. For such problems, Papadimitriou and Yannakakis define the complexity classes $\log\text{-NP}$ and $\log\text{-SNP}$ [29]. It would be interesting to put the max-coloring problem on trees into one of these classes.

The max-coloring problem aims to minimize the *makespan* of a batched schedule of jobs. Another natural objective is to minimize the *average completion time* of the jobs. This objective has been considered by Epstein et al. [11], who obtain approximation algorithms for various graph classes, most notably, the class of perfect graphs and subclasses thereof. There is a third objective that is more natural and that is of the *flow-time* of the jobs. Each job comes with a release date and cannot be processed before this start date. The cost of a job is the difference between its completion time and release time, that is, the amount of time the job stays in the system. Almost nothing is known about this objective when there is an underlying conflict graph on the jobs.

Cross-References

- ▶ [Equitable Coloring of Graphs](#)
- ▶ [Greedy Approximation Algorithms](#)
- ▶ [On Coloring Problems](#)

Recommended Reading

1. B.S. Baker, E.G. Coffmann Jr., Mutual exclusion scheduling. *Theor. Comput. Sci.* **162**, 225–243 (1996)
2. E. Bampis, A. Kononov, G. Lucarelli, I. Milis, Bounded max-colorings of graphs, in *Proceedings of the 21st International Symposium on Algorithms and Computation* (Springer, 2010), pp. 353–365
3. H.L. Bodlaender, K. Jansen, Restrictions of graph partition problems. Part I. *Theor. Comput. Sci.* **148**(1), 93–109 (1995)
4. N. Bourgeois, G. Lucarelli, I. Milis, V.Th. Paschos, Approximating the max-edge-coloring problem. *Theor. Comput. Sci.* **411**(34–36), 3055–3067 (2010)
5. D.G. Corneil, S. Olariu, L. Stewart, The ultimate interval graph recognition algorithm? (extended abstract), in *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms* (SIAM, 1998), pp. 175–180
6. J. Correa, N. Megow, R. Raman, K. Suchan, Cardinality constrained graph partitioning into cliques with submodular costs, in *Proceedings of the 8th Cologne Twente Workshop on Graphs and Combinatorial Optimization* (2009), pp. 347–350
7. D. de Werra, M. Demange, B. Escoffier, J. Monnot, V.Th. Paschos, Weighted coloring on planar, bipartite and split graphs: Complexity and approximation. *Discret. Appl. Math.* **157**(4), 819–832 (2009)
8. M. Demange, D. de Werra, J. Monnot, V.Th. Paschos, Weighted node coloring: When stable sets are expensive, in *Proceedings of the 28th International Workshop on Graph-Theoretic Concepts in Computer Science* (Springer, 2002), pp. 114–125
9. M. Demange, D. de Werra, J. Monnot, V.Th. Paschos, Time slot scheduling of compatible jobs. *J. Sched.* **10**(2), 111–127 (2007)
10. Z. Dvorak, K.-I. Kawarabayashi, R. Thomas, Three-coloring triangle-free planar graphs in linear time. *ACM Trans. Algorithms* **7**(4), 41 (2011)
11. L. Epstein, M. M. Halldorsson, A. Levin, H. Shachnai, Weighted sum coloring in batch scheduling of conflicting jobs. *Algorithmica* **55**(4), 643–665 (2009)
12. L. Epstein, A. Levin, On the max coloring problem, in *Proceedings of the 2th Workshop on Approximation and Online Algorithms* (Springer, 2007), pp. 142–155
13. L. Epstein, M. Levy, Online interval coloring and variants, in *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming* (Springer, 2005), pp. 602–613
14. B. Escoffier, J. Monnot, V.Th. Paschos, Weighted coloring: further complexity and approximability results. *Inf. Process. Lett.* **97**(3), 98–103 (2006)
15. F. Gardi, Mutual exclusion scheduling with interval graphs or related classes. Part I. *Discret. Appl. Math.* **157**(1), 19–35 (2009)
16. M.R. Garey, D.S. Johnson, L.J. Stockmeyer, Some simplified np-complete graph problems. *Theor. Comput. Sci.* **1**(3), 237–267 (1976)
17. M.C. Golumbic, *Algorithmic graph theory and perfect graphs* (Academic, New York, 1980)
18. D.J. Guan, X. Zhu, A coloring problem for weighted graphs. *Inf. Process. Lett.* **61**(2), 77–81 (1997)
19. M.M. Halldórsson, H. Shachnai, Batch coloring flat graphs and thin, in *Proceedings of the 11th Scandinavian Workshop on Algorithm Theory* (Springer, 2008), pp. 198–209
20. I. Holyer, The np-completeness of edge-coloring. *SIAM J. Comput.* **10**(4), 718–720 (1981)
21. K. Jansen, The mutual exclusion scheduling problem for permutation and comparability graphs. *Inf. Comput.* **180**(2), 71–81 (2003)
22. T. Kavitha, J. Mestre, Max-coloring paths: Tight bounds and extensions, in *Proceedings of the 20th International Symposium Algorithms and Computation* (Springer, 2009), pp. 87–96
23. A. Kesselman, K. Kogan, Nonpreemptive scheduling of optical switches. *IEEE Trans. Commun.* **55**(6), 1212–1219 (2007)
24. A. Kierstead, W.T. Trotter, An extremal problem in recursive combinatorics. *Congr. Numerantium* **33**, 143–153 (1981)

25. J. Kratochvil, Precoloring extensions with a fixed color bound. *Acta Math. Univ. Comen.* **62**, 139–153 (1993)
26. G. Lucarelli, I. Milis, Improved approximation algorithms for the max edge-coloring problem. *Inf. Process. Lett.* **111**(16), 819–823 (2011)
27. N.S. Narayanswamy, R. Subhash Babu, A note on first-fit coloring of interval graphs. *Order* **25**, 49–53 (2008)
28. T. Nonner, Clique-clustering yields PTAS for max-coloring interval graphs, in *Proceedings of the 38th International Colloquium on Automata, Languages and Programming* (Springer, 2011), pp. 183–194
29. C.H. Papadimitriou, M. Yannakakis, On limited nondeterminism and the complexity of the v-c dimension. *J. Comput. Syst. Sci.* **53**(2), 161–170 (1996)
30. S.V. Pemmaraju, R. Raman, Approximation algorithms for the max-coloring problem, in *Proceedings of the 32th International Colloquium on Automata, Languages and Programming* (Springer, 2005), pp. 1064–1075
31. S.V. Pemmaraju, R. Raman, K. Varadarajan, Buffer minimization using max-coloring, in *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms* (SIAM, 2004), pp. 562–571
32. S.V. Pemmaraju, R. Raman, K. Varadarajan, Max-coloring and online coloring with bandwidths on interval graphs. *ACM Trans. Algorithms* **7**(3) (2011)
33. C.N. Potts, M.Y. Kovalyov, Scheduling with batching: A review. *Eur. J. Oper. Res.* **120**, 228–249 (2000)
34. R. Raman, Chromatic Scheduling. PhD thesis, University of Iowa, 2007
35. N. Robertson, D.P. Sanders, P.D. Seymour, R. Thomas, The four-colour theorem. *J. Comb. Theory B* **70**(1), 2–44 (1997)
36. A. Schrijver, *Combinatorial Optimization* (Springer, Berlin/Heidelberg, 2003)
37. S. Vishwanathan, Randomized online graph coloring. *J. Algorithms* **13**(4), 657–669 (1992)

Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs

Donatella Granata, Raffaele Cerulli, Maria Grazia Scutellà and
Andrea Raiconi

Contents

1	Introduction	1914
2	The Maximum Flow Problem.....	1915
2.1	Augmenting Path-Based Algorithms.....	1918
2.2	Preflow-Push Algorithms.....	1922
3	Edge-Labeled Graphs.....	1927
4	Maximum Flow with the Minimum Number of Labels (MF-ML).....	1933
4.1	Basic Notations and Definitions.....	1933
4.2	Time Complexity.....	1935
4.3	A Mathematical Formulation.....	1936
4.4	Skewed Variable Neighborhood Search.....	1938
4.5	Computational Results.....	1939
5	Conclusion.....	1945
	Cross-References.....	1945
	Recommended Reading.....	1945

D. Granata (✉)

Department of Statistics, Probability and Applied Statistics, University of Rome “La Sapienza”,
Rome, Italy

e-mail: d.granata@na.iac.cnr.it

R. Cerulli • A. Raiconi

Department of Mathematics, University of Salerno, Fisciano (SA), Italy
e-mail: raffaele@unisa.it; araiconi@unisa.it

M.G. Scutellà

Department of Computer Science, University of Pisa, Pisa, Italy
e-mail: scut@di.unipi.it

Abstract

The aim of this chapter is to present an overview of the main results for a well-known optimization problem and an emerging optimization area, as well as introducing a new problem which is related to both of them. The first part of the chapter presents an overview of the main existing results for the classical maximum flow problem. The maximum flow problem is one of the most studied optimization problems in the last decades. Besides its many practical applications, it also arises as a subproblem of several other complex problems (e.g., min cost flow, matching, covering on bipartite graphs). Subsequently, the chapter introduces some problems defined on edge-labeled graphs by reviewing the most relevant results in this field. Edge-labeled graphs are used to model situations where it is crucial to represent qualitative differences (instead of quantitative ones) among different regions of the graph itself. Finally, the *maximum flow problem with the minimum number of labels* (MF-ML) problem is presented and discussed. The aim is to maximize the network flow as well as the homogeneity of the solution on a capacitated network with logic attributes.

This problem finds a practical application, for example, in the process of water purification and distribution.

1 Introduction

Network flow problems arise in several fields of inquiry, including applied mathematics, engineering, management, operations research, and computer science.

One of the most relevant network flow problems is the maximum flow problem (MFP) that can be stated as follows: given an arc-capacitated graph and two special nodes, namely, a source s and a sink t , the problem is to send as much flow as possible from s to t , by respecting the arc capacities. In the standard maximum flow problem formulation, capacities are real values, while integral and unit capacity cases reflect special scenarios.

There is a wide presence of flow networks in the real world. For instance, think about a network of pipes used to transport fluids like gas, water, or oil, in which the arcs represent the pipes and the nodes are their junctions. Other examples can be found in transportation networks, where arcs and nodes represent roads and crosses, or in communication networks, where the arcs represent cables with different bandwidths connecting a set of terminals. Whatever is the kind of the considered network, the maximum flow on them represents the maximum rate at which it is possible to ship commodities from the source to the sink. Besides its practical applications, the MFP arises as a subproblem of more complex optimization problems, such as minimum cost flow, matching, and covering on bipartite graphs as well as NP-hard network design and routing problems. This justifies the continuous efforts carried out throughout the years by researchers to improve the efficiency of the existing algorithms and to design new ones.

Extensive discussion on the maximum flow problem and its applications can be found in the books of Even [29], Ford and Fulkerson [33], Lawler [43], Papadimitriou and Steiglitz [52], Ahuja et al. [7], Tarjan [59], and in Goldberg et al., survey [38].

Recently, an NP-hard variant of the classical MFP has been introduced, namely, the maximum flow with minimum number of labels (MF-ML). MF-ML belongs to the class of problems defined on arc-labeled (or colored) graphs, which has met a growing interest in the last decade. Labeled graphs are used to model situations where it is necessary to represent qualitative differences among different regions of the graph itself. Typically, each label represents a different property (or a set of properties) to model.

The MF-ML problem consists in finding the maximum flow solution that minimizes the number of different labels on a graph where a label and a capacity are assigned to each arc. As for the original version of the problem, this variant finds real-world applications in several types of networks, although this chapter just shows an application to water purification systems.

The work is organized as follows: In Sect. 2, the maximum flow problem is formally defined, and the most relevant solution approaches for the problem are introduced. Section 3 presents an overview of the literature and the most important contributions in the emerging field of the edge-labeled graph problems. Finally, Sect. 4 addresses the *maximum flow with the minimum number of labels* (MF-ML), by presenting both mathematical formulations and metaheuristic algorithms to solve it.

2 The Maximum Flow Problem

The maximum flow problem can be defined both on directed and undirected graphs, but it is necessary to distinguish between them, although the directed and undirected variants are closely related. Ford and Fulkerson [33] provided a reduction from undirected graphs to directed graphs with comparable size and capacities. Picard and Ratli [53] gave the reduction from directed to undirected graphs, allowing to find the maximum flow value in a directed graph by solving a minimum cut problem on an undirected graph. However, a significant drawback occurs when this reduction is applied to a unit capacity instance, since it produces an instance with capacities that are proportional to the number of nodes. No efficient algorithms are available, so far, for the reduction from the unit capacity undirected problem to the unit capacity directed problem. For the sake of simplicity, from now on, only the maximum flow problem defined on directed graphs is considered.

Let $G = (V, A)$ be a directed graph where V is the *set of nodes* and A is the *set of arcs*, with $n = |V|$ and $m = |A|$. Let $u_{ij} \geq 0$ be a capacity assigned to each arc $(i, j) \in A$, and let $U = \max\{u_{ij} : u_{ij} < \infty \forall (i, j) \in A\}$. It is now made the assumption that whenever an arc (i, j) belongs to A , arc (j, i) also belongs to it; this assumption is not a limiting one, since arcs with zero capacity are allowed. Let $\delta_G^+(i) = \{(i, j) : (i, j) \in A\}$ and $\delta_G^-(i) = \{(k, i) : (k, i) \in A\}$ be the forward

star and the backward star of node $i \in V$, respectively. A graph $G' = (V', A')$ is a subgraph of G if $V' \subseteq V$ and $A' \subseteq A$. A path $P = (v_1, \dots, v_k)$ is a sequence of distinct nodes belonging to V such that, for $1 \leq i \leq k - 1$, (v_i, v_{i+1}) belongs to A . Finally, two special nodes s and t are distinguished (called source and sink, respectively). Every other node is called *intermediate*.

A function $x : A \rightarrow \mathbb{R}$ is a *feasible flow* of G with value $\hat{f} \geq 0$ if the following constraints are satisfied:

$$\sum_{(i,j) \in \delta_G^+(i)} x_{ij} - \sum_{(k,i) \in \delta_G^-(i)} x_{ki} = \begin{cases} \hat{f} & i = s \\ 0 & \forall i \in V \setminus \{s, t\} \\ -\hat{f} & i = t \end{cases} \quad (1)$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A \quad (2)$$

The variable x_{ij} represents the amount of flow sent on arc (i, j) . Flow conservation constraints (1) make sure that for each intermediate node the total ingoing flow is equal to the total outgoing flow, while the net outgoing flow of the source is equal to the net ingoing flow of the sink. Capacity constraints (2) impose that the flow sent on each arc is nonnegative and does not exceed its capacity.

The maximum flow problem consists in finding a feasible flow with the maximum value f^* , and can therefore be formulated as

$$\max \quad \hat{f} \quad (3)$$

$$s.t. \text{ (1) and (2)}$$

Note that the feasibility region of this LP problem is nonempty since $x_{ij} = 0 \quad \forall (i, j) \in A$ is a feasible solution and is bounded by constraints (2).

From now on, the expressions *feasible flow* and *flow* will be indistinguishably used since only feasibility will be considered unless differently specified.

Given a flow x of G , let the *residual network of G induced by x* , denoted as $G^x = (V, A^x)$, be a graph composed of the same set of vertices of G and the set of arcs A^x such that:

- For each $(i, j) \in A$ such that $x_{ij} = 0$, $(i, j) \in A^x$ and has capacity $r_{ij} = u_{ij}$.
- For each $(i, j) \in A$ such that $0 < x_{ij} < u_{ij}$, $(i, j) \in A^x$ and has capacity $r_{ij} = u_{ij} - x_{ij}$; moreover $(j, i) \in A^x$ with capacity $r_{ji} = x_{ij}$.

- For each $(i, j) \in A$ such that $x_{ij} = u_{ij}$, $(j, i) \in A^x$ and has capacity $r_{ji} = u_{ij}$.
- Each arc $(i, j) \in A^x$ has residual capacity $r_{ij} > 0$; this means that it is possible to send (or subtract) a positive amount of flow on (i, j) . An example of residual network is shown in Fig. 1.

An *s-t cut* of G , denoted by $[V_s, V_t]$, is a partition of its set of vertices V in two subsets V_s and V_t such that $s \in V_s$ and $t \in V_t$. Moreover, let (V_s, V_t) denote the set of forward arcs going from V_s to V_t , and let (V_t, V_s) be the set of backward arcs going from V_t to V_s . The capacity of an *s-t cut* (V_s, V_t) , denoted by $u[V_s, V_t]$, is given by the sum of the capacities of its forward arcs, that is, $u[V_s, V_t] = \sum_{(i,j) \in (V_s, V_t)} u_{ij}$.

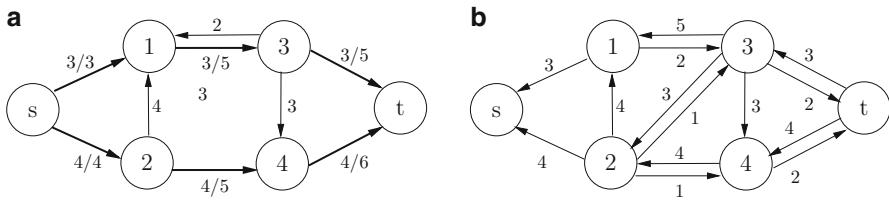


Fig. 1 (a) An example of a flow network with a maximum flow value $f^* = 7$. The source is s , and the sink t . The numbers denote flow and capacity, respectively. (b) Corresponding residual network

Now, consider the linear dual of the maximum flow formulation (10)–(12):

$$\min \sum_{(i,j) \in A} u_{ij} h_{ij} \quad (4)$$

s.t.

$$w_t - w_s = 1 \quad (5)$$

$$w_i - w_j + h_{ij} \geq 0 \quad \forall (i, j) \in A \quad (6)$$

$$h_{ij} \geq 0 \quad \forall (i, j) \in A, \quad (7)$$

Where each variable w_i corresponds to the flow conservation constraint for node i belonging to (1), and each variable h_{ij} corresponds to constraint $x_{ij} < u_{ij}$. The dual constraint (5) is associated with the primal flow variable \hat{f} , while constraints (6) are related to the variables x_{ij} . It can be noted that this dual formulation models the problem of finding the $s-t$ cut with minimum capacity. Indeed, given a feasible solution $[V_s, V_t]$, let

$$w_i = \begin{cases} 1 & \forall i \in V_t \\ 0 & \forall i \in V_s \end{cases}$$

$$h_{ij} = \begin{cases} 1 & \forall (i, j) \in (V_s, V_t) \\ 0 & \forall (i, j) \in A \setminus (V_s, V_t) \end{cases}$$

This assignment of values provides a feasible solution for the dual formulation, and the objective function value (4) is equal to the capacity value $u[V_s, V_t]$. Hence, from the weak duality theorem, it can be derived that the capacity of any cut of G is an upper bound to the maximum flow that can be sent from s to t , while as a consequence of the strong duality theorem, the following result can be stated:

Theorem 1 (Maximum Flow – Minimum Cut Theorem) *The maximum flow value in G is equal to the minimum capacity of the $s-t$ cuts of G .*

In the following sections, two general classes of algorithms to solve the maximum flow problem are presented, namely:

- **Augmenting path-based algorithms:** During their execution, these algorithms maintain the flow conservation constraints for each node of the network, and iteratively increase the value along augmenting paths from s to t .
- **Preflow-push algorithms:** Algorithms belonging to this class do not guarantee that flow conservation constraints are respected during their execution. Therefore, there could be flow *excesses* in some intermediate nodes. Preflow-push algorithms send these flow excesses toward nodes close to the sink, or close to the source when the former is not possible anymore.

2.1 Augmenting Path-Based Algorithms

The first known augmenting path procedure was introduced by Ford and Fulkerson in [31]. This algorithm is based on the concepts of residual graphs, cuts (as defined above) and augmenting paths. Given a graph G and a flow x , an augmenting path is a path (v_1, v_2, \dots, v_k) from the source s to the sink t in the residual network G^x , that is, $v_1 = s$, $v_k = t$.

The residual capacity r of an augmenting path P is defined as the minimum among all capacities of the arcs in P . From the definition of residual network, r is always a positive value. Therefore, if there is an augmenting path in G^x , then it is possible to send additional flow from s to t . Indeed, it is possible to prove that:

Theorem 2 (Ford-Fulkerson) *A flow x is maximum if and only if there is no augmenting path; that is, t is not reachable from the source s in the residual graph G^x .*

From [Theorem 2](#), the basic scheme for augmenting path algorithms is derived; that is, find an augmenting path P into the residual graph and send r flow units along P , were r is its capacity. This operation is repeated until there are no augmenting paths available in the residual graph. The steps of this generic augmenting path algorithm are subsumed in the [Algorithm 1](#).

[Algorithm 1](#) does not specify how to compute the augmenting path at each iteration. This is a crucial aspect on which the time complexity of the algorithm depends. One of the first algorithms developed to compute the augmenting path is the *labeling algorithm*, proposed by Ford and Fulkerson in [32]. This algorithm explores the residual network starting from the source s in order to find all the reachable vertices. During the exploration, vertices are divided into *labeled* (i.e., reachable from s) and *non-labeled*. The algorithm iteratively selects one of the labeled vertices, checks its forward star in the residual network, and labels all its neighbors. When the sink t is labeled, an augmenting path P is found and a flow equal to the residual capacity of P is sent along it; then, all the labels are deleted for the next iteration. If after checking the forward stars of all labeled vertices the sink

Algorithm 1 Generic augmenting path algorithm**Procedure Augmenting Path****for** each $(u, v) \in A$ **do** $x(u, v) \leftarrow 0$ **end for****while** there is a path P from s to t in G^x **do** $r \leftarrow \infty$ **for** each (u, v) in the path P **do** $r \leftarrow \min\{r, r_{uv}\}$ **end for** **for** each (u, v) in the path P **do** $x(u, v) \leftarrow x(u, v) + r$ $x(v, u) \leftarrow x(v, u) - r$ **end for****end while****return** x

is not labeled, the algorithm stops because no augmenting paths are available in the residual network. In Fig. 2 an execution of the algorithm is shown.

A worst-case time complexity analysis of the labeling algorithm is now provided. Note that each iteration except the last one increases the flow along an augmenting path, and requires $O(m)$ time. In fact, since each node is visited only if it is not labeled and is labeled after that, each forward star (and therefore each arc) is explored at most once. Sending the flow, as well as deleting all the labels, requires $O(n)$ time. Therefore, the complexity of the algorithm depends on the number of augmenting paths found. If all capacities are integer values and limited by an upper bound U , the capacity of an $s-t$ cut can be at most $(n-1)U$. The augmenting path algorithm will send at least one flow unit at each iteration; therefore, the overall time complexity of the algorithm is $O(nmU)$.

Unfortunately, there are some cases in which the Ford-Fulkerson algorithm may loop. In particular, if all the arc capacities are rational, then the Ford-Fulkerson algorithm eventually halts, but if they are irrational, the algorithm may loop forever, since it could perform an infinite sequence of small flow augmentations, and in the worst case it may not even converge to the maximum flow. Zwick [68] showed three small acyclic sample networks on which the Ford-Fulkerson procedure may fail to terminate.

Edmonds and Karp in [28] suggested two specializations of the Ford-Fulkerson algorithm. The first one consists in computing, at each iteration, the augmenting path of minimum length, which can be found by means of a breadth-first search (BFS) visit of the graph starting from s . It was proved that using shortest augmenting paths, the algorithm performs $O(nm)$ augmentations and, therefore, the whole procedure runs in $O(nm^2)$ time.

The second one is based on the idea of *scaling* that is based on computing augmenting paths with a residual capacity which is at least equal to a parameter Δ . In this way, it is guaranteed that at each iteration the flow value is increased of at

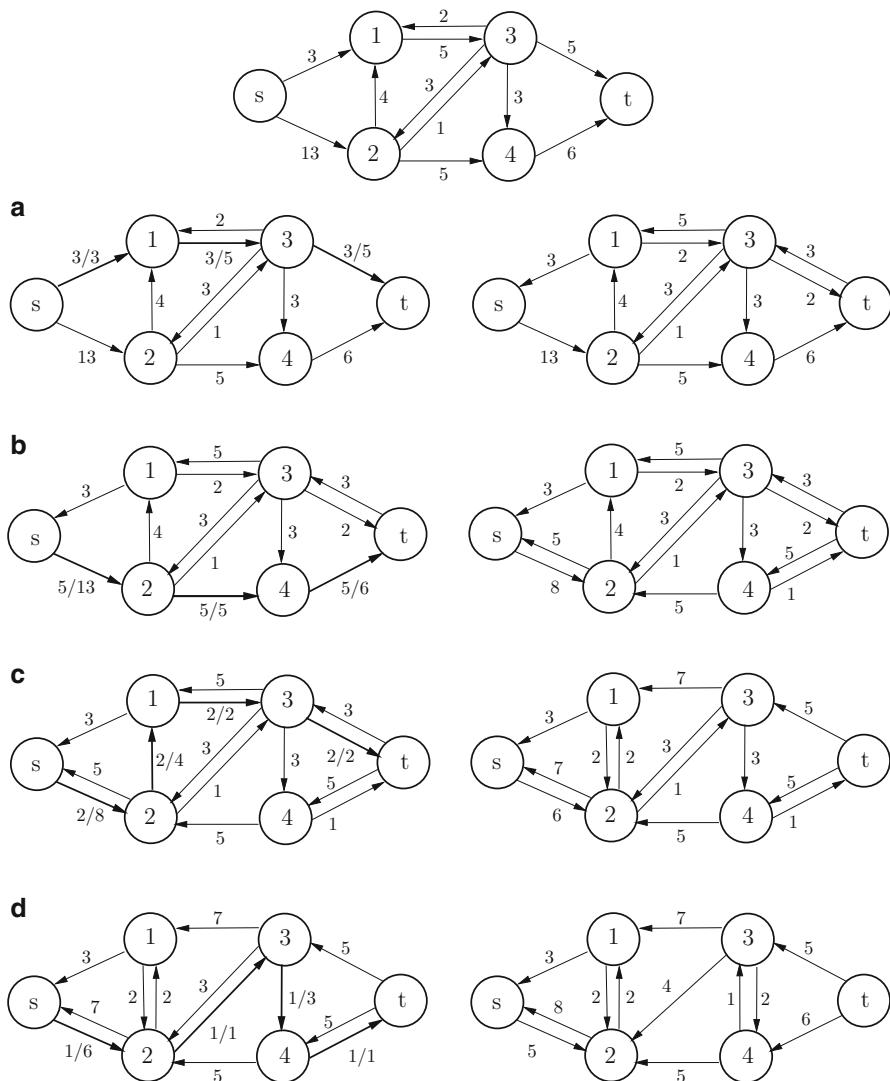


Fig. 2 Execution of the Ford-Fulkerson algorithm. (a)–(d) The *left side* figures show the augmenting path P on the residual network G^x , represented by *bold arrows*. The notation *flow/capacity* is used for each arc. The figures on the *right side* show the obtained residual graph from adding f to the current flow. The case (d) shows the residual network when there does not exist any augmenting path, and the flow x shown in (d) is therefore a maximum flow

least Δ units. To this end, the algorithm starts with $\Delta = 2^{\lfloor \log U \rfloor}$ and builds the residual network $G^{x,\Delta}$ containing only arcs with a residual capacity greater than or equal to Δ . Then, it computes an augmenting path in $G^{x,\Delta}$. If this path exists, then a quantity of flow greater than or equal to Δ is sent along this path, and the residual graph is updated. Otherwise, the algorithm sets $\Delta = \Delta/2$ and rebuilds $G^{x,\Delta}$. The algorithm stops when on the residual network $G^{x,\Delta}$, with $\Delta = 1$, no augmenting paths are found. Its time complexity is equal to $O(m^2 \log U)$.

A refinement of the shortest augmenting path algorithm was proposed by Ahuja and Orlin in [5], and it relies on the concept of *distance labels*. Given a flow x , a distance function d assigning a nonnegative integer to each node is *valid* if it satisfies the following conditions:

$$d(t) = 0 \quad (8)$$

$$d(v) \leq d(w) + 1 \quad \forall (v, w) \in A^x \quad (9)$$

As a consequence of the distance function definition, the following results can be stated:

Corollary 1 *If the distance labels are valid, the distance label $d(u)$ of a node u is a lower bound on the length of the shortest directed path from u to the sink t in the residual graph.*

Corollary 2 *If the distance labels are valid and $d(s) \geq n$, there are no paths from s to t in the residual graph.*

Corollary 1 can be easily proved by considering a path to t of length k in G^x , $(u_1, u_2, \dots, u_{k-1}, u_k, t)$. By conditions (8) and (9), it follows that

$$\begin{aligned} d(u_k) &\leq d(t) + 1 = 1 \\ d(u_{k-1}) &\leq d(u_k) + 1 \leq 2 \\ &\dots \\ d(u_1) &\leq d(u_2) + 1 \leq k \end{aligned}$$

Corollary 2 is an immediate consequence of Corollary 1.

Distance labels are defined *exact* if, for each node u , $d(u)$ is exactly the length of the shortest directed path from u to t in the residual graph. Exact distance labels can be determined in $O(m)$ time by means of a reverse breadth first search starting from t . Moreover, an arc $(u, v) \in A^x$ is defined *admissible* if $d(u) = d(v) + 1$, and a path is admissible if it is only composed of feasible arcs. It is straightforward to observe that an admissible $s-t$ path is an augmenting path with minimum length.

The improved shortest augmenting path algorithm starts by computing the exact distance labels for the nodes of G . At each iteration, it tries to build at each iteration

an admissible $s-t$ path, gradually extending an admissible path from s to some intermediate node u . If an admissible arc (u, v) exists, the path is extended (*advance* step); otherwise, the algorithm backtracks on the decision of including u in the path, and its distance label $d(u)$ is updated to $1 + \min\{d(v) : (u, v)A^x\}$ (*retreat* step). If the admissible path reaches t , an augmentation is performed using the path. The algorithm stops when $d(s) \geq n$, by returning a maximum flow in $O(n^2m)$.

In 1970, Dinitz [27] introduced his algorithm for the MFP, which became known as the *Dinic's algorithm*. Dinic's algorithm is based on the observation that many consecutive steps in the shortest path version of the augmenting path algorithm involve augmenting paths of the same length. Therefore, its aim is to push flow along all the shortest augmenting paths simultaneously. In fact, Dinic's algorithm considers the graph as a layered network, in which a given layer k contains all the vertices whose distance from the source s is k ; for example, layer 0 contains just s . The algorithm works in phases and uses the classical *breadth-first search (BFS)*. At the beginning of each phase, the algorithm constructs the residual network G^x induced by the current flow x and runs the *BFS* on it, starting from s . If the sink t is not reached, then x is maximal, and the algorithm stops. Otherwise, the *BFS* constructs a layered network L^x . When the algorithm reaches t , at distance k from s , it puts t in layer k and discards all remaining vertices. The procedure keeps in L^x all those arcs that go from one layer to the next one (discarding arcs within the same layer and backward arcs). The algorithm runs in $O(n^2m)$ time and in networks with unit capacities it terminates in $O(m\sqrt{n})$ time.

In conclusion, it is important to emphasize that the search for more efficient maximum flow algorithms led to the development of sophisticated data structures for implementing Dinic's algorithm. Some data structures were suggested by Shiloach [54] and Galil and Namaad [36]; the resulting implementations run in $O(nm \log^2 n)$ time, but the most efficient one is the *dynamic tree data structure* introduced by Sleator and Tarjan [55, 56], which yields an $O(nm \log n)$ time maximum flow algorithm.

2.2 Preflow-Push Algorithms

Given a graph G , a preflow x is a flow such that constraints (2) are satisfied whereas the net flow at node $v \in N \setminus \{s, t\}$ can be positive, that is $e(v) = \sum_{(v,w) \in \delta_G^+(v)} x_{vw} - \sum_{(k,v) \in \delta_G^-(v)} x_{kv} \geq 0$, where $e(v)$ is called the *excess flow* of node v . That is, in this class of algorithms the capacity constraints are satisfied, but the total amount of incoming flow for an intermediate vertex can exceed the total amount outgoing from it. A node with a strictly positive flow excess is defined *active*.

The first algorithm in this class is due to Karzanov [41]. The algorithm proposed by Karzanov considers layered networks, in the sense already introduced by describing the Dinic's algorithm. It starts by sending as much preflow as possible from s to its adjacent nodes (i.e., nodes in layer 1). The preflow is then propagated

Algorithm 2 Generic preflow-push algorithm

Procedure Preflow-Push

Preprocess

while there is at least an active node **do** Select active node u Push/Relabel(u)**end while**

to t : each intermediate node considers its arcs to the next layer according to a predefined order and saturates them one at a time (with the possible exception of the last considered arc), until its excess becomes zero. After this phase, defined *advance*, the algorithm considers the highest layer containing active nodes. For each of these nodes, the incoming flow is reduced until the excess is equal to 0, and all its incoming arcs are marked as closed; this phase is defined *balance*. Subsequently, a new advance phase is executed, sending as much preflow as possible on arcs that are not closed. The algorithm iterates advance and balance phases until there are no intermediate nodes with positive excess. Karzanov's algorithm runs in $O(n^3)$ time. Cherkassky [21] and Galil [35] presented further improvements of Karzanov's algorithm, which allow one to improve the time complexity to $O(n^2m^{1/2})$ and $O(n^{5/3}m^{2/3})$, respectively.

Goldberg and Tarjan [37] proposed a class of preflow-push algorithms using the concept of distance labels. These algorithms work by examining the active intermediate nodes and pushing excess from them to nodes that are estimated to be closer to the sink t , by sending flow along admissible arcs in the residual network (*push* operation). If the currently considered active node does not have any admissible arc, then its distance label is increased (*relabel* operation), in order to refine the distance approximation and to generate, as a consequence, new admissible arcs. The algorithms terminate when there are no active nodes. The steps of the generic preflow-push algorithm by Goldberg and Tarjan are presented in [Algorithm 2](#), while [Algorithm 3](#) shows the preprocess and the push/relabel subroutines. Implementing the procedure requires to specify in which order the active vertices will be visited.

The preprocess phase performs some initialization operations. More in detail, it sets the initial flow to 0 for each arc of the network, computes the exact distance labels, and sends flow from the source to its neighbors. Since all the arcs outgoing from the source are saturated, the procedure also updates the value of $d(s)$ to n , which by [Corollary 2](#) implies that the residual graph does not contain any directed path from s to t . The push-relabel operation on a generic active node u checks whether there is an admissible arc in the residual graph on which a push operation can be performed. A push along (u, v) decreases both r_{uv} and $e(u)$ and increases both $e(v)$ and r_{vu} . It is straightforward to observe that if the push operation does not saturate the selected arc, then the excess of u is reduced to 0. The relabel operation aims at creating at least one admissible arc on which the algorithm will be able to

Algorithm 3 Preprocess and push/relabel operations**Procedure Preprocess****for** each $(u, v) \in A$ **do** $x(u, v) \leftarrow 0$ **end for****for** each $u \in V$ **do** compute the exact distance label $d(u)$ **end for****for** each $(s, j) \in \delta_G^+(s)$ **do** $x(s, j) \leftarrow u_{sj}$ **end for** $d(s) \leftarrow n$ **Procedure Push/Relabel(u)****if** there is an admissible $(u, v) \in \delta_{G^x}^+(u)$ **then** *push:* $\delta \leftarrow \min\{e(u), r_{uv}\}$ $x(u, v) \leftarrow x(u, v) + \delta$ $e(u) \leftarrow e(u) - \delta$ $e(v) \leftarrow e(v) + \delta$ $r_{uv} \leftarrow r_{uv} - \delta$ $r_{vu} \leftarrow r_{vu} + \delta$ **else** *relabel:* $d(u) \leftarrow \min\{d(v) + 1 : (u, v) \in \delta_{G^x}^+(u)\}$ **end if**

perform new pushes. Eventually, no more flow can be sent to the sink, and relabeling will send flow back to the source.

When the algorithm terminates, there is no excess in the intermediate nodes, and therefore the preflow is a feasible flow. Since the algorithm maintains valid distance labels and $d(s) = n$ throughout the algorithm execution, then there are no augmenting paths from s to t in the residual graph. Therefore, the termination criterion of the augmenting path algorithm is met, and the flow sent to t is a maximum flow. Goldberg and Tarjan in [37] proved that the generic version of the preflow-push algorithm terminates in $O(n^2m)$ time. The computational time is dominated by nonsaturating pushes.

In [38], the authors have proposed an efficient implementation where each vertex v maintains a list of incident arcs, in a fixed but arbitrary order; the vertex also keeps track of its *current* arc, which will be the next one used for pushing operations originating from v . The implementation iteratively performs the *discharge* operation, shown in [Algorithm 4](#). The operation attempts iteratively to push excess of flow from an active node along its current arc. If the arc is saturated, the current arc is replaced with the next one in the list; if the last arc of the list has

Algorithm 4 Discharge operator

```

procedure Discharge(u)
let  $(u, v)$  be the current edge for  $u$ 
time_to_relabel  $\leftarrow \text{false}$ 
while  $e(v) > 0$  or  $\text{time\_to\_relabel} == \text{false}$  do
    if  $(u, v)$  is admissible then
        push on  $(u, v)$ 
    else
        if  $(u, v)$  is not the last edge of the list of  $u$  then
            update the current edge with the next one
        else
            update the current edge with the first one
            time_to_relabel  $\leftarrow \text{true}$ 
        end if
    end if
end while
if  $\text{time\_to\_relabel} == \text{true}$  then
    relabel  $u$ 
end if

```

been saturated, the current arc is updated with the first one of the list and the node is relabeled. The operation terminates when either the node is no more active or it has been relabeled.

A crucial factor for the performance of a push-relabel algorithm is the order in which active nodes are examined. Two possible orders are suggested and analyzed in [37] and [44]. In the First In, First Out selection strategy (FIFO), the data structure implemented for the set of active vertices is a queue Q ; therefore, the front vertex of Q is chosen for discharge operations, and all the new active vertices are pushed to the back of it. The other selection strategy is called largest label. According to this strategy, the vertex with the bigger distance label is selected for the discharge operation. The FIFO algorithm runs in $O(n^3)$ time, as shown by Goldberg and Tarjan in [37], while the largest label algorithm runs in $O(n^2m^{1/2})$ time as proved by Cheriyan and Maheshwari [44].

The implementation of the largest label algorithm uses an array of sets L_i , $0 \leq i \leq 2n - 1$ and an index l into the array. The set L_i is composed of all the active vertices labeled with i , which are represented as a doubly linked list. Therefore, the insertion and deletion operations take $O(1)$ time. The index l is the largest label of an active vertex. After the initialization phase, all arcs outgoing from the source are saturated, and all the resulting active vertices (i.e., the vertices adjacent to s) are inserted into L_0 , and l is set to 0. At each iteration, the algorithm removes a vertex from L_l , processes it using the discharge operation, and updates l . The algorithm loops until l becomes negative, that is, when all the vertices are not active. The steps of the process-vertex procedure by Goldberg et al. [38] are shown in [Algorithm 5](#).

Algorithm 5 Process-vertex procedure

Procedure Process-vertexremove vertex u from the current L_l $last_label \leftarrow d(u)$ *Discharge*(u)add to $L_{d(w)}$ each vertex w that became active after the discharge operation**if** $d(u) \neq last_label$ **then** $l \leftarrow d(u);$ add u to $L_l;$ **else** **if** $L_l = \emptyset$ **then** $l \leftarrow l - 1;$ **end if****end if**

This algorithm correctly maintains l , since for each node v $discharge(v)$ either relabels v or reduces to zero its excess, but doesn't do both at the same time. In the first case, v is largest distance label active vertex, and then l must be increased to $d(l)$. In the other case, if the excess at v is zero, then the excess quantity has been moved to vertices with distance labels equal to $l - 1$, so if L_l is empty, l must be decreased by one.

Cheriyan and Mehlhorn [20] presented an alternative proof that the complexity of the preflow-push method by Goldberg and Tarjan using the largest label policy is $O(n^2 m^{1/2})$. The proof is based on a potential function argument, and unlike the one in [44] does not depend on the implementation of the algorithm using current edges.

Ahuja and Orlin [4] proposed a variant of the algorithm that incorporates the concept of *excess scaling* in order to limit the number of nonsaturating pushes. This is done by pushing flow from nodes with a *sufficiently large* excess to nodes with a *sufficiently small* one, and never letting excesses to become too large. The algorithm makes use a parameter Δ called *excess dominator*, which is a power of 2 larger than any excess in the residual graph, and guarantees that each nonsaturating push sends at least $\Delta/2$ units of flow. The preflow-push algorithm with excess scaling computes a maximum flow in $O(nm + n^2 \log U)$ time, where U is a bound on arc capacities. The authors also presented a parallel implementation using the PRAM model and requiring $p = \lceil n/m \rceil$ processors that runs in $O(n^2 \log U \log p)$ time. Subsequently, Ahuja et al. [6] developed two improved versions of the excess-scaling algorithms, namely, the *stack-scaling algorithm* that runs in $O(nm + n^2 \log U / \log \log U)$ and the *wave-scaling algorithm* that runs in $O(nm + n^2 (\log U)^{1/2})$. They also demonstrated that the time complexity of the latter algorithm can be lowered to $O(nm \log((n/m)(\log U)^{1/2} + 2))$ by using dynamic trees in the implementation.

Mazzoni et al. [45] proposed an algorithm that first finds a maximum preflow (the so-called first phase), and then converts the preflow in a maximum flow (the second phase). In the first phase, the algorithm selects an active node v , detects

an augmenting path from v to t , and moves some amount of flow along such a path. Then, it repeats this process until no augmenting path exists from any active node to the sink. Once a maximum preflow is found, in the second phase, the generic algorithm sends to the source the excess flow of each active node along augmenting paths from active nodes to s .

Cerulli et al. [18] analyzed a potential drawback of the preflow-push algorithm, defined *ping pong effect*. This issue, deriving from the local nature of the choices made by the procedure, occurs when flow units are iteratively exchanged between a node u and a node v along a sequence of push-relabel operations. Interrupting this sequence might require the distance label of one of the two nodes to become greater than the one of the source, which can correspond to $\Omega(n)$ flow exchanges between the nodes and $\Omega(n^2)$ overall push operations. To overcome the ping pong effect, the authors proposed a procedure called budget algorithm. The algorithm combines the preflow-push and the augmenting path approaches in order to make sure that, starting from a node u , instead of sending flow to node v with $d(u) = d(v) + 1$, it is sent to a node v' such that $d(u) - d(v') \geq k$ for a given constant k .

3 Edge-Labeled Graphs

In the graph theory literature, a considerable amount of work has been spent to face problems related to colored (labeled) graphs. Labeled graphs, as opposed to weighted graphs, are used to model situations where it is crucial to represent qualitative differences (instead of quantitative ones) among different regions of the graph itself. Each label represent a different property (or set of properties) to model. More in detail, considering colors or labels in association to the edges of a graph, it is possible to divide the literature contributions in two main classes: The works related to the assignment of such colors (edge coloring problems), and the ones whose aim is to find homogeneous or heterogeneous substructures in the case in which such an assignment is already given (problems on edge-labeled graphs). The maximum flow problem variant presented in Sect. 4 belongs to the second class. Some of the main existing results related to this class are summarized in this section. Problems defined on edge-labeled graphs usually are stated as an input graph $G = (V, E, L)$ such that V is the set of the vertices, E is the set of the edges, and L is the set of labels in the graph. Each edge $e \in E$ has an associated label $l_e \in L$. Moreover, let $n = |V|$ and $m = |E|$. Problems belonging to this field of research have application in many fields, like communication networks, multimodal transportation networks, and molecular biology, among others.

One of the most widely studied problem in this area is the minimum labeling spanning tree problem (MLST). Given an undirected edge-labeled graph $G = (V, E, L)$, the problem consists in finding the spanning tree with the minimum number of colors. The problem was initially addressed by Broersma and Li [9]. They proved, on the one hand, that the MLST is NP-hard by reduction from the minimum dominating set problem, and, on the other hand, that the “opposite” problem of looking for a spanning tree with the maximum number of colors is polynomially solvable. Independently, Chang and Leu [19] provided a different NP-hardness proof

of the problem by reduction from the Set Covering problem. They also developed two heuristics to find feasible solutions of the problem and tested the performance of these heuristics by comparison with the results of an exponential exact approach based on an A* algorithm, reporting very good performances for one of the heuristics (the maximum vertex covering algorithm, or MVCA). Krumke and Wirth [42] presented a modification of MVCA and demonstrated that the algorithm approximates the optimal solution with a worst case ratio of at most $2 \ln n + 1$. They also showed that the other heuristic in [19] can perform arbitrarily bad and that the problem cannot be approximated within a constant factor unless $P = NP$. Wan et al. [60] provided a better analysis of the MVCA variant given in [42] by showing that its worst case performance ratio is at most $\ln(n - 1) + 1$. More recently, Xiong et al. [62] obtained the better bound of $1 + \ln(b)$, where each color appears at most b times.

The pseudocode for the procedure is given in [Algorithm 6](#). It starts from a subgraph H composed by all the nodes of G and an empty set of edges. Iteratively, it adds to the subgraph all the edges with a given label, until H is connected. At each iteration, the label whose insertion would produce the smallest number of connected components is selected.

There also exist some works in literature presenting metaheuristic comparisons for the resolution of the MLST problem, including genetic, pilot method, simulated annealing, VNS, tabu search, and GRASP [15, 23, 61, 63].

Captivo et al. [11] presented a mixed integer linear formulation for MLST based on single-commodity flow. Moreover, the authors demonstrated that by relaxing the binary variables related to the selection of the edges, an optimal solution for the original problem is always found, while the computational time required to solve the formulation by means of a solver is greatly reduced.

A variant of the problem has been studied by Brüggemann et al. [10], where the MLST with bounded color classes has been addressed, in which each color appears a maximum of r times in the graph. This special case of the MLST is polynomially solvable for $r = 2$, and NP-hard and APX-complete for r greater than or equal to 3; moreover, they showed that the problem can be approximated with a factor equal to $r/2$ through local search.

A different variant, where the edges are both labeled and weighted, was proposed by Xiong et al. [65]. The authors investigated the label-constrained minimum spanning tree problem (LCMST), where given a positive integer k , the aim to find a minimum weight spanning tree that uses at most k distinct labels. The problem was demonstrated to be NP-complete. Moreover, the authors proposed two different local search schemes, a genetic algorithm and two ILP formulations based on single-commodity and multi-commodity flow. The authors also introduced the related problem consisting in finding the spanning tree with the minimum number of labels which respects a bound on the total weight, defined cost-constrained minimum label spanning tree (CCMLST). They presented an algorithmic framework based on bisection search that, given a generic algorithm for LCMST, can be used to find heuristic solutions for CCMLST.

The minimum labeling Steiner problem is a variant of the classical Steiner problem and is an extension of the MLST that has also been object of study; in

Algorithm 6 Modified maximum vertex covering algorithm**Procedure MVCA**

Let $L' \leftarrow \emptyset$ be a set of labels

Let $E' \leftarrow \emptyset$ be a set of edges

Let $H \leftarrow (V, E')$ be an undirected edge-labeled graph

while H is not connected **do**

$mincomp \leftarrow$ number of connected components in H

for all $i \in L \setminus L'$ **do**

$H_i \leftarrow (V, E' \cup \{\text{edges labeled with } i\})$

$comp_i \leftarrow$ number of connected components in H_i

if $comp_i < mincomp$ **then**

$mincomp \leftarrow comp_i$

$minlabel \leftarrow i$

end if

end for

$H \leftarrow (V, E' \cup \{\text{edges labeled with } minlabel\})$

$L' \leftarrow L' \cup \{minlabel\}$

end while

return L'

this problem, a subset of *basic* nodes is considered as in the case of the Steiner tree problem, and the problem consists in finding a subgraph covering all the basic nodes using the minimum number of labels. The problem was originally proposed by Cerulli et al. [17], where the authors presented a heuristic algorithm based on MVCA and several metaheuristics, namely, VNS, reactive tabu search, simulated annealing, and pilot method. Consoli et al. [22, 24, 25] proposed GRASP, discrete particle swarm, and VNS metaheuristics, as well as a hybrid local search combining VNS and simulated annealing.

Carr et al. [12] analyzed a possible generalization of MLST, which is the problem of finding a subgraph with the minimum number of nodes that satisfies certain connectivity constraints. They called this problem the min-color generalized forest and showed that even the simplest case, in which given a couple of nodes $\{s, t\}$ the aim is to find the $s-t$ path with fewer labels (MLP), is at least as hard to approximate as the red-blue set cover problem. In the red-blue set cover problem, given two sets of elements R and B and a family $S \subseteq 2^{R \cup B}$, the aim is to find a subfamily $C \subseteq S$ that covers all elements in B and the minimum number of elements in R . Unless $P = NP$, the red-blue set cover problem cannot be approximated within $O(2^{\log^{1-\delta} |S|})$, where $\delta = 1/\log \log^c |S|$ for any $c < 1/2$.

In Yuan et al. [66] the authors took into account reliability issues related to mesh networks in scenarios when several connections can fail due to the same event, by associating colors to failures. Therefore (assuming that each failure event could occur with the same probability), they also proposed the problem of finding the $s-t$ path with the minimum number of labels, that they called minimum-color

single-path (MCSiP). Furthermore, they introduced two problems in which the aim is to find two link-disjoint $s-t$ paths that minimize either the total number of colors (minimum total color disjoint-path or MTCDiP) or the number of overlapping colors (minimum overlapping color disjoint-path or MOCDiP). MTCDiP aims at reducing the overall possibility of a failure, while MOCDiP minimizes the possibility that both paths could fail due to a single failure. The authors proved that the problems are NP-complete and proposed integer linear formulations as well as heuristic algorithms for each of them.

Other results regarding MLST and MLP can be found in Hassin et al. [39]. In this work, the authors proposed a variant of MLST where a weight is associated to each label and showed an extension of the modified MVCA that yields an approximation guarantee equal to H_{n-1} , that is, the $n - 1$ th harmonic number. Moreover, by terminating prematurely MVCA and switching to the exact algorithm proposed in [10] when each label does not appear more than twice, they obtained an algorithm with $H(r) - 1/6$ approximation for the MLST when each label appears at most r times in the graph. They also proposed an $O(\sqrt{m})$ approximation algorithm for MLP and proved that the algorithm cannot be approximated within a polylogarithmic factor unless $P = NP$.

In Zhang et al. [67], the authors investigated the minimum label $s-t$ cut problem (MLC). Given a couple of nodes $\{s, t\}$, the objective of the problem is to select a subset of labels with minimum cardinality such that the removal of the corresponding edges would disconnect s from t . The authors presented an $O(\sqrt{m})$ approximation algorithm and showed that the problem cannot be approximated within $2^{\log^{1-1/\log \log^c n} n}$. They also proved that the same approximation hardness holds for MLP, improving previous approximation results.

A variant of the classical Hamiltonian cycle problem, in which the aim is find a Hamiltonian cycle with the minimum number of colors, has also been object of studies. The problem can be considered, for instance, a variant of the traveling salesman problem in cases in which connections belong to different transportation providers adopting fixed-rate charging. The problem was introduced by Cerulli et al. [16] with the name minimum labelling Hamiltonian cycle problem (MLHC). They proved the problem to be NP-complete both on general and complete edge-colored graphs and proposed a heuristic algorithm as well as a Tabu Search metaheuristic. Independently from their work, Xiong et al. [64] introduced the same problem limiting their scope to complete graphs, with the name colorful traveling salesman problem (CTSP). This work also includes a constructive heuristic and a metaheuristic, namely, a genetic algorithm.

Couëtoux et al. [26] presented approximation algorithms and hardness approximation results for the TSP with both the maximum and the minimum number of colors on complete edge-colored graphs. Regarding the maximization version, they introduced an algorithm with $\frac{1}{2}$ -approximation based on local improvements and showed the problem to be APX-hard. The minimization version was proved to be not approximable within $n^{1-\epsilon}$ for any fixed $\epsilon > 0$, where n is the number of nodes in the graph. Moreover, the authors demonstrated that if each color does not appear more than r times, where r is an increasing function of n , the problem cannot be approximated within a factor that is less than $O(r^{1-\epsilon})$ for any $\epsilon > 0$. Finally,

for the special case where color frequency is a constant, a greedy algorithm with approximation factor equal to $\frac{r+H_r}{2}$ was shown. For each algorithm, the authors demonstrated the tightness of their analysis by means of worst-case test instances.

Similarly to what was done in [65] for the MLST problem, in [40] the authors introduced two variants of MLHC defined on graphs that have both costs and labels associated to the edges. In the first variant, a maximum cost for the tour is imposed as constraint, and the aim is to minimize the number of labels. In the second variant, the budget constraint is on the maximum number of labels that can be used, and the Hamiltonian cycle with minimum cost has to be found. The authors presented branch-and-cut optimal algorithms for MLHC and both the variants.

Monnot [50] proposed the labeled perfect matching problem in bipartite graphs. Given an input graph $G = (V, E, L)$, with $|V| = 2n$ and such that E contains a perfect matching of size n , the problem consists in finding the perfect matching with either the minimum (labeled MinPM) or the maximum (labeled MaxPM) number of colors. The author proved that on two-regular bipartite graph (i.e., consisting in a collection of pairwise disjoint cycles with even length), if each label appears at most r times, both the problems are APX-complete for each $r \geq 2$. Using reductions from MinSAT and MaxSAT, he also proved that the minimization variant is 2-approximable and the maximization one is 0.7846-approximable. On complete bipartite graphs, it was proven that the labeled MinPM is APX-complete for $r \geq 6$ and can't be approximated within a factor of $O(\log n)$. The author also proposed a greedy algorithm for labeled MinPM which selects in each iteration a monochromatic matching with maximum size and showed that it provides a $\frac{r+H_r}{2}$ -approximation. Subsequently, in Monnot [51] the author showed that labeled MinPM is not in APX whenever it is defined on a bipartite graph with maximum degree equal to 3 and that unless $P = NP$ the problem is not in polyLog-APX.

Carrabs et al. [13] presented the labeled maximum matching (LMM) problem, in which the aim is to determine the maximum matching with the minimum number of different colors. They proved that the problem is NP-complete and presented four mathematical formulations, as well as a branch-and-bound approach.

In [30] several of the above mentioned problems were analyzed in terms of parameterized complexity. More in detail, the authors considered MLST, MLHC, MLP, MLC, labeled MinPM, LMM, and the minimum label edge dominating set (MLEDS), in which the objective is the edge-dominating set that uses the minimum number of labels. When parameterized by the number of used labels, all the problems are W[2]-hard. On the other hand, when parameterized by the solution size, MLP and MLC are W[1]-hard, while all the other problems are fixed-parameter tractable.

Several other works in this area are focused on the determination of properly edge-colored subgraphs respecting a specific pattern. A subgraph $G' = (V', E', L')$ of $G = (V, E, L)$ is *properly edge-colored* if and only if $|E'| \geq 2$ and any two successive edges differ in color. A *properly edge-colored path* does not allow that two adjacent edges have the same color and vertex repetitions. A *properly edge-colored trail* does not allow edge repetitions and every successive couple of edges must differ in color.

The determination of properly edge-colored $s-t$ paths is known to be solvable in polynomial time for general graphs with two colors as showed by Bang-Jensen and Gutin [8]. Their results were extended by Szeider [58], who proved that the problem of finding a properly colored path in a graph with any number of colors can be solved in linear time. Abouelaoualim et al. proved in [1] the equivalence between properly edge-colored path and properly edge-colored trails. They also stated that the existence of properly edge-colored $s-t$ trails can be checked in polynomial time (as a straightforward consequence of the results in [58]), presenting a polynomial procedure for both of them. Finally, they proved that is NP-complete to check the existence of k pairwise vertex/edge disjoint properly edge-colored $s-t$ paths or trails in an edge-colored graph $G = (V, E, L)$ even for $k = 2$ and $|L| = O(|V|^2)$, and that these problems are still NP-complete on graphs with $|L| = O(|V|)$ that do not contain properly edge-colored cycles.

Abouelaoualim et al. [2] gave sufficient degree conditions for the existence of properly edge-colored cycles and paths in edge-colored graphs, multigraphs and random graphs. In particular, they proved that an edge-colored multigraph of order n with at least three colors and with minimum color degree greater or equal to $\lceil(n + 1)/2\rceil$ has properly edge-colored cycles of all possible lengths, including Hamiltonian cycles.

Agueda et al. [3], introduced the properly edge-colored spanning tree problem. There exist two types of properly edge-colored tree, called weak (WST) and strong (SST), respectively; the main difference among them is that in a weak tree adjacent edges may have the same color. More formally, a tree T in an edge-colored graph is a SST if every two adjacent edges differ in color, while given a root node r it is a WST if any path in T from r to a leaf is a properly edge-colored one.

The authors presented sufficient conditions for the existence of spanning trees with both patterns.

Finally, it's worth reporting the main results related to problems involving edge-colored graphs and connection (or reload) costs applied to each vertex, which depend on the colors of its adjacent edges.

The reload cost model was introduced by Wirth and Steffan [57]. In a given undirected edge-colored graph G , with edges of the same color constituting a subgraph, let a *reload cost matrix* R represent the costs of moving from one subgraph to another, that is, the cost of switching between the corresponding colors. The reload cost of a path in G is the sum of the reload costs that arise at its internal nodes from passing from the color of an incident edge to the next one. Wirth and Steffan [57] proved that the *Min-Diam* problem (i.e., the problem of looking for a spanning tree of G having a minimum diameter and taking into account the reload cost) is not approximable at all, even when the maximum degree of the graph is five. If, instead, the reload costs satisfy the triangle inequality, the problem is named Δ -*Min-Diam*. They proved that this problem is not approximable within a logarithmic bound. Galbiati [34] showed that *Min-Diam* on graphs having maximum degree 4 cannot be approximated within any constant $\epsilon < 2$, and Δ -*Min-Diam* cannot be approximated within any constant $\epsilon < 5/3$.

4 Maximum Flow with the Minimum Number of Labels (MF-ML)

In this section, a variant of the classical maximum flow problem on edge-labeled graphs, which has been introduced by Cerulli and Granata [14], will be presented, namely, the *maximum flow with the minimum number of labels* (MF-ML). Given a directed capacitated and labeled graph $G = (N, A, L)$, where a capacity and a label belonging to the set L are associated with each arc, the aim is to look for a maximum flow on G , from a source s to a sink t , using the minimum number of different labels.

A practical application of MF-ML concerns the purification of water during the distribution process. Contamination in distribution systems may result from many factors including cross connections to sewers, contamination at service reservoirs and poor hygiene practice during repairs. All the water distribution systems can be contaminated by ingress of pollutants through leaks present in the pipes, pumps, valves, or storage tanks with consequences for the public health. Therefore, it is crucial to maintain water quality and reduce the possibility to deliver water with pollutants to households. To this end, the water is collected in cisterns, before the distribution to customers, where it is subjected to a decontamination process. Now, suppose to identify each pollutant with a different color and to associate a color with the pipe of the distribution network according to the pollutant, it could be subjected to (in case of more pollution contaminations associated with a single pipe, it is possible to model the pipe with a chain of arcs labeled with different pollution colors). Minimizing the number of colors used to bring the maximum flow from sources to destinations (cistern) means to minimize the pollutions eventually involved in a possible contamination and then minimizing the number of treatments to use during the decontamination process.

In the sequel of the section, a more formal definition of the problem and an analysis of its time complexity are initially presented. Then, a mathematical model and some possible enhancements are proposed. Finally, since the problem is NP-hard, a metaheuristic procedure based on the skewed *VNS* scheme is described, and related preliminary computational results are reported.

4.1 Basic Notations and Definitions

Let $G = (N, A, L)$ be a directed graph where a label $l_{ij} \in L$ and a capacity u_{ij} are assigned to each arc $(i, j) \in A$. Two special nodes s and t in G are given. Given a feasible flow $x = \{x_{ij}\}$ on G , denote with G^x the subgraph of G containing only edges (i, j) with $x_{ij} > 0$, that is $G^x = (N, A^x, L^x)$ with $A^x = \{(i, j) \in A : x_{ij} > 0\}$ and $L^x = \{l_{ij} : (i, j) \in A^x\}$. Moreover, given a label $l \in L$, let $A^l = \{(i, j) \in A : l_{ij} = l\}$. The couple (x, \hat{f}) is used to denote a feasible flow x with value \hat{f} . From now on, denote by f^* the value of the maximum flow on G from s to t . The MF-ML problem consists in looking for, among all the admissible

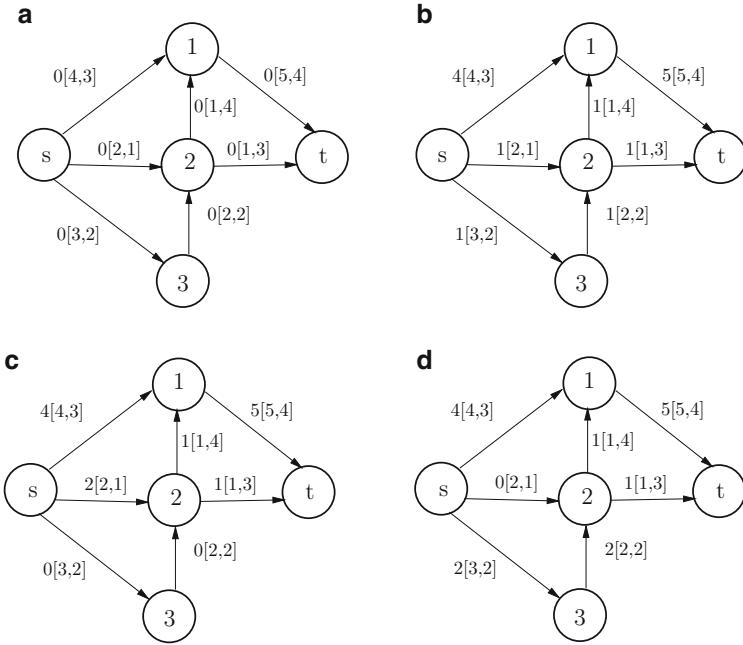


Fig. 3 (a) The starting graph G with $f^* = 6$. (b) A feasible solution with four labels $\{1, 2, 3, 4\}$. (c)–(d) Two optimal solutions with three labels, $\{1, 3, 4\}$ and $\{2, 3, 4\}$, respectively. Note that only the labels $\{3, 4\}$ are necessary as well nodes 1 and 2

flows with value f^* , a flow x such that $|L^x|$ is minimized. In the following, two useful definitions are introduced.

Definition 1 (Necessary label) A label $k \in L$ is a *necessary label* if and only if there is no feasible solution (x, f^*) in the subgraph $G_k = \{N, A_k, L \setminus \{k\}\}$ with $A_k = \{(i, j) \in A : l_{ij} \in L \setminus \{k\}\}$.

Definition 2 (Necessary node) A node $i \in N \setminus \{s, t\}$ is a *necessary node* if and only if the value of the maximum flow f_i^* on the subgraph of G induced by the nodes in $N \setminus \{i\}$ is strictly lower than f^* .

Denote by $\mathcal{CL} \subseteq L$ the set of the necessary labels and by $\mathcal{CN} \subseteq N$ the set of the necessary nodes. Moreover, let $f_i^- = f^* - f_i^* \forall i \in \mathcal{CN}$; it is straightforward to understand that for any max flow solution at least f_i^- units of flow have to pass through any necessary node i .

In the example in Fig. 3, labels $\{3, 4\}$ as well as nodes $\{1, 2\}$ are necessary. Moreover, $f_1^* = 1$ and $f_2^* = 4$.

4.2 Time Complexity

In this section, a proof that the decisional version of problem MF-ML is NP-complete is given. Consider the decisional version of the problem, namely, the *bounded labeled maximum flow problem (BLMFP)*:

Bounded Labeled Maximum Flow Problem: Given a directed, capacitated, and edge-labeled graph $G = (N, A, L)$, with a source node $s \in N$, a sink node $t \in N$, and a positive integer k , is there a maximum flow x from s to t in G such that the total number of different labels corresponding to arcs (i, j) with $x_{ij} > 0$ is less than or equal to k ?

Theorem 3 *The bounded labeled maximum flow problem is NP-complete.*

Proof It is easy to see that BLMFP is NP since it can be checked in polynomial time whether a given flow is feasible and maximum and whether the number of different labels of the arcs with a positive flow is less than or equal to k .

The theorem is proved by reduction from the minimum labeled spanning tree problem (MLST [9]). Let $G = (N, A, L)$ be an undirected and edge-labeled graph. Given an instance of MLST, let us generate a directed and arc-labeled graph $G' = (N', A', L')$ with a capacity associated with each arc and show that there exists a spanning tree of G with at most k labels if and only if there exists a maximum flow x from s to t in G' such that the total number of different labels of the arcs with a positive flow is at most $k' = k + 1$ (Fig. 4).

Let G' be the directed graph obtained from G in the following way:

- Replace any edge $(i, j) \in A$ with two directed arcs (i, j) and (j, i) having the same label of the original arc and associate a capacity $n - 1$, where $n = |N|$, with both of them.
- Add two new nodes in G' , respectively, the source node s and the sink node t ($N' = N \cup \{s, t\}$).
- Connect the source s to a generic node, say node w , by adding the arc (s, w) with capacity $n - 1$ and a new label z .
- Connect any node, but node w , to the sink node t , with z -labeled arcs, (j, t) whose capacity is equal to 1.

Therefore, G' contains $|N'| = n + 2$ vertices, $|A'| = 2|A| + n$ arcs, and $|L'| = |L| + 1$ labels. This construction can be accomplished in polynomial time. Note that the maximum flow value in G' is equal to $n - 1$, and all the z -labeled arcs must have a positive flow in each maximum flow solution.

Consider now a spanning tree T of G with at most k labels. A maximum flow in G' from s to t can be defined by sending a positive flow on all the z -labeled arcs and along the arcs of G' generated starting from the edges of T (see Fig. 4). Note that such a flow has at most $k' = k + 1$ labels. Vice versa, consider a maximum flow from s to t in G' with at most k' labels. This flow is such that all the z -labeled arcs have a positive flow and all the arcs with a positive flow whose label other than z form a spanning tree of G with at most $k = k' - 1$ labels. \square

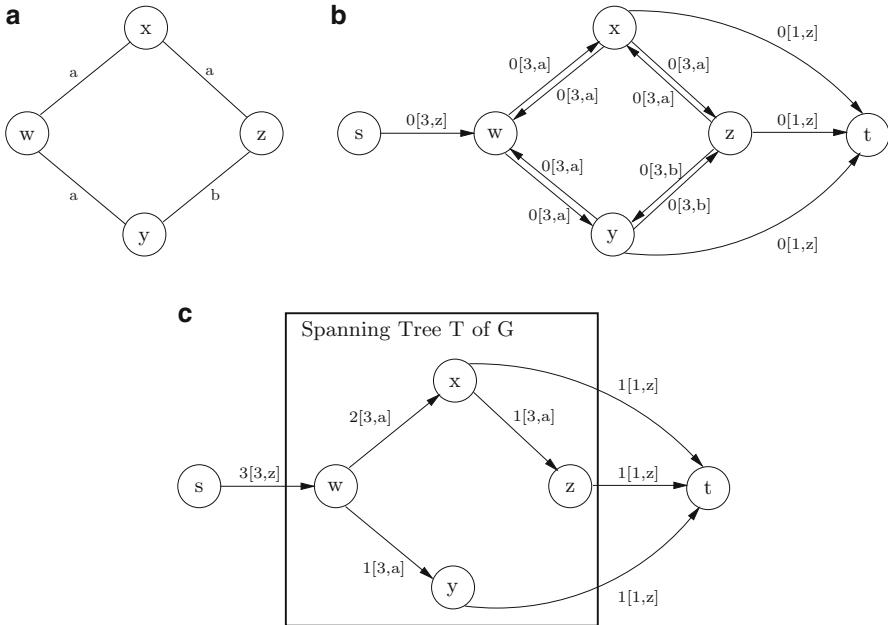


Fig. 4 (a) Graph G (generic MLST instance). (b) Graph G' obtained from G . (c) Graph G' where only arcs with positive flow and the related spanning tree T of G are shown

4.3 A Mathematical Formulation

Given a capacitated and arc-labeled directed graph $G = (N, A, L)$ and a feasible flow (x, f^*) , for each label $k \in L$, introduce a binary variable y_k whose value is equal to 1 if $k \in L^x$ and 0 otherwise. Let B be an input matrix where the entry b_{ij}^k ($(i, j) \in A$ and $k \in L$) is equal to 1 if $l(i, j) = k$ and 0 otherwise.

An integer linear programming formulation (ILP1) for MF-ML is the following:

$$(ILP1) \quad \text{ZILP1} = \min \sum_{k \in L} y_k \quad (10)$$

subject to

$$u_{ij} y_k \geq b_{ij}^k x_{ij} \quad \forall (i, j) \in A, k \in L \quad (11)$$

$$\sum_{(i,j) \in \delta_G^+(i)} x_{ij} - \sum_{(j,i) \in \delta_G^-(i)} x_{ji} = \begin{cases} f^*, & i = s; \\ 0, & \forall i \in N \setminus \{s, t\}; \\ -f^*, & i = t. \end{cases} \quad (12)$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A \quad (13)$$

$$y_k \in \{0, 1\} \quad \forall k \in L \quad (14)$$

Constraints (11) force the variable y_k to be equal to 1 if there exists at least one arc with positive flow whose associated label is k . Otherwise the variable y_k is not constrained and will be set to 0 to minimize the objective function. Constraints (12) and (13) are the classical flow conservation and capacity constraints once the flow value is fixed to f^* .

The (ILP1) formulation can be strengthened by incorporating information about the necessary nodes and labels, obtaining (ILP2):

$$(ILP2) \quad z_{ILP2} = |\mathcal{CL}| + \text{Min} \sum_{k \in L \setminus \mathcal{CL}} y_k \quad (15)$$

subject to

(12) and (13)

$$u_{ij} y_k \geq b_{ij}^k x_{ij} \quad \forall (i, j) \in A, \quad k \in L \setminus \mathcal{CL} \quad (16)$$

$$\sum_{(j,i) \in \delta_G^-(i)} x_{ji} \geq f_i^- \quad \forall i \in \mathcal{CN} \quad (17)$$

$$\sum_{(i,j) \in \delta_G^+(i)} x_{ij} \geq f_i^+ \quad \forall i \in \mathcal{CN} \quad (18)$$

$$y_k \in \{0, 1\} \quad \forall k \in L \setminus \mathcal{CL} \quad (19)$$

By introducing necessary labels, it is possible to reduce the cardinality of the set the binary variables, since variables corresponding to such labels would be always set to 1. Moreover, as already discussed, positive flow will pass through each necessary node, which brings to constraints (17) and (18).

A further improvement could be obtained by considering an additional class of valid inequalities defined as follows.

A *label cutting set* is a set of labels CS such that at least one of them is in every optimal solution. A cutting set CS is minimal if there does not exist another cutting set CS' such that $CS' \subset CS$. For example, in Fig. 3 the label set {1,2} is a minimal cutting set. Note that, by definition, a single necessary label is a minimal cutting set. Moreover, it can be also observed that for each necessary node, the set of labels belonging to its ingoing arcs as well as the ones of the outgoing arcs define two cutting sets.

For each cutting set CS, the valid inequality $\sum_{k \in CS} y_k \geq 1$ can be derived, which can be further strengthened by considering minimal cutting sets.

In order to obtain good lower bounds using the above presented formulation, one possibility is to relax the binary variables y . However, it is experimentally verified that this continuous relaxation brings to trivial lower bounds. Better lower bounds can be obtained by keeping integrality for variables y_k and by relaxing the flow conservation constraints for intermediate nodes, replacing them with

$$\sum_{(i,j) \in \delta_G^+(i)} x_{ij} - \sum_{(j,i) \in \delta_G^-(i)} x_{ji} \geq -\Delta \quad \forall i \in N \setminus \{s, t\} \quad (20)$$

$$\sum_{(i,j) \in \delta_G^+(i)} x_{ij} - \sum_{(j,i) \in \delta_G^-(i)} x_{ji} \leq \Delta \quad \forall i \in N \setminus \{s, t\} \quad (21)$$

where Δ is a positive chosen value.

4.4 Skewed Variable Neighborhood Search

Skewed variable neighborhood search (SVNS) [49] is a metaheuristic, introduced as an improvement of the simple variable neighborhood search (VNS). The idea was driven by the necessity to explore more fully valleys which are far away from the incumbent solution. Recall that the VNS is based on dynamically changing neighborhood structures during a local search process, without following a trajectory, but searching new solutions in increasingly distant neighborhoods from the current solution, jumping to the next local minimum only if it is a better solution than the incumbent one [46–48].

On the contrary, the SVNS accepts a solution close to the best one known but not necessarily better. In particular, let z be the objective function of a minimization problem (i.e., $z(x) = |L^x|$ in the MF-ML case), x be the current solution, x^* be the incumbent solution, and x'' be the local optimum of a given iteration. Then, SVNS accepts x'' if it improves the incumbent solution (i.e., $z(x'') < z(x^*)$) or if $z(x'') - \alpha\rho(x, x'') < z(x)$, where $\rho(x, x'')$ denotes a distance between the solution x and the new one x'' and α is a given parameter. Whenever the solutions are described by boolean vectors (as in the MF-ML case, by considering the boolean variables y_k), the Hamming distance can be used as a metric for evaluating the distance between the solutions. As for the value of the α parameter, after a tuning phase, in the SVNS algorithm the parameter α is set as follows: $\alpha = 0.5$ if $d \leq 0.5$ and $\alpha = 1.5$ otherwise, where d is a measure of the density of G (the computation of this measure will be discussed in Sect. 4.5).

A starting feasible solution is determined by solving the following problem, which minimizes the sum of the flows along the arcs of the network, by then recovering the corresponding number of used labels z_{UB} :

$$(UB) \quad \min \sum_{(i,j) \in A} x_{ij} \quad (22)$$

s.t. (12) and (13)

The underlying idea of this upper bound method is to find a maximum flow of G with a low number of used arcs since, intuitively, the number of used colors tends to decrease if the number of used arcs decreases as well.

Fig. 5 In this graph example, the maximum flow f can be sent using: (i) a path of two arcs $((s, 1, t)$ with $z_{UB} = 2$); (ii) a path of four arcs $((s, 2, 3, 1, t)$ with $z_{UB} = 4$; (iii) using both the paths (i) and (ii) (with $(z_{UB} = 5)$)

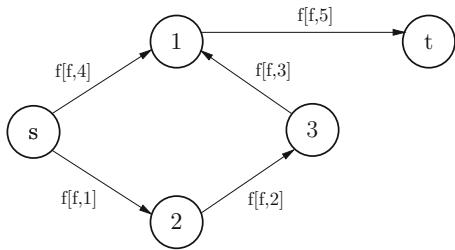


Figure 5 shows that minimizing $\sum_{(i,j) \in A} x_{ij}$ is likely to reduce the number of used arcs.

Moreover, in order to implement a SVNS procedure for the problem, a set of parametric neighborhoods have been defined. These neighborhoods are based on two integer parameters k and β :

- $N_{k,\beta}(x)$: a solution x' belongs to the neighborhood if the label set $L^{x'}$ can be obtained by removing up to k labels in $L^x \setminus \mathcal{CL}$ and adding up to k new labels, and moreover, the new labels associated with arcs which are incident to at least β nodes belonging to \mathcal{CN} .

Let $l_{\max} \in L$ be a label such that $A^{l_{\max}}$ has the maximum cardinality. The SVNS algorithm considers a neighborhood for each value of k in $\{1, \dots, |L \setminus \mathcal{CL}|\}$ and $\beta = 0$; moreover, it considers the neighborhood obtained by the previous definition by choosing $k = \lfloor |A^{l_{\max}}|/2 \rfloor$ and $\beta = 3$. That is, the SVNS algorithm takes into account a set of neighborhoods N_h , $h = 1, \dots, h_{\max}$ such that $N_1 \equiv N_{1,0}, \dots, N_{|L \setminus \mathcal{CL}|} \equiv N_{|L \setminus \mathcal{CL}|,0}$, $N_{|L \setminus \mathcal{CL}|+1} \equiv N_{\lfloor |A^{l_{\max}}|/2 \rfloor,3}$.

In Algorithm 7, the pseudocode of the SVNS procedure is given. In particular, in (a) a random solution is chosen from the current neighborhood; in (b) the algorithm tries to improve this solution by using a simple local search as described in the next subsection; in (c) is checked if the local optimum found is an improvement of the current incumbent, while in (d) it is checked if the found solution respects the SVNS criterion of acceptance.

4.4.1 Local Search Phase

Given a solution x' , at most 200 neighbors belonging to the current neighborhood $N_k(x')$ are examined. The local search phase tries to find a better solution by removing k labels and adding up to $k - 1$ new ones one at time until a new feasible solution is obtained and the β parameter is satisfied. The algorithm stops as soon as a neighbor x'' such that $z(x'') < z(x')$ is found; if no such solution is found after the considered iterations limit, the solution $x'' = x'$ is returned.

4.5 Computational Results

The instances used in the tests have been randomly generated by using three parameters: the number of nodes n , the density of the graph d , and the number of

Algorithm 7 Skewed variable neighborhood search for MF-ML

Procedure Initialize

1. Select the neighborhood structures N_k , for $k = 1, \dots, k_{max}$, where $k_{max} = |L \setminus \mathcal{CL}| + 1$ that will be used in the search;
2. find an initial solution z and its value $z(x)$;
3. $x^* \leftarrow x$
4. $z^* \leftarrow z(x)$

Procedure SVNS

1. $k \leftarrow 1$
 2. **Repeat** the following steps **until** $k = k_{max}$
 - (a) $x' \leftarrow$ Random solution $\in N_k(x)$
 - (b) $x'' \leftarrow LocalSearch(x')$
 - (c) **if** $z(x'') < z^*$ **then**
 $x^* \leftarrow x''$
 $z^* \leftarrow z(x'')$
end if
 - (d) **if** $z(x'') - \alpha\rho(x, x'') < z(x)$
 $x \leftarrow x''$
 $k \leftarrow 1$
else
 $k \leftarrow k + 1$
end if
-

labels $|L|$. Two collections of scenarios have been generated by considering two different ranges for the value of n . In the first one (*small* instances), the value of n ranges from 20 to 80 with a step equal to 20, while in the second one (*big* instances), the value of n ranges from 100 to 200 using the same step. Parameter d varies in the set 0.2, 0.5, 0.7; the total number of arcs in a given graph is $m = dn(n - 1)$. As for parameter $|L|$, it is set to 20, 30, and 40 % m . For each scenario, three instances have been generated, for a total of 90 scenarios (36 small and 54 big scenarios, respectively) and 270 instances (108 small and 162 big instances, respectively). Tables 1 and 2 report average results for each scenario. In the tables, the first three columns report the scenario characteristics: number of nodes, number of arcs, and number of labels. Columns 4 and 5 report average values for two instance characteristics, that is, maximum flow value f^* and number of necessary labels $|\mathcal{CL}|$. In Table 1, columns with headers RF, ILP1, and ILP2 represent the average optimum values of the relaxed model and the of integer programming formulations, respectively, computed using the *CPLEX* solver. The average computational time for ILP1 and ILP2 are also reported. For each run, the time limit has been set to 3,600 s.

Since some of the small instances are not solved by the solver in the considered time limit, in those cases the best upper bounds found by the solver are used to compute the averages. In particular, it is used the notation sol^{*0} , sol^{*1} or sol^{*2} in

Table 1 Computational result on small instances

n	m	L	f*	CL	RF		ILP1		ILP2		SVNS	
					Sol	Time	Sol	Time	Sol	Time	Sol	Time
20	76	15	42	5.00	6.33	7.33	0.00	7.33	0.00	7.33	0.00	0.00
20	76	22	26	4.67	6.33	6.67	0.00	6.67	0.00	6.67	0.00	0.00
20	76	30	8	1.00	2.00	2.00	0.00	2.00	0.00	2.00	0.00	0.00
20	190	38	91	8.33	13.33	13.67	0.01	13.67	0.01	13.67	1.28	0.00
20	190	57	106	9.67	16.00	18.00	0.21	18.00	0.02	18.00	0.49	0.00
20	190	76	99	18.33	20.33	22.33	0.39	22.33	0.26	22.33	0.84	0.00
20	265	53	162	21.33	24.00	24.00	0.01	24.00	0.01	24.00	2.16	0.00
20	265	79	167	13.33	24.00	25.00	0.01	25.00	0.01	25.00	0.37	0.00
20	265	106	172	29.33	29.67	30.67	0.01	30.67	0.01	30.67	0.59	0.00
40	312	62	198	12.00	19.67	21.33	0.53	21.33	0.11	21.33	1.12	0.00
40	312	93	78	4.00	9.67	10.67	1.81	10.67	1.00	10.67	5.20	0.00
40	312	124	98	6.00	15.67	17.67	1.21	17.67	4.51	17.67	3.76	0.00
40	780	156	374	20.33	32.00	32.00	0.07	32.00	0.06	32.00	3.42	0.00
40	780	234	498	38.00	42.67	44.00	0.48	44.00	0.2	44.00	2.98	0.00
40	780	312	283	13.67	27.67	29.33	5.34	29.33	3.39	30.00	3.23	2.26
40	1,091	218	481	28.67	44.00	44.33	0.06	44.33	0.06	44.33	3.15	0.00
40	1,091	327	440	25.67	41.67	42.33	0.36	42.33	0.2	42.33	3.40	0.00
40	1,091	436	591	27.00	51.00	52.33	7.51	52.33	2.78	53.33	4.63	1.89
60	708	141	308	10.67	23.33	24.00	10.17	24.00	6.48	24.33	2.50	1.36
60	708	212	329	20.67	28.33	30.00	69.03	30.00	32.7	30.67	3.22	2.22

(continued)

Table 1 (Continued)

n	m	L	f*	CL	RF		ILP1		ILP2		SVNS	
					Sol	Time	Sol	Time	Sol	Time	Sol	Time
60	708	283	335	10.33	25.33	27.00	19.03	27.00	12.42	27.33	3.43	1.26
60	1,770	354	778	27.00	43.67	44.00	0.35	44.00	0.23	44.67	4.96	1.52
60	1,770	531	820	27.67	50.00	50.67	46.62	50.67	55.43	51.00	8.79	0.63
60	1,770	708	826	30.67	56.00	57.67	145.38	57.67	51.41	60.00	9.44	3.95
60	2,477	495	1,198	33.67	62.67	62.67	0.36	62.67	0.36	62.67	10.60	0.00
60	2,477	743	1,177	59.67	77.00	78.00	6.61	78.00	1.35	78.33	17.09	0.42
60	2,477	991	1,278	73.33	85.67	87.00	122.98	87.00	19.7	87.67	21.18	0.75
80	1,264	252	492	13.67	25.00	26.67	446.95	26.67	129.67	27.00	5.03	1.26
80	1,264	379	448	11.00	27.00	28.33	1,781.07	28.33	793.42	29.00	8.05	2.30
80	1,264	505	645	15.00	37.67	40.00* ¹	1,240.09	39.67	1,536.47	41.67	16.12	4.92
80	3,160	632	1,525	62.67	67.67	67.67	0.83	67.67	0.38	67.67	20.94	0.00
80	3,160	948	1,656	37.00	71.67	73.67* ²	2,155.64	73.00	558.06	75.00	32.34	2.70
80	3,160	1,264	1,169	34.00	62.67* ¹	65.67* ⁰	3,600	64.67* ²	3,205.36	67.00	27.16	3.56
80	4,423	884	2,024	52.67	94.00	94.00	0.72	94.00	0.87	94.00	44.60	0.00
80	4,423	1,327	2,231	53.00	95.00	95.67	37.56	95.67	52.16	96.67	41.52	1.04
80	4,423	1,769	2,014	51.67	92.67	94.67* ²	1,980.49	94.67* ²	2,278.04	97.67	50.65	3.09

Table 2 Computational result on big instances

n	m	L	f^*	CL	SVNS	
					Sol	Time
100	1,980	396	1,095	22.00	48.00	19.83
100	1,980	594	1,095	22.00	54.33	23.64
100	1,980	792	1,095	22.00	57.33	24.31
100	4,950	990	2,290	42.00	76.33	52.73
100	4,950	1,485	2,290	42.00	86.67	65.18
100	4,950	1,980	2,290	42.00	92.00	71.15
100	6,930	1,386	3,044	69.00	110.00	90.62
100	6,930	2,079	3,044	69.00	116.67	119.23
100	6,930	2,772	3,044	69.00	123.67	139.57
120	2,856	571	1,020	22.00	41.33	19.07
120	2,856	856	1,151	22.67	49.00	48.61
120	2,856	1,142	1,278	21.00	61.00	36.10
120	7,140	1,428	3,347	56.33	97.33	109.72
120	7,140	2,142	3,432	55.00	106.67	146.12
120	7,140	2,856	2,776	51.67	105.00	111.27
120	9,995	1,999	4,483	80.67	128.33	179.58
120	9,995	2,998	5,649	86.00	162.67	217.27
120	9,995	3,998	4,414	73.33	148.00	322.99
140	3,892	778	1,811	25.00	58.33	65.65
140	3,892	1,167	1,671	28.00	63.33	55.92
140	3,892	1,556	1,942	42.00	70.67	76.67
140	9,730	1,946	5,210	67.00	126.33	249.80
140	9,730	2,919	4,799	64.33	125.33	261.40
140	9,730	3,892	4,629	72.00	144.33	354.12
140	13,621	2,724	7,000	97.00	179.33	414.90
140	13,621	4,086	6,585	93.33	178.33	558.91
140	13,621	5,448	7,253	99.00	199.33	515.56
160	5,088	1,017	2,594	34.00	68.33	65.33
160	5,088	1,526	2,201	27.00	63.67	73.66
160	5,088	2,035	2,239	28.00	71.33	103.87
160	12,720	2,544	5,805	91.00	144.67	242.31
160	12,720	3,816	5,003	71.33	130.33	304.83
160	12,720	5,088	5,696	92.00	168.33	621.72
160	17,807	3,561	8,836	200.00	209.00	362.02
160	17,807	5,342	7,746	103.67	193.67	891.52
160	17,807	7,123	7,490	99.67	191.00	943.74
180	6,444	1,288	2,761	34.33	74.67	122.57
180	6,444	1,933	3,615	41.00	94.67	172.06
180	6,444	2,577	3,080	36.00	93.33	170.89
180	16,110	3,222	8,556	92.00	174.00	641.89
180	16,110	4,833	7,525	85.67	167.00	815.80
180	16,110	6,444	7,205	81.00	172.67	911.43

(continued)

Table 2 (Continued)

n	m	$ L $	f^*	$ CL $	SVNS	
					Sol	Time
180	22,553	4,510	11,516	125.67	233.67	1,236.32
180	22,553	6,766	9,971	115.33	217.33	1,110.90
180	22,553	9,021	10,747	129.33	244.67	1,488.15
200	7,960	1,592	4,096	37.00	83.67	215.47
200	7,960	2,388	2,681	29.00	69.33	138.73
200	7,960	3,184	3,570	37.00	98.00	241.02
200	19,900	3,980	9,605	104.33	183.00	519.58
200	19,900	5,970	10,191	95.00	191.00	747.56
200	19,900	7,960	9,427	96.00	198.33	1,191.62
200	27,859	5,572	12,449	135.67	257.67	1,978.20
200	27,859	8,358	12,587	135.67	233.67	1,954.76
200	27,859	11,144	13,769	136.67	253.33	2,145.93

order to indicate whether 0, 1, or 2 instances were solved to optimality. The same notation is also used in a scenario for the relaxed model, where two instances have not been solved in the time limit; therefore, in this case, the average value reported in the table is not a certified average lower bound.

The last three columns in [Table 1](#) are related to the SVNS procedure. In particular, the average solution values are reported as well as running times and percentage gaps with respect to the optimal solution value (or the best upper bound available).

[Table 2](#) contains average results for the big instances. For these scenarios, the results obtained by CPLEX for RF, ILP1, and ILP2 are not reported; in fact, the solver was not able to find solutions within the time limit for most of the instances using these models. Therefore, the SVNS solutions are compared with the trivial lower bound given by $|CL|$.

Looking at [Table 1](#), it can be noticed that the RF formulation returns very good lower bounds. Indeed, in 6 out of 36 scenarios, it always finds certified optimal solutions, while the worst average upper bound is 13.6 % bigger than the optimum and the average gap is 3.79 %. Now, consider the two ILP formulations. It can be noticed that introducing necessary labels and nodes improves effectively the model, since in general the running times are reduced and, as a consequence, ILP2 is able to find five optimal solutions more than ILP1. In total, ILP1 fails to return a certified optimal value in 7 out of 108 instances, while ILP2 fails only twice. The SVNS also returns very good results in fast computational times. The instances with 20 and 40 nodes, it always returns optimal solutions in 52 out of 54 scenarios. In general, the maximum percentage gap is 4.92 %, and the average gap is under 1 %. The maximum average computational time is just 50.65 s.

Regarding [Table 2](#), it is possible to observe that the average gap among the very trivial lower bound $|CL|$ and the SVNS solution values is 89.13 %. This might seem a weak result. However, it should be noticed that the average gap between $|CL|$ and the optimal solution values in [Table 1](#) is 63.5 %, and therefore a good performance

of SVNS on these instances as well can be expected. Further research will be spent in finding nontrivial lower bounds for big instances.

5 Conclusion

In this chapter, the well-known maximum flow problem is reviewed, and an interesting variant on edge-labeled graphs is introduced. Since their introduction, flow problems defined on capacitated networks have been widely studied in the literature because of their great relevance. In fact, they can be used to effectively model and study many situations occurring in the real world. Furthermore, network-flow substructures are often used to model complex optimization problems. In the first part of this work, the most relevant contributions on the maximum flow problem are reviewed. In particular, the attention is focused on the description of augmenting path and preflow-push methods, which are the most commonly used classes of algorithms to face the problem. In the second part of the chapter, the maximum flow problem with the minimum number of Labels (MF-ML) is discussed, which aims at maximizing the amount of flow pushed from a given source to a given destination as well as the homogeneity of the solution on a capacitated network with logic attributes (usually known as colors or labels). This problem finds a practical application, for example, in the process of water purification and distribution. Our discussion starts with an overview of the class of optimization problems defined on labeled graphs, an area that has met a relevant interest in the last few years, in which many new problems have been defined and studied. Then the MF-ML problem is deeply explained by introducing some mathematical formulations and a skewed variable neighborhood search metaheuristic, whose efficiency is supported by a pool of preliminary computational results on randomly generated MF-ML instances.

Cross-References

- ▶ [Max-Coloring](#)
- ▶ [Network Optimization](#)
- ▶ [On Coloring Problems](#)
- ▶ [Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work](#)

Recommended Reading

1. A. Abouelaoualim, K. Ch. Das, L. Faria, Y. Manoussakis, C. Martinhon, R. Saad, Paths and trails in edge-colored graphs. *Theor. Comput. Sci.* **409**, 497–510 (2008)
2. A. Abouelaoualim, K.C. Das, M. Karpiński, Y. Manoussakis, C.A. Martinhon, R. Saad, W.F. de la Vega, Cycles, paths and trails in edge-colored graphs with given degrees. *J. Graph Theory* **64**, 63–86 (2010)
3. R. Agueda, V. Borozan, Y. Manoussakis, G. Mendy, R. Muthu, Sufficient conditions for the existence of spanning colored trees in edge-colored graphs. *Discret. Math.* (2012). doi:10.1016/j.disc.2012.01.031

4. R.K. Ahuja, J.B. Orlin, A fast and simple algorithm for the maximum flow problem. *Oper. Res.* **37**, 748–759 (1989)
5. R.K. Ahuja, J.B. Orlin, Distance-directed augmenting path algorithms for maximum flow and parametric maximum flow problems. *Nav. Res. Logist.* **38**, 413–430 (1991)
6. J.B. Ahuja, R.K. Orlin, R.E. Tarjan, Improved time bounds for the maximum flow problem. *SIAM J. Comput.* **18**, 939–954 (1989)
7. R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications* (Prentice-Hall, Englewood Cliffs, 1993)
8. J. Bang-Jensen, G. Gutin, Alternating cycles and trails in 2-edge-coloured complete multi-graphs. *Discret. Math.* **188**(1–3), 61–72 (1998)
9. H. Broersma, X. Li, Spanning trees with many or few colors in edge-colored graphs. *Discuss. Math. Graph Theory* **17**(2), 259–269 (1997)
10. T. Brüggemann, J. Monnot, G.J. Woeginger, Local search for the minimum label spanning tree problem with bounded color classes. *Oper. Res. Lett.* **31**, 195–201 (2003)
11. M. Captivo, J.C. Climaco, M.M. Pascoal, A mixed integer linear formulation for the minimum label spanning tree problem. *Comput. Oper. Res.* **36**, 3082–3085 (2009)
12. R.D. Carr, S. Doddi, G. Konjedov, M. Marathe, On the red-blue set cover problem, in *In Proceedings of the 11th ACN-SIAM Symposium on Discrete Algorithms*, 2000, pp. 345–353. ISBN: 0-89871-453-2
13. F. Carrabs, R. Cerulli, M. Gentili, The labeled maximum matching problem. *Comput. Oper. Res.* **36**(6), 1859–1871 (2009)
14. R. Cerulli, D. Granata, Varianti colorate del problema del massimo flusso. Technical report, University of Salerno, 2009
15. R. Cerulli, A. Fink, M. Gentili, S. Voß, Metaheuristics comparison for the minimum labelling spanning tree problem, in *The Next Wave in Computing, Optimization, and Decision Technologies*, ed. by B.L. Golden, S. Raghavan, E.A. Wasil (Springer, New York, 2005), pp. 93–106
16. R. Cerulli, P. Dell’Olmo, M. Gentili, A. Raiconi, Heuristic approaches for the minimum labelling hamiltonian cycle problem. *Electronic Notes Discret. Math.* **25**, 131–138 (2006). *CTW2006 – Cologne-Twente Workshop on Graphs and Combinatorial Optimization*
17. R. Cerulli, A. Fink, M. Gentili, S. Voß, Extensions of the minimum labeling spanning tree problem. *J. Telecommun. Inf. Technol.* **4**, 39–45 (2006)
18. R. Cerulli, M. Gentili, A. Iossa, Efficient preflow push algorithms. *Comput. Oper. Res.* **35**, 2694–2708 (2008)
19. R.S. Chang, S.J. Leu, The minimum labeling spanning trees. *Inf. Process. Lett.* **63**(5), 277–282 (1997)
20. J. Cheriyan, K. Mehlhorn, An analysis of the highest-level selection rule in the preflow-push max-flow algorithm. *Inf. Process. Lett.* **69**, 239–242 (1999)
21. R.V. Cherkasky, An algorithm for constructing maximal flows in networks with complexity of $o(v^2\sqrt{E})$ operations. *Math. Methods Solut. Econ. Probl.* **7**, 112–125 (1977) (In Russian)
22. S. Consoli, J.A. Moreno-Perez, K. Darby-Dowman, N. Mladenovic, Discrete particle swarm optimization for the minimum labeling steiner tree problem, in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2007)*, ed. by N. Krasnogor, G. Nicosia, M. Pavone, D. Pelta (Springer, Berlin, 2008), pp. 313–322
23. S. Consoli, K. Darby-Dowman, N. Mladenovic, J.A. Moreno-Perez, Greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem. *Eur. J. Oper. Res.* **196**, 440–449 (2009)
24. S. Consoli, K. Darby-Dowman, N. Mladenovic, J.A. Moreno-Perez, Variable neighbourhood search for the minimum labelling steiner tree problem. *Ann. Oper. Res.* **172**, 71–96 (2009)
25. S. Consoli, J.A. Moreno-Perez, K. Darby-Dowman, N. Mladenovic, Discrete particle swarm optimization for the minimum labeling steiner tree problem. *Nat. Comput.* **9**, 29–46 (2010)
26. B. Couëtoux, L. Gourvès, J. Monnot, O.A. Telelis, Labeled traveling salesman problems: complexity and approximation. *Discret. Optim.* **7**(1–2), 74–85 (2010)

27. E.A. Dinic, Algorithm for solution of a problem of maximum flow in networks with power estimation. Sov. Math. Dokl. **II**, 1277–1280 (1970)
28. J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM **19**, 248–264, (1972)
29. S. Even, *Graph Algorithms*. Computer Science Press, Potomac (1979)
30. M.R. Fellows, J. Guo, I.A. Kanj, The parameterized complexity of some minimum label problems, in *Graph-Theoretic Concepts in Computer Science*, ed. by C. Paul, M. Habib (Springer, Berlin/New York, 2009), pp. 88–99
31. L.R. Ford, D.R. Fulkerson, Maximal flow through a network. Can. J. Math. **8**, 399–404 (1956)
32. L.R. Ford, D.R. Fulkerson, A simple algorithm for finding maximal network flows and an application to the hitchcock problem. Can. J. Math. **9**, 210–218 (1957)
33. L.R. Ford, D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, 1962)
34. G. Galbiati, The complexity of a minimum reload cost diameter problem. Discret. Appl. Math. **156**(18), 3494–3497 (2008)
35. Z. Galil, An $\text{O}(n^{5/3}m^{2/3})$ algorithm for the maximal flow problem. Acta Inform. **14**, 221–242 (1980)
36. Z. Galil, A. Naamad, An $\text{O}(nm(\log n)^2)$ algorithm for the maximal flow problem. J. Comput. Syst. Sci. **21**, 203–217 (1980)
37. A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem. J. ACM **35**, 921–940 (1988)
38. A.V. Goldberg, E. Tardos, R.E. Tarjan, Network flow algorithms, in *Flows, Paths and VLSI*, ed. by B. Korte, L. Lovasz, H.J. Promel, A. Schrijver (Springer, Berlin, 1990), pp. 101–164 (1990)
39. R. Hassin, J. Monnot, D. Segev, Approximation algorithms and hardness results for labeled connectivity problem. J. Comb. Optim. **14**, 437–453 (2007)
40. N. Jozefowicz, G. Laporte, F. Semet, A branch-and-cut algorithm for the minimum labeling hamiltonian cycle problem and two variants. Comput. Oper. Res. **38**, 1534–1542 (2011)
41. A.V. Karzanov, Determining the maximal flow in a network by the method of preflows. Sov. Math. Dokl. **15**, 434–437 (1974)
42. S.O. Krumke, H.-C. Wirth, On the minimum label spanning tree problem. Inf. Process. Lett. **66**, 81–85 (1998)
43. E. Lawler, *Combinatorial Optimization: Networks and Matroids* (Holt, Reinhart, and Winston, New York, 1976)
44. S.N. Maheshwari, J. Cheriyan, Analysis of preflow push algorithms for maximum network flow. SIAM J. Comput. **18**(6), 1057–1086 (1989)
45. G. Mazzoni, S. Pallottino, M.G. Scutellá, The maximum flow problem: a max-preflow approach. Eur. J. Oper. Res. **53**, 257–278 (1991)
46. N. Mladenović, P. Hansen, Variable neighborhood search. Comput. Oper. Res. **24**(11), 1097–1100 (1997)
47. N. Mladenović, P. Hansen, Variable neighbourhood search: principles and applications. Eur. J. Oper. Res. **130**, 449–467 (2001)
48. N. Mladenović, P. Hansen, Variable neighbourhood search. Eur. J. Oper. Res. **130**(11), 145–185 (2003)
49. N. Mladenović, P. Hansen, B. Jaumard, A. Parreira, Variable neighborhood search for weighted maximum satisfiability problem. Les Cahiers du GERAD G-2000-62, Montréal, 2000
50. J. Monnot, The labeled perfect matching in bipartite graphs. Inf. Process. Lett. **96**(3), 81–88 (2005)
51. J. Monnot, A note on the hardness results for the labeled perfect matching problems in bipartite graphs. RAIRO **42**, 315–324 (2008)
52. C.H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Dover, Mineola, 1998)
53. J.C. Picard, H.D. Ratliff, Minimum cuts and related problems. Networks **5**, 357–370 (1975)
54. Y. Shiloach, An $\text{O}(nI(\log^2 I))$ maximum-flow algorithm. Technical report, STAN-CS-78-802, Computer Science Department, Stanford University, Stanford, 1978

55. D.D. Sleator, An $O(nm \log n)$ algorithm for maximum network flow. Technical report, STAN-CS-80-831, Computer Science Department, Stanford University, Stanford, 1980
56. D.D. Sleator, R.E. Tarjan, A data structure for dynamic trees. *J. Comput. Syst. Sci.* **26**, 362–391 (1983)
57. J. Steffan, H.-C. Wirth, Reload cost problems: minimum diameter spanning tree. *Discret. Appl. Math.* **113**(1), 73–85 (2001)
58. S. Szeider, Finding paths in graphs avoiding forbidden transitions. *Discret. Appl. Math.* **126**, 261–273 (2003)
59. R.E. Tarjan, *Data Structures and Network Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, 1983)
60. Y. Wan, G. Chen, Y. Xu, A note on the minimum label spanning tree. *Inf. Process. Lett.* **84**, 84–99 (2002)
61. Y. Xiong, B. Golden, E. Wasil, A one-parameter genetic algorithm for the minimum labeling spanning tree problem. *IEEE Trans. Evolut. Comput.* **9**(1), 55–60 (2005)
62. Y. Xiong, B. Golden, E. Wasil, Worst-case behavior of the MVCA heuristic for the minimum labeling spanning tree problem. *Oper. Res. Lett.* **33**(1), 77–80 (2005)
63. Y. Xiong, B. Golden, E. Wasil, Improved heuristics for the minimum labelling spanning tree problem. *IEEE Trans. Evolut. Comput.* **10**(6), 700–703 (2006)
64. Y. Xiong, B. Golden, E. Wasil, The colorful traveling salesman problem, in *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, ed. by E.K. Baker et al. (Springer, New York, 2007), pp. 115–123
65. Y. Xiong, B. Golden, E. Wasil, S. Chen, The label-constrained minimum spanning tree problem, in *Telecommunications Modeling, Policy, and Technology*, ed. by S. Raghavan, B.L. Golden, E.A. Wasil (Springer, New York, 2008), pp. 39–50
66. S. Yuan, S. Varma, J.P. Jue, Minimum-color path problems for reliability in mesh networks, in INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 4, Miami, 2005, pp. 2658–2669
67. P. Zhang, J.-Y. Cai, L.-Q. Tang, W.-B. Zhao, Approximation and hardness results for label cut and related problem. *J. Comb. Optim.* **21**, 192–208 (2011)
68. U. Zwick, The smallest networks on which the ford-fulkerson maximum flow procedure may fail to terminate. *Theor. Comput. Sci.* **148**, 165–170 (1995)

Modern Network Interdiction Problems and Algorithms

J. Cole Smith, Mike Prince and Joseph Geunes

Contents

1	Introduction	1950
2	Classical Interdiction Study	1951
2.1	General Problem Descriptions	1952
2.2	Applications	1964
3	Emerging Areas of Interdiction Research	1970
3.1	Fortification Problems	1970
3.2	Stochastic Interdiction	1972
3.3	Asymmetric Information	1973
3.4	Multi-objective Interdiction	1974
4	Competition, Interdiction, and Optimization Models	1975
4.1	Industrial Competition as an Interdiction Problem	1975
4.2	Optimization Models for Competition with Interdiction	1977
5	Connection to Robust Optimization and Survivable Network Design	1979
5.1	Robust Optimization	1980
5.2	Survivable Network Design	1983
6	Conclusion	1984
	Cross-References	1984
	Recommended Reading	1984

Abstract

A network interdiction problem usually involves two players who compete in a min–max or max–min game. One player, the network owner, tries to optimize its objective over the network, for example, as measured by a shortest path, maximum flow, or minimum cost flow. The opposing player, called the interdictor, alters the owner’s network to maximally impair the owner’s objective (e.g., by destroying arcs that maximize the owner’s shortest path). This chapter

J.C. Smith (✉) • M. Prince • J. Geunes

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: cole@ise.ufl.edu; mikeprince4@ufl.edu; geunes@ise.ufl.edu

summarizes the impressive recent development of this field. The first part of this chapter emphasizes interdiction foundations, applications, and emerging research areas. The links between this field and business competition models are then developed, followed by a comparison of interdiction research with parallel developments in robust optimization and survivable network design.

1 Introduction

This chapter discusses classical and emerging network interdiction problems, along with two- and three-stage mathematical programming models and solution techniques that are often employed to solve these problems. It is useful to think of these problems in terms of defenders and attackers of a network. A defender (sometimes called an “operator” or an “owner” of the network) may be a player who wishes to protect the network from damage or optimally push flows across the network using the infrastructure that exists. An attacker (“enemy,” “interdictor,” or “adversary” is also used) seeks to disrupt the defender’s objective to the maximum extent possible. Hence, the attacker may seek to optimally impair the defender’s infrastructure to limit the defender’s overall utility from the network.

For instance, as discussed later in this chapter, a classical (though stylized) example may consist of a defender that builds a pipeline network with the intention of pushing as much flow as possible from an origin to a destination node. The attacker may attempt to destroy a strategic set of links (pipes connecting junction points), which would then minimize the defender’s maximum flow on the resulting network. As such, optimization models for this class of problems are sometimes referred to as *minimax* optimization models. Similarly, *maximin* models exist in which, for example, the attacker may wish to maximize the (optimal) cost that the defender must incur in operating its network.

Most of the standard interdiction problems that have been proposed in the literature regard Stackelberg games [65], wherein the two entities (i.e., a leader and a follower) operate in turn with full knowledge of each other’s actions. In the example given above, the defender is presumably aware of which links have been attacked in the pipeline and adjusts flow accordingly. This assumption can be rather strong, though, and several emerging research areas have targeted problems that relax this assumption within a framework where both players act simultaneously.

The role of “defender” and “attacker” is sometimes given in the context of security and antiterrorism applications, where the attacker plays the role of an adversary. However, this viewpoint is narrow with respect to the overall scope of interdiction research. For instance, it is common to model an attacker as a natural disaster such as a hurricane or earthquake, an accidental failure of infrastructure, or uncertainty about resource availability. None of these entities is, literally speaking, an intelligent and optimizing force that seeks to impede a defender. However, a defender wishing to protect itself in the worst case (restricted to whatever interdiction resources have been allocated to the attacker) must treat accidents as if they are malicious in nature. Hence, an analyst may wish to adopt the philosophy

that the worst possible event will transpire in order to guarantee a minimum level of network performance.

The defender may additionally be capable of combatting an attacker by fortifying its network infrastructure or when that is not practically achievable, installing enough redundancy in the network so that no small simultaneous collection of failures will substantially cripple the network's functionality. One can thus consider a three-stage "defender–attacker–defender" game in which the players act sequentially (again with full knowledge of each other's actions) in a Stackelberg fashion. These problems take the form of min–max–min (or max–min–max) games and are typically substantially more difficult than min–max or max–min games. The fortification actions that take place can be broadly conceived and may involve the fortification or design decisions mentioned above, a reduction of the attacker's resources, deception regarding the defender's infrastructure, and so on.

There are several additional survey treatments of interdiction in the literature. An excellent starting point was provided by Brown et al. [16, 17] in survey and tutorial articles that explain the breadth of interdiction actions, particularly from the standpoint of security. See also [55] for an introductory treatment accessible to beginning optimization researchers and [57] for a recent discussion of the interdiction area.

[Section 2](#) presents some of the founding research principles behind network interdiction, along with key examples that demonstrate common mathematical programming approaches for solving these problems. Emerging areas of interdiction are discussed in [Sect. 3](#). These areas include consideration of uncertain defense/attack strategies, uncertain or differing perceptions of network data, multi-objective interdiction, and fractional interdiction problems where defense/attack actions can be applied at intermediate levels of effectiveness. Applications of these problems pertaining to competition outside traditional interdiction application domains are presented in [Sect. 4](#). The chapter then examines robust optimization theory and survivable network design techniques in [Sect. 5](#) and compares their philosophies to that found in network interdiction studies. Finally, the chapter concludes in [Sect. 6](#).

2 Classical Interdiction Study

This section describes four general types of interdiction problems: shortest path, maximum flow, facility interdiction, and minimum cost flow. These problems are defined on a graph $G(N, A)$, where N represents the vertex set and A represents a set of directed arcs. For each arc $(i, j) \in A$, let c_{ij} represent a per-unit flow cost, and let u_{ij} represent the capacity of arc (i, j) . The *forward star* of node i is the set of arcs that leave node i . The *reverse star* of node i is the set of nodes that enter node i . Formally, the forward star $FS(i) = \{j \in N : (i, j) \in A\}$, and the reverse star $RS(i) = \{j \in N : (j, i) \in A\}$, for all $i \in N$.

In Sect. 2.1, each of the four problem types is described, formulated, and then illustrated with a numerical example. Section 2.2 includes various applications for interdiction problems.

2.1 General Problem Descriptions

First, the general formulation of a *minimax* network interdiction problem is given as follows:

$$\min_{\mathbf{x} \in \mathbf{X}} \max \mathbf{p}(\mathbf{x})^\top \mathbf{y} \quad (1a)$$

$$\text{s.t. } \mathbf{Dy} \leq \mathbf{r}(\mathbf{x}) \quad (1b)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (1c)$$

where \mathbf{x} and \mathbf{y} are the leader and follower variables, respectively, and where $\mathbf{p}(\mathbf{x})$ and $\mathbf{r}(\mathbf{x})$ represent the follower's flow profits and available resources, respectively, as a function of the first-stage decisions. The \mathbf{y} -vector usually represents the flow, and $\mathbf{p}(\mathbf{x})$ and $\mathbf{r}(\mathbf{x})$ may or may not depend on the decisions made by the attacker. The constraints $\mathbf{Dy} \leq \mathbf{r}(\mathbf{x})$ usually consist of flow conservation and capacity constraints. The attacker's feasible region, \mathbf{X} , is typically of the form $\mathbf{X} = \{\mathbf{x} : f_i(\mathbf{x}) \leq b_i \text{ for } i = 1, \dots, m; \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}\}$ where m is the number of constraints, f_i is a function (usually linear) of \mathbf{x} , and \mathbf{e} is a vector of ones. For instance, \mathbf{X} often consists of a single knapsack constraint, for example, an interdiction budget. The attacker is also commonly restricted to making binary decisions in which they must fully attack an arc, or not attack it at all.

2.1.1 Shortest Path Interdiction

The shortest path problem seeks a path from a source node s to a destination node t such that the sum of the arc weights on the path is minimized. In the shortest path interdiction problem, the attacker seeks to maximize the defender's shortest path cost on the network by increasing the cost of some arcs $(i, j) \in A$ from c_{ij} to $c_{ij} + d_{ij}$. Indeed, this construct permits the development of models in which either (a) the cost (or time) to traverse an arc is actually increased by d_{ij} or (b) arc (i, j) is completely destroyed by making d_{ij} a prohibitively large number that prevents traversal of (i, j) at optimality.

Define $N_0 = N \setminus \{s, t\}$ to be the set of intermediate nodes. The shortest path interdiction problem is formulated as follows:

$$\max_{\mathbf{x} \in \mathbf{X}} \min \sum_{(i,j) \in A} (c_{ij} + d_{ij} x_{ij}) y_{ij} \quad (2a)$$

$$\text{s.t. } \sum_{j \in FS(i)} y_{ij} - \sum_{j \in RS(i)} y_{ji} = \begin{cases} 1 & \text{for } i = s, \\ -1 & \text{for } i = t, \\ 0 & \text{for } i \in N_0 \end{cases} \quad (2b)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (2c)$$

where \mathbf{X} again constrains the leader decisions. In the objective function, the cost of arc (i, j) is increased by $d_{ij}x_{ij}$. The constraints in (2b) represent the flow conservation constraints. In order to efficiently solve this two-stage problem, the dual of the minimization problem can be used to combine the problems into a single maximization problem:

$$\max \pi_s - \pi_t \quad (3a)$$

$$\text{s.t. } \pi_i - \pi_j - d_{ij}x_{ij} \leq c_{ij} \quad \forall (i, j) \in A \quad (3b)$$

$$\mathbf{x} \in \mathbf{X}, \boldsymbol{\pi} \text{ unrestricted.} \quad (3c)$$

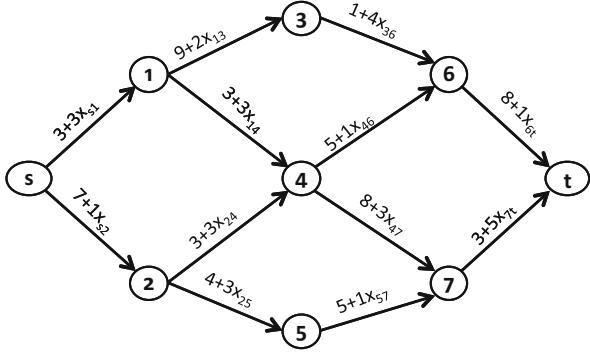
This problem is a linear program so long as \mathbf{X} contains only linear constraints on the \mathbf{x} -variables. However, the literature predominantly regards applications in which the interdictor must destroy all or none of an arc, which require the imposition of binary restrictions on the \mathbf{x} -variables.

Example 1 An evader seeks escape from a starting point s to a safe destination point t in a network, where practically speaking, t may represent a dummy node that represents the union of all safe points in a particular region. A police force seeks to maximize the evader's shortest path by blocking roads. In this example, the police force is the interdicting entity. Suppose that the police force is capable of blocking four roads, subject to some constraints, and that the decisions are binary: a roadblock can either be placed on a road or not placed. Moreover, a road block is effective in one direction on a road, which is equivalent to blocking one (directed) arc in a network. The road blocks serve to delay the evader's escape route as depicted in Fig. 1, which shows the shortest path network (with travel times labeled on each arc).

This shortest path interdiction problem is given as follows:

$$\begin{aligned} \max_{\mathbf{x} \in \mathbf{X}} \min & (3 + 3x_{s1})y_{s1} + (7 + 1x_{s2})y_{s2} + (9 + 2x_{13})y_{13} + (3 + 3x_{14})y_{14} \\ & +(3 + 3x_{24})y_{24} + (4 + 3x_{25})y_{25} + (1 + 4x_{36})y_{36} + (5 + 1x_{46})y_{46} \\ & +(8 + 3x_{47})y_{47} + (5 + 1x_{57})y_{57} + (8 + 1x_{67})y_{67} \\ & +(3 + 5x_{7y})y_{7y} \end{aligned} \quad (4a)$$

$$\text{s.t. } y_{s1} + y_{s2} = 1 \quad (4b)$$

Fig. 1 Shortest path network

$$-y_{s1} + y_{13} + y_{14} = 0 \quad (4c)$$

$$\vdots \quad (4d)$$

$$-y_{6t} - y_{7t} = -1 \quad (4e)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (4f)$$

where $\mathbf{X} = \{\mathbf{x} : x_{s1} + x_{s2} \leq 1, x_{13} + x_{14} + x_{24} + x_{25} \leq 1, x_{36} + x_{46} + x_{47} + x_{57} \leq 1, x_{6t} + x_{7t} \leq 1, \mathbf{x} \in \{0, 1\}^{|A|}\}$. If the police did not set up any road blocks, the optimal path would be $s - 1 - 4 - 7 - t$ with a travel time of 17. After taking the dual of the inner minimization problem, the inner and outer problems can be combined into a single maximization problem, resulting in the following formulation:

$$\max \pi_s - \pi_t \quad (5a)$$

$$\text{s.t. } \pi_s - \pi_1 - 3x_{s1} \leq 3 \quad (5b)$$

$$\pi_s - \pi_2 - 1x_{s2} \leq 7 \quad (5c)$$

$$\vdots \quad (5d)$$

$$\pi_7 - \pi_t - 5x_{7t} \leq 3 \quad (5e)$$

$$x_{s1} + x_{s2} \leq 1 \quad (5f)$$

$$x_{13} + x_{14} + x_{24} + x_{25} \leq 1 \quad (5g)$$

$$x_{36} + x_{46} + x_{47} + x_{57} \leq 1 \quad (5h)$$

$$x_{6t} + x_{7t} \leq 1 \quad (5i)$$

$$\mathbf{x} \in \{0, 1\}^{|A|}. \quad (5j)$$

In an optimal solution, the police interdict arcs $(s, 1)$, $(1, 4)$, $(4, 6)$, and $(7, t)$, which results in a shortest travel time of 24 by changing the shortest path to $s - 1 - 3 - 6 - t$. \square

Remark 1 Note that the evader problem described here is a simplified version of the nuclear smuggling interdiction problem (see, e.g., [40, 43]). In this problem, there exist a probability a_{ij} that an evader can escape undetected through the network if arc (i, j) is not monitored and a probability b_{ij} that the evader escapes if the arc is monitored, where $0 < b_{ij} < a_{ij} < 1$. In this case, the evader seeks a *longest* path, but where this longest path distance is given by

$$\prod_{(i,j) \in A} (a_{ij} - (a_{ij} - b_{ij})x_{ij})^{y_{ij}},$$

where once again, x_{ij} is the degree to which arc (i, j) is interdicted, and y_{ij} is a flow variable that equals 1 if the evader uses arc (i, j) in its evasion path, and $y_{ij} = 0$ otherwise.

One can equivalently maximize the natural log of this function instead (noting that the natural log of a function is increasing over the domain of positive numbers). This problem's objective is given by

$$\begin{aligned} \ln \left(\prod_{(i,j) \in A} (a_{ij} - (a_{ij} - b_{ij})x_{ij})^{y_{ij}} \right) &= \sum_{(i,j) \in A} \ln ((a_{ij} - (a_{ij} - b_{ij})x_{ij})^{y_{ij}}) \\ &= \sum_{(i,j) \in A} \ln (a_{ij} - (a_{ij} - b_{ij})x_{ij}) y_{ij}, \end{aligned}$$

where each coefficient of y_{ij} in the latter expression is negative, by the assumption that $0 < a_{ij} - (a_{ij} - b_{ij})x_{ij} < 1$ for each $(i, j) \in A$ (and because $\ln(q) < 0$ for $0 < q < 1$). The optimization challenge of the evader is therefore a shortest path problem with positive costs. This exercise demonstrates that the more practical version of the evader-interdiction problem in which one minimizes the probability of escape is exactly transformable to the shortest path interdiction problem covered above. \square

2.1.2 Maximum Flow Interdiction

The maximum flow problem seeks to maximize the flow from node s to node t on a capacitated network. In the maximum flow interdiction problem, the interdictor's goal is to minimize the operator's maximum flow. Letting u_{ij} denote the capacity of arc $(i, j) \in A$ and adding a dummy arc (t, s) with $u_{ts} = \infty$, the problem is formulated as follows:

$$\min_{\mathbf{x} \in \mathbf{X}} \max_{\mathbf{y} \in \mathbf{Y}} y_{ts} \quad (6a)$$

$$\text{s.t. } \sum_{j \in FS(i)} y_{ij} - \sum_{j \in RS(i)} y_{ji} = 0, \quad \forall i \in N \quad (6b)$$

$$y_{ij} \leq u_{ij}(1 - x_{ij}) \quad \forall (i, j) \in A \setminus \{(t, s)\} \quad (6c)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (6d)$$

where \mathbf{x} is the interdictor's decision vector and \mathbf{y} is the flow vector. Constraints (6b) are the flow conservation constraints, (6c) are the capacity constraints, which are a function of the interdictor's decisions, and constraints (6d) are the nonnegativity restrictions. Since the inner maximization problem is always feasible (a zero flow is always feasible), the two-stage problem can be combined into a single minimization problem by taking the dual of the inner problem. Letting α and β be dual variables corresponding to (6b) and (6c), respectively, we combine the two stages to obtain the following formulation:

$$\min \sum_{(i, j) \in A \setminus \{(t, s)\}} u_{ij}(1 - x_{ij})\beta_{ij} \quad (7a)$$

$$\text{s.t. } \alpha_i - \alpha_j + \beta_{ij} \geq 0 \quad \forall (i, j) \in A \setminus \{(t, s)\} \quad (7b)$$

$$\alpha_t - \alpha_s \geq 1 \quad (7c)$$

$$\beta \geq \mathbf{0} \quad (7d)$$

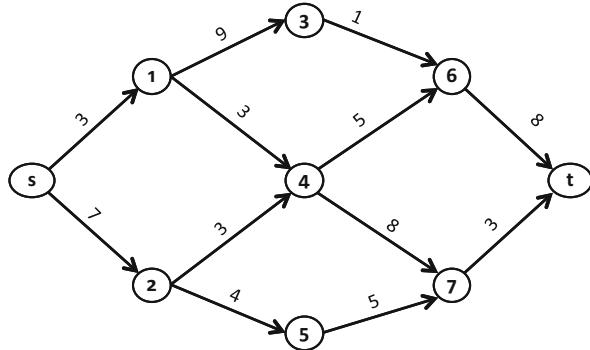
$$\mathbf{x} \in \mathbf{X}. \quad (7e)$$

Unlike the shortest path interdiction problem, this formulation is nonlinear. However, the dual variables β can be further restricted to be binary, because changing the capacity constraints by one will only change the maximum flow by either zero or one unit (i.e., β_{ij} is zero or one, $\forall (i, j) \in A \setminus \{(t, s)\}$). This allows a standard linearization technique to be used in which $x_{ij}\beta_{ij}$ is replaced by a single variable γ_{ij} and the following constraints are added: $\gamma_{ij} \leq x_{ij}$, $\gamma_{ij} \leq \beta_{ij}$, $\gamma_{ij} \geq x_{ij} + \beta_{ij} - 1$, and $\gamma_{ij} \geq 0$. In fact, only the first two constraints are needed because the bilinear terms appear in the objective function with coefficients of -1 .

Example 2 Consider a military conflict in which one of the sides (the “transporter”) has a goal of sending as much supply as possible from its base (denoted by s) to its front lines (denoted by t). The transporter has smaller, intermediate bases at which it can stop on the way to the front lines, but a limited amount of supply can be sent between each pair of points. This network is represented in Fig. 2 with the arc capacities labeled on each arc.

The opposition (interdictor) is capable of disrupting the transporter's flow by removing any arc in the network (e.g., by destroying a bridge). The goal of the

Fig. 2 Maximum flow network



interdictor is to minimize the transporter's maximum flow, and their optimization problem can be formulated as follows:

$$\min_{\mathbf{x} \in \mathbf{X}} \max_{\mathbf{y} \in \mathbf{Y}} y_{ts} \quad (8a)$$

$$\text{s.t. } y_{s1} + y_{s2} - y_{ts} = 0 \quad (8b)$$

$$\vdots \quad (8c)$$

$$-y_{6t} - y_{7t} + y_{ts} = 0 \quad (8d)$$

$$y_{s1} \leq 3(1 - x_{s1}) \quad (8e)$$

$$\vdots \quad (8f)$$

$$y_{7t} \leq 3(1 - x_{7t}) \quad (8g)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (8h)$$

where $\mathbf{X} = \{\mathbf{x} : \sum_{(i,j) \in A} x_{ij} \leq 1, \mathbf{x} \in \{0, 1\}^{|A|}\}$. Without interdiction, the maximum flow is 9 units, which is obtained by sending one unit of flow on the path $s - 1 - 3 - 6 - t$, two units of flow on the path $s - 1 - 4 - 6 - t$, and three units of flow on paths $s - 2 - 4 - 6 - t$ and $s - 2 - 5 - 7 - t$. Taking the dual of the maximization problem and combining the two-stage problem into a single minimization problem gives the following:

$$\begin{aligned} \min & \quad 3(1 - x_{s1})\beta_{s1} + 7(1 - x_{s2})\beta_{s2} + 9(1 - x_{13})\beta_{13} + 3(1 - x_{14})\beta_{14} \\ & + 3(1 - x_{24})\beta_{24} + 4(1 - x_{25})\beta_{25} + 1(1 - x_{36})\beta_{36} + 5(1 - x_{46})\beta_{46} \\ & + 8(1 - x_{47})\beta_{47} + 5(1 - x_{57})\beta_{57} + 8(1 - x_{6t})\beta_{6t} + 3(1 - x_{7t})\beta_{7t} \end{aligned} \quad (9a)$$

$$\text{s.t. } \alpha_s - \alpha_1 + \beta_{s1} \geq 0 \quad (9b)$$

$$\vdots \quad (9c)$$

$$\alpha_7 - \alpha_t + \beta_{7t} \geq 0 \quad (9d)$$

$$\alpha_t - \alpha_s \geq 1 \quad (9e)$$

$$\beta \geq \mathbf{0} \quad (9f)$$

$$\mathbf{x} \in \mathbf{X}. \quad (9g)$$

Using the linearization technique discussed above, the following linear mixed-integer program is obtained:

$$\min 3(\beta_{s1} - \gamma_{s1}) + 7(\beta_{s2} - \gamma_{s2}) + 9(\beta_{13} - \gamma_{13}) + 3(\beta_{14} - \gamma_{14}) \quad (10a)$$

$$+ 3(\beta_{24} - \gamma_{24}) + 4(\beta_{25} - \gamma_{25}) + 1(\beta_{36} - \gamma_{36}) + 5(\beta_{46} - \gamma_{46}) \quad (10b)$$

$$+ 8(\beta_{47} - \gamma_{47}) + 5(\beta_{57} - \gamma_{57}) + 8(\beta_{6t} - \gamma_{6t}) + 3(\beta_{7t} - \gamma_{7t}) \quad (10c)$$

$$\text{s.t. } \gamma_{s1} \leq x_{s1} \quad (10d)$$

$$\gamma_{s1} \leq \beta_{s1} \quad (10e)$$

$$\gamma_{s2} \leq x_{s2} \quad (10f)$$

$$\gamma_{s2} \leq \beta_{s2} \quad (10g)$$

$$\vdots \quad (10h)$$

$$\gamma_{7t} \leq x_{7t} \quad (10i)$$

$$\gamma_{7t} \leq \beta_{7t} \quad (10j)$$

$$\beta \text{ binary} \quad (10k)$$

$$(9b) - (9g). \quad (10l)$$

The optimal solution for the interdictor is to interdict arc (4, 6), which reduces the maximum flow to 4 units. \square

2.1.3 Facility Assignment Interdiction

In the facility assignment problem, a set of facilities is responsible for providing some good or service to a set of demand points. The goal is to minimize the weighted distance required to supply each demand point, where the weight of each demand point may be some function of the amount of demand at the node, or the relative importance of supplying demand to that node. Assuming that the capacity of each facility is large enough to satisfy the total demand of any subset of demand points, an optimal solution is obtained by assigning each demand point to its closest facility. In the facility assignment interdiction problem, the goal is to find a subset of r facilities, which when removed, maximizes the minimum weighted facility assignment objective.

The model for this interdiction problem was introduced by Church et al. [20]. Let N denote the set of demand points (indexed by i), F denote the set of p facilities (indexed by j), a_i denote the demand at node i , d_{ij} denote the distance between facility j and node i , r denote the number of facilities to be interdicted or eliminated, and $T_{ij} = \{k \in F : d_{ik} > d_{ij}\}$ be the set of existing facilities that are farther than j from demand i . The decision variables include interdiction variables x_j , $\forall j \in F$, and assignment variables y_{ij} , $\forall i \in N$, $j \in F$. The variable x_j equals 1 if facility j is interdicted, and 0 otherwise, and y_{ij} equals 1 if demand i is assigned to facility j , and 0 otherwise. The facility assignment interdiction formulation is given as follows:

$$\max \sum_{i \in N} \sum_{j \in F} a_i d_{ij} y_{ij} \quad (11a)$$

$$\text{s.t. } \sum_{j \in F} y_{ij} = 1 \quad \forall i \in N \quad (11b)$$

$$\sum_{j \in F} x_j = r \quad (11c)$$

$$\sum_{k \in T_{ij}} y_{ik} \leq x_j \quad \forall i \in N, j \in F \quad (11d)$$

$$y_{ij} \in \{0, 1\} \quad \forall i \in N, j \in F \quad (11e)$$

$$x_j \in \{0, 1\} \quad \forall j \in F. \quad (11f)$$

The objective function (11a) is the sum of the weighted distances of the facility-demand point assignments. Constraint (11b) requires each demand point to be assigned to a facility, and constraint (11c) requires r facilities be interdicted. Constraint (11d) ensures that demand i is assigned to the closest non-interdicted facility. Constraints (11e) and (11f) give the binary restrictions. It is important to note that if the x -variables are binary, then the y -variables are binary at optimality (given a binary x , there exists an optimal solution in which each demand is fully assigned to its closest non-interdicted facility). Therefore, it is possible to reduce the number of binary variables to p —one for each facility—while equivalently relaxing (11e) to $y_{ij} \geq 0$, $\forall i \in N$, $j \in F$ (with the unitary upper bounds on y_{ij} implied by (11b)).

Example 3 An army has five primary bases that must be assigned to cover nine remote outposts. For simplicity, suppose that each outpost is assigned to the closest primary base and that the effectiveness of the army's operations is given by the sum of base-to-outpost distances. To assess the vulnerability of their coverages, the army would like to know which two large bases, if destroyed, would lead to the largest increase in minimum base-to-outpost distance assignments. Figure 3 shows the network representation along with the d -values and the optimal solution for the case in which no interdiction occurs (resulting an objective function value

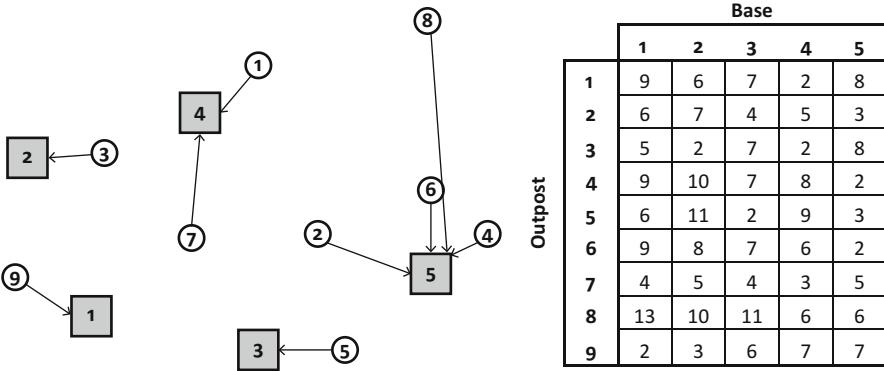


Fig. 3 Facility location example

of 24). For this particular case, $a_i = 1, \forall i = 1, \dots, 9$. The following formulation corresponds to the illustration in Fig. 3, in which $r = 2$:

$$\max \sum_{i \in N} \sum_{j \in F} d_{ij} y_{ij} \quad (12a)$$

$$\text{s.t. } y_{11} + y_{12} + y_{13} + y_{14} + y_{15} = 1 \quad (12b)$$

⋮

$$y_{91} + y_{92} + y_{93} + y_{94} + y_{95} = 1 \quad (12c)$$

$$x_1 + x_2 + x_3 + x_4 + x_5 = 2 \quad (12d)$$

$$0 \leq x_1 \quad (12e)$$

$$y_{11} + y_{13} + y_{15} \leq x_2 \quad (12f)$$

$$y_{11} + y_{15} \leq x_3 \quad (12g)$$

$$y_{11} + y_{12} + y_{13} + y_{15} \leq x_4 \quad (12h)$$

$$y_{11} \leq x_5 \quad (12i)$$

⋮

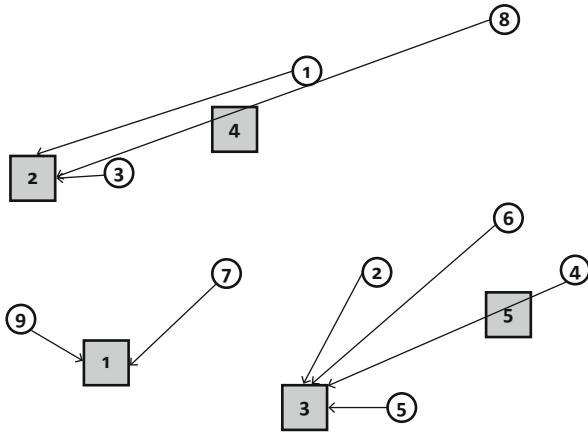
$$y_{92} + y_{93} + y_{94} + y_{95} \leq x_1 \quad (12j)$$

$$y_{93} + y_{94} + y_{95} \leq x_2 \quad (12k)$$

$$y_{94} + y_{95} \leq x_3 \quad (12l)$$

$$0 \leq x_4 \quad (12m)$$

Fig. 4 Facility location solution



$$0 \leq x_5 \quad (12n)$$

$$y_{ij} \geq 0 \quad \forall i = 1, \dots, 9, j = 1, \dots, 5 \quad (12o)$$

$$x_j \in \{0, 1\} \quad \forall j = 1, \dots, 5. \quad (12p)$$

The optimal solution has $x_4 = x_5 = 1$, resulting in an objective function value of 44. [Figure 4](#) shows the revised assignments after the interdiction.

2.1.4 Minimum Cost Flow Interdiction

The minimum cost flow problem seeks to satisfy all node demand requirements in a network at minimum cost and subsumes the cases discussed prior to now. In the minimum cost flow interdiction problem, the interdictor's goal is to maximize the follower's minimum flow cost by reducing or eliminating the capacity on some subset of arcs. Letting d_i be the supply (when positive) or demand (when negative) of node $i \in N$, the minimum cost flow interdiction model is formulated as follows:

$$\max_{\mathbf{x} \in \mathbf{X}} \min \sum_{(i,j) \in A} c_{ij} y_{ij} \quad (13a)$$

$$\text{s.t. } \sum_{j \in FS(i)} y_{ij} - \sum_{j \in RS(i)} y_{ji} = d_i \quad \forall i \in N \quad (13b)$$

$$y_{ij} \leq u_{ij}(1 - x_{ij}) \quad \forall (i, j) \in A \quad (13c)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (13d)$$

where $\sum_{i \in N} d_i = 0$. The goal for the follower is to minimize cost while satisfying the flow conservation constraints (13b) and capacity constraints (13c).

Let α and $-\beta$ be the dual variables corresponding to (13b) and (13c), respectively. The dual formulation to (13) is

$$\max \sum_{i \in N} d_i \alpha_i - \sum_{(i,j) \in A} u_{ij} (1 - x_{ij}) \beta_{ij} \quad (14a)$$

$$\text{s.t. } \alpha_i - \alpha_j - \beta_{ij} \leq c_{ij} \quad \forall (i, j) \in A \quad (14b)$$

$$\beta \geq \mathbf{0} \quad (14c)$$

$$\mathbf{x} \in \mathbf{X}. \quad (14d)$$

The form of the objective function, (14a), leads to a nonlinear optimization problem that seeks to optimally interdict a minimum cost flow.

Remark 2 Unfortunately, it is not generally possible to restrict β to be binary as in the maximum flow interdiction problem because the change in the objective function value as a function of capacity is dependent on the cost vector \mathbf{c} . If \mathbf{x} is binary, then linearization is still possible with suitable upper bounds on β . Moreover, in many cases, \mathbf{X} is given by a set of constraints $\{\mathbf{x} : P\mathbf{x} = \mathbf{q}, \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}\}$. If P is a totally unimodular matrix and \mathbf{q} is integral (see e.g., [4]), then in fact an optimal solution exists in which the \mathbf{x} -variables take binary values. This fact is due to the observation that, given values of α and β , problem (14) is a linear program, and an optimal solution would exist at an extreme point of \mathbf{X} . Because the extreme points of \mathbf{X} would be all binary (due to the total unimodularity of P and integrality of \mathbf{q}), \mathbf{x} is binary in some optimal solution. \square

Example 4 A company has three production facilities and three retail facilities. The company's goal is to satisfy the demand at the retail facilities with the supply from the production facilities at the lowest cost. The company would like to know the worst-case cost associated with transportation in the event that up to two of the roads may be closed (or partially closed). In the network depicted in Fig. 5, the production facilities are labeled with their supply quantities (positive values) and the retail facilities are labeled with their demands (negative values). The table shows, for each arc $(i, j) \in A$, the cost c_{ij} of sending one unit on the arc and the capacity u_{ij} on the arc. Without any road closures, an optimal solution is $y_{16} = 4$, $y_{24} = 3$, $y_{34} = 1$, and $y_{36} = 2$, with all other arcs having zero flow, and a total cost of 37. The interdiction problem is formulated as follows:

$$\begin{aligned} \max_{\mathbf{x} \in \mathbf{X}} \min & 9y_{14} + 6y_{15} + 3y_{16} + 4y_{24} + 6y_{25} + 5y_{26} \\ & + 5y_{34} + 8y_{35} + 4y_{36} + y_{54} + y_{56} \end{aligned} \quad (15a)$$

$$\text{s.t. } y_{14} + y_{15} + y_{16} = 4 \quad (15b)$$

$$\vdots \quad (15c)$$

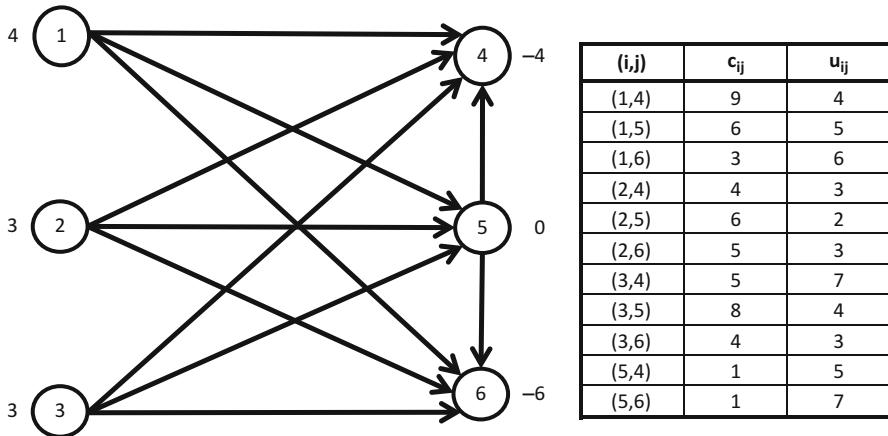


Fig. 5 Minimum cost flow network

$$-y_{16} - y_{26} - y_{36} - y_{56} = -6 \quad (15d)$$

$$y_{14} \leq 4(1 - x_{14}) \quad (15e)$$

$$\vdots \quad (15f)$$

$$y_{56} \leq 7(1 - x_{56}) \quad (15g)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (15h)$$

where $\mathbf{X} = \{\mathbf{x} : \sum_{(i,j) \in A} x_{ij} \leq 2, \mathbf{0} \leq \mathbf{x} \leq \mathbf{e}\}$. After taking the dual of the minimization problem and combining it with the first-stage problem, the following maximization problem is obtained:

$$\begin{aligned} & \max 4\alpha_1 + 3\alpha_2 + 3\alpha_3 - 4\alpha_4 - 6\alpha_6 - 4(1 - x_{14})\beta_{14} \\ & -5(1 - x_{15})\beta_{15} - 6(1 - x_{16})\beta_{16} - 3(1 - x_{24})\beta_{24} - 2(1 - x_{25})\beta_{25} \\ & -3(1 - x_{26})\beta_{26} - 7(1 - x_{34})\beta_{34} - 4(1 - x_{35})\beta_{35} - 3(1 - x_{36})\beta_{36} \\ & -5(1 - x_{54})\beta_{54} - 7(1 - x_{56})\beta_{56} \end{aligned} \quad (16a)$$

$$\text{s.t. } \alpha_1 - \alpha_4 - \beta_{14} \leq 9 \quad (16b)$$

$$\vdots \quad (16c)$$

$$\alpha_5 - \alpha_6 - \beta_{56} \leq 1 \quad (16d)$$

$$\beta \geq \mathbf{0} \quad (16e)$$

$$\mathbf{x} \in \mathbf{X}. \quad (16f)$$

In this case, because the structural constraints of the set \mathbf{X} consist only of the cardinality constraint enforcing the condition that only two interdictions are allowed, along with the unit upper bounds on the x -variables, the constraints defining \mathbf{X} are totally unimodular. Hence, as explained in Remark 2, the \mathbf{x} -variables are binary, and the nonlinear product terms can be linearized. The optimal solution in this case occurs when the roads corresponding to arcs $(1, 5)$ and $(1, 6)$ are fully closed, resulting in a cost of 63. \square

2.2 Applications

This subsection presents several cases in which interdiction has been applied in real-world settings, as opposed to the simpler illustrative settings mentioned in the previous section.

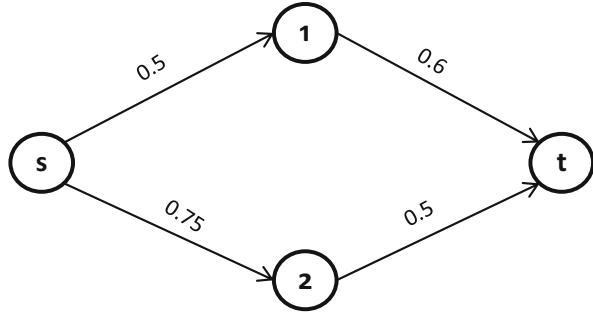
2.2.1 Nuclear Smuggling (Shortest Path Interdiction)

Morton et al. [40] give a shortest path interdiction model for a nuclear smuggling scenario. A joint effort between the United States and Russia, the goal was to determine the optimal locations for nuclear sensors to prevent the smuggling of nuclear material out of the former Soviet Union. Hence, in this case, the interdictors are the American/Russian governments, and the evaders are unknown criminals (as discussed in Remark 1). There is an underlying network on which the evaders travel, and there exists some probability of evasion on each arc. The authors developed a stochastic model in which the origin–destination pairs for evaders are unknown, but follow a known probability distribution. This results in a model where the objective of the interdictor is to minimize the maximum probability of evasion by the criminals. A directed graph $G(N, A)$ is used where $FS(i)$ is the forward star of node i and $RS(i)$ is the reverse star of node i . The interdictor has a total budget, b , with which it can install the sensors, each at a cost of c_{ij} , for all $(i, j) \in A$. Associated with each arc $(i, j) \in A$ is a probability, p_{ij} , that the evader can traverse it undetected when no sensor is installed, as well as a probability, q_{ij} , that the evader can traverse it undetected with a sensor installed. The set of scenarios is denoted by Ω , and each scenario, $\omega \in \Omega$, has a probability of p^ω and an origin–destination realization of (s^ω, t^ω) . The interdictor has one set of decision variables for which x_{ij} takes a value of 1 if a sensor is placed on arc (i, j) , and 0 otherwise. The evader has two sets of decision variables for which y_{ij} is positive only if the evader traverses arc (i, j) and a sensor is not installed, and z_{ij} is positive if the evader traverses arc (i, j) and a sensor is installed. This results in the following model:

$$\min_{\mathbf{x} \in \mathbf{X}} \sum_{\omega \in \Omega} p^\omega h(x, (s^\omega, t^\omega)) \quad (17)$$

where $\mathbf{X} = \{\mathbf{x} : \sum_{(i,j) \in AD} c_{ij} x_{ij} \leq b; x_{ij} \in \{0, 1\}, \forall (i, j) \in A\}$, and $h(x, (s^\omega, t^\omega))$ is the optimal value of

Fig. 6 Nuclear smuggling example



$$\max_{y,z} y_{t^\omega} \quad (18a)$$

$$\text{s.t. } \sum_{j \in FS(s^\omega)} (y_{s^\omega j} + z_{s^\omega j}) = 1 \quad (18b)$$

$$\sum_{j \in RS(i)} (p_{ji} y_{ji} + q_{ji} z_{ji}) = \sum_{j \in FS(i)} (y_{ij} + z_{ij}) \quad \forall i \in N \setminus \{s^\omega, t^\omega\} \quad (18c)$$

$$y_{t^\omega} = \sum_{j \in RS(t^\omega)} (p_{jt^\omega} y_{jt^\omega} + q_{jt^\omega} z_{jt^\omega}) \quad (18d)$$

$$0 \leq y_{ij} \leq 1 - x_{ij}, \quad \forall (i, j) \in A \quad (18e)$$

$$z_{ij} \geq 0 \quad \forall (i, j) \in A. \quad (18f)$$

The objective function in (17) is the expected value of evasion, taken over all possible scenarios. The optimal value of (18) is the probability of evasion given the origin–destination pair (s^ω, t^ω) . Constraint (18b) ensures that one unit of flow leaves the origin, constraint (18c) enforces flow conservation at the intermediate nodes, and constraint (18d) defines the probability of evasion, y_{t^ω} . Constraint (18e) ensures that y_{ij} must be zero if the arc is interdicted. If the evader uses an arc that is not interdicted, since $p_{ij} > q_{ij} \forall (i, j)$, the corresponding y_{ij} -variable would be positive in order to maximize evasion probability. Therefore, given a path P_{s^ω, t^ω} from s^ω to t^ω , this results in:

$$y_{t^\omega} = \prod_{(i,j) \in P_{s^\omega, t^\omega}} [p_{ij}(1 - x_{ij}) + q_{ij}x_{ij}]. \quad (19)$$

To illustrate how y_{t^ω} is calculated, consider the following example illustrated in Fig. 6 in which there is no interdiction (the p_{ij} -values are located on the arcs.) Clearly, the optimal path is $s - 2 - t$, which has an evasion probability of $(0.75)(0.5) = 0.375$. In satisfying constraint (18b), the values $y_{s1} = 0$ and $y_{s2} = 1$ are obtained. In satisfying constraint (18c) for node 2, the value $y_{2t} = 0.75$ is

obtained, and in satisfying constraint (18d), the value $y_t = 0.6y_{1t} + 0.5y_{2t} = 0 + (0.5)(0.75) = 0.375$ is obtained.

The authors also developed a model in which information is asymmetric, that is, the interdictor and the smuggler have different perceptions of the network parameters. Reformulation and duality techniques were used as well as the introduction of step inequalities to improve and tighten the formulations. CPLEX was used to solve the resulting linear mixed-integer programs. See Sect. 3 for further discussion on handling challenges posed by stochasticity and information asymmetry.

2.2.2 Electrical Grids

One highly researched area for applications of network interdiction models arises in electrical grid analysis (see, e.g., [23, 51, 52]). The goal is to determine the best way to operate a power system when the system is subject to disruptions. These disruptions may or may not be malicious. In the case of malicious disruptions, the interdictor attempts to maximize the amount of “load shedding” in the system. Load shedding occurs when the demand of the power grid exceeds the capacity, which results in significant costs to the electricity providers (as well as to those not receiving power). In [23], a two-stage model is introduced in which the interdictor (or terrorist in this case) attacks or destroys some of the power lines in the grid, and the operator attempts to minimize the amount of load shedding over the system. A simplified version of the formulation in the paper is as follows:

$$\max_v \sum_{n \in N} \Delta_n \quad (20a)$$

$$\text{s.t. } \sum_{l \in L} (1 - v_e) = M \quad (20b)$$

$$v_e \in \{0, 1\} \quad \forall e \in E, \quad (20c)$$

where E is the set of transmission lines; v_e is a binary variable that takes a value of 0 if line e is destroyed, and 1 otherwise; M is the number of lines to be destroyed; N is the set of buses; and Δ_n is the load shed at bus n , which is obtained from the operator’s load-shedding minimization problem:

$$\min_{\Delta, \mathbf{P}, \mathbf{w}, \delta} \sum_{n \in N} \Delta_n \quad (21a)$$

$$\text{s.t. } F(\mathbf{P}, \delta, \mathbf{w}, \mathbf{v}) = 0 \quad (21b)$$

$$A^\top \mathbf{P} = \Delta \quad (21c)$$

$$\mathbf{0} \leq \Delta \leq \Delta^u \quad (21d)$$

$$\mathbf{P}^l \leq \mathbf{P} \leq \mathbf{P}^u \quad (21e)$$

$$\boldsymbol{\delta}^l \leq \boldsymbol{\delta} \leq \boldsymbol{\delta}^u \quad (21f)$$

$$w_e \in \{0, 1\} \quad \forall e \in E. \quad (21g)$$

Here \mathbf{P} is the vector of variables corresponding to line power flows and generator power outputs; \mathbf{w} is the vector of variables corresponding to the binary decision of disconnecting ($w_e = 0$) or not disconnecting ($w_e = 1$) line $e \in E$; $\boldsymbol{\delta}$ is the vector of variables corresponding to the phase angles at each of the buses; Δ^u , \mathbf{P}^l , \mathbf{P}^u , $\boldsymbol{\delta}^l$, and $\boldsymbol{\delta}^u$ are vectors of lower and upper bounds for the variables; given \mathbf{v} , $F(\mathbf{P}, \boldsymbol{\delta}, \mathbf{w}, \mathbf{v})$ is a set of nonlinear functions of \mathbf{P} , $\boldsymbol{\delta}$, and \mathbf{w} , which correspond to transmission line flows; and $A^\top \mathbf{P} = \Delta$ is the set of linear constraints representing the power balance in each bus of the system.

The authors develop a Benders' decomposition technique for solving the model. Since the problem is nonconvex, the solution method incorporates multiple restarts in order to search a majority of the solution space. As a result, the algorithm is able to obtain optimal or near-optimal solutions in a reasonable amount of time.

In [16], a defender–attacker–defender model is formulated in which the operator is able to prevent interdiction on some of the lines before the interdictor is able to act. Through duality techniques, they are able to turn the three-stage formulation into a single-level master problem that is conducive to Benders' decomposition (see also the discussion in Sect. 3.1).

2.2.3 Drug Enforcement (Maximum Flow Interdiction)

In [39], a maximum flow interdiction model is used to model city-level drug enforcement decisions. In the drug trade, there is usually some multilevel hierarchy of drug dealers and drug users. Assuming that the goal of the dealer, or “enemy,” at the highest level is to maximize the flow of drugs, the optimization problem from this dealer’s point of view is a maximum flow problem. A network is used to represent the hierarchy of dealers and users with source node α that represents the “origin” of the drugs and a sink node ω that represents the sink of the flow. The origin is essentially the main dealer that supplies the city with drugs, and the sink captures all flow that reaches the users. Each dealer or user is represented in the network by a node. An arc $(i, j) \in A$ exists if the person represented by node i deals to the person represented by node j . Each arc has a capacity, and the formulation consists of the typical flow conservation and capacity constraints, with the goal of the “enemy” being to deliver as much flow from the source node to the sink node over a time horizon $1, \dots, T$. The goal of law enforcement is to minimize the maximum flow over the time horizon. One of the interesting ideas in this chapter is the way that law enforcement’s feasible region was characterized. The law enforcement officials are constrained not only by the resources of their departments but also by the real-life aspects of the drug trade. They discuss the idea that law enforcement must “climb the ladder” of the drug supply chain network in order to reach the dealers at the upper levels of the hierarchy. For example, arresting an individual may require that law enforcement officers monitor this

individual for some period of time. Also, to be able to monitor a dealer that is high up in the hierarchy, it might be necessary to arrest several of the individuals that purchase from this dealer. The resource allocation of the department is constrained as follows:

$$\sum_{(i,j) \in A} (a_{ij} w_{ijt} + b_{ij} y_{ijt}) \leq B \quad \forall t = 1, \dots, T, \quad (22)$$

where B is the number of available officers, a_{ij} is the number of officers required to arrest/remove arc (i, j) , b_{ij} is the number of officers required to monitor arc (i, j) , w_{ijt} is a binary variable representing the decision to arrest/remove arc (i, j) from the network in time period t , and y_{ijt} is a binary variable that represents the decision to monitor/target arc (i, j) in time period t . To restrict law enforcement to only be able to arrest an individual after they have been monitored for a set period of time, the following constraints are used:

$$\sum_{t'=1}^{t-1} y_{ijt'} \geq \tau_{ij} w_{ijt} \quad \forall t = 1, \dots, T \quad (23a)$$

$$(1 - z_{ijt}) \leq (1 - z_{ij,t-1}) + w_{ijt} \quad \forall t = 1, \dots, T, \quad (23b)$$

where τ_{ij} is the number of time periods that (i, j) must be targeted before it is interdicted and z_{ijt} is a binary variable representing the availability of arc (i, j) in time period t . If it is necessary to monitor the target in consecutive time periods, then the summation in (23a) can be from $t' = t - \tau_{ij}$ to $t - 1$. In order to model the fact that law enforcement may have to arrest lower-level individuals in order to start to target a higher-level individual, each dealer is actually represented with two nodes, i and i' . The capacity of the arc between them, $\mu_{ii'}$, represents the number of arrests that are necessary in order to target the dealer. This is modeled with the following constraint set:

$$\sum_{j \in A(i')} (1 - z_{jj't}) \geq \mu_{ii'} y_{ii't} \quad \forall t = 1, \dots, T, \quad (24)$$

where $A(i')$ is the adjacency list of node i' (or the set of buyers from dealer i .) The authors also detail further considerations that face law-enforcement agents in gaining cooperation from the criminals, and how these considerations affect the mathematical models they prescribe.

Through the use of duality and linearization techniques, the authors develop a single mixed-integer linear program that can be solved with a commercial solver such as CPLEX.

2.2.4 Supply Chains (Facility Assignment Interdiction)

Supply chains are vulnerable to a number of disruptions, many of them unavoidable such as hurricanes or other natural disasters. A great deal of research has focused on

designing supply chains that are resilient to these types of disruptions. Snyder et al. [61] analyze the design problem for such supply chains, including a worst-case cost model. This model takes into account various scenarios and determines a design that minimizes the worst-case cost of operating the supply chain. Let I denote the set of customers, J the set of possible facility locations, and S a set of possible failure scenarios for the facilities. Not every possible scenario belongs to S —if so, the worst-case scenario would simply consist of the failure of all facilities—because the simultaneous failure of some combination of facilities may be sufficiently unlikely and hence not worthy of consideration. Instead, this set may, for instance, only include scenarios in which no more than four facilities are disrupted. Each customer i has an annual demand of h_i units, and each unit shipped from facility j to customer i has a shipment cost of d_{ij} . There is a fixed cost f_j for opening each facility j , and a_{js} represents whether facility j was disrupted ($a_{js} = 1$) in scenario s , or not ($a_{js} = 0$). The model uses two sets of binary decisions variables: x_j which takes a value of 1 if facility j is opened, and 0 otherwise; and y_{ijs} which takes a value of 1 if customer i is assigned to facility j in scenario s , and 0 otherwise. The model is formulated as follows:

$$\min U \quad (25a)$$

$$\text{s.t. } U \geq \sum_{j \in J} f_j x_j + \sum_{i \in I} \sum_{j \in J} h_i d_{ij} y_{ijs} \quad \forall s \in S \quad (25b)$$

$$\sum_{j \in J} y_{ijs} = 1 \quad \forall i \in I, s \in S \quad (25c)$$

$$y_{ijs} \leq (1 - a_{js})x_j \quad \forall i \in I, j \in J, s \in S \quad (25d)$$

$$x_j \in \{0, 1\} \quad \forall j \in J \quad (25e)$$

$$y_{ijs} \in \{0, 1\} \quad \forall i \in I, \forall j \in J, \forall s \in S. \quad (25f)$$

The objective minimizes U , where Constraint (25b) forces U to be greater than or equal to the minimum cost of operation in each of the scenarios (the right-hand side of the constraint is forced to be the minimum cost for each scenario, since this is a minimization problem.) Constraint (25c) ensures that each customer is assigned to a facility in every scenario. Constraint (25d) ensures that customer i can only be assigned to facility j in scenario s if the facility is opened and not disrupted in that scenario. It is important to note that there are clear differences between this model and the general facility assignment interdiction model presented earlier. Instead of an interdictor acting with a goal of maximizing the cost of operating the supply chain, there is a set of possible scenarios that must be considered. However, since the supply chain designer is preparing for the worst-case scenario, this results in the same solution as if there were an interdictor choosing the worst-case from the set of possible scenarios.

3 Emerging Areas of Interdiction Research

This section describes a few research thrusts that have arisen primarily in the past decade. Fortification optimization problems are discussed in Sect. 3.1. Models and algorithms for stochastic interdiction problems are then presented in Sect. 3.2, followed by the consideration of asymmetric information in Sect. 3.3. Lastly, solution strategies are explored for multi-objective problems in Sect. 3.4.

3.1 Fortification Problems

Several fortification problems, such as those explored in [17, 19, 53, 59], seek to fortify a network in anticipation of an optimal attacker. These fortification actions are essentially taken in advance of the mathematical programming models described in Sect. 2. In many settings, it may be possible to integrate the presence of fortification actions directly within the mathematical programming interdiction model. However, for the examples mentioned here, a cutting-plane approach to solving these problems is evidently required.

In the model described by (11), Church and Scaparra [19] prescribe a two-phased algorithm to optimally fortify facilities, where a fortified facility cannot be destroyed by the interdictor. In this approach, a master problem is solved first that prescribes fortification actions and computes a lower bound on the optimal objective function value. Fortification actions are constrained by $\mathbf{w} \in \mathbf{W}$, where $\mathbf{W} = \{\mathbf{w} : \mathbf{e}^\top \mathbf{w} = q, \mathbf{w} \in \{0, 1\}^{|A|}\}$ restricts the fortification actions to be binary and enforces the condition that q fortifications are made. The approach taken is then to minimize some value function variable η , subject to $\mathbf{w} \in \mathbf{W}$, along with an exponentially sized set of affine inequalities that relate η to the fortification decisions \mathbf{w} . Sets of these inequalities are generated one at a time, as needed, from a subproblem of the form (11), which has the additional restrictions that $x_i \leq (1 - z_i)$ for each $i \in N$. To generate inequalities at iteration k of this process, consider any feasible interdiction solution \mathbf{x}^h to (11) at iteration h of the algorithm, which induces a solution \mathbf{y}^h . Define $d_i^h = \sum_{j \in F} d_{ij} y_{ij}^h$ as the distance from i to its closest non-interdicted facility in the solution given by $(\mathbf{x}^h, \mathbf{y}^h)$. If $d_{ij} < d_i^h$ for some $j \in F$, then by fortifying facility j , the total objective value may decrease by as much as $(d_i^h - d_{ij})$. Let B_i^h be the set of all closer facilities $j \in F$ such that $d_{ij} < d_i^h$. One possible valid inequality is given by

$$\eta \geq \sum_{i \in N} a_i \left(d_i^h - \sum_{j \in B_i^h} (d_i^h - d_{ij}) w_j \right),$$

but this inequality is quite weak due to the fact that for some $i \in N$, fortifying multiple facilities in $B' \subseteq B_i^h$ promises a reduction of $\sum_{j \in B'} (d_i^h - d_{ij})$ rather than the actual (maximum possible) reduction of $(d_i^h - \min_{j \in B'} \{d_{ij}\})$. To strengthen this formulation and avoid the weakness presented by the foregoing inequality, Church

and Scaparra [19] instead define additional variables v_{ij}^h at iteration h , which equal 1 if facility j is the closest facility to i in B_i^h that is fortified, and 0 otherwise. This permits the inclusion of the stronger, though larger, inequality system for every iteration h enumerated by the algorithm:

$$\eta \geq \sum_{i \in N} a_i \left(d_i^h - \sum_{j \in B_i^h} (d_i^h - d_{ij}) v_{ij}^h \right) \quad (26a)$$

$$v_{ij}^h \leq w_j \quad \forall i \in N, j \in B_i^h \quad (26b)$$

$$\sum_{j \in B_i^h} v_{ij}^h \leq 1 \quad (26c)$$

$$v_{ij}^h \in \{0, 1\} \quad \forall i \in N, j \in B_i^h. \quad (26d)$$

Even though (26) increases the number of variables and constraints for each solution h , because they are added only as needed during the two-phase process (and not in general for every possible interdiction solution), the routine is able to solve real-world instances in acceptable computational limits. The process terminates when the predicted objective η found in the master problem matches the objective computed in the subproblem.

The approach of Brown et al. [17] and Smith et al. [59] is tailored toward the more general problem of minimum cost network flow interdiction. Again letting \mathbf{W} be the feasible region for fortification actions, the objective would be to minimize $v(\mathbf{w})$ over $\mathbf{w} \in \mathbf{W}$, where $v(\mathbf{w})$ solves problem (14) with the additional objective term $-M(w_{ij} x_{ij})$ for some large number M . This term serves as a penalty function to prohibit setting $x_{ij} = 1$ at optimality when $w_{ij} = 1$. Note that the feasible region of (14) is invariant with respect to \mathbf{w} and that with \mathbf{w} fixed, (14) is a disjointly constrained mixed-integer bilinear program, assuming that \mathbf{X} contains only linear and (possibly) integrality constraints. (That is, α and β are constrained only by the polyhedron induced by (14b) and (14c), and \mathbf{x} is only constrained to lie in \mathbf{X} . If either set of variables is fixed, then assuming that \mathbf{X} is defined by a set of linear constraints, (14) becomes a linear program.) Therefore, let Θ denote the set of extreme points to (14b) and (14c), and let Λ denote the set of extreme points to \mathbf{X} (or to $\text{conv}(\mathbf{X})$ if \mathbf{X} includes binary restrictions). Because the problem is bilinear, an optimal solution lies at a combination of extreme points in Θ and Λ . Then, the interdictor's objective function can be rewritten as

$$\max_{(\alpha^k, \beta^k) \in \Theta, \mathbf{x} \in \Lambda} \sum_{i \in N} d_i \alpha_i^k - \sum_{(i, j) \in A} u_{ij} (1 - x_{ij}^k) \beta_{ij}^k - M(w_{ij} x_{ij}^k).$$

The strategy for solving this class of problems constructs a master problem and subproblem as mentioned above, where the master problem contains a value function η , and the subproblem solves (14) augmented with the fortification penalty

term. If η does not match the subproblem objective term, an optimal solution (α^*, β^*, x^*) is obtained from the subproblem and used to generate a (single) cut of the form

$$\eta \geq \sum_{i \in N} d_i \alpha_i^* - \sum_{(i,j) \in A} u_{ij} (1 - x_{ij}^*) \beta_{ij}^* - M(w_{ij} x_{ij}^*).$$

3.2 Stochastic Interdiction

Interdiction and fortification problems are further complicated in the presence of uncertainty. In practice, some of the data present in the model may be stochastic, and the success of interdiction actions may be uncertain. The former case is considered by Morton et al. [40] and Pan and Morton [44]. These papers consider an evader/interdictor, where the interdictor seeks to monitor network nodes and/or arcs with the purpose of minimizing the evader's maximum probability of escape. The origin and destination of the evader are unknown in these models, which results in an NP-hard interdiction problem even if the underlying network is bipartite [41].

Cormican et al. [21] consider a stochastic maximum flow interdiction problem, which is subject to uncertainty of interdiction actions. In their base model, the network data is known with certainty, but each interdiction action is successful only with some probability p_{ij} , for each arc (i, j) . The authors extend their results for the case in which arc capacities are also uncertain, interdiction success is varied among several discrete levels, and multiple interdiction actions are possible.

The approach taken in [21] is to consider a problem similar to the form of (6), where constraints (6c) are replaced with

$$y_{ij} \leq u_{ij} (1 - \tilde{I}_{ij} x_{ij}) \quad \forall (i, j) \in A \setminus \{(t, s)\}, \quad (27)$$

and where \tilde{I}_{ij} is a random variable that equals 1 if an interdiction on arc (i, j) would be successful (which occurs with probability p_{ij}), and otherwise equals 0 if an interdiction would be unsuccessful (occurring with probability $(1 - p_{ij})$). Both successful and unsuccessful interdiction actions contribute to the interdictor's overall budget. Hence, the interdictor is faced with the challenge of trading off the selection of arc interdictions that are maximally effective when all interdictions are successful, with those that correspond to large p_{ij} -values.

The problem is now to identify the interdiction actions $x \in S$ that minimize the network owner's maximum flow. Janjarassuk and Linderoth [34] approach this problem via sampling strategies as used in scenario-based stochastic problems. Cormican et al. [21] prescribe a sequential partitioning approach, where lower and upper bounds on the expected objective are gradually tightened over the partitioned space. The essence of the algorithm is to note that, given interdiction actions $\mathbf{x} \in \mathbf{X}$, the maximum flow problem given by (6) (which includes (27) instead of (6c)) is a

concave function over the set of possible outcomes of \tilde{I} . This problem is denoted here as SMF-1. An equivalent formulation, SMF-2, is given by

$$\min_{\mathbf{x} \in \mathbf{X}} \max y_{ts} - \sum_{(i,j) \in A} (x_{ij} \tilde{I}_{ij}) y_{ij} \quad (28a)$$

$$\text{s.t. Constraints (6b)} - \text{(6d).} \quad (28b)$$

Here, equivalence is claimed in the sense that an optimal solution to SMF-1 is also optimal to SMF-2, and although an optimal solution to SMF-2 need not be optimal to SMF-1, the optimal objective function values of the two problems match. Given a fixed interdiction action in \mathbf{X} , the inner maximum flow problem of SMF-1 is a concave function of the random variables \tilde{I} , while the inner maximum flow problem of SMF-2 is a convex function of these random variables. (Note that stochasticity occurs in the constraints of the inner problem in SMF-1 and in the objective of the inner problem in SMF-2.)

As a result, an initial lower bound (LB_0) is given by replacing the random variables with their expectation in SMF-2, and an initial upper bound (UB_0) is given by doing the same in SMF-1. These bounds are valid due to Jensen's inequality (see, e.g., Kall and Wallace [35] for a discussion of this inequality's use in stochastic programming). To refine this bound, one could partition the state space of \tilde{I} into K spaces $\mathcal{I}_1, \dots, \mathcal{I}_K$, where the probability of \tilde{I} belonging to \mathcal{I}_k is given by ρ_k , $\forall k = 1, \dots, K$. Then in the same manner above, one can compute LB_k and UB_k for each $k = 1, \dots, K$ and obtain new lower and upper bounds as

$$LB^K = \sum_{k=1}^K \rho_k LB_k \geq LB_0$$

$$UB^K = \sum_{k=1}^K \rho_k UB_k \leq UB_0.$$

Moreover, as K becomes large, LB^K and UB^K both converge to the optimal expected interdiction objective. See [21] for methods that dynamically partition the original state space into subregions and terminate once $UB^K - LB^K$ is sufficiently small.

3.3 Asymmetric Information

Bayrak and Bailey [3] and Morton et al. [40, 41] examine interdiction problems in which there exists so-called information asymmetry between the opposing players. In the context of interdiction problems, “asymmetric information” refers to the situation in which the opposing sides have different perceptions about the parameters of the model. Bayrak and Bailey [3] define c_{ij} as the length of arc (i, j) and d_{ij} as the amount that the arc length is increased if the arc is interdicted. The correct data values are known to the interdictor. The evader, on the other hand, only has estimates of these values: \bar{c}_{ij} and \bar{d}_{ij} . (Whether or not the interdictor in fact knows the correct data is immaterial, but in any case, the interdictor is optimizing

with respect to c - and d -values, with knowledge that the evader optimizes with respect to \bar{c} - and \bar{d} -values.)

Variables x_{ij} , for each $(i, j) \in A$, are the interdictor's decision variables, which take a value of 1 if arc (i, j) is interdicted, and 0 otherwise. Variables y_{ij} are the evader's decision variables, $\forall (i, j) \in A$, which take a value of 1 if arc (i, j) is on the shortest path, and 0 otherwise. The shortest path network interdiction problem with asymmetric information is formulated as follows:

$$\max_{\mathbf{x}} \sum_{(i,j) \in A} (c_{ij} + d_{ij}x_{ij})y_k^* \quad (29a)$$

$$\text{s.t. } \sum_{(i,j) \in A} b_{ij}x_{ij} \leq B \quad (29b)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A, \quad (29c)$$

where

$$y^* \in \operatorname{argmin}_{\mathbf{y}} \left\{ \min_{\mathbf{y}} \sum_{(i,j) \in A} (\bar{c}_{ij} + \bar{d}_{ij}x_{ij})y_{ij} \right\} \quad (30a)$$

$$\text{s.t. } \sum_{j \in FS(i)} y_{ij} - \sum_{j \in RS(i)} y_{ji} = \begin{cases} 1 & \text{for } i = s, \\ -1 & \text{for } i = t, \\ 0 & \text{for } i \in N_0 \end{cases} \quad (30b)$$

$$\mathbf{y} \geq \mathbf{0}, \quad (30c)$$

and where $N_0 = N \setminus \{s, t\}$. Since the evader's problem is a linear program, it is possible to reformulate it as another linear program with no objective function and a constraint set consisting of the constraints in (30), the constraints from the dual of (30), as well as the constraint: [primal objective]=[dual objective]. This allows the two-stage problem to be formulated as a single mixed-integer program that can be linearized and solved by standard commercial solvers.

3.4 Multi-objective Interdiction

Royset and Wood [50] examine a maximum flow interdiction problem of the form (6), but with the aim of finding all *Pareto-optimal* solutions with respect to interdiction effectiveness and interdiction cost (see also the work of Rocco and Ramirez-Marquez [49], which takes an evolutionary algorithm-based approach for solving these problems). As such, the interdictor's feasible region is relaxed to contain only binary restrictions on the \mathbf{x} -variables, and the quality of an interdiction solution is measured by both the follower's maximum possible flow (measured

by y_{ts}) and the cost of interdiction, given by $\sum_{(i,j) \in A} c_{ij} x_{ij}$, where c_{ij} is the cost of interdicting arc $(i, j) \in A$. While this could be achieved by enumerating all possible budget values between 0 and R , where R is the minimum directed cut of the graph (equal to the maximum flow of the graph, see, e.g., [1]), this process would require the solution of $O(R)$ integer programming problems, and $O(R)$ itself is not a polynomial function of the problem's input size.

An alternative approach instead employs a scheme that is interpreted as Lagrangian dual optimization to identify several Pareto-optimal solutions. The scheme places objective weights on the two criteria and uses Lagrangian dual values to intelligently adjust these weights in order to compute more of the Pareto-optimal frontier. After this process has completed, a customized branch-and-bound search is then performed to compute all remaining Pareto-optimal solutions.

4 Competition, Interdiction, and Optimization Models

This section discusses the application of interdiction models to contexts involving industrial or organizational competition. The following subsection describes the view of competitive commerce problems as a class of interdiction problems. After explaining this view, the subsequent subsection explores the structure of the game-theoretic problems that result, as well as the optimization problems that must often be solved in order to characterize the actions that rational players of these games will take.

4.1 Industrial Competition as an Interdiction Problem

While the term interdiction is typically defined in warfare-related terms (e.g., contexts involving an *enemy*), this term may be applied more liberally to general competitive settings. That is, instead of defining an interdictor as an enemy in the combative sense, any competitor whose goals conflict with those of a principal operator or agent may be thought of as an interdictor. Thus, given a principal operator (referred to as an “operator”) with some set of goals, any separate party who takes action(s) that impede the operator’s ability to meet its goals may be characterized as an interdictor. As a result, given an operator organization that wishes to engage in commercial activity by selling a product or providing a service to a market of finite size, any competitor selling a substitutable product or service to that market can effectively act as an interdictor by hampering the operator’s ability to meet market demands. The concepts, models, and methods applied to network interdiction problems, therefore, often readily apply to competitive commercial operations settings. This use of warfare analogies in competitive business is hardly new. For example, Kotler and Singh [36] provide an interesting overview of the ways in which military strategies can assist in developing marketing strategies for businesses.

Cartwright [18] provided a framework discussing the elements of corporate warfare and argued that three important dimensions of corporate operational warfare strategy exist: development, defense, and attack. A competitive firm uses development strategies to create a network of relationships vital for achieving its goals. This network creates linkages among suppliers and customers to enable the flow of goods and/or services. This development phase of corporate warfare strategy is akin to the network design phase in communications or other physical network. Defense strategies focus on fortifying the network of relationships once it has been developed and may include strategic partnerships and contractual agreements that protect this network. These defense strategies are analogous to the techniques applied in survivable network design (see, e.g., Smith et al. [59]). Attack strategies are employed to disrupt a competitor's network. When a firm employs such strategies, it acts as an interdictor with respect to another firm's network. Cartwright [18] argues that the weapons employed in network development, defense, and attack (interdiction) strategies include products, information, financial capital, and human capital. Firms thus utilize information and financial and human capital to develop a network of products and alliances that enable them to compete effectively.

Numerous practical examples exist in the literature involving forms of strategic corporate warfare. For example, in response to the rise and market power of Amazon.com in the 1990s, Barnes and Noble sought to purchase its distributor, although this proposal never came to fruition as a result of opposition from the Federal Trade Commission [18]. Instead, Barnes and Noble spun off bn.com and partnered with Yahoo to enter the Internet-direct sales business. Barnes and Noble now successfully employs a dual-channel strategy using both physical stores and its web presence. Examples of competitive reaction are also prevalent in service industries. When a small start-up company called Legend Airlines began to offer 56-seat flights from Love field in Dallas, American Airlines, headquartered in Dallas, also began to offer flights using planes of the same capacity [58]. Examples of location-based competitive responses abound as well. Fast food and pharmaceutical chains often respond to competitors' strategic location decisions by building a competing store next to a newly opened store, which can undercut the original store's competitive position.

As these examples illustrate, interdiction in commerce may manifest itself in numerous ways. The most obvious and visible manifestation occurs when an operator's competitor (interdictor) introduces a product or service that is intended to compete with and take market share away from an existing product or service offered by the operator. Smith et al. [58] model product line offering decisions for an operator and an interdictor, the latter of which wishes to minimize the operator's profit. The operator must determine which among a finite set of products to offer in order to meet a set of customer needs in a market. The interdictor introduces a product line after the operator introduces its own and may thus take away customers that would have otherwise preferred a product offered by the operator. This problem is modeled as a two-stage Stackelberg game in which the operator acts first and the interdictor follows. By anticipating the interdictor's best response to its actions, the operator can effectively minimize the potential damage inflicted by the interdictor.

Competitive location-based models have a long history in the research literature, beginning with Hotelling [32] in 1929, who modeled a two-player simultaneous location game with customers uniformly distributed along a line. Since that time, researchers have modeled competitive location decisions under numerous assumptions on the potential facility locations and the factors that drive customer and competitive response to location choices (see Eiselt et al. [24] and Plastria [47] for more detailed discussions of this body of literature). This work tends to model competition based on customer preference factors, such as price, service level, and location, rather than addressing the possibility of direct attacks on a network of facilities. More recent work has taken a worst-case view of location problems that is more in line with the literature on network interdiction, identifying and fortifying the most critical locations in a facility network subject to attacks that may be man-made or natural (see Church et al. [20] and Scaparra and Church [53]).

In addition to competition based on new product/service introduction and strategic location choice, firms may engage in competitive warfare through advertising/promotions and using strategic partnerships. A well-known example of advertising competition occurred in the 1980s when Coca-Cola and Pepsi targeted each other in advertising over an extended period of time, resulting in the so-called Cola Wars (see Pearson and Irwin [46]). A recent example of an effective strategic partnership for competitive advantage can be seen in the (formerly) exclusive resale contract between AT&T and Apple for Apple's iPhone.

These are a few of the numerous ways in which firms may act in order to either fortify their strategic networks or act as an interdictor who seeks to disrupt a competitor's network. AT&T, for example, fortified their cellular phone network by partnering with Apple, while American Airlines acted as an interdictor and disrupted the network of Legend Airlines, who remained in business for only 5 years. Competitive strategic decisions such as those described can often be modeled using the tools of game theory in general and network interdiction in particular. The following subsection explores the structure of these game-theoretic models and the optimization problems that they imply.

4.2 Optimization Models for Competition with Interdiction

The vast majority of interdiction modeling efforts in the literature that deal with business operations settings assume that the “attack” comes in the form of a natural disaster, terrorist attack, or other phenomenon that is unrelated to competitors’ actions. The facility assignment and facility-to-retailer flow interdiction problems discussed in Sect. 2.1, for example, assume a diminished network capacity for flows due to a loss of a subset of facilities and transportation links, respectively. Modeling realistic competitive scenarios, on the other hand, requires a different view of interdiction. The focus on competitive models in this section assumes an absence of illegal activities such as deliberate corporate sabotage. This removes the focus of the corresponding models from the loss of facilities and transportation links

and emphasizes the ways in which firms may interfere with the profit-making goals of competitors using legal competitive strategies.

A discussion of three particular competitive models illustrates the kinds of optimization models that arise in competitive interdiction. An early example of the use of interdiction ideas in competition can be found in Parlar [45], who looked at the inventory stocking problem for two firms that stock substitutable products. He modeled a simultaneous game in which both firms make stock level decisions at the same time. Each player faces a probability distribution representing its uncertain demand, and when a firm runs out of stock, customers will seek to fill demands at the other firm if they have available stock. While the primary purpose of the paper was to characterize equilibrium stock levels when both firms seek to maximize profit (the paper demonstrates that a unique Nash equilibrium exists), it also examines the case in which one of the players acts “irrationally” in an attempt to inflict as much damage as possible on the competitor. Parlar [45] shows that when this occurs, the irrational player achieves its lowest possible profit, while the other player faces a problem that is equivalent to one without a competitor.

Smith et al. [58] provided the second illustrative model, which demonstrates how a competitor may interdict by offering a product line that competes with that of another firm. They consider two firms, both of whom can offer a subset of N candidate products that target a customer population or market, consisting of a number of market segments. Each of M market segments has an ordered preference list of a subset of the N products—customers in a given segment will choose the first available product on their ordered preference list (and if none of the products in their preference list are offered, customers from the segment will not purchase). The two firms offering the products are considered a leader and a follower in a Stackelberg game in which the leader first determines its product line offering and the follower subsequently determines which products to offer. Each firm has a limited budget and incurs a product- and firm-specific fixed cost for offering any product. If both firms offer the same product, then $100\% \times \rho_{ij}$ of the demand from market segment i for product j will go to the leader, and the remaining percentage to the follower. Each unit of product j sold to market segment i produces a revenue of r_{ij} .

The model proposed in [58] takes a worst-case approach for the leader. That is, the leader seeks to maximize its profit in the worst case (i.e., maximize its minimum profit), assuming the follower strictly seeks to introduce a set of products that minimizes the leader’s revenue. The result is a two-stage optimization model. The first-stage problem maximizes the leader’s minimum profit by determining which subset of products to offer (indicated by an N -vector \mathbf{x} of binary variables, where $x_j = 1$ if the leader offers product j). This first-stage problem encodes the follower’s product-line decision via an N -vector \mathbf{y} of binary variables, where $y_j = 1$ if the follower offers product j . The second-stage (or follower’s) problem requires the follower to choose the vector \mathbf{y} that minimizes the leader’s revenue based on the leader’s choice of the \mathbf{x} vector. Solving the leader’s problem, therefore, requires characterizing the follower’s best response to any given \mathbf{x} vector chosen by the leader and then choosing the \mathbf{x} vector that effectively minimizes the

potential damage. Smith et al. [58] show that the follower's (second-stage) problem is NP-hard in the strong sense. They thus reformulate their model as a three-stage optimization problem that requires solving a bilinear optimization problem. Using linearization and decomposition techniques (employing cutting planes), they provide an exact method for solving the problem. They also propose heuristic solution methods that can be used for solving problems of practical size.

The third example setting that illustrates interdiction in business competition is from Prince et al. [48], who consider two competitive firms that seek to secure a common raw material supply from a common supplier base. In this setting, suppliers have finite capacities and offer customers quantity discounts. An operator firm and an interdictor firm each have a supply quantity that they wish to procure from the supplier base. In addition, the operator may enter into *exclusivity* contracts with suppliers, under which the supplier agrees to provide supply only to the operator. Under an exclusivity agreement, the supplier incurs a fixed cost that is supplier dependent (which may correspond to a franchise fee paid to the supplier).

The sequence of decisions proceeds in three stages. In the first stage, the operator determines which suppliers to secure through exclusivity agreements. In the second stage, the interdictor allocates demand to available suppliers with a goal of maximizing the operator's supply cost. In the third stage, the supplier allocates its demand to the available remaining capacity in the supply base. The authors propose a two-stage reformulation of the problem in which the second stage is a bilinear integer optimization problem, and they propose a cutting-plane algorithm for the problem's solution.

Clearly numerous variants of this game are possible. For example, the exclusivity agreements may be available to the interdictor, and the interdictor may have a different objective, such as profit maximization. These variants provide a set of interesting and high-value research directions. Moreover, additional high-value directions are available at the intersection of competition and interdiction that directly consider a multitude of ways in which competitors may impede each other's ability to meet its objectives.

5 Connection to Robust Optimization and Survivable Network Design

This chapter has thus far given a sense of the breadth of interdiction application research, with a particular emphasis on network survivability (regardless of whether the networks represent physical systems or are abstractions of, e.g., supply chains or transportation operations). A natural question thus regards the relationship of interdiction models to methods designed in parallel to interdiction research over the last few decades. This subsection discusses robust and survivable network design approaches to constructing resilient networks. (Note that this discussion is *not* intended to cover the range and depth of these fields but rather to connect the interdiction literature discussed so far to the principles established in these bodies of literature.)

5.1 Robust Optimization

The theme of modern robust optimization research is to elegantly accommodate data uncertainty arising in mathematical programs. Essentially, robust optimization models define an *uncertainty set* over which data exists (as opposed to the point estimates of data in deterministic models) and seek a solution that is feasible to all constraints, regardless of the actual outcome of the data. The objective is measured according to the worst-case observation of data with respect to the given solution. For instance, if the objective is to minimize cost, then a solution's objective is the maximum possible objective function value over all objective data that belong to the uncertainty set.

The result is a fairly pessimistic model that minimizes the maximum objective function value that could arise (again in case of a minimization problem), with a guarantee of feasibility, all under the assumption that the problem data lies in the uncertainty set given above. (An alternative viewpoint is given in the field of stochastic programming (see, e.g., [15, 35]), which typically seeks a solution that is sufficiently likely to be feasible. The objective usually optimizes the expected objective function value over a given distribution of data.) In order to control the degree of robustness, the key is to strategically control the uncertainty sets that are modeled for robust optimization problems, given the actual distribution of uncertain data. See [7–10, 25, 26] for foundational work on robust optimization, an introductory chapter in [56], and the excellent recent reference book [6] and tutorial [13] treatment of the subject.

To compare the interdiction and robust optimization viewpoints, consider the following linear programming problem:

$$\min \mathbf{c}^\top \mathbf{x} \quad (31a)$$

$$\text{s.t. } A\mathbf{x} \geq \mathbf{b} \quad (31b)$$

$$\mathbf{x} \geq \mathbf{0}, \quad (31c)$$

where A is an $m \times n$ matrix and where \mathbf{c} and \mathbf{b} have conforming dimensions, but where A is uncertain. In fact, the assumption that \mathbf{c} and \mathbf{b} are deterministic, with uncertainty limited to A , can be made without loss of generality by suitable variable substitutions. (Much of the robust optimization literature assumes a lower bound l on the \mathbf{x} -variables, where l is not necessarily nonnegative; however, the brief exposition here is simplified with the assumption that \mathbf{x} is nonnegative, and moreover, any linear programming problem can easily be transformed to the form (31).) Suppose that A belongs to the uncertainty set \mathcal{U} . The *robust counterpart* formulation guarantees feasibility of \mathbf{x} to (31b) for every $A \in \mathcal{U}$, that is, by changing (31b) to $\bar{A}\mathbf{x} \geq \mathbf{b}$, $\forall \bar{A} \in \mathcal{U}$.

There is now potentially an infinite number of constraints to the problem, depending on \mathcal{U} . However, this problem can be cast as a finite optimization problem by focusing only on certain forms of the set \mathcal{U} and employing mathematical

programming techniques reminiscent of those developed for interdiction models. It is first useful to state the following observations regarding robust optimization problems:

- Let \mathbf{a}^i denote the i th row vector of A and define \mathcal{U}_i as the set of all A -matrices whose i th row is \mathbf{a}^i . Because \mathbf{x} must be feasible for any $A \in \mathcal{U}$, the following constraints are valid:

$$\mathbf{a}^i \mathbf{x} \geq b_i \quad \forall i = 1, \dots, m, \quad \mathbf{a}^i \in \mathcal{U}_i. \quad (32)$$

Therefore, optimizing over the uncertainty set \mathcal{U} is equivalent to optimizing over the possibly larger uncertainty set $\mathcal{U}_1 \times \dots \times \mathcal{U}_m$. This is referred to as row-wise uncertainty, in which randomness is independent over the rows of A .

- Similarly, optimization over \mathcal{U} is equivalent to optimization over the convex hull of \mathcal{U} and to the closure of this set (see [42]).
- If \mathcal{U} independently constrains the *columns* of A (referred to as column-wise uncertainty), then randomness is independent over both rows and columns. Randomness is thus independent over each *element* of A , and the robust counterpart replaces each element of A with its smallest possible value (in the case of \geq constraints). See [62] for an early exposition of this case. The remaining discussion in this section regards the more complex case of intercolumn dependence in \mathcal{U} .

For robust optimization problems, one can again employ minmax (or maxmin) optimization techniques as done in the context of Stackelberg games. Here, the first player (leader) selects a solution $\hat{\mathbf{x}}$. The follower will, for each row $i = 1, \dots, m$, minimize $\mathbf{a}^i \hat{\mathbf{x}}$ over all $\mathbf{a}^i \in \mathcal{U}_i$ with the aim of forcing $\hat{\mathbf{x}}$ to be infeasible. Just as in the development of the interdiction models of Sect. 2, the inner problem can be dualized to uncover a single-level optimization problem. However, the order in which the defender (now the leader) and attacker (now the follower) play is reversed. The attacker's problem is now quite simple, which admits a far simpler optimization model.

Consider the case of *budgeted uncertainty* [12], where $a_{ij} \in [\bar{a}_{ij} - \hat{a}_{ij}, \bar{a}_{ij}]$ for all elements of A and no more than some specified (nonnegative) value Γ_i of the coefficients a_{i1}, \dots, a_{in} are allowed to simultaneously deviate from \bar{a}_{ij} , for all $i = 1, \dots, m$. To minimize the left-hand side of the structural constraints, given the leader's solution $\hat{\mathbf{x}}$, the follower minimizes

$$\sum_{j=1}^n \bar{a}_{ij} \hat{x}_j - \sum_{j=1}^n \hat{a}_{ij} \hat{x}_j z_{ij} \quad \forall i = 1, \dots, m, \quad (33)$$

which is equivalent to solving

$$\max \quad \sum_{j=1}^n \hat{a}_{ij} \hat{x}_j z_{ij} \quad (34a)$$

$$\text{s.t. } \sum_{j=1}^n z_{ij} \leq \Gamma_i \quad (34\text{b})$$

$$0 \leq z_{ij} \leq 1 \quad \forall j = 1, \dots, n. \quad (34\text{c})$$

Associating the dual variable π_i with (34b) and μ_{ij} with the constraint $z_{ij} \leq 1$, $\forall j = 1, \dots, n$, the dual formulation of (34) is

$$\min \Gamma_i \pi_i + \sum_{j=1}^n \mu_{ij} \quad (35\text{a})$$

$$\text{s.t. } \pi_i + \mu_{ij} \geq \hat{a}_{ij} \hat{x}_j \quad \forall j = 1, \dots, n \quad (35\text{b})$$

$$\pi_i \geq 0, \mu_{ij} \geq 0 \quad \forall j = 1, \dots, n. \quad (35\text{c})$$

Because $\sum_{j=1}^n \hat{a}_{ij} \hat{x}_j z_{ij}$ is given by the smallest possible value of (35a) and is constrained by (35b) and (35c), each constraint (32) is equivalent to

$$\bar{a}^i x - \left(\Gamma_i \pi_i + \sum_{j=1}^n \mu_{ij} \right) \geq b_i \quad \forall i = 1, \dots, m \quad (36\text{a})$$

$$\pi_i + \mu_{ij} \geq \hat{a}_{ij} \hat{x}_j \quad \forall j = 1, \dots, n \quad (36\text{b})$$

$$\pi, \mu \geq 0. \quad (36\text{c})$$

Note that the robust counterpart is also a linear program, as opposed to the models of Sect. 2, which become nonlinear (although they can be linearized when at least one of the bilinear terms is restricted to be binary). This class of robust models is therefore considerably easier to solve than the previously developed interdiction models. However, robust models are also far more pessimistic, as they permit the attacker to essentially act not only with full knowledge of any defender design decisions but also with full knowledge of the defender's *recourse* decisions. By contrast, in the interdiction models, the defender has the capability of choosing its recourse based on the attacker's actions. The most appropriate model for a given application should therefore take into consideration to what extent recourse decisions are allowed after an attack is made. (It is worth noting that some progress has also recently been made on robust optimization with recourse, see, e.g., [2, 13, 14].)

Three other observations are relevant before continuing. First, the foregoing analysis can be applied when A belongs to a general polyhedral set. Second, much of the traditional robust optimization theory regards the case in which \mathcal{U} is ellipsoidal, and for this case, the robust counterpart becomes a nonlinear (but still convex) program. See [11] for analogous derivations of robust counterparts with respect to more general uncertainty sets. Third, when the "true" range of possible A -values

is restricted to lie in a constrained uncertainty set (e.g., budgeted or ellipsoidal uncertainty), it is possible to derive bounds on the probability that the constraint is in fact violated [6].

5.2 Survivable Network Design

Finally, it is worthwhile to note the relationship among the perspectives of other survivable network design research and interdiction studies. The survivable network design literature encompasses a broad study of network design and operations problems that remain functional in the face of failures. Hence, it is harder to put as precise a definition on this field (which some would consider to be encompassed by interdiction and/or robust optimization) than on the previously discussed lines of research.

As a starting point, one may consider the case in which robustness is defined with respect to a collection of possible failure scenarios. Each failure scenario defines a simultaneous realization of uncertain data, and design decisions must be feasible with respect to every possible failure scenario. (The objective would thus be defined as the worst-case scenario in a robust modeling framework and perhaps as an expected cost from a stochastic programming perspective.) Kouvelis and Yu [37] show that with just two scenarios, the shortest path problem with nonnegative data becomes NP-hard, in contrast to the polynomial solvability of the deterministic version of this problem. Garg and Smith [27] provide another version of this problem, in which many failure scenarios exist, each of which represents a set of possible link failures. The authors employ Benders' decomposition to mitigate the computational effort imposed by encountering a large set of failure scenarios.

Several studies, for example, [22, 33, 63], assume that network failures are rare, especially when applied to accidental failure scenarios on equipment whose functionality is very reliable but will occasionally fail. Polyhedral integer programming studies by Grötschel and Monma [30] and Grötschel et al. [31] regard connectivity problems on networks. Other network systems implicitly guarantee survivability with respect to single points of failure. In the past two decades, synchronous optical networking (SONET) ring networks [66, 67] received much research attention from the combinatorial optimization community [5, 28, 29, 38, 54, 60, 64]. These network design problems can be seen within either the interdiction or robust viewpoint. From the interdiction perspective, the attacker destroys any link in an attempt to maximize damage (dropped data), while the recourse action of the defender reroutes data in response to the attack. From the robust perspective, the design decisions build the network with the implicit understanding that *any* interdiction action can be mitigated in the future (without explicitly stating variables and constraints that prove the existence of such recourse actions).

6 Conclusion

Although network interdiction models have existed for some time, the past decade has seen an accelerated development within this field. This has resulted from a confluence of several factors. First, several major destructive events have occurred since the year 2000, including the terrorist attacks on the World Trade Center on September 11, 2001, and Hurricane Katrina in New Orleans in 2005. These events, and the state of the world in their aftermath, motivated researchers to intensify efforts on problems involving major catastrophic disruptions. Second, the operations research community and its available tools have seen advances in the development of models for handling singular events, uncertainties, and relatively large-scale network optimization problems in the past 20 years. Third, computing power and availability has increased to a degree that has permitted an ability to solve large-scale problems on desktop computers.

In a sense, however, the field of network interdiction modeling is in its early stages. The models discussed in this chapter remain somewhat limited in their abilities to handle dynamic changes in the data that drive these models. That is, the majority of the models this chapter has described take a snapshot of a problem in time and solve that snapshot or single-period version of the problem. Moreover, the limited action sets and information symmetry assumptions contained in the majority of existing models can limit their practical application. Finally, multiple interdictors with varying interdiction mechanisms serve to compound problem complexity. Accounting for a richer set of assumptions on the dynamic evolution of data, the potential actions of multiple interdictors, and various states of information will require greater modeling sophistication and computing power than currently available. Thus, while great advances have been made in both computing and operations research methods, a great deal of opportunity exists for developing models and methods for solving realistic versions of dynamic problems.

Cross-References

- ▶ [Combinatorial Optimization in Transportation and Logistics Networks](#)
 - ▶ [Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs](#)
 - ▶ [Network Optimization](#)
-

Recommended Reading

1. R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, Upper Saddle River, 1993)
2. A. Atamtürk, M. Zhang, Two-stage robust network flow and design under demand uncertainty. *Oper. Res.* **55**(4), 662–673 (2007)
3. H. Bayrak, M.D. Bailey, Shortest path network interdiction with asymmetric information. *Networks* **52**(3), 133–140 (2008)

4. M.S. Bazaraa, J.J. Jarvis, H.D. Sherali, *Linear Programming and Network Flows*, 2nd edn. (Wiley, New York, 1990)
5. G. Belvaux, N. Boissin, A. Sutter, L.A. Wolsey, Optimal placement of add/drop multiplexers: static and dynamic models. *Eur. J. Oper. Res.* **108**, 26–35 (1998)
6. A. Ben-Tal, L. El-Ghaoui, A. Nemirovski, *Robust Optimization* (Princeton University Press, Princeton, 2009)
7. A. Ben-Tal, A. Nemirovski, Stable truss topology design via semidefinite programming. *SIAM J. Optim.* **7**, 991–1016 (1997)
8. A. Ben-Tal, A. Nemirovski, Robust convex optimization. *Math. Oper. Res.* **23**, 769–805 (1998)
9. A. Ben-Tal, A. Nemirovski, Robust solutions to uncertain linear programs. *Oper. Res. Lett.* **25**, 1–13 (1999)
10. A. Ben-Tal, A. Nemirovski, Robust solutions of linear programming problems contaminated with uncertain data. *Math. Program.* **88**(3), 411–424 (2000)
11. A. Ben-Tal, A. Nemirovski, Robust optimization – methodology and applications. *Math. Program. Ser. B* **92**, 453–480 (2002)
12. D. Bertsimas, M. Sim, The price of robustness. *Oper. Res.* **52**(1), 35–53 (2004)
13. D. Bertsimas, A. Thiele, Robust and data-driven optimization: Modern decision-making under uncertainty, in *Tutorials in Operations Research: Models, Methods, and Applications for Innovative Decision Making*, ed. by M.P. Johnson, B. Norman, N. Secomandi. INFORMS, 2006, pp. 95–122
14. D. Bertsimas, A. Thiele, A robust optimization approach to inventory theory. *Oper. Res.* **54**(1), 150–168 (2006)
15. J.R. Birge, F.V. Louveaux, *Introduction to Stochastic Programming* (Springer, New York, 1997)
16. G.G. Brown, W.M. Carlyle, J. Salmerón, K. Wood, Analyzing the vulnerability of critical infrastructure to attack and planning defenses, in *Tutorials in Operations Research: Emerging Theory, Methods, and Applications*, ed. by H.J. Greenberg, J.C. Smith. INFORMS, Hanover, 2005, pp. 102–123
17. G.G. Brown, W.M. Carlyle, J. Salmerón, K. Wood, Defending critical infrastructure. *Interfaces* **36**(6), 530–544 (2006)
18. S.D. Cartwright, Supply chain interdiction and corporate warfare. *J. Bus. Strat.* **21**(2), 30–35 (2000)
19. R.L. Church, M.P. Scaparra, The r -interdiction median problem with fortification. *Geogr. Anal.* **39**, 129–146 (2007)
20. R.L. Church, M.P. Scaparra, R. Middleton, Identifying critical infrastructure: The median and covering interdiction models. *Ann. Assoc. Am. Geogr.* **94**(3), 491–502 (2004)
21. K.J. Cormican, D.P. Morton, R.K. Wood, Stochastic network interdiction. *Oper. Res.* **46**(2), 184–197 (1998)
22. G. Dahl, M. Stoer, A cutting plane algorithm for multicommodity survivable network design problems. *INFORMS J. Comput.* **10**(1), 1–11 (1998)
23. A. Delgadillo, J.M. Arroyo, N. Alguacil, Analysis of electric grid interdiction with line switching. *IEEE Trans. Power Syst.* **25**(2), 633–641 (2010)
24. H.A. Eiselt, G. Laporte, J.-F. Thisse, Competitive location models: A framework and bibliography. *Transport. Sci.* **27**(1), 44–54 (1993)
25. L. El-Ghaoui, H. Lebret, Robust solutions to least-square problems with uncertain data matrices. *SIAM J. Matrix Anal. Appl.* **18**, 1035–1064 (1997)
26. L. El-Ghaoui, F. Oustry, H. Lebret, Robust solutions to uncertain semidefinite programs. *SIAM J. Optim.* **9**, 33–52 (1998)
27. M. Garg, J.C. Smith, Models and algorithms for the design of survivable multicommodity flow networks with general failure scenarios. *Omega* **36**(6), 1057–1071 (2008)
28. O. Goldschmidt, D.S. Hochbaum, A. Levin, E.V. Olinick, The SONET edge-partition problem. *Networks* **41**(1), 13–23 (2003)
29. O. Goldschmidt, A. Laugier, E.V. Olinick, SONET/SDH ring assignment with capacity constraints. *Discrete Appl. Math.* **129**(1), 99–128 (2003)

30. M. Grötschel, C.L. Monma, Integer polyhedra arising from certain network design problems with connectivity constraints. *SIAM J. Discrete Math.* **3**(4), 502–523 (1990)
31. M. Grötschel, C.L. Monma, M. Stoer, Polyhedral and computational investigations for designing communication networks with high survivability requirements. *Oper. Res.* **43**(6), 1012–1024 (1995)
32. H. Hotelling, Stability in competition. *Econ. J.* **39**(153), 41–57 (1929)
33. M. Iri, Extension of the maximum-flow minimum-cut theorem to multicommodity flows. *J. Oper. Res. Soc. Jpn.* **13**(3), 129–135 (1971)
34. U. Janjarassuk, J. Linderroth, Reformulation and sampling to solve a stochastic network interdiction problem. *Networks* **52**(3), 120–132 (2008)
35. P. Kall, S.W. Wallace, *Stochastic Programming* (Wiley, Chichester, 1994)
36. P. Kotler, R. Singh, Marketing warfare in the 1980s. *J. Bus. Strat.* **1**(3), 30–41 (1980)
37. P. Kouvelis, G. Yu, *Robust Discrete Optimization and Its Applications* (Kluwer Academic, Norwell, 1997)
38. Y. Lee, H.D. Sherali, J. Han, S. Kim, A branch-and-cut algorithm for solving an intra-ring synchronous optical network design problem. *Networks* **35**(3), 223–232 (2000)
39. A. Malaviya, C. Rainwater, T. Sharkey, Multi-period network interdiction models with applications to city-level drug enforcement. Working paper, Department of Industrial and Systems Engineering, Rensselaer Polytechnic Institute, Troy, 2011
40. D.P. Morton, F. Pan, K.J. Saeger, Models for nuclear smuggling interdiction. *IIE Trans.* **39**(1), 3–14 (2007)
41. D.P. Morton, F. Pan, J.C. Smith, K. Sullivan, Inequalities for an asymmetric stochastic network interdiction problem. Technical report, Department of Industrial and Systems Engineering, University of Florida, Gainesville, 2011
42. A. Nemirovski, Lectures on robust convex optimization. <http://www2.isye.gatech.edu/~nemirovs/>
43. F. Pan, W. Charlton, D.P. Morton, Interdicting smuggled nuclear material, in *Network Interdiction and Stochastic Integer Programming*, ed. by D.L. Woodruff (Kluwer Academic, Boston), 2003, pp. 1–20
44. F. Pan, D.P. Morton, Minimizing a stochastic maximum-reliability path. *Networks* **52**, 111–119 (2008)
45. M. Parlar, Game theoretic analysis of the substitutable product inventory problem with random demands. *Nav. Res. Logist.* **35**(3), 397–409 (1988)
46. A.E. Pearson, C.L. Irwin, Coca-Cola vs. Pepsi-Cola (A). Case study, Harvard Business School Case No. 9-387-108, Cambridge, 1988
47. F. Plastria, Static competitive facility location: An overview of optimisation approaches. *Eur. J. Oper. Res.* **129**(3), 461–470 (2001)
48. M. Prince, J.C. Smith, J. Geunes, Optimizing exclusivity agreements in a three-stage procurement game. Working paper, Department of Industrial and Systems Engineering, The University of Florida, Gainesville, 2011
49. C.M. Rocco, J.E. Ramirez-Marquez, A bi-objective approach for shortest-path network interdiction. *Comput. Ind. Eng.* **59**, 232–240 (2010)
50. J.O. Royset, R.K. Wood, Solving the bi-objective maximum-flow network-interdiction problem. *INFORMS J. Comput.* **19**, 175–184 (2007)
51. J. Salmerón, K. Wood, R. Baldick, Analysis of electric grid security under terrorist threat. *IEEE Trans. Power Syst.* **19**, 905–912 (2004)
52. J. Salmerón, K. Wood, R. Baldick, Worst-case interdiction analysis of large-scale electric power grids. *IEEE Trans. Power Syst.* **24**, 96–104 (2009)
53. M.P. Scaparra, R.L. Church, A bilevel mixed-integer program for critical infrastructure protection planning. *Comput. Oper. Res.* **35**(6), 1905–1923 (2008)
54. H.D. Sherali, J.C. Smith, Y. Lee, Enhanced model representations for an intra-ring synchronous optical network design problem allowing demand splitting. *INFORMS J. Comput.* **12**(4), 284–298 (2000)

55. J.C. Smith, Basic interdiction models, in *Wiley Encyclopedia of Operations Research and Management Science*, ed. by J. Cochran (Wiley, Hoboken, 2011)
56. J.C. Smith, S. Ahmed, Introduction to robust optimization, in *Wiley Encyclopedia of Operations Research and Management Science*, ed. by J. Cochran (Wiley, Hoboken, 2011)
57. J.C. Smith, C. Lim, Algorithms for network interdiction and fortification games, in *Pareto Optimality, Game Theory and Equilibria*, ed. by A. Migdalas, P.M. Pardalos, L. Pitsoulis, A. Chinchuluun, *Nonconvex Optimization and Its Applications Series* (Springer, New York, 2008), pp. 609–644
58. J.C. Smith, C. Lim, A. Alptekinoğlu, New product introduction against a predator: A bilevel mixed-integer programming approach. *Nav. Res. Logist.* **56**, 714–729 (2009)
59. J.C. Smith, C. Lim, F. Sudargho, Survivable network design under optimal and heuristic interdiction scenarios. *J. Global Optim.* **38**, 181–199 (2007)
60. J.C. Smith, A.J. Schaefer, J.W. Yen, A stochastic integer programming approach to solving a synchronous optical network ring design problem. *Networks* **44**(1), 12–26 (2004)
61. L.V. Snyder, M.P. Scaparra, M.S. Daskin, R.L. Church, Planning for disruptions in supply chain networks, in *Tutorials in Operations Research: Models, Methods, and Applications for Innovative Decision Making*, ed. by M.P. Johnson, B. Norman, N. Secomandi. (INFORMS, 2006), pp. 234–257
62. A.L. Soyster, Convex programming with set-inclusive constraints and applications to inexact linear programming. *Oper. Res.* **21**, 1154–1157 (1973)
63. M. Stoer, G. Dahl, A polyhedral approach to multicommodity survivable network design. *Numer. Math.* **68**(1), 149–167 (1994)
64. A. Sutter, F. Vanderbeck, L.A. Wolsey, Optimal placement of add/drop multiplexers: heuristic and exact algorithms. *Oper. Res.* **46**(5), 719–728 (1998)
65. H. von Stackelberg, *The Theory of the Market Economy* (William Hodge, London, 1952)
66. T.-H. Wu, *Fiber Network Service Survivability* (Artech House, Boston, MA, 1992)
67. T.-H. Wu, M.E. Burrowes, Feasibility study of a high-speed SONET self-healing ring architecture in future interoffice networks. *IEEE Comm. Mag.* **28**(11), 33–43 (1990)

Network Optimization

Samir Khuller and Balaji Raghavachari

Contents

1	Introduction.....	1990
2	The Matching Problem.....	1991
2.1	Matching Problem Definitions.....	1992
2.2	Matchings and Augmenting Paths.....	1992
2.3	Bipartite Matching Algorithm.....	1993
2.4	Sample Execution.....	1994
2.5	Analysis.....	1995
2.6	The Matching Problem in General Graphs.....	1995
2.7	Assignment Problem.....	1996
3	Applications of Matchings.....	1998
3.1	Two Processor Scheduling.....	1999
3.2	Weighted Edge-Cover Problem.....	2000
3.3	Chinese Postman Problem.....	2001
4	The Network Flow Problem.....	2002
4.1	Network Flow Problem Definitions.....	2002
4.2	Blocking Flows.....	2006
5	The Min-Cut Problem.....	2007
5.1	Finding an $s-t$ Min Cut.....	2008
5.2	Finding All-Pair Min Cuts.....	2008
6	Applications of Network Flows (and Min Cuts).....	2008
6.1	Connectivity.....	2008
6.2	Finding a Minimum-Weight Vertex Cover in Bipartite Graphs.....	2009
6.3	Maximum-Weight Closed Subset in a Partial Order.....	2009
7	Finding Densest Subgraphs.....	2010
7.1	Algorithm for Densest Subgraph Without a Specified Subset.....	2010
7.2	Algorithm for Densest Subgraph with a Specified Subset C	2011

S. Khuller (✉)

Department of Computer Science, University of Maryland, College Park, MD, USA
e-mail: samir@cs.umd.edu

B. Raghavachari

Department of Computer Science, University of Texas at Dallas, Richardson, TX, USA
e-mail: rbk@utdallas.edu

8	Minimum-Cost Flows.....	2012
8.1	Min-Cost Flow Problem Definitions.....	2012
9	The Multi-Commodity Flow Problem.....	2014
9.1	Local Control Algorithm.....	2015
10	Minimum Spanning and Other Light-Weight Trees.....	2016
10.1	Minimum-Weight Branchings.....	2017
10.2	Spanners.....	2018
10.3	Light Approximate Shortest Path Trees.....	2019
11	Coloring Problems.....	2020
11.1	Vertex Coloring.....	2020
11.2	Edge Coloring.....	2020
12	Approximation Algorithms for Hard Problems.....	2021
13	Conclusion.....	2022
	Further Reading.....	2023
	Glossary.....	2023
	Cross-References.....	2024
	Recommended Reading.....	2025

Abstract

Many real-life problems can be modeled as optimization problems in networks. Examples include finding shortest paths, scheduling classes in classrooms, and finding the vulnerability of networks to disconnection because of link and node failures. In this chapter, a number of interesting problems and their algorithms are surveyed. Some of the problems studied are matching, weighted matching, maximum flow in networks, blocking flows, minimum cuts, densest subgraphs, minimum-cost flows, multi-commodity flows, minimum spanning trees, spanners, Light approximate shortest path trees, and graph coloring. Some interesting applications of matching, namely, two processor scheduling, edge covers, and the Chinese postman problem, are also discussed.

1 Introduction

The optimization of a given objective, while working with limited resources, is a fundamental problem that occurs in all walks of life, and therefore, optimization problems have been widely studied in Computer Science and Operations Research. Problems in discrete optimization vary widely in their complexity, and efficient solutions are derived for many of these problems by studying their combinatorial structure and understanding their fundamental properties. In this chapter, a number of problems arising in networks and advanced algorithmic techniques for solving them are studied.

One of the basic topics in this field is network flow. The related optimization problems are fundamentally important. They have applications in various disciplines and provide a fundamental framework for solving problems. For example, the problem of efficiently moving entities, such as bits, people, or products, from one place to another in an underlying network, can be modeled as a network

flow problem. Applications of flow include matching, scheduling, and connectivity problems. Because of its central role in the fields of Operations Research and Computer Science, considerable emphasis has been placed on the design of efficient algorithms for solving it.

Many practical combinatorial problems such as the assignment problem, and the problem of finding the susceptibility of networks to failure due to faulty links or nodes, are special instances of the max-flow problem. There are several variations and generalizations of the max-flow problem including the vertex capacitated max-flow problem, the minimum-cost max-flow problem, the minimum-cost circulation problem, and the multi-commodity flow problem.

In this chapter, a number of interesting and important optimization problems in networks are surveyed. Some of them such as network flow and matchings have been known for several decades, while others have recent results. The chapter starts with a discussion of matchings in Sect. 2. The single-commodity maximum flow problem is introduced in Sect. 4. The minimum-cut problem is discussed in Sect. 5. In Sect. 6 some applications of network flows are discussed. Section 7 discusses an interesting application of max flows to obtain the densest subgraph of a graph. Section 8 discusses the min-cost flow problem. The multi-commodity flow problem is discussed in Sect. 9. Section 10 introduces the problem of computing optimal branchings. Section 11 discusses the problem of coloring the edges and vertices of a graph. Section 12 discusses approximation algorithms for NP-hard problems. At the end, references to current research in graph algorithms are provided.

2 The Matching Problem

First, the matching problem, which is a special case of the max-flow problem, is introduced. Then, the assignment problem, a generalization of the matching problem, is studied.

A matching is a set of edges such that no two of them are incident to the same vertex. Finding a matching of maximum cardinality within a given graph is called the matching problem. A matching is perfect if every vertex of the graph is matched. An entire book [35] has been devoted to the study of various aspects of the matching problem, ranging from necessary and sufficient conditions for the existence of perfect matchings to algorithms for solving the matching problem. Many of the basic algorithms such as breadth-first search, depth-first search, and shortest paths play important roles in developing efficient implementations for network flow and matching algorithms.

The maximum matching problem is discussed in detail only for bipartite graphs. The same principles are used to design efficient algorithms to solve the matching problem in arbitrary graphs. The algorithms for general graphs are complex due to the presence of odd-length cycles called blossoms, and the reader is referred to [38, Chap. 10] or [43, Chap. 9] for detailed treatments on how blossoms are handled.

2.1 Matching Problem Definitions

Given a graph $G = (V, E)$, a matching M is a subset of the edges such that no two edges in M share a common vertex. In other words, the problem is that of finding a set of independent edges that have no incident vertices in common. The cardinality of M is usually referred to as its size.

The following terms are defined with respect to a matching M . The edges in M are called matched edges, and edges not in M are called free edges. Likewise, a vertex is a matched vertex if it is incident to a matched edge. A free vertex is one that is not matched. The mate of a matched vertex v is its neighbor w that is at the other end of the matched edge incident to v . A matching is called perfect if all vertices of the graph are matched in it. When the number of vertices is odd, one vertex is permitted to remain unmatched. The objective of the maximum matching problem is to maximize $|M|$, the size of the matching. If the edges of the graph have weights, then the weight of a matching is defined to be the sum of the weights of its edges. A path $p = [v_1, v_2, \dots, v_k]$ is called an alternating path if the edges (v_{2j-1}, v_{2j}) , $j = 1, 2, \dots$ are free and the edges (v_{2j}, v_{2j+1}) , $j = 1, 2, \dots$ are matched. An augmenting path $p = [v_1, v_2, \dots, v_k]$ is an alternating path in which both v_1 and v_k are free vertices. Observe that an augmenting path is defined with respect to a specific matching. The symmetric difference of a matching M and an augmenting path P , $M \oplus P$, is defined to be $(M - P) \cup (P - M)$. It can be shown that $M \oplus P$ is also a matching. [Figure 1](#) shows an augmenting path $p = [a, b, c, d, g, h]$ with respect to the given matching. The symmetric difference operation can also be used as above between two matchings. In this case, the resulting graph consists of a collection of paths and cycles with alternate edges from each matching.

2.2 Matchings and Augmenting Paths

The following theorem gives necessary and sufficient conditions for the existence of a perfect matching in a bipartite graph.

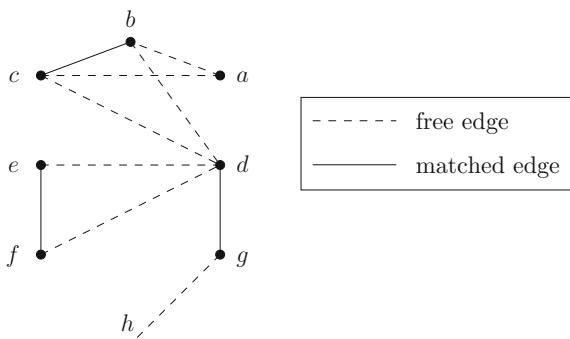


Fig. 1 An augmenting path p with respect to a matching

Theorem 1 (Hall's Theorem) A bipartite graph $G = (X, Y, E)$ with $|X| = |Y|$ has a perfect matching if and only if $\forall S \subseteq X, |N(S)| \geq |S|$, where $N(S) \subseteq Y$ is the set of vertices that are neighbors of some vertex in S .

Although the above theorem captures exactly the conditions under which a given bipartite graph has a perfect matching, it does not lead to an algorithm for finding perfect matchings directly. The following lemma shows how an augmenting path with respect to a given matching can be used to increase the size of a matching. An efficient algorithm will be described later that uses augmenting paths to construct a maximum matching incrementally.

Lemma 1 Let P be the edges on an augmenting path $p = [v_1, \dots, v_k]$ with respect to a matching M . Then, $M' = M \oplus P$ is a matching of cardinality $|M| + 1$.

Proof Since P is an augmenting path, both v_1 and v_k are free vertices in M . The number of free edges in P is one more than the number of matched edges in it. The symmetric difference operator replaces the matched edges of M in P by the free edges in P . Hence the size of the resulting matching, $|M'|$, is one more than $|M|$.

The following theorem provides a necessary and sufficient condition for a given matching M to be a maximum matching.

Theorem 2 A matching M in a graph G is a maximum matching if and only if there is no augmenting path in G with respect to M .

Proof If there is an augmenting path with respect to M , then M cannot be a maximum matching, since by Lemma 1 there is a matching whose size is larger than that of M . To prove the converse, it is shown that if there is no augmenting path with respect to M , then M is a maximum matching. Suppose that there is a matching M' such that $|M'| > |M|$. Consider the subgraph of G induced by the edges $M \oplus M'$. Each vertex in this subgraph has degree at most two, since each node has at most one edge from each matching incident to it. Hence each connected component of this subgraph is either a path or a simple cycle. For each cycle, the number of edges of M is the same as the number of edges of M' . Since $|M'| > |M|$, one of the paths must have more edges from M' than from M . This path is an augmenting path in G with respect to the matching M , contradicting the assumption that there were no augmenting paths with respect to M .

2.3 Bipartite Matching Algorithm

High-Level Description: The algorithm starts with the empty matching $M = \emptyset$ and augments the matching in phases. In each phase, an augmenting path with respect to the current matching M is found, and it is used to increase the size of the matching.

An augmenting path, if one exists, can be found in $O(|E|)$ time, using a procedure similar to breadth-first search.

The search for an augmenting path proceeds from the free vertices. At each step, when a vertex in X is processed, all its unvisited neighbors are also searched. When a matched vertex in Y is considered, only its matched neighbor is searched. This search proceeds along a subgraph referred to as the Hungarian tree.

The algorithm uses a queue to hold vertices that are yet to be processed. Initially, all free vertices in X are placed in the queue. The vertices are removed one by one from the queue and processed as follows. In turn, when vertex v is removed from the queue, all edges incident to it are scanned. If it has a neighbor in the vertex set Y that is free, then the search for an augmenting path is successful; update the matching, and the algorithm proceeds to its next phase. Otherwise, add the mates of all the matched neighbors of v to the queue if they have never been added to the queue, and continue the search for an augmenting path. If the algorithm empties the queue without finding an augmenting path, its current matching is a maximum matching, and it terminates.

2.4 Sample Execution

[Figure 2](#) shows a sample execution of the matching algorithm. It depicts a partial matching and the structure of the resulting Hungarian tree. In this example, the search starts from the free vertex b . Its neighbors are v and z , and both are matched.

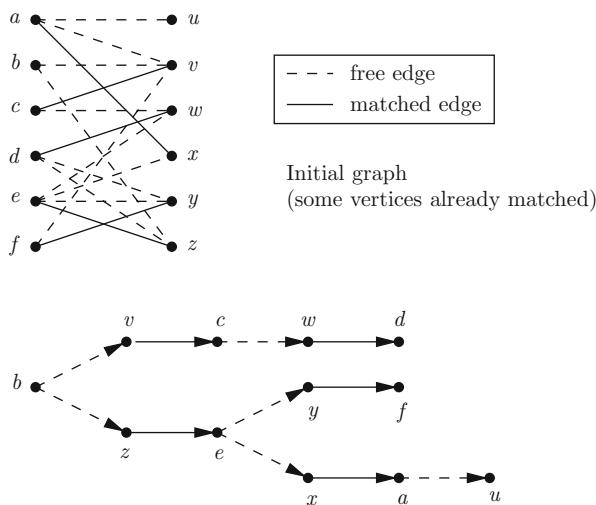


Fig. 2 Sample execution of matching algorithm

Their mates, c and e , are added to a queue Q . When c is explored, d is added to Q and then f and a . Since a has a free neighbor u , an augmenting path from vertex b to vertex u is found by the algorithm.

2.5 Analysis

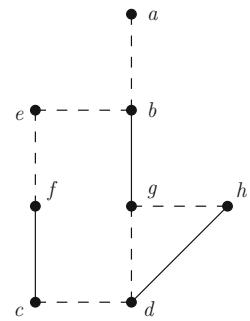
If there are augmenting paths with respect to the current matching, the algorithm will find at least one of them. Hence, when the algorithm terminates, the graph has no augmenting paths with respect to the current matching, and the current matching is optimal. Each iteration of the main while loop of the algorithm runs in $O(|E|)$ time. The construction of the auxiliary graph A and computation of the array $free$ also take $O(|E|)$ time. In each iteration, the size of the matching increases by one, and thus, there are at most $\min(|X|, |Y|)$ iterations of the while loop. Therefore, the algorithm solves the matching problem for bipartite graphs in time $O(\min(|X|, |Y|)|E|)$. Hopcroft and Karp (see [38]) showed how to improve the running time by finding a maximal set of disjoint augmenting paths in a single phase in $O(|E|)$ time. They also proved that the algorithm runs in only $O(\sqrt{|V|})$ phases, yielding a worst-case running time of $O(\sqrt{|V|}|E|)$.

2.6 The Matching Problem in General Graphs

The techniques used to solve the matching problem in bipartite graphs do not extend directly to non-bipartite graphs. The notion of augmenting paths and their relation to maximum matchings (Theorem 2) remain the same. Therefore, it is still possible to use the natural algorithm of starting with an empty matching and increasing its size repeatedly with augmenting paths until no augmenting path exists in the graph. But the problem of finding augmenting paths in non-bipartite graphs is harder. The main trouble is due to odd-length cycles known as blossoms that appear along alternating paths explored by the algorithm when it is looking for augmenting paths. This difficulty is illustrated by the example in Fig. 3. The search for an augmenting path from an unmatched vertex, such as e , could go through the following sequence of vertices $[e, b, g, d, h, g, b, a]$. Even though the augmenting path satisfies the “local” conditions for being an augmenting path, it is not a valid augmenting path since it is not simple. The reason for this is that the odd-length cycle (g, d, h, g) causes the path to “fold” on itself – a problem that does not arise in the bipartite case. In fact, the matching does contain a valid augmenting path $[e, f, c, d, h, g, b, a]$. Not all odd cycles cause this problem, but odd cycles that are as dense in matched edges as possible, that is, it depends on the current matching. By “shrinking” blossoms to single nodes, the algorithm can get rid of them [38]. Subsequent work focussed on efficient implementation of this method.

Edmonds (see [38]) gave the first polynomial-time algorithm for solving the maximum matching problem in general graphs. The current fastest algorithm for this

Fig. 3 Difficulty in dealing with blossoms



problem is due to Micali and Vazirani [36] and their algorithm runs in $O(|E|\sqrt{|V|})$ steps, which are the same bound obtained by the Hopcroft-Karp algorithm for finding a maximum matching in bipartite graphs.

2.7 Assignment Problem

In this section, the assignment problem – that of finding a maximum-weight matching in a given bipartite graph in which edges are given nonnegative weights – is studied. There is no loss of generality in assuming that the graph is a complete bipartite graph, since zero-weight edges may be added between pairs of vertices that are nonadjacent in the original graph without affecting the weight of a maximum-weight matching. The minimization version of the weighted version is the problem of finding a minimum-weight perfect matching in a complete bipartite graph. Both versions of the weighted matching problem are equivalent, and it is shown below how to reduce the minimum-weight perfect matching to maximum-weight matching. Choose a constant W that is larger than the weight of any edge, and assign each edge a new weight of $w'(e) = W - w(e)$. Observe that maximum-weight matchings with the new weight function are minimum-weight perfect matchings with the original weights.

In this section, attention is restricted to the study of the maximum-weight matching problem for bipartite graphs. Similar techniques have been used to solve the maximum-weight matching problem in arbitrary graphs (see [33, 38]).

The input is a complete bipartite graph $G = (X, Y, X \times Y)$, and each edge e has a nonnegative weight of $w(e)$. The following algorithm is known as the Hungarian method (see [1, 35, 38]). The method can be viewed as a primal-dual algorithm in the framework of linear programming [38]. No knowledge of linear programming is assumed here.

A feasible vertex-labeling ℓ is defined to be a mapping from the set of vertices in G to the real numbers such that for each edge (x_i, y_j) , the following condition holds:

$$\ell(x_i) + \ell(y_j) \geq w(x_i, y_j).$$

The following can be verified to be a feasible vertex labeling. For each vertex $y_j \in Y$, set $\ell(y_j)$ to be 0, and for each vertex $x_i \in X$, set $\ell(x_i)$ to be the maximum weight of an edge incident to x_i :

$$\begin{aligned}\ell(y_j) &= 0, \\ \ell(x_i) &= \max_j w(x_i, y_j).\end{aligned}$$

The equality subgraph, G_ℓ , is defined to be the spanning subgraph of G which includes all vertices of G but only those edges (x_i, y_j) which have weights such that

$$\ell(x_i) + \ell(y_j) = w(x_i, y_j).$$

The connection between equality subgraphs and maximum-weighted matchings is established by the following theorem.

Theorem 3 *If the equality subgraph, G_ℓ , has a perfect matching, M^* , then M^* is a maximum-weight matching in G .*

Proof Let M^* be a perfect matching in G_ℓ . By definition,

$$w(M^*) = \sum_{e \in M^*} w(e) = \sum_{v \in X \cup Y} \ell(v).$$

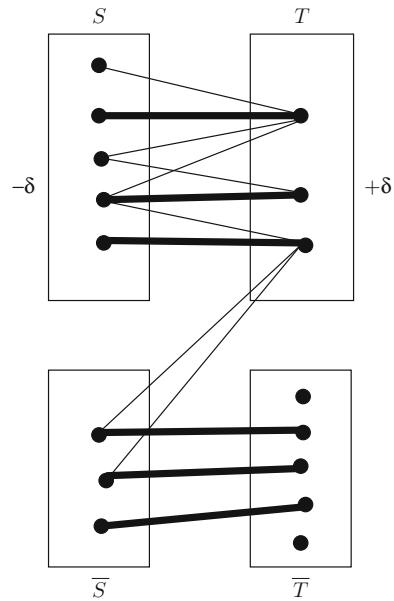
Let M be any perfect matching in G . Then,

$$w(M) = \sum_{e \in M} w(e) \leq \sum_{v \in X \cup Y} \ell(v) = w(M^*).$$

Hence, M^* is a maximum-weight perfect matching.

High-Level Description: The above theorem is the basis of the following algorithm for finding a maximum-weight matching in a complete bipartite graph. The algorithm starts with a feasible labeling, then computes the equality subgraph and a maximum cardinality matching in this subgraph. If the matching found is perfect, by [Theorem 3](#), the matching must be a maximum-weight matching, and the algorithm returns it as its output. Otherwise the matching is not perfect, and more edges need to be added to the equality subgraph by revising the vertex labels. The revision should ensure that edges from the current matching do not leave the equality subgraph. After more edges are added to the equality subgraph, the algorithm grows the Hungarian trees further. Either the size of the matching increases because an augmenting path is found, or a new vertex is added to the Hungarian tree. In the former case, the current phase terminates, and the algorithm starts a new phase since the matching size has increased. In the latter case, new nodes are added to the Hungarian tree. In $|X|$ phases, the tree includes all the nodes, and therefore, there are at most $|X|$ phases before the size of the matching increases.

Fig. 4 Sets S and T as maintained by the algorithm



It is now described in more detail how the labels are updated and which edges are added to the equality subgraph. Suppose M is a maximum matching found by the algorithm. Hungarian trees are grown from all the free vertices in X . Vertices of X (including the free vertices) that are encountered in the search are added to a set S , and vertices of Y that are encountered in the search are added to a set T . Let $\bar{S} = X - S$ and $\bar{T} = Y - T$. Figure 4 illustrates the structure of the sets S and T . Matched edges are shown in bold; the other edges are the edges in G_ℓ . Observe that there are no edges in the equality subgraph from S to \bar{T} , even though there may be edges from T to \bar{S} . The algorithm now revises the labels as follows. Decrease all the labels of vertices in S by a quantity δ (to be determined later), and increase the labels of the vertices in T by δ . This ensures that edges in the matching continue to stay in the equality subgraph. Edges in G (not in G_ℓ) that go from vertices in S to vertices in \bar{T} are candidate edges to enter the equality subgraph, since one label is decreasing and the other is unchanged. The algorithm chooses δ to be the smallest value such that some edge of $G - G_\ell$ enters the equality subgraph. Suppose this edge goes from $x \in S$ to $y \in \bar{T}$. If y is free, then an augmenting path has been found. On the other hand, if y is matched, the Hungarian tree is grown by moving y to T and its matched neighbor to S , and the process of revising labels is continued.

3 Applications of Matchings

Matchings lie at the heart of many optimization problems, and the problem has many applications. The name comes from the marriage problem, where the objective is to find a maximum number of compatible pairs of men and women for marriage,

given their pairwise compatibility information. Applications of matching include assigning workers to jobs, assigning a collection of jobs with precedence constraints to two processors such that the total execution time is minimized, determining the structure of chemical bonds in Chemistry, matching moving objects based on a sequence of snapshots, and localization of objects in space after obtaining information from multiple sensors (see [1]). A few applications of matching are discussed.

3.1 Two Processor Scheduling

First, the problem of scheduling jobs on two processors is studied. The original paper is by Fujii, Kasami, and Ninomiya [17].

There are two identical processors and a collection of n jobs that need to be scheduled. Each job requires unit time. There is a precedence graph associated with the jobs (also called the DAG). If there is an edge from i to j , then i must be finished before j is started by either processor. How should the jobs be scheduled on the two processors so that all the jobs are completed as quickly as possible. The jobs could represent courses that need to be taken (courses have prerequisites), and if a student is taking at most two courses in each semester, the question really is: How quickly can he graduate?

This is a good time to note that even though the two processor scheduling problem can be solved in polynomial time, the three processor scheduling problem is not known to be solvable in polynomial time and is known to be NP-complete. In fact, the complexity of the k processor scheduling problem when k is *fixed* is not known.

From the acyclic graph G , a *compatibility* graph G^* is constructed as follows. G^* has the same nodes as G , and there is an (undirected) edge (i, j) if there is no directed path from i to j or from j to i in G . In other words, i and j are jobs that can be done together.

A maximum matching in G^* indicates the maximum number of pairs of jobs that can be processed simultaneously. Clearly, a solution to the scheduling problem can be used to obtain a matching in G^* . More interestingly, a solution to the matching problem can be used to obtain a solution to the scheduling problem!

Suppose a maximum matching M in G^* is found. An unmatched vertex is executed on a single processor while the other processor is idle. If the maximum matching has size m^* , then $2m^*$ vertices are matched and $n - 2m^*$ vertices are left unmatched. The time to finish all the jobs will be $m^* + n - 2m^* = n - m^*$. Hence a maximum matching will minimize the time required to schedule all the jobs.

It is now shown that, given a maximum matching M^* , a schedule of size $n - m^*$ can be extracted from it. The key idea is that each matched job is scheduled in a slot together with another job (which may be different from the job it was matched to). This ensures that no more than $n - m^*$ slots are used.

Let S be the subset of vertices that have indegree 0. The following cases show how a schedule can be constructed from G and M^* . Basically, it must be ensured

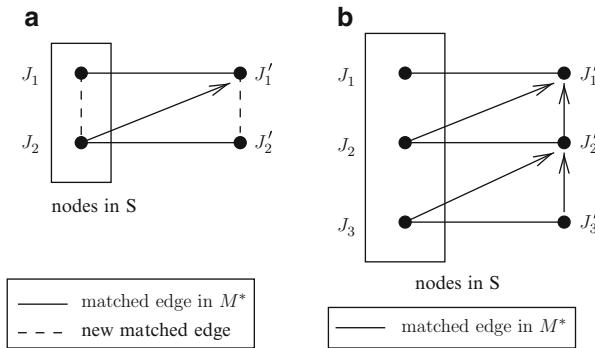


Fig. 5 (a) Changing the schedule (b) Continuing the proof

that for all jobs that are paired up in M^* , they are also paired up in the schedule. The following rules can be applied repeatedly.

1. If there is an unmatched vertex in S , schedule it and remove it from G .
2. If there is a pair of jobs in S that are matched in M^* , then schedule the pair and delete both the jobs from G .

If none of the above rules are applicable, then all jobs in S are matched; moreover each job in S is matched with a job not in S .

Let $J_1 \in S$ be matched to $J'_1 \notin S$. Let J_2 be a job in S that has a path to J'_1 . Let J'_2 be the mate of J_2 . If there is path from J'_1 to J'_2 , then J_2 and J'_2 could not be matched to each other in G^* (this cannot be an edge in the compatibility graph). If there is no path from J'_2 to J'_1 , then the matching is *changed* to match (J_1, J_2) and (J'_1, J'_2) since (J'_1, J'_2) is an edge in G^* (see Fig. 5a).

The only remaining case is when J'_2 has a path to J'_1 . Recall that J_2 also has a path to J'_1 . The above argument is repeated with J'_2 in place of J'_1 . This will yield a new pair (J_3, J'_3) , etc (see Fig. 5b). Since the graph G has no cycles at each step, a distinct pair of jobs is generated; at some point of time this process will stop (since the graph is finite). At that time a pair of jobs in S have been found that can be scheduled together.

3.2 Weighted Edge-Cover Problem

Another application of matching is the problem of finding a minimum-weight edge cover of a graph. Given a graph $G = (V, E)$ with weights on the edges, a set of edges $C \subseteq E$ form an edge cover of G if every vertex in V is incident to at least one edge in C . The weight of the cover C is the sum of the weights of its edges. The objective is to find an edge cover C of minimum weight. The problem can be reduced to the minimum-weight perfect matching problem as follows: Create an identical copy $G' = (V', E')$ of graph G , except the weights of edges in E' are all 0. Add an edge from each $v \in V$ to $v' \in V'$ with weight $w_{\min}(v) = \min_{x \in N(v)} w(v, x)$. The final graph H is the union of G and G' , together with the edges connecting the

two copies of each vertex; H contains $2|V|$ vertices and $2|E| + |V|$ edges. There exist other reductions from minimum-weight edge cover to minimum-weight perfect matching; the reduction outlined above has the advantage that it creates a graph with $O(|V|)$ vertices and $O(|E|)$ edges. The minimum-weight perfect matching problem may be solved using the techniques described earlier for the bipartite case, but the algorithm is more complex [33, 38].

Theorem 4 *The weight of a minimum-weight perfect matching in H is equal to the weight of a minimum-weight edge cover in G .*

Proof Consider a minimum-weight edge cover C in G . There is no loss of generality in assuming that a minimum-weight edge cover has no path of three edges, since the middle edge can be removed from the cover, thus reducing its weight further. Hence the edge cover C is a union of “stars” (trees of height 1). For each star that has a vertex of degree more than one, one of its leaf nodes can be made to match to its copy in G' , with weight at most the weight of the edge incident on the leaf vertex, thus reducing the degree of the star. This is repeated until each vertex has degree at most one in the edge cover, that is, it is a matching. In H , select this matching, once in each copy of G . Observe that the cost of the matching within the second copy G' is 0. Thus, given C , a perfect matching in H can be found, whose weight is no larger.

To argue the converse, it is now shown how to construct a cover C from a given perfect matching M in H . For each edge $(v, v') \in M$, add to C a least weight edge incident to v in G . Also include in C any edge of G that was matched in M . It can be verified that C is an edge cover whose weight equals the weight of M .

3.3 Chinese Postman Problem

One of the first problems to be studied in graph theory was the Euler tour problem. It asks whether a given graph has a tour that traverses each edge of the graph exactly once. Graphs that have Euler tours are called Eulerian graphs. It is well known that a graph is Eulerian if and only if it is connected and the degree of every vertex is even. Similarly, a necessary and sufficient condition for a directed graph to have an Euler tour is that it is strongly connected and the number of incoming edges at each vertex is equal to the number of outgoing edges from it. Given an Eulerian graph, whether undirected or directed, an Euler tour can be found in linear time.

A natural optimization problem, known as the Chinese postman problem (CPP), arises in non-Eulerian graphs, which are that of finding a shortest tour that traverses each edge of a given graph *at least* once. This problem and its generalizations were studied by Edmonds and Johnson [13]. CPP can be solved by using algorithms for weighted matching as follows. It is assumed that the underlying graph is connected, since otherwise, the problem is not feasible.

CPP in undirected graphs: Here, the given graph G needs to be augmented by adding additional edges such that the degree of every vertex becomes even. The subgraph that is added, known as a T-join, has odd degree at odd-degree nodes

of G and has even degree (possibly zero) at other nodes. It is easy to show that any graph has an even number of odd-degree nodes and that a minimum-weight T-join can be decomposed into a collection of edge-disjoint paths connecting odd-degree nodes. This leads us to the following algorithm for finding a shortest CPP in undirected graphs. Construct a new graph containing just the odd-degree nodes of G . Add edges connecting each pair of these nodes, with the weight of an edge being the length of a shortest path in G connecting those two nodes. Now, find a minimum-weight perfect matching in this graph. For each edge in this matching, a shortest path in G is added between its end vertices. The resulting multigraph has even degree at all nodes, and an Euler tour can be found in it.

CPP in directed graphs: In this case, it must be ensured that every node is balanced, that is, the number of outgoing edges is equal to the number of its incoming edges. For a vertex v , its imbalance, $b(v)$, is defined to be its outdegree minus its indegree. Construct an auxiliary graph with just the vertices that have $b(v) \neq 0$. In this graph, it is necessary to add as many copies of a vertex as its imbalance, that is, there are $|b(v)|$ copies of vertex v . The vertices are partitioned into those with positive imbalance (V_p) and vertices with negative imbalance (V_n). Take the complete bipartite graph between V_p and V_n , where the weight of an edge (p, n) is equal to the length of a shortest path in G from $p \in V_p$ to $n \in V_n$. Now, find a minimum-weight perfect matching M in this bipartite graph, and when G is augmented by adding the paths in G corresponding to M , the resulting multigraph is Eulerian. Note that it is also possible to solve the problem by using a solution for min-cost flow (see Sect. 8), by defining demands to be $b(v)$, by setting the weight of each edge to be its cost, and by finding a min-cost flow (without capacities) that satisfies all the demands.

4 The Network Flow Problem

A number of polynomial-time flow algorithms have been developed over the last two decades. The reader is referred to the books by Ahuja, Magnanti, and Orlin [1] and Tarjan [43] for a detailed account of the historical development of the various flow methods. An excellent survey on network flow algorithms has been written by Goldberg, Tardos, and Tarjan [23]. The book by Cormen, Leiserson, Stein, and Rivest [11] describes the preflow-push method, and to complement their coverage, an implementation of the “blocking flow” technique of Malhotra, Kumar, and Maheshwari (see [38]) is discussed here.

4.1 Network Flow Problem Definitions

Flow Network: A flow network $G = (V, E)$ is a directed graph, with two specially marked nodes, namely, the source s and the sink t . A capacity function $u : E \mapsto \mathbb{R}^+$ maps edges to positive real numbers.

Max-Flow Problem: A flow function $f : E \mapsto \mathbb{R}$ maps edges to real numbers. For an edge $e = (v, w)$, $f(v, w)$ refers to the flow on edge e , which is also called the net flow from vertex v to vertex w . This notation is extended to sets of vertices as follows: If X and Y are sets of vertices, then $f(X, Y)$ is defined to be $\sum_{x \in X} \sum_{y \in Y} f(x, y)$. A flow function is required to satisfy the following constraints:

- (Capacity constraint) For all edges e , $f(e) \leq u(e)$.
- (Skew symmetry constraint) For an edge $e = (v, w)$, $f(v, w) = -f(w, v)$.
- (Flow conservation) For all vertices $v \in V - \{s, t\}$, $\sum_{w \in V} f(v, w) = 0$.

The capacity constraint states that the total flow on an edge does not exceed its capacity. The skew symmetry condition states that the flow on an edge is the negative of the flow in the reverse direction. The flow conservation constraint states that the total net flow out of any vertex other than the source and sink is zero.

The value of the flow is defined to be the net flow out of the source vertex:

$$|f| = \sum_{v \in V} f(s, v).$$

In the maximum flow problem, the objective is to find a flow function that satisfies the above three constraints and also maximizes the total flow value $|f|$.

Remark This formulation of the network flow problem is powerful enough to capture generalizations where there are many sources and sinks (single-commodity flow) and where both vertices and edges have capacity constraints. To reduce multiple sources to a single source, add a new source vertex with edges connecting it to the original source vertices. To reduce multiple sinks to a single sink, add a new sink vertex with edges from the original sinks to the new sink. It is easy to reduce the vertex capacity problem to edge capacities by “splitting” a vertex into two vertices and making all the incoming edges come into the first vertex and the outgoing edges come out of the second vertex. An edge is added between them with the capacity of the corresponding vertex, so that the entire flow through the vertex is forced through this edge. The problem for undirected graphs can be solved by treating each undirected edge as two directed edges with the same capacity as the original undirected edge.

First, the notion of cuts is defined, and then, the max-flow min-cut theorem is introduced. Then, residual networks, layered networks, and the concept of blocking flows are introduced. It is then shown how to reduce the max-flow problem to the computation of a sequence of blocking flows. Finally an efficient algorithm for finding a blocking flow is described.

A cut is a partitioning of the vertex set into two subsets S and $V - S$. An edge crosses the cut if it connects a vertex $x \in S$ to a vertex $y \in V - S$. An $s-t$ cut of the graph is a partitioning of the vertex set V into two sets S and $T = V - S$ such that $s \in S$ and $t \in T$. If f is a flow, then the net flow across the cut is defined as $f(S, T)$. The capacity of the cut is defined as $u(S, T) = \sum_{x \in S} \sum_{y \in T} u(x, y)$. The

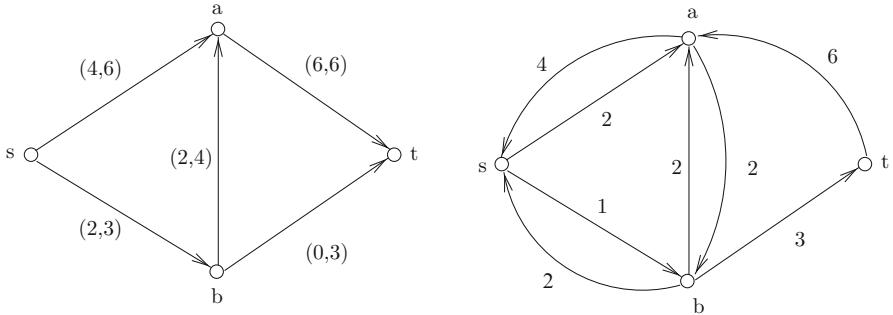


Fig. 6 A network with a given flow function and its residual network. A label of (f, c) for an edge indicates a flow of f and capacity of c

net flow across a cut includes negative flows between vertices (flow from T to S), but the capacity of the cut includes only the capacities of edges from S to T .

Using the flow conservation principle, it can be shown that the net flow across an s - t cut is exactly the flow value $|f|$. By the capacity constraint, the flow across the cut cannot exceed the capacity of the cut. Thus, the value of the maximum flow is no greater than the capacity of a minimum s - t cut. The max-flow min-cut theorem states that the two quantities are actually equal. In other words, if f^* is a maximum flow, then there is some cut (X, \bar{X}) with $s \in X$ and $t \in \bar{X}$, such that $|f^*| = u(X, \bar{X})$. The reader is referred to [11, 43] for further details.

The residual capacity of an edge (v, w) with respect to a flow f is defined as follows:

$$u'(v, w) = u(v, w) - f(v, w).$$

The quantity $u'(v, w)$ is the number of additional units of flow that can be pushed from v to w without violating the capacity constraints. An edge e is saturated if $u(e) = f(e)$, that is, if its residual capacity $u'(e) = 0$. The residual graph, $G_R(f)$, for a flow f , is the graph with vertex set V , source and sink s and t , respectively, and those edges (v, w) for which $u'(v, w) > 0$. Figure 6 shows a network on the left with a given flow function. Each edge is labeled with two values: the flow through that edge and its capacity. The figure on the right depicts the residual network corresponding to this flow.

An augmenting path for f is a path P from s to t in $G_R(f)$. The residual capacity of P , denoted by $u'(P)$, is the minimum value of $u'(v, w)$ over all edges (v, w) in the path P . The flow can be increased by $u'(P)$, by increasing the flow on each edge of P by this amount. Whenever $f(v, w)$ is changed, $f(w, v)$ is also correspondingly changed to maintain skew symmetry.

Most flow algorithms are based on the concept of augmenting paths pioneered by Ford and Fulkerson [16]. They start with an initial zero flow and augment the flow in stages. In each stage, a residual graph $G_R(f)$ with respect to the current flow function f is constructed, and an augmenting path in $G_R(f)$ is found, thus increasing the value of the flow. Flow is increased along this path until an edge in

this path is saturated. The algorithms iteratively keep increasing the flow until there are no more augmenting paths in $G_R(f)$ and return the final flow f as their output. Edmonds and Karp [14] suggested two possible improvements to this algorithm to make it run in polynomial time. The first was to choose shortest possible augmenting paths, where the length of the path is simply the number of edges on the path. This method can be improved using the approach due to Dinitz (see [43]) in which a sequence of blocking flows is found. The second strategy was to select a path which can be used to push the maximum amount of flow. The number of iterations of this method is bounded by $O(|E| \log C)$ where C is the largest edge capacity. In each iteration, a path is found through which a maximum amount of flow can be pushed. This step is implemented by a suitable modification of Dijkstra's shortest path algorithm.

The following lemma is fundamental in understanding the basic strategy behind these algorithms.

Lemma 2 *Let f be a flow and f^* be a maximum flow in G . The value of a maximum flow in the residual graph $G_R(f)$ is $|f^*| - |f|$.*

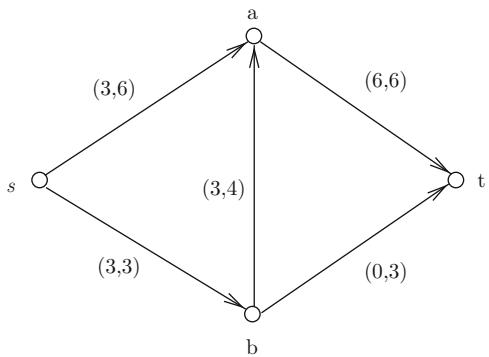
Proof Let f' be any flow in $G_R(f)$. Define $f + f'$ to be the flow $f(v, w) + f'(v, w)$ for each edge (v, w) . Observe that $f + f'$ is a feasible flow in G of value $|f| + |f'|$. Since f^* is the maximum flow possible in G , $|f'| \leq |f^*| - |f|$. Similarly define $f^* - f$ to be a flow in $G_R(f)$ defined by $f^*(v, w) - f(v, w)$ in each edge (v, w) , and this is a feasible flow in $G_R(f)$ of value $|f^*| - |f|$, and it is a maximum flow in $G_R(f)$.

Blocking Flow: A flow f is a blocking flow if every path from s to t in G contains a saturated edge. Blocking flows are also known as maximal flows. [Figure 7](#) depicts a blocking flow. Any path from s to t contains at least one saturated edge. For example, in the path $[s, a, t]$, the edge (a, t) is saturated. It is important to note that a blocking flow is not necessarily a maximum flow. There may be augmenting paths that increase the flow on some edges and decrease the flow on other edges (by increasing the flow in the reverse direction). The example in [Figure 7](#) contains an augmenting path $[s, a, b, t]$, which can be used to increase the flow by 3 units.

Layered Networks: Let $G_R(f)$ be the residual graph with respect to a flow f . The level of a vertex v is the length of a shortest path from s to v in $G_R(f)$. The level graph L for f is the subgraph of $G_R(f)$ containing vertices reachable from s and only the edges (v, w) such that $\text{level}(w) = 1 + \text{level}(v)$. L contains all shortest length augmenting paths and can be constructed in $O(|E|)$ time.

The maximum flow algorithm proposed by Dinitz starts with the zero flow and iteratively increases the flow by augmenting it with a blocking flow in $G_R(f)$ until t is not reachable from s in $G_R(f)$. At each step, the current flow is replaced by the sum of the current flow and the blocking flow. This algorithm terminates in $|V| - 1$ iterations, since in each iteration the shortest distance from s to t in the residual graph increases. The shortest path from s to t is at most $|V| - 1$, and this gives an upper bound on the number of iterations of the algorithm.

Fig. 7 Example of blocking flow – all paths from s to t are saturated. A label of (f, c) for an edge indicates a flow of f and capacity of c



An algorithm to find a blocking flow that runs in $O(|V|^2)$ time is described here (see [38] for details), and this yields an $O(|V|^3)$ max-flow algorithm. There are a number of $O(|V|^2)$ blocking flow algorithms available, some of which are described in [43].

4.2 Blocking Flows

Dinitz's algorithm finds a blocking flow as follows. The main step is to find paths from the source to the sink and push as much flow as possible on each path and thus saturate an edge in each path. It takes $O(|V|)$ time to compute the amount of flow that can be pushed on a path. Since there are $|E|$ edges, this yields an upper bound of $O(|V||E|)$ steps on the running time of the algorithm. The following algorithm shows how to find a blocking flow more efficiently.

Malhotra-Kumar-Maheshwari Blocking Flow Algorithm: The algorithm has a current flow function f and its corresponding residual graph $G_R(f)$. Define for each node $v \in G_R(f)$ a quantity $tp[v]$ that specifies its maximum throughput, that is, either the sum of the capacities of the incoming arcs or the sum of the capacities of the outgoing arcs, whichever is smaller. The quantity $tp[v]$ represents the maximum flow that could pass through v in any feasible blocking flow in the residual graph. Vertices with zero throughput are deleted from $G_R(f)$.

The algorithm selects a vertex x with least throughput. It then greedily pushes a flow of $tp[x]$ from x towards t , level by level in the layered residual graph. This can be done by creating a queue which initially contains x , which is assigned the task of pushing $tp[x]$ out of it. In each step, the vertex v at the front of the queue is removed, and the arcs going out of v are scanned one at a time, and as much flow as possible is pushed out of them until v 's allocated flow has been pushed out. For each arc (v, w) that the algorithm pushed flow through, it updates the residual capacity of the arc (v, w) and places w on a queue (if it is not already there) and increments the net incoming flow into w . Also $tp[v]$ is reduced by the amount of flow that was sent through it now. The flow finally reaches t , and the algorithm never comes across a vertex that has incoming flow that exceeds its outgoing capacity since x was chosen

as a vertex with least throughput. The above idea is repeated to pull a flow of $tp[x]$ from the source s to x . Combining the two steps yields a flow of $tp[x]$ from s to t in the residual network that goes through x . The flow f is augmented by this amount. Vertex x is deleted from the residual graph, along with any other vertices that have zero throughput.

The above procedure is repeated until all vertices are deleted from the residual graph. The algorithm has a blocking flow at this stage since at least one vertex is saturated in every path from s to t . In the above algorithm, whenever an edge is saturated, it may be deleted from the residual graph. Since the algorithm uses a greedy strategy to pump flows, at most $O(|E|)$ time is spent when an edge is saturated. When finding flow paths to push $tp[x]$, there are at most $|V|$ times (once for each vertex) when the algorithm pushes a flow that does not saturate the corresponding edge. After this step, x is deleted from the residual graph. Hence the above algorithm to compute blocking flows terminates in $O(|E| + |V|^2) = O(|V|^2)$ steps.

Karzanov (see [43]) used the concept of preflows to develop an $O(|V|^2)$ time algorithm for developing blocking flows as well. This method was then adapted by Goldberg and Tarjan (see [11]), who proposed a preflow-push method for computing a maximum flow in a network. A preflow is a flow that satisfies the capacity constraints, but not flow conservation. Any node in the network is allowed to have more flow coming into it than there is flowing out. The algorithm tries to move the excess flows towards the sinks without violating capacity constraints. The algorithm finally terminates with a feasible flow that is a max flow. It finds a max flow in $O\left(|V||E|\log \frac{|V|^2}{|E|}\right)$ time.

5 The Min-Cut Problem

Two problems are studied in this section. The first is the s - t min-cut problem: Given a directed graph with capacities on the edges and two special vertices s and t , the problem is to find a cut of minimum capacity that separates s from t . The value of a min-cut's capacity can be computed by finding the value of the maximum flow from s to t in the network. Recall that the max-flow min-cut theorem shows that the two quantities are equal. It is now shown how to find the sets S and T that provides an s - t min cut from a given s - t max flow.

The second problem is to find a smallest cut in a given graph G . In this problem on undirected graphs, the vertex set must be partitioned into two sets such that the total weight of the edges crossing the cut is minimized. This problem can be solved by enumerating over all $\{s, t\}$ pairs of vertices and finding a s - t min cut for each pair. This procedure is rather inefficient. However, Hao and Orlin [24] showed that this problem can be solved in the same order of running time as taken by a single max-flow computation. Stoer and Wagner [41] have given a simple and elegant algorithm for computing minimum cuts based on a technique due to Nagamochi and Ibaraki [37]. Recently, Karger [27] has given a randomized algorithm that runs in almost linear time for computing minimum cuts without using network flow methods.

5.1 Finding an s - t Min Cut

Let f^* be a maximum flow from s to t in the graph. Recall that $G_R(f^*)$ is the residual graph corresponding to f^* . Let S be the set of all vertices that are reachable from s in $G_R(f^*)$, that is, vertices to which s has a directed path. Let T be all the remaining vertices of G . By definition $s \in S$. Since f^* is a max flow from s to t , $t \in T$. Otherwise the flow can be increased by pushing flow along this path from s to t in $G_R(f^*)$. All the edges from vertices in S to vertices in T are saturated and form a minimum cut.

5.2 Finding All-Pair Min Cuts

For an undirected graph G with $|V|$ vertices, Gomory and Hu (see [1]) showed that the flow values between each of the $|V|(|V| - 1)/2$ pairs of vertices of G can be computed by solving only $|V| - 1$ max-flow computations. Furthermore, they showed that the flow values can be represented by a weighted tree T on $|V|$ nodes. Each node in the tree represents one of the vertices of the given graph. For any pair of nodes (x, y) , the maximum flow value from x to y (and hence the x - y min cut) in G is equal to the weight of the minimum-weight edge on the unique path in T between x and y .

6 Applications of Network Flows (and Min Cuts)

There are numerous applications of the maximum flow algorithm in scheduling problems of various kinds. It is used in open-pit mining, vehicle routing, etc. A few interesting applications of the flow problem are discussed. See [1] for further details.

6.1 Connectivity

Given a graph G , its connectivity is a measure of its susceptibility to disconnection upon the removal of some of its edges or nodes. It is an important concept in the area of telecommunication networks, where failure of nodes (called hubs) and edges (called links) can lead to network separation, resulting in millions of dollars in losses.

For two nodes v and w , define their *edge connectivity* to be the maximum number of *edge-disjoint* paths between v and w in G . Similarly, their *vertex connectivity* or simply *connectivity* is the maximum number of *vertex-disjoint* (also known as *openly disjoint*) paths connecting them. One can also define the terms using the minimum number of edges/vertices that have to be removed to disconnect v from w . Menger showed that these two definitions are equivalent: The maximum number of edge-disjoint and openly disjoint paths from v to w are, respectively, equal to the

minimum number of edges and vertices that must be removed to separate v from w in G .

Given a graph G , the edge connectivity of its vertices v and w is computed as follows. Replace each edge of G by two (antiparallel) directed edges between its end points. Assign the capacity of each edge to be 1. In the resulting flow network, find a maximum flow from v to w . The value of the maximum flow is equal to the edge connectivity of v and w in G . A min cut separating v and w is a minimum set of edges of G , whose removal separates v from w in G . For vertex connectivity, openly disjoint paths are required. This can be done by setting capacities of vertices also to be 1, which can be accomplished by replacing each vertex u by two vertices u_{in} and u_{out} , adding the edge from u_{in} to u_{out} of capacity 1. All incoming edges to u are sent to u_{in} . All outgoing edges from u are sent out of u_{out} . Now, a maximum flow from v_{out} to w_{in} gives us the maximum number of openly disjoint paths from v to w in G .

6.2 Finding a Minimum-Weight Vertex Cover in Bipartite Graphs

A vertex cover of a graph is a set of vertices that is incident to all its edges. The weight of a vertex cover is the sum of the weights of its vertices. Finding a vertex cover of minimum weight is NP-hard for arbitrary graphs (see Sect. 12 for more details). For bipartite graphs, the problem is solvable efficiently using the maximum flow algorithm.

Given a bipartite graph $G = (X, Y, E)$ with weights on the vertices, the goal is to find a subset C of vertices of minimum total weight, so that for each edge, at least one of its end vertices is in C . The weight of a vertex v is denoted by $w(v)$. Construct a flow network N from G as follows: Add a new source vertex s , and for each $x \in X$, add a directed edge (s, x) , with capacity $w(x)$. Add a new sink vertex t , and for each $y \in Y$, add a directed edge (y, t) , with capacity $w(y)$. The edges in E are given infinite capacity, and they are oriented from X to Y . Let C be a subset of vertices in G . In the following discussion, X_C denotes $X \cap C$, and Y_C denotes $Y \cap C$. Let $\bar{X}_C = X - X_C$ and $\bar{Y}_C = Y - Y_C$.

If C is a vertex cover of G , then there are no edges in G connecting \bar{X}_C and \bar{Y}_C , since such edges would not be incident to C . A cut in the flow network N , whose capacity is the same as the weight of the vertex cover, can be constructed as follows: $S = \{s\} \cup \bar{X}_C \cup Y_C$ and $T = \{t\} \cup X_C \cup \bar{Y}_C$. Each vertex cover of G gives rise to a cut whose capacity is equal to the weight of the cover. Similarly, any cut of finite capacity in N corresponds to a vertex cover of G whose weight is equal to the capacity of the cut. Hence the minimum-weight $s-t$ cut of N corresponds to a minimum-weight vertex cover of G .

6.3 Maximum-Weight Closed Subset in a Partial Order

Consider a directed acyclic graph (DAG) $G = (V, E)$. Each vertex v of G is given a weight $w(v)$ that may be positive or negative. A subset of vertices $S \subseteq V$ is said to be closed in G if for every $s \in S$, the predecessors of s are also in S .

The predecessors of a vertex s are vertices that have directed paths to s . Consider the objective of computing a closed subset S of maximum weight. This problem occurs, for example, in open-pit mining, and it can be solved efficiently by reducing it to computing the minimum cut in the following network N :

- The vertex set of $N = V \cup \{s, t\}$, where s and t are two new vertices.
- For each vertex v of negative weight, add the edge (s, v) with capacity $|w(v)|$ to N .
- For each vertex v of positive weight, add the edge (v, t) with capacity $w(v)$ to N .
- All edges of G are added to N , and each of these edges has infinite capacity.

Consider a minimum s - t cut in N . Let S be the positive weight vertices whose edges to the sink t are not in the min cut. It can be shown that the union of the set S and its predecessors is a maximum-weight closed subset.

7 Finding Densest Subgraphs

The density of a subgraph is defined as the ratio of the number of edges in the induced graph to the number of vertices in the subgraph. The problem can be extended to the weighted case when edges and vertices have positive integer weights defined by functions w' and w , respectively, by defining the density of a subgraph as the ratio of the total weight of its edges to the total weight of its vertices. Given a graph G , the objective is to find a subgraph of G , whose density is the largest among all its subgraphs. Despite the fact that there are exponentially many subgraphs, it is possible to find a maximum density subgraph in polynomial time using a procedure for the max-flow (min-cut) problem. An interesting extension of the problem is when a subset C of vertices is given, and the goal is to find a densest subgraph that must include all nodes of C .

7.1 Algorithm for Densest Subgraph Without a Specified Subset

First, the basic algorithm for finding a densest subgraph by a series of max-flow (min-cut) computations [33] is discussed. Guess α , the density of the maximum density subgraph, and then refine the guess by doing a network flow computation. Suppose a maximum density subgraph S^* has density α^* . Let the current guess be α . By appropriately defining a flow network as shown below and examining its min-cut structure, it is possible to determine if $\alpha = \alpha^*$ or $\alpha < \alpha^*$ or $\alpha > \alpha^*$. It can be verified that α^* lies between 0 and $w'(E)$ (total weight of all edges), and therefore, α^* can be found by using binary search. Since densities are rational numbers, once the interval size drops below $\frac{1}{|V|^2}$, the algorithm can stop.

The construction of the flow network for a given guess α is described below. Create a flow network G' with a source s and sink t . Each edge in G has a node in G' , and let E' be the set of these nodes of G' . In addition, each node in G also has its own node in G' , and let V' be the set of these nodes in G' . Add edges from s to

$e \in E'$ of capacity $w'(e)$ and an edge from $v \in V'$ to t with capacity $\alpha w(v)$. Add edges from $e = (x, y) \in E'$ to both $x \in V'$ and $y \in V'$ with capacity ∞ . In fact, this method works even in the case of hyper-graphs, and in this case, E' is a set of hyper-edges, and such edges are added from e to all $x \in V'$ such that $x \in e$. Since infinite capacity edges were added from e to x and y , any finite capacity (s, t) cut must have all of e , x , and y on the same side of the cut whenever e is on the same side as s . Therefore, all such finite cuts correspond to subgraphs of G .

If the problem is given without a specified subset ($C = \emptyset$), then the algorithm proceeds as follows. First, note that there is a s - t cut of value $w'(E)$. This is obtained by setting $S = \{s\}$ and $T = \{t\} \cup E' \cup V'$. Suppose the max density subset has density α^* . Suppose the guess $\alpha < \alpha^* = \frac{w'(S^*)}{w(S^*)}$, then it follows that $\alpha w(S^*) < w'(S^*)$.

Now consider an s - t cut $(s \cup V_1, t \cup V_2)$ in the flow network G' , then let $S = V_1 \cap V'$. The cut includes all the edges from nodes in S to t of capacity $\alpha w(S)$ as well as edges from s to nodes in E' that are not in V_1 . All edges $e = (x, y) \in E$ with one end in $V \setminus S$ must be in V_2 since otherwise there will be an edge of ∞ capacity across the cut. If all the edges in the induced graph formed by S are not in V_1 , a smaller cut can be found by moving such edges from V_2 to V_1 . Therefore, s - t min cuts have all edges of the induced subgraph of S in V_1 . The capacity of such a cut is exactly $w'(E \setminus E(S)) + \alpha w(S)$. Note that $w'(E \setminus E(S))$ includes the weight of all edges that are incident on some node in $E \setminus S$, and so, $w'(E \setminus E(S)) + \alpha w(S) = w'(E) - w'(S) + \alpha w(S) = w'(E) - (w'(S) - \alpha w(S))$. But for the optimal subset S^* , $w'(S^*) - \alpha w(S^*) > 0$, and thus, there is a cut whose capacity is smaller than $w'(E)$. Therefore, if the capacity of a min cut happens to be smaller than $w'(E)$, then it implies that the guess for α is smaller than α^* . Similarly, if the guess for α is greater than α^* , then the (unique) min cut has value $w'(E)$. When $\alpha = \alpha^*$, then there could be multiple min cuts of value $w'(E)$. Any min cut other than the trivial cut gives a correct solution.

7.2 Algorithm for Densest Subgraph with a Specified Subset C

It is now shown (see [40]) how to modify the above construction when $C \neq \emptyset$. In the following discussion, $w(C)$ denotes the sum of the weights of the vertices in C , and $w'(C)$ denotes the sum of the weights of edges in the induced subgraph of C . Create a new source s' and add an edge to s with capacity $w'(E) - \alpha w(C)$. In addition, remove C from V' . Again suppose that $\alpha < \alpha^*$. In this case, a subset S^* exists such that $\frac{w'(E(S^*)) + w'(C) + w'(S^*, C)}{w(S^*) + w(C)} > \alpha$. Thus, $w'(E(S^*)) + w'(C) + w'(S^*, C) - \alpha(w(S^*) + w(C)) > 0$. Replace $w'(E) - w'(\overline{E}(V \setminus (S^* \cup C)))$ for the first term. Note that $w'(\overline{E}(V \setminus (S^* \cup C)))$ includes all edges incident on nodes in $V \setminus (S^* \cup C)$ and not only the edges induced by those nodes. Hence,

$$w'(E) - w'(\overline{E}(V \setminus (S^* \cup C))) - \alpha(w(S^*) + w(C)) > 0.$$

$$(w'(E) - \alpha w(C)) - (w'(\overline{E}(V \setminus (S^* \cup C))) + \alpha w(S^*)) > 0.$$

$$(w'(E) - \alpha w(C)) > w'(\overline{E}(V \setminus (S^* \cup C))) + \alpha w(S^*).$$

This means that a min cut exists (e.g., defined by the subset S^*) that is smaller than $w'(E) - \alpha w(C)$.

So again, by looking at the min-cut structure, it is possible to know that $\alpha < \alpha^*$. If $\alpha > \alpha^*$, then the trivial min cut separating s' from the rest of the graph is unique. Again a binary search for α can be done.

Note: A simple method that does *not* work is to snap the edges to C as self loops and to then compute the densest subgraph in G with C removed. If the density of the densest subgraph found is lower than $w'(C)/w(C)$, then the algorithm just returns C as the answer. Otherwise, it returns $S \cup C$. The main problem is that the density of S could get lowered when C is added back. The level of dilution depends on the size of the densest subgraph in G with C removed; hence a subgraph with slightly lower density than the optimal solution but of much larger size could be a better choice.

8 Minimum-Cost Flows

In this section, one of the most important problems in combinatorial optimization, namely, the minimum-cost network flow problem, is studied. Some of the basic ideas behind the problem are outlined, and the reader can find a wealth of information in [1] about efficient algorithms for this problem.

The min-cost flow problem can be viewed as a generalization of the max-flow problem, the shortest path problem, and the minimum-weight perfect matching problem. To reduce the shortest path problem to the min-cost flow problem, notice that if just one unit of flow needs to be shipped, it is best to send it on a shortest path. The minimum-weight perfect matching problem on a bipartite graph $G = (X, Y, X \times Y)$ is reduced to min-cost flow as follows. Introduce a special source s and sink t , and add unit capacity edges from s to vertices in X and from vertices in Y to t . Compute a flow of value $|X|$ to obtain a minimum-weight perfect matching.

8.1 Min-Cost Flow Problem Definitions

The flow network $G = (V, E)$ is defined as before. Other than the capacity function u , a cost function $c : E \rightarrow \mathbb{R}$ is also given. The cost function specifies the cost of shipping one unit of flow through an edge. Associated with each vertex v , there is a supply $b(v)$. If $b(v) > 0$, then v is a supply node, and if $b(v) < 0$, then it is a demand node. It is assumed that $\sum_{v \in V} b(v) = 0$; if the supply exceeds the demand, an artificial sink can be added to absorb the excess flow. As before, a flow function satisfies the capacity and skew symmetry constraints. The flow conservation is rewritten to indicate that each vertex v has an outgoing flow of $b(v)$:

- (Flow Conservation) For all vertices $v \in V$, $\sum_{w \in V} f(v, w) = b(v)$.

The objective is to find a flow function that minimizes the cost of the flow,

$$\min z(f) = \sum_{e \in E} c(e) \cdot f(e).$$

Before studying any specific algorithm to solve the problem, it is noted that a feasible solution (not necessarily optimal) can be found, if one exists, by finding a max flow and ignoring costs. To see this, add two new vertices, a source vertex s and a sink vertex t . Add edges from s to all vertices v with $b(v) > 0$ of capacity $b(v)$ and edges from all vertices w with $b(w) < 0$ to t of capacity $|b(w)|$. Find a max flow of value $\sum_{b(v)>0} b(v)$. If such a flow does not exist, then the flow problem is not feasible.

The algorithm for the min-cost flow problem also uses the concept of residual networks. As before, the residual network $G_R(f)$ is defined with respect to a specific flow. Edges in the residual graph have both capacity and cost. If there is a flow $f(v, w)$ on edge $e = (v, w)$, then its capacity in the residual network is $u(e) - f(e)$ and its residual cost is $c(e)$. The reverse edge (w, v) has residual capacity $f(e)$, but its residual cost is $-c(e)$. Note that sending a unit of flow on the reverse edge actually reduces the original amount of flow that was sent, thereby decreasing the total cost, and hence its residual cost is negative. Only edges with strictly positive residual capacity are retained in the residual network.

The following theorem characterizes optimal solutions.

Theorem 5 *A feasible solution f is an optimal solution if and only if the residual network $G_R(f)$ has no directed cycles of negative cost.*

The above theorem can be used to develop an algorithm for finding a min-cost flow. As indicated earlier, a feasible flow can be found by the techniques developed in the previous section. To improve the cost of the solution, the algorithm identifies negative-cost cycles in the residual graph and pushes as much flow around them as possible and reduces the cost of the flow. This is repeated until there are no negative cycles left, and a min-cost flow has been found. A negative-cost cycle may be found by using an algorithm like the Bellman-Ford algorithm, which takes $O(|E||V|)$ time.

An important issue that affects the running time of the algorithm is the choice of a negative-cost cycle. For fast convergence, one should select a cycle that decreases the cost as much as possible but finding such a cycle is NP-hard. Goldberg and Tarjan [22] showed that selecting a cycle with minimum mean cost (the ratio of the cost of cycle to the number of arcs on the cycle) yields a polynomial-time algorithm. There is a parametric search algorithm [1] to find a min-mean cycle that works as follows. Let μ^* be the average edge weight of a min-mean cycle. Then μ^* is the largest value such that reducing the weight of every edge by μ^* does not introduce a negative-weight cycle into the graph. One can search for the value of μ^* using binary search: Given a guess μ , decrease the weight of each edge by μ and test if

the graph has no negative-weight cycles. If the smallest cycle has zero weight, it is a min-mean cycle. If all cycles are positive, the guess is increased by μ . If the graph has a negative-weight cycle, the guess is decreased by μ . Karp (see [1]) has given an algorithm based on dynamic programming for this problem that runs in $O(|E||V|)$ time.

The first polynomial-time algorithm for the min-cost flow problem was given by Edmonds and Karp [14]. Their idea is based on scaling capacities, and the running time of their algorithm is $O(|E| \log U(|E| + |V| \log |V|))$, where U is the largest capacity. The first strongly polynomial algorithm (whose running time is only a function of $|E|$ and $|V|$) for the problem was given by Tardos [42].

9 The Multi-Commodity Flow Problem

A natural generalization of the max-flow problem is the multi-commodity flow problem. In this problem, there are a number of commodities $1, 2, \dots, k$ that have to be shipped through a flow network from source vertices s_1, s_2, \dots, s_k to sink vertices t_1, t_2, \dots, t_k , respectively. The amount of demand for commodity i is specified by d_i . Each edge and each vertex has a nonnegative capacity that specifies the total maximum flow of all commodities flowing through that edge or vertex. A flow is feasible if all the demands are routed from their respective sources to their respective sinks while satisfying flow conservation constraints at the nodes and capacity constraints on the edges and vertices. A flow is an ϵ -feasible flow for a positive real number ϵ if for each demand i , it satisfies at least $\frac{d_i}{1+\epsilon}$ of that demand. There are several variations of this problem as noted below:

In the simplest version of the multi-commodity flow problem, the goal is to decide if there exists a feasible flow that routes all the demands of the k commodities through the network. The flow corresponding to each commodity is allowed to be split arbitrarily (i.e., even fractionally) and routed through multiple paths. The concurrent flow problem is identical to the multi-commodity flow problem when the input instance has a feasible flow that satisfies all the demands. If the input instance is not feasible, then the objective is to find the smallest fraction ϵ for which there is an ϵ -feasible flow. In the minimum-cost multi-commodity flow problem, each edge has an associated cost for each unit of flow through it. The objective is to find a minimum-cost solution for routing all the demands through the network. All of the above problems can be formulated as linear programs and, therefore, have polynomial-time algorithms using either the ellipsoid algorithm or the interior point method [28].

A lot of research has been done on finding approximately optimal multi-commodity flows using more efficient algorithms. There are a number of papers that provide approximation algorithms for the multi-commodity flow problem. For a detailed survey of these results, see [12, Chap. 5].

In this section, a paper that introduced a new algorithm for solving multi-commodity flow problems is discussed. Awerbuch and Leighton [3] gave a simple and elegant algorithm for the concurrent flow problem. Their algorithm is based

on the “liquid-flow paradigm.” Flow is pushed from the sources on edges based on the “pressure difference” between the end vertices. The algorithm does not use any global properties of the graph (such as shortest paths) and uses only local changes based on the current flow. Due to its nature, the algorithm can be implemented to run in a distributed network since all decisions made by the algorithm are local in nature. They extended their algorithms to dynamic networks, where edge capacities are allowed to vary [4]. The best implementation of this algorithm runs in $O(k|V|^2|E|\epsilon^{-3}\ln^3(|E|/\epsilon))$ time.

In the integer multi-commodity flow problem, the capacities and flows are restricted to be integers. Unlike the single-commodity flow problem, for problems with integral capacities and demands, the existence of a feasible fractional solution to the multi-commodity flow problem does not guarantee a feasible integral solution. An extra constraint that may be imposed on this problem is to restrict each commodity to be sent along a single path. In other words, this constraint does not allow one to split the demand of a commodity into smaller parts and route them along independent paths (see the recent paper by Kleinberg [31] for approximation algorithms for this problem). Such constraints are common in problems that arise in telecommunication systems. All variations of the integer multi-commodity flow problem are NP-hard.

9.1 Local Control Algorithm

In this section, Awerbuch and Leighton’s local control algorithm is discussed. It finds an approximate solution for the concurrent flow problem.

First, the problem is converted to the continuous flow problem. In this version of the problem, d_i units of commodity i is added to the source vertex s_i in each phase. Each vertex has queues to store the commodities that arrive at that node. The node tries to push the commodities stored in the queues towards the sink nodes. As the commodities arrive at the sinks, they are removed from the network. A continuous flow problem is stable if the total amount of all the commodities in the network at any point in time is bounded (i.e., independent of the number of phases completed). The following theorem establishes a tight relationship between the multi-commodity flow problem (also known as the static problem) and its continuous version.

Theorem 6 *A stable algorithm for the continuous flow problem can be used to generate an ϵ -feasible solution for the static concurrent flow problem.*

Proof Let R be the number of phases of the stable algorithm for the continuous flow problem at which $\frac{1}{1+\epsilon}$ fraction of each of the commodities that have been injected into the network have been routed to their sinks. It is known that R exists since the algorithm is stable. The average behavior of the continuous flow problem in these R phases generates an ϵ -feasible flow for the static problem. In other words,

the flow of a commodity through a given edge is the average flow of that commodity through that edge over R iterations, and this flow is ϵ -feasible.

In order to get an approximation algorithm using the fluid-flow paradigm, there are two important challenges. First, it must be shown that the algorithm is stable, that is, that the queue sizes are bounded. Second, the number of phases that it takes the algorithm to reach “steady state” (specified by R in the above theorem) must be minimized. Note that the running time of the approximation algorithm is R times the time it takes to run one phase of the continuous flow algorithm.

Each vertex v maintains one queue per commodity for each edge to which it is incident. Initially all the queues are empty. In each phase of the algorithm, commodities are added at the source nodes. The algorithm works in four steps:

1. Add $(1 + \epsilon)d_i$ units of commodity i at s_i . The commodity is spread equally to all the queues at that vertex.
2. Push the flow across each edge so as to balance the queues as much as possible.
If Δ_i is the discrepancy of commodity i in the queues at either end of edge e , then the flow f_i crossing the edge in that step is chosen so as to maximize $\sum_i f_i(\Delta_i - f_i)/d_i^2$ without exceeding the capacity constraints.
3. Remove the commodities that have reached their sinks.
4. Balance the amount of each commodity in each of the queues at each vertex.

The above 4 steps are repeated until $\frac{1}{1+\epsilon}$ fraction of the commodities that have been injected into the system have reached their destination.

It can be shown that the algorithm is stable and the number of phases is $O(|E|^2 k^{1.5} L \epsilon^{-3})$, where $|E|$ is the number of edges, k is the number of commodities, and $L \leq |V|$ is the maximum path length of any flow. The proof of stability and the bound on the number of rounds are not included here. The actual algorithm has a few other technical details that are not mentioned here, and the reader is referred to the original paper for further details [3].

10 Minimum Spanning and Other Light-Weight Trees

Efficient algorithms are well known for finding minimum-weight spanning trees of undirected graphs. There are several greedy algorithms known for solving the problem. The problem turns out to be more difficult to solve in directed graphs, and the greedy algorithm fails since not all maximal, acyclic subgraphs are trees in directed graphs. One solution for finding a minimum-weight branching, as minimum spanning trees are called in directed graphs, is by using an algorithm for matroid intersection, which is similar to the algorithm for weighted matching. A simpler and more efficient algorithm for finding optimal branchings is discussed.

Another well-studied problem in networks is that of finding shortest paths. One may be interested in finding shortest paths from a single-source node or

between all pairs of nodes. For the single-source case, if all edges have nonnegative weights, then the problem of finding shortest paths from the source to all nodes of the graph can be solved efficiently using Dijkstra's algorithm. The algorithm also outputs a “shortest path tree” such that for any vertex, the shortest path to it from the root is the path in the tree from the root to that vertex. In general, shortest path trees can be much heavier than minimum spanning trees. It is useful in practice if it is possible to find light subgraphs in which shortest path distances are approximately preserved with respect to the given graph. Spanners are light subgraphs in which the distance between every pair of vertices is approximately preserved. Light approximate shortest path trees (LAST) do the same for single-source shortest paths.

10.1 Minimum-Weight Branchings

A natural analog of a spanning tree in a directed graph is a branching (also called an arborescence). For a directed graph G and a vertex r , a branching rooted at r is an acyclic subgraph of G in which each vertex but r has exactly one outgoing edge, and there is a directed path from any vertex to r . This is also sometimes called an in-branching. Replacing “outgoing” in the above definition of a branching to “incoming” defines an out-branching. An optimal branching of an edge-weighted graph is a branching of minimum total weight. Unlike the minimum spanning tree problem, an optimal branching cannot be computed using a greedy algorithm. Edmonds gave the first polynomial-time algorithm to find optimal branchings (see [20]).

Let $G = (V, E)$ be an arbitrary graph, let r be the root of G , and let $w(e)$ be the weight of edge e . Consider the problem of computing an optimal branching rooted at r . In the following discussion, assume that all edges of the graph have nonnegative weights.

Two key ideas are discussed below that can be converted into a polynomial-time algorithm for computing optimal branchings. First, for any vertex $v \neq r$ in G , suppose that all outgoing edges of v are of positive weight. Let $\epsilon > 0$ be a number that is less than or equal to the weight of any outgoing edge from v . Suppose the weight of every outgoing edge from v is decreased by ϵ . Observe that since any branching has exactly one edge out of v , its weight decreases by exactly ϵ . Therefore, an optimal branching of G with the original weights is also an optimal branching of G with the new weights. In other words, decreasing the weight of the outgoing edges of a vertex uniformly leaves optimal branchings invariant. Second, suppose there is a cycle C in G consisting only of edges of zero weight. Suppose the vertices of C are combined into a single vertex and the resulting multigraph is made into a simple graph by replacing each multiple edge by a single edge whose weight is the smallest among them and by discarding self loops. Let G_C be the resulting graph. It can be shown that the weights of optimal branchings of G and G_C are the same. An optimal branching of G_C can also be converted into an optimal branching

of G by adding sufficiently many edges from C without introducing cycles. The above two ideas can be used to design a recursive algorithm for finding an optimal branching. The fastest implementation of a branching algorithm is due to Gabow, Galil, Spencer, and Tarjan [18].

10.2 Spanners

Given a graph $G = (V, E)$ and a positive number $t \geq 1$, a t -spanner is a spanning subgraph $H = (V, E')$ of G with the property that for all pairs of vertices v and w ,

$$\text{dist}_H(v, w) \leq t \cdot \text{dist}_G(v, w).$$

In other words, a spanner is a subgraph that approximates shortest paths between every pair of vertices. The factor t is called the stretch factor. The goal is to minimize the size and weight of H for a given value of t . The size of H refers to the number of edges in H , and the weight of H refers to the total weight of the edges in H . The subject of graph spanners has received much attention recently [2, 9, 39]. The following elegant algorithm computes a t -spanner [2].

```

SPANNER ( $G, t$ )
1  $H \leftarrow (V, \emptyset)$ .
2 Sort all edges in increasing weight,  $w(e_1) \leq w(e_2) \leq \dots \leq w(e_m)$ .
3 for  $i = 1$  to  $m$  do
4   let  $e_i = (v, w)$ .
5   let  $P(v, w)$  be the shortest path in  $H$  from  $v$  to  $w$ .
6   if  $w(P(v, w)) > t \cdot w(e_i)$ , then
7      $H \leftarrow H \cup e_i$ .
8   end-if
9 end-for
end-proc
```

It is not difficult to show that this algorithm outputs a t -spanner. The following property is used to establish the size of the spanner output by the algorithm.

Lemma 3 *Any cycle in the graph H output by algorithm SPANNER has at least $t + 2$ edges.*

Proof Suppose for contradiction that H contains a cycle C with at most $t + 1$ edges. Let $e_{\max} = (v, w)$ be a heaviest weight edge on the cycle C . When the algorithm considers e_{\max} , all the other edges on C have already been added. This implies that there is a path of length at most $t \cdot w(e_{\max})$ from v to w , since the path contains t edges each of weight at most $w(e_{\max})$. Thus, the edge e_{\max} is not added to H , a contradiction.

Using this property, it can be shown that H has at most $O(|V|^{1+\frac{2}{t-1}})$ edges by applying a theorem from extremal graph theory relating the number of edges in a graph and its girth (length of a shortest cycle) [8]. In a subsequent chapter, Chandra et al. [9] proved the following theorem.

Theorem 7 *Let G be a connected edge-weighted graph. For all $\epsilon > 0$, algorithm SPANNER constructs a t -spanner of G with weight $O(|V|^{\frac{2+\epsilon}{t-1}}) \cdot w(MST)$. The constant depends only on t and ϵ .*

10.3 Light Approximate Shortest Path Trees

In this section, the problem of computing light-weight spanning trees that approximate shortest paths to every vertex from a single fixed root vertex is considered.

Let G be a graph and let r be the root vertex in G . Let T_{\min} be a minimum spanning tree of G . For any vertex v , let $d(r, v)$ be the length of a shortest path from r to v in G . An (α, β) -light approximate shortest path tree $((\alpha, \beta)\text{-LAST})$ of G is a spanning tree T of G with the property that the distance from the root to any vertex v in T is at most $\alpha \cdot d(r, v)$, and the weight of T is at most β times the weight of T_{\min} .

Awerbuch, Baratz, and Peleg [5], motivated by applications in broadcast-network design, made a fundamental contribution by showing that every graph has a *shallow-light* tree – a tree whose *diameter* is at most a constant times the diameter of G and whose total weight is at most a constant times the weight of a minimum spanning tree. Cong et al. [10] studied the same problem and showed that the problem has applications in VLSI-circuit design; they improved the approximation ratios obtained in [5] and also studied variations of the problem such as bounding the *radius* of the tree instead of the diameter.

Khuller, Raghavachari, and Young [29] modified the algorithm in [5] and obtained a stronger result; they showed that the distance from the root to each vertex can be approximated within a constant factor. Their algorithm also runs in linear time if a minimum spanning tree and a shortest path tree are provided. The algorithm computes an $(\alpha, 1 + \frac{2}{\alpha-1})$ -LAST. Independently, Awerbuch, Baratz, and Peleg [6] modified their algorithm from [5]. They obtained the same algorithm as in [10] but gave a stronger analysis, proving that the algorithm computes an $(\alpha, 1 + \frac{4}{\alpha-1})$ -LAST.

The basic idea is as follows: Initialize a subgraph H to be a minimum spanning tree T_{\min} . The vertices are processed in a preorder traversal of T_{\min} . When a vertex v is processed, its distance from r in H is compared to $\alpha \cdot d(r, v)$. If the distance exceeds the required threshold, then the algorithm adds to H a shortest path in G from r to v . When all the vertices have been processed, the distance in H from r to any vertex v meets its distance requirement. A shortest path tree in H is returned by the algorithm as the required LAST. An analysis shows that the entire subgraph H satisfies the weight guarantee.

11 Coloring Problems

11.1 Vertex Coloring

A vertex coloring of a graph G is a coloring of the vertices of G such that no two adjacent vertices of G receive the same color. The objective of the problem is to find a coloring that uses as few colors as possible. The minimum number of colors needed to color the vertices of a graph is known as its chromatic number. The register allocation problem in compiler design and the map coloring problem are instances of the vertex coloring problem.

The problem of deciding if the chromatic number of a given graph is at most a given integer k is NP-complete. The problem is NP-complete even for fixed $k \geq 3$. For $k = 2$, the problem is to decide if the given graph is bipartite, and this can be solved in linear time using depth-first or breadth-first search. The vertex coloring problem in general graphs is a notoriously hard problem, and it has been shown to be intractable even for approximating within a factor of n^ϵ for some constant $\epsilon > 0$ [12, Chap. 10]. A greedy coloring of a graph yields a coloring that uses at most $\Delta + 1$ colors, where Δ is the maximal degree of the graph. Unless the graph is an odd cycle or the complete graph, it can be colored with Δ colors (known as Brooks' theorem) [20]. A celebrated result on graph coloring is that every planar graph is 4-colorable. However, checking if a planar graph is 3-colorable is NP-complete [19]. For more information on recent results on approximating the chromatic number, see [26].

In the equitable coloring problem, the objective is to find a coloring of the nodes of a graph with the additional constraint that each color class should have roughly the same cardinality. Chapter ▶A Unified Approach for Domination Problems on Different Network Topologies surveys results in this area. Coloring problems are also discussed in Chap. ▶Resource Allocation Problems.

11.2 Edge Coloring

The edge coloring problem is similar to the vertex coloring problem. In this problem, the goal is to color the edges of a given graph using the fewest colors such that no two edges incident to a common vertex are assigned the same color. The problem finds applications in assigning classroom to courses and in scheduling problems. The minimum number of colors needed to color a graph is known as its chromatic index. Since any two incident edges must receive distinct colors, the chromatic index of a graph with maximal degree Δ is at least Δ . In a remarkable theorem, Vizing (see [20]) has shown that every graph can be edge colored using at most $\Delta + 1$ colors. Deciding whether the edges of a given graph can be colored using Δ colors is NP-complete (see [12]). Special classes of graphs such as bipartite graphs and planar graphs of large maximal degree are known to be Δ -edge-colorable.

12 Approximation Algorithms for Hard Problems

Many graph problems are known to be NP-complete (see Sects. A.1 and A.2 of [19]). The area of approximation algorithms explores intractable (NP-hard) optimization problems and tries to obtain polynomial-time algorithms that generate feasible solutions that are close to optimal. In this section, a few fundamental NP-hard optimization problems in graphs that can be approximated well are discussed. For more information about approximating these and other problems, see the chapter on approximation algorithms ([►Algorithmic Aspects of Domination in Graphs](#)). We also mention several books on this topic under “Further Information and Recommended Reading” in [Sect. 13](#).

In the following discussion, $G = (V, E)$ is an arbitrary undirected graph, and k is a positive integer.

Vertex cover: A set of vertices $S \subset V$ such that every edge in E is incident to at least one vertex of S . The minimum vertex cover problem is that of computing a vertex cover of minimum cardinality. If the vertices of G have weights associated with them, a minimum-weight vertex cover is a vertex cover of minimum total weight. Using the primal-dual method of linear programming, one can obtain a 2-approximation.

Dominating set: A set of vertices $S \subset V$ such that every vertex in the graph is either in S or adjacent to some vertex in S . There are several versions of this problem such as the total dominating set (every vertex in G must have a neighbor in S , irrespective of whether it is in S or not), the connected dominating set (induced graph of S must be connected) and the independent dominating set (induced graph of S must be empty). The minimum dominating set problem is to compute a minimum cardinality dominating set. The problems can be generalized to the weighted case suitably. All but the independent dominating set problem can be approximated within a factor of $O(\log n)$. For more information on domination problems, see Chaps. [►Combinatorial Optimization Techniques for Network-Based Data Mining](#) and [►Fuzzy Combinatorial Optimization Problems](#).

Steiner tree problem: A tree of minimum total weight that connects a set of terminal vertices S in a graph $G = (V, E)$. There is a 2-approximation algorithm for the general problem. There are better algorithms when the graph is defined in Euclidean space. For more information, see [12, 25]. Gradient constrained Steiner trees are discussed in Chap. [►Online Frequency Allocation and Mechanism Design for Cognitive Radio Wireless Networks](#) and Steiner minimal trees in Chap. [►Neural Network Models in Combinatorial Optimization](#).

Minimum k -connected graph: A graph G is k -vertex-connected, or simply k -connected, if the removal of up to $k - 1$ vertices along with their incident edges leaves the remaining graph connected. It is k -edge-connected if the removal of up to $k - 1$ edges does not disconnect the graph. In the minimum k -connected graph problem, the objective is to compute a k -connected spanning subgraph of G of minimum cardinality. The problem can be posed in the context of vertex or edge connectivities, and with edge weights. The edge-connectivity problems can be approximated within a factor of 2 from optimal, and a factor smaller than 2 when the

edges do not have weights. The unweighted k -vertex-connected subgraph problem can be approximated within a factor of $1 + 2/k$, and the corresponding weighted problem can be approximated within a factor of $O(\log k)$. When the edges satisfy the triangle inequality, the vertex-connectivity problem can be approximated within a factor of about $2 + 2/k$. These problems find applications in fault-tolerant network design. For more information on approximation algorithms for connectivity problems, see [12, Chap. 6].

Traveling salesman problem (TSP): Finding a shortest tour that visits all the vertices in a given graph with weights on the edges. It has received considerable attention in the literature [34]. The problem is known to be computationally intractable (NP-hard). Several heuristics are known to solve practical instances. Considerable progress has also been made for finding optimal solutions for graphs with a few thousand vertices.

Degree constrained spanning tree: A spanning tree of maximal degree k . This problem is a generalization of the traveling salesman path problem. There is a polynomial-time approximation algorithm that returns a spanning tree of maximal degree at most $k + 1$ if G has a spanning tree of degree k (see [12, Chap. 7]).

Max cut: A partition of the vertex set into (V_1, V_2) such that the number of edges in $E \cap (V_1 \times V_2)$ (edges between a vertex in V_1 and a vertex in V_2) is maximized. It is easy to compute a partition in which at least half the edges of the graph cross the cut. This is a 2-approximation. Semi-definite programming techniques can be used to derive an approximation algorithm with a performance guarantee of about 1.15 (see [12, Chap. 11]).

13 Conclusion

Some of the recent research efforts in graph algorithms have been in the areas of dynamic algorithms, graph layout and drawing, and approximation algorithms and randomized algorithms. The methods illustrated in this chapter find use in algorithms for many optimization problems. For approximation algorithms see the books by Hochbaum [12], Vazirani [44], and Williamson and Shmoys [45]. More information on graph layout and drawing can be found in an annotated bibliography by DiBattista et al. [7].

An exciting result of Karger [27] is a randomized, near-linear time algorithm for computing minimum cuts. Finding a deterministic algorithm with similar bounds is a major open problem.

Iterative rounding [32] has emerged and has a very useful paradigm to design approximation algorithms for a collection of NP-hard optimization problems dealing with networks.

Computing a maximum flow in $O(|E||V|)$ time in general graphs is still open. Several recent algorithms achieve these bounds for certain graph densities. A recent result by Goldberg and Rao [21] breaks this barrier, by paying an extra $\log U$ factor

in the running time, when the edge capacities are in the range $[1 \dots U]$. Finding a maximum matching in time better than $O(|E|\sqrt{|V|})$ and obtaining nontrivial lower bounds are open problems.

Further Reading

The area of graph algorithms continues to be a very active field of research. There are several journals and conferences that discuss advances in the field. Some of the important meetings are “ACM Symposium on Theory of Computing (STOC)”; “IEEE Conference on Foundations of Computer Science (FOCS)”; “ACM-SIAM Symposium on Discrete Algorithms (SODA)”; “International Colloquium on Automata, Languages and Programming (ICALP)”; and the “European Symposium on Algorithms (ESA).” There are many other regional algorithms/theory conferences that carry research papers on graph algorithms. The primary journals that carry articles on current research in graph algorithms are “Journal of the ACM,” “SIAM Journal on Computing,” “SIAM Journal on Discrete Mathematics,” “ACM Transactions on Algorithms,” “Algorithmica,” “Journal of Computer and System Sciences,” “Information and Computation,” “Information Processing Letters,” “Networks,” “Internet Mathematics,” “Journal of Graph Algorithms and Applications,” and “Theoretical Computer Science.”

To find more details about some of the graph algorithms described in this chapter, the reader is referred to the books by Cormen, Leiserson, Stein, and Rivest [11]; Even [15]; Gibbons [20]; Kleinberg and Tardos [30]; and Tarjan [43]. Ahuja, Magnanti, and Orlin [1] have provided a comprehensive book on the network flow problem. The survey chapter by Goldberg, Tardos, and Tarjan [23] provide an excellent survey of various min-cost flow and generalized flow algorithms. A detailed survey describing various approaches for solving the matching and flow problems can be found in Tarjan’s book [43]. Papadimitriou and Steiglitz [38] discuss the solution of many combinatorial optimization problems using a primal-dual framework. For understanding recent methods on iterative rounding, the book by Lau, Ravi, and Singh [32] is suggested. For the area of approximation algorithms with an extensive coverage of graph algorithms, books by Hocbaum [12], Vazirani [44], and Williamson and Shmoys [45] are available.

Glossary

Assignment problem: the problem of finding a matching of maximum (or minimum) weight in an edge-weighted graph.

Augmenting path: a path used to augment (increase) the size of a matching or a flow.

Blocking flow: a flow function in which any directed path from s to t contains a saturated edge.

Blossoms: odd-length cycles that appear during the course of the matching algorithm on general graphs.

Branching: a spanning tree in a rooted graph, such that each vertex has a path to the root (also known as in-branching). An out-branching is a rooted spanning tree in which the root has a path to every vertex in the graph.

Capacity: the maximum amount of flow that is allowed to be sent through an edge or a vertex.

Chromatic index: the minimum number of colors with which the edges of a graph can be colored.

Chromatic number: the minimum number of colors needed to color the vertices of a graph.

Chinese postman problem: the problem of finding a minimum length tour that traverses each edge at least once.

Concurrent flow: a multi-commodity flow in which the same fraction of the demand of each commodity is satisfied.

Edge coloring: an assignment of colors to the edges of a graph such that no two edges incident to a common vertex receive the same color.

Euler tour: a tour of the graph that visits each of its edges exactly once.

Eulerian graph: a graph that has an Euler tour.

Integer multi-commodity flow: a multi-commodity flow in which the flow through each edge of each commodity is an integral value. The term is also used to capture the multi-commodity flow problem in which each demand is routed along a single path.

Matching: a subgraph in which every vertex has degree at most one.

Maximum flow: the maximum amount of flow that can be sent from a source vertex to a sink vertex in a given network.

Multi-commodity flow: a network flow problem involving multiple commodities, in which each commodity has an associated demand and source-sink pairs.

Network flow: an assignment of flow values to the edges of a graph that satisfies flow conservation, skew symmetry, and capacity constraints.

Traveling salesman problem: the problem of computing a minimum length tour that visits all vertices exactly once.

s-t cut: a partition of the vertex set into S and T such that $s \in S$ and $t \in T$.

Vertex coloring: an assignment of colors to the vertices of a graph such that no two adjacent vertices receive the same color.

Cross-References

- ▶ [A Unified Approach for Domination Problems on Different Network Topologies](#)
- ▶ [Combinatorial Optimization Techniques for Network-Based Data Mining](#)
- ▶ [Fuzzy Combinatorial Optimization Problems](#)
- ▶ [Neural Network Models in Combinatorial Optimization](#)
- ▶ [Online Frequency Allocation and Mechanism Design for Cognitive Radio Wireless Networks](#)

- ▶ Optimal Partitions
- ▶ Quadratic Assignment Problems
- ▶ Resource Allocation Problems
- ▶ Binary Unconstrained Quadratic Optimization Problem

Recommended Reading

1. R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network flows* (Prentice Hall, Englewood Cliffs, 1993)
2. I. Althöfer, G. Das, D. Dobkin, D. Joseph, J. Soares, On sparse spanners of weighted graphs. *Discret. Comput. Geom.* **9**(1), 81–100 (1993)
3. B. Awerbuch, T. Leighton, A simple local-control approximation algorithm for multicommodity flow, in *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, Palo Alto, California, 3–5 November 1993, pp. 459–468
4. B. Awerbuch, T. Leighton, Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks, in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, Montréal, 23–25 May 1994, pp. 487–496
5. B. Awerbuch, A. Baratz, D. Peleg, Cost-sensitive analysis of communication protocols, in *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, Québec City, 22–24 Aug 1990, pp. 177–187
6. B. Awerbuch, A. Baratz, D. Peleg, Efficient broadcast and light-weight spanners. Manuscript (1991)
7. G.D. Battista, P. Eades, R. Tamassia, I.G. Tollis, Annotated bibliography on graph drawing algorithms. *Comput. Geom.* **4**, 235–282 (1994)
8. B. Bollobás, *Extremal Graph Theory* (Academic, New York, 1978)
9. B. Chandra, G. Das, G. Narasimhan, J. Soares, New sparseness results on graph spanners, in *Proceedings of the 8th Annual ACM Symposium on Computational Geometry* (ACM, Berlin, Germany, 1992), pp. 192–201
10. J. Cong, A.B. Kahng, G. Robins, M. Sarrafzadeh, C.K. Wong, Provably good performance-driven global routing. *IEEE Trans. CAD* **11**(6), 739–752 (1992)
11. T.H. Cormen, C.E. Leiserson, C. Stein, R.L. Rivest, *Introduction to Algorithms* (MIT, Cambridge, 1989)
12. D.S. Hochbaum (ed.) *Approximation Algorithms for NP-Hard Problems* (PWS Publishing, Boston, 1996)
13. J. Edmonds, E.L. Johnson, Matching, Euler tours and the Chinese postman. *Math. Program.* **5**, 88–124 (1973)
14. J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems. *J. Assoc. Comput. Mach.* **19**, 248–264 (1972)
15. S. Even, *Graph Algorithms* (Computer Science Press, Rockville, 1979)
16. L.R. Ford, D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, 1962)
17. M. Fujii, T. Kasami, K. Ninomiya, Optimal sequencing of two equivalent processors. *SIAM J. Appl. Math.* **17**(4), 784–789 (1969)
18. H.N. Gabow, Z. Galil, T. Spencer, R.E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica* **6**, 109–122 (1986)
19. M.R. Garey, D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness (Freeman, San Francisco, 1979)
20. A.M. Gibbons, *Algorithmic Graph Theory* (Cambridge University Press, New York, 1985)
21. A. Goldberg, S. Rao, Beyond the flow decomposition barrier, in *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, Miami Beach, Florida, 20–22 October 1997, pp. 2–11
22. A.V. Goldberg, R.E. Tarjan, Finding minimum-cost circulations by canceling negative cycles. *J. Assoc. Comput. Mach.* **36**(4), 873–886 (1989)

23. A.V. Goldberg, É. Tardos, R.E. Tarjan, Network flow algorithms, in *Algorithms and Combinatorics Volume 9: Flows, Paths and VLSI Layout*, ed. by B. Korte, L. Lovász, H.J. Prömel, A. Schrijver (Springer, Berlin, 1990), pp. 101–164
24. J. Hao, J.B. Orlin, A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithm.* **17**(3), 424–446 (1994)
25. F. Hwang, D.S. Richards, P. Winter, *The Steiner Tree Problem* (North-Holland, Amsterdam, 1992)
26. D. Karger, R. Motwani, M. Sudan, Approximate graph coloring by semidefinite programming, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* Santa Fe, New Mexico, 20–22 November 1994, pp. 2–13
27. D.R. Karger, Minimum cuts in near-linear time, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, Philadelphia, 22–24 May 1996, pp. 56–63
28. H. Karloff, *Linear Programming* (Birkhäuser, Boston, 1991)
29. S. Khuller, B. Raghavachari, N. Young, Balancing minimum spanning trees and shortest-path trees. *Algorithmica* **14**(4), 305–321 (1995)
30. J. Kleinberg, É. Tardos, *Algorithm Design* (Pearson Education, Boston, 2006)
31. J.M. Kleinberg, Single-source unsplittable flow, in *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, Burlington, Vermont, 14–16 October 1996, pp. 68–77
32. L-C. Lau, R. Ravi, M. Singh, *Iterative Methods in Combinatorial Optimization* (Cambridge University Press, Cambridge, 2011)
33. E. L. Lawler, *Combinatorial Optimization: Networks and Matroids* (Holt, Rinehart and Winston, New York, 1976)
34. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization* (Wiley, New York, 1985)
35. L. Lovász, M.D. Plummer, *Matching Theory* (Elsevier, New York, 1986)
36. S. Micali, V.V. Vazirani, An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs, in *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, Syracuse, New York, 13–15 October 1980, pp. 17–27
37. H. Nagamochi, T. Ibaraki, Computing edge-connectivity in multi-graphs and capacitated graphs. *SIAM J. Disc. Math.* **5**(1), 54–66, 1992.
38. C.H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice Hall, Englewood Cliffs, 1982)
39. D. Peleg, A. Schäffer, Graph spanners. *J. Graph Theory* **13**, 99–116 (1989)
40. B. Saha, A. Hoch, S. Khuller, L. Raschid, X-N. Zhang, Dense subgraphs with restrictions and applications to gene annotation graphs, in *RECOMB*, ed. by B. Berger. Volume 6044 of Lecture Notes in Computer Science (Springer, Berlin, 2010)
41. M. Stoer, F. Wagner, A simple min-cut algorithm. *J. Assoc. Comput. Mach.*, **44**(4), 585–590 (1997)
42. É. Tardos, A strongly polynomial minimum cost circulation algorithm. *Combinatorica* **5**, 247–255 (1985)
43. R.E. Tarjan, *Data and Network Algorithms* (Society for Industrial and Applied Mathematics, Philadelphia, 1983)
44. V.V. Vazirani, *Approximation Algorithms* (Springer, Berlin, 2001)
45. D.P. Williamson, D.S. Shmoys, *Design of Approximation Algorithms* (Cambridge University Press, New York, 2011)

Neural Network Models in Combinatorial Optimization

Mujahid N. Syed and Panos M. Pardalos

Contents

1	Introduction	2028
1.1	Objective	2029
1.2	Outline	2029
2	Review	2030
2.1	Artificial Neural Networks (ANNs)	2030
2.2	Example: Implementation	2033
2.3	Hopfield and Tank (H-T) Models	2037
2.4	Durbin-Willshaw (D-W)'s Model	2040
2.5	Kohonen Model	2041
2.6	Lagrangian Model	2042
2.7	General Methodology	2044
2.8	Example: Mapping	2045
3	Optimality Conditions	2047
3.1	System Dynamics	2048
3.2	Lyapunov Energy Function	2049
3.3	Penalty-Based Energy Functions	2057
3.4	Lagrange Energy Functions	2059
4	Escaping Local Minima	2061
4.1	Stochastic Extensions	2061
4.2	Chaotic Extensions	2063
4.3	Convexification	2065
4.4	Hybridization	2066
5	General Optimization Problems	2066
5.1	Linear Programming	2066

M.N. Syed (✉)

Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: smujahid@ufl.edu; snumujahid@gmail.com

P.M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: pardalos@ufl.edu

5.2 Convex Programming.....	2069
5.3 Quadratic Programming.....	2070
5.4 Nonlinear Programming.....	2071
5.5 Complementarity Problem.....	2072
5.6 Mixed Integer Programming Problems (MIPs).....	2073
6 Discrete Optimization Problems.....	2076
6.1 Graph Problems.....	2076
6.2 Shortest Path Problems.....	2081
6.3 Number Partitioning Problems (NPP).....	2082
6.4 Assignment Problems.....	2082
6.5 Sorting Problems.....	2083
6.6 Traveling Salesman Problems (TSP).....	2084
7 Criticism.....	2085
8 Conclusion.....	2087
Cross-References.....	2087
Recommended Reading.....	2087

Abstract

This chapter reviews the theory and application of artificial neural network (ANN) models with the intention of solving combinatorial optimization problems (COPs). Brief introductions to the theory of ANNs and to the classical models of ANNs applied to COPs are presented at the beginning of this chapter. Since the classical ANN models follow gradient-based search, they usually converge to a local optimal solution. To overcome this, several methods that extend the capability of ANNs to avoid the local minima have been reviewed in this chapter. Apart from that, not all the ANNs converge to a local minimum; thus, stability and/or convergence criteria of various ANNs have been addressed. The thin wafer that divides continuous and discrete optimization problems while applying ANNs to solve the COPs is highlighted. Applications of ANNs to solve the general optimization problems and to solve the discrete optimization problems have been surveyed. To conclude, issues regarding the performance behavior of the ANNs are discussed at the end of this chapter.

1 Introduction

Combinatorial optimization problems (COPs) are special problems in the field of operations research, where a real-valued mathematical function is optimized over a given domain (feasible set). Either some or all the variables of these problems are discrete in nature. The main criterion in proposing a solution method to solve the COPs is to provide a solution strategy better than the complete enumeration method. The conventional methods that are used to find the solution of a COP include branching, bounding, cutting planes, etc. Usually, COPs are NP-Hard in nature; the solution time of the conventional solution methods grows exponentially with the problem size. A curiosity to improve the solution time of finding an optimal solution of a COP directed the research toward unconventional methods. The primary unconventional methods for solving the COPs are the heuristics, and the most

prominent ones among them are the metaheuristics, such as simulated annealing, genetic and evolutionary algorithms, tabu search, and greedy randomized adaptive search procedure [15, 38, 44, 46, 53, 87, 94, 98, 115]. On the other hand, there were simultaneous research [32, 112] to use the analog circuits instead of the conventional computers for solving COPs. A method for solving the linear and the quadratic programs using resistors, diodes, transformers, current sources, and voltage sources (the basic building blocks of the analog circuit) was proposed in [32]. However, this method was impractical due to the unavailability of an ideal diode. Thus, this method remained in the theory until improved methods were proposed in [23, 62]. The first known analog circuit implementations which were capable of solving linear and quadratic programming problems were presented in [24, 131]. However, these methods were not ideal due to the use of negative resistors [131]. Later in 1985, Hopfield and Tank [121] proposed an analog circuit consisting of neurons,¹ called artificial neural networks (ANNs), to solve the COPs. This method caught the attention of many researchers, when Hopfield and Tank [121] solved some sufficiently large instances of the symmetric traveling salesman problem (TSP). The proposed method illustrated an alternate approach to solve the COPs, which is independent of the digital computers. After this illustration, many researchers were curious to utilize ANNs as a tool to solve various COPs.

1.1 Objective

The main objective of this chapter is to present different ideas of applying ANNs in solving the COPs. Furthermore, the issues related to the optimality and/or the stability analysis of various ANNs will be dealt. In addition to that, the implementation and the mapping of ANNs to solve the COPs will be illustrated by examples. Nevertheless, a brief history about the ANNs will be reviewed. In addition to that, usage of ANNs in solving the general optimization problems will be addressed.

1.2 Outline

In this chapter, ANN models which are applied in solving the COPs are described. This chapter is organized as follows: an introduction of ANNs in general and a review of the classical models in ANNs that are suitable for solving the COPs in particular are presented in Sect. 2. A systematic method, which illustrates the stability and convergence analysis of ANNs, is described in Sect. 3. Subsequently, the extensions and the improvements that can be incorporated to the classical ANN models are presented in Sect. 4. In addition to that, a methodology of mapping and

¹A term similar to the smallest processing unit in the brain, in ANNs it consist of a small independent circuit with resistors, diodes, capacitors etc.

solving some of the general optimization problems is discussed in Sect. 5. Moreover, the applications of ANNs for solving various discrete programming problems are surveyed in Sect. 6. Nonetheless, in Sect. 7, the criticism of using ANNs to solve the COPs is presented. Finally, this chapter ends with the concluding remarks in Sect. 8.

2 Review

Human brain and its functionality has been the topic of research for many years, and until now, there has been no model that could aptly imitate the human brain. Curiosity of studying human brain lead to the development of ANNs. Henceforth, ANNs are the mathematical models mimicking the interactions among the neurons² within the human brain. Initially, ANNs were developed to provide a novel solution approach to solve the problems which do not have any algorithmic solution approach. The first model that depicted a working ANN used the concept of perceptron³ [82, 101]. After the invention of perceptron, ANNs were the area of interest for many researchers for almost 10 years. However, this area of research was crestfallen due to the results shown by Minsky and Papert in [85]. Their results showed that the perceptron is incapable to model the simple logical operations. However, their work also indirectly suggested the usage of multilayer ANNs. After a decade, as a requital, this area of research became popular with Hopfield's feedback⁴ ANN [56, 57]. A pioneer practical approach to solve the COPs was presented in [58, 121]. The mathematical analysis illustrated by Hopfield demonstrated that ANNs can be used as a gradient descent method in solving optimization problems. Moreover, some instances of traveling salesman problem (TSP) were mapped and solved using ANNs. This approach kindled enthusiasm among many researchers to provide an efficient solution methodology for solving NP-Hard problems. Since then, ANNs were modified and applied in numerous ways to solve the COPs.

2.1 Artificial Neural Networks (ANNs)

Neurons are the primary elements of any ANN. They posses some memory represented by their state. The state of a neuron is represented either by a single state or a dual state (internal and external state). The state of a neuron can take any real value between the interval $[0, 1]$.⁵ Furthermore, the neurons interact with each other via links to share the available information. In general, the external state of the

²Biological neurons are the basic building blocks of the nervous system.

³A simple single-layered artificial neural network invented by Frank Rosenblatt.

⁴Feedback and feed-forward are two main classes of neural networks and will be explained in the following subsections.

⁵Some authors prefer to use the values between $[-1, 1]$; however, both the definition are interchangeable and have the same convergence behavior. Hence, in this work $[0, 1]$, representation is followed.

neuron takes part in the outer interactions, whereas the internal state of the neuron takes part in the firing⁶ decision. The intensity and sense of interactions between any two connecting neurons are represented by the weight (or synaptic⁷ weight) of the links.

A single neuron itself is incapable to process any complex information. However, it is the collective behavior of the neurons that results in the intelligent information processing. Usually, neurons update their states based on the weighted cumulative effect of the states of the surrounding neurons. The update rule for the states is

$$\mathcal{I}_i = \sum_{j=1, j \neq i}^n w_{ij} x_j - U_i \quad (1)$$

$$x_i = \Psi(\mathcal{I}_i) \quad (2)$$

where x_i and \mathcal{I}_i represent the external and internal states of an i th neuron, respectively, w_{ij} represents the weight between the i th neuron and the j th neuron, and U_i represents the threshold level of the i th neuron. Function $\Psi()$ is called the transfer function.⁸ If the collective input to the i th neuron $(\sum_{j=1, j \neq i}^n w_{ij} x_j)$ is greater than the threshold (U_i), then the neuron will fire (value of x_i is set as 1). This is the basic mechanism of any ANN. With appropriate network structure and link weights, ANNs are capable of making any logical operations.

Example 1 Consider the following neural networks:

In Fig. 1, a simple neural network, which performs logical AND and logical OR operations, is depicted. The number over the links represents the weight of the link. The number inside the neurons represents the threshold value. Each neural network has two binary input signals and one binary output signal. Similarly, in Fig. 2, another elementary logical operator XOR is presented (where XOR is logical exclusive OR operator). From this figure, it can be seen that for a specific operation, neural network may not have a unique representation.

From Figs. 1 and 2, it can be seen that although a single neuron does not possess the computation ability, collectively, they can perform any logical operations very easily. Moreover, these are just the elementary form of logical operations that can be performed using simple neural networks. ANNs can be used for the complex logical operations as well. In order to use ANNs for complex logical operations, feedback method is used. Based on the feed mechanism, ANNs can be classified as feed-forward neural networks and Feedback neural networks as shown in Fig. 3.

⁶Whether to set the value of the corresponding external state to 0 or 1

⁷The term synaptic is related to the nervous system and is used in ANNs to indicate the weight between any two neurons.

⁸This function may be continuous or discontinuous based on the type of the neuron. That is, for a discrete case, this function is similar to a signum function, whereas for continuous case, this function is similar to a sigmoidal function.

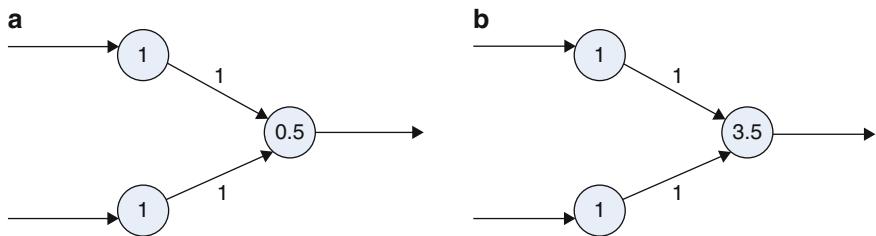


Fig. 1 Example:1 simple logical operations in ANNs. (a) OR. (b) AND

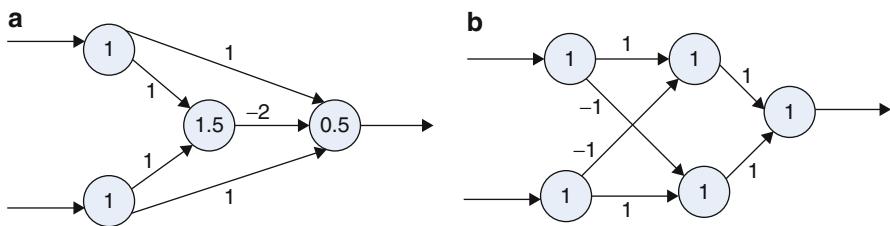


Fig. 2 Example:2 simple logical operations in ANNs. (a) XOR. (b) XOR

Fig. 3 Major categories of ANNs. (a) Feed-forward ANN. (b) Feedback ANN



In the feed-forward neural networks, information is processed only in one direction. Whereas in the case of feedback neural networks, information is processed in both the directions. From the literature, typically, there are three main types of applications of ANNs in information processing. They are categorized as:

Type 1: Pattern Recognition

- This is the most explored application of the ANNs [14, 19, 88, 89]. In pattern recognition, two sets of data, namely, *training data*, and *testing data* are given. Generally, the data sets consist of input parameters which are believed to be the cause of the output features. The aim of pattern recognition is to find a relation that maps the input parameters with the output features. This is obtained by learning, where the weights of the neural network are updated by error backpropagation. When the error between the generated features and the required output features falls below some acceptable limit, the weights are frozen or set to the current value. These weight are used to validate the performance of the obtained neural network using the testing data. This approach is used in function approximation, data prediction, curve fitting, etc.

Type 2: Associative Memory

- Advancement in the theory of ANNs from feed-forward neural networks to feedback neural networks led to this application [18, 84, 93]. In associative memory application, a neural network is set to the known target memory pattern. Later, if an incomplete or partly erroneous pattern is presented to the trained

ANN, then the ANN is capable to generate or repair the input pattern to the most resembling target pattern.

This approach is used in handwriting recognition, image processing, etc.

Type 3: Optimization

- Further extensions of associative memory approach led to the application of ANNs in solving the optimization problems. This application is fundamentally different from the pattern recognition application. In this approach, weights of the neural network are fixed, and the inputs are randomly (synchronously or asynchronously) updated such that the total energy of the network is minimized. This minimization is achieved if the network is stable. Based on the different optimality conditions, the system may converge to the local (or global) optimal solution. ANNs have been applied not only for solving the COPs [3, 43, 64, 97, 110, 111] but also for solving the optimization problems in general [39, 86, 90, 104]

In the following subsection, an example implementation of ANNs for solving a general linear programming (LP) problem is presented.

2.2 Example: Implementation

One of the conventional methods of solving any COP is to relax the problem into a series of continuous problems, and solve them until the desired discrete solution is obtained [47]. Although simplex is the most widely used method to solve the linear relaxation of COPs, it has been shown in the literature that other methods like dual-simplex, barrier methods, and interior point methods perform better than simplex on certain occasions [66, 99, 133]. Apart from that, the convergence time of simplex method is exponential in time [69], yet in practice, it performs better than polynomial time interior point method. Thus, finding a polynomial time algorithm (polynomial both in theory and in practice) for solving linear programming problems is still an attractive area of research. Therefore, it becomes an attractive choice to study the method of solving an LP problem using an ANN. Apart from that, the purpose of selecting an LP model to show the implementation of ANNs is due to the simple nature of LP models. Thus, a reader familiar with the theory of LP will find it easy to understand the analog circuit of the LP problem.

Consider a standard form of a linear programming problem described as

minimize :

$$\mathbf{c}^T \mathbf{x}$$

subject to :

$$A\mathbf{x} = \mathbf{b}$$

$$0 \leq x_i \leq x_{\max} \quad \forall i = 1, 2, \dots, n \quad (3)$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, and $\mathbf{x} \in \mathbb{R}^n$. In order to apply an ANN to solve this problem, the above stated problem has to be transformed into an unconstrained optimization problem using either a penalty function or a Lagrange function method. For the sake of simple illustration, a penalty function approach will be presented. Let $\mathbf{x}(t)$ represent the instantaneous or dynamic value of the solution vector at time t . Let $E(\mathbf{x}(t), t)$ represent the modified (penalized) objective function, which is given as

$$E(\mathbf{x}(t), t) = \frac{C_1}{2} (\mathbf{A}\mathbf{x}(t) - \mathbf{b})^T (\mathbf{A}\mathbf{x}(t) - \mathbf{b}) + C_2 \mathbf{c}^T (\bar{\mathbf{x}} + \mathbf{x}(t) e^{-C_3 t}) \quad (4)$$

where C_1 , C_2 , and C_3 represent the positive scaling parameters of the system. Let the system dynamics be defined as

$$\nabla_t \mathcal{I} = -\nabla_{\mathbf{x}(t)} E(\mathbf{x}(t), t) \quad (5)$$

where \mathcal{I} represents the internal energy of the neuron. Based on the system dynamics defined in Eq. (5), the system can be described as

$$\nabla_t \mathcal{I} = -C_1 \mathbf{A}^T \mathbf{A} \mathbf{x}(t) + C_1 \mathbf{A}^T \mathbf{b} - C_2 \mathbf{c} e^{-C_3 t} \quad (6)$$

and

$$x_i(t) = \frac{x_{\max}}{1 + e^{(-C_4 \mathcal{I}(t))}} \quad (7)$$

where C_4 is a positive scaling parameter,

$$\nabla_t \mathcal{I} = \left(\frac{d\mathcal{I}_1}{dt}, \frac{d\mathcal{I}_2}{dt}, \dots, \frac{d\mathcal{I}_n}{dt} \right)^T, \quad \mathcal{I} \in \mathbb{R}^n$$

and

$$\nabla_{\mathbf{x}(t)} E(\mathbf{x}(t), t) = \left(\frac{\partial E(\mathbf{x}(t), t)}{\partial x_1}, \frac{\partial E(\mathbf{x}(t), t)}{\partial x_2}, \dots, \frac{\partial E(\mathbf{x}(t), t)}{\partial x_n} \right)^T$$

Equations (6) and (7) are the main equations which are used to design the ANN. The first thing in designing the neural network is to know the number of neurons. Since each variable is represented by a neuron, there will be altogether n neurons required for the construction of ANN. Let \mathcal{I}_i represent the internal state of an i th neuron and x_i represent the external state of an i th neuron. At any given time, both the states of an i th neuron can be represented by $[\mathcal{I}_i(t), x_i(t)]$. The next step will be to connect the network using links. These links will have resistors, which represent the weights (synaptic weights) of the link. Based on the coefficients in Eq. (6), the weights are taken as the corresponding coefficients of the first-order term. The constant term in Eq. (6) is taken as the input bias to the neurons. From the Eq. (6), the following weight matrix and bias vector is obtained:

$$W = -C_1 \mathbf{A}^T \mathbf{A} \quad \text{and} \quad \theta(t) = C_1 \mathbf{A}^T \mathbf{b} - C_2 \mathbf{c} e^{-C_3 t} \quad (8)$$

or

$$w_{ij} = -C_1 \sum_{k=1}^m a_{ki} a_{kj} \quad \text{and} \quad \theta_i(t) = C_1 \sum k = 1^m a_{ki} b_k - C_2 c_i e^{-C_3 t} \quad (9)$$

where a_{ij} is the i th row j th column element of A and w_{ij} is the i th row j th column element of W .

Now, in order to set the appropriate weights, we use resistors. The absolute value of synaptic weight w_{ij} (between i th and j th neuron) is obtained as the ratio of the resistors R_f and R_{ij} , that is,

$$|w_{ij}| = \frac{R_f}{R_{ij}} \quad (10)$$

The next step is to assign input bias to each neuron. This can be done using the voltage E_i and the resistor R_i , given as

$$\theta_i = \frac{R_f E_i}{R_i} \quad (11)$$

The other part of the input bias, that is, $C_2 c_i e^{-C_3 t}$ is obtained by providing a discharging loop, with an additional set of capacitor C_d , and resistor R_d . The configuration is shown in Fig. 4. The values of R_d and R_f are taken arbitrarily such that $R_d \ll R_f$. The initial value of C_d is assigned as $-C_2 c_i \forall i$. Once these values are set, the other values are given as

$$R_{ij} = \frac{R_f}{w_{ij}} \quad (12)$$

$$C_3 \approx 1/(R_d C_d) \quad (13)$$

The positive (or negative) value of weight is obtained by using the x_i th terminal (or $-x_i$ th terminal) of the neuron. For the sake of simplicity, the neuron in Fig. 4 can be represented as depicted in Fig. 5. Once the structure of a single neuron is formed, it is connected to the other neurons to form the ANN which can be represented as in Fig. 6. As the time proceeds, the neurons update their internal and external energies based on the update rules and based on the dynamics of selecting the neurons for the update. If the network is stable and the dynamics is appropriate, the network will converge to a limit point(local minimum), which will give the optimal solution of the LP [39]. Applying ANNs to solve the LP problems has been studied and analyzed in [25–28, 35, 67, 100, 136].

The objective of depicting the method of developing analog circuit for solving LP was to illustrate the reader various complexities involved in the deployment of an analog ANN for solving any COP. In the following subsections, some of the classical models of the ANNs that are applied in solving COPs will be described.

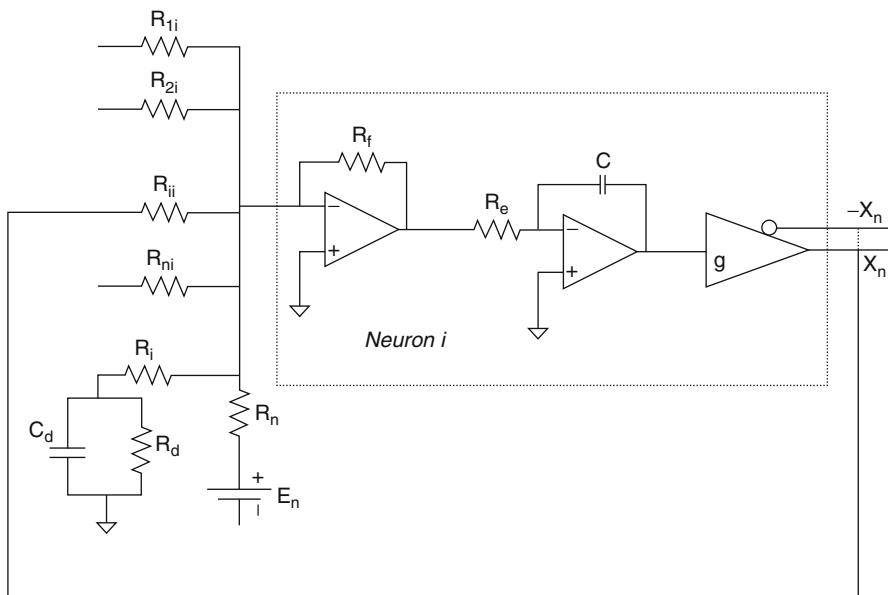


Fig. 4 Circuit representation of a neuron

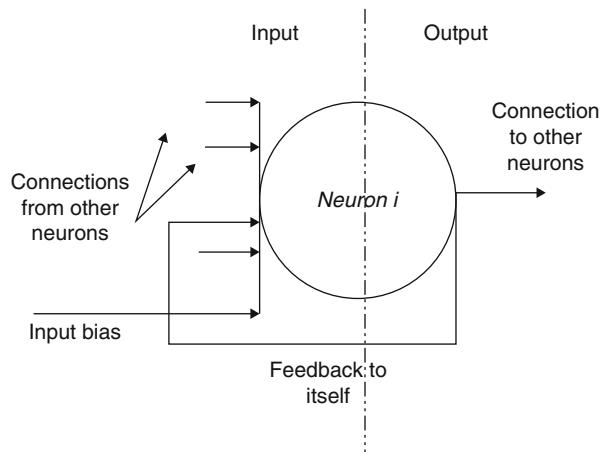
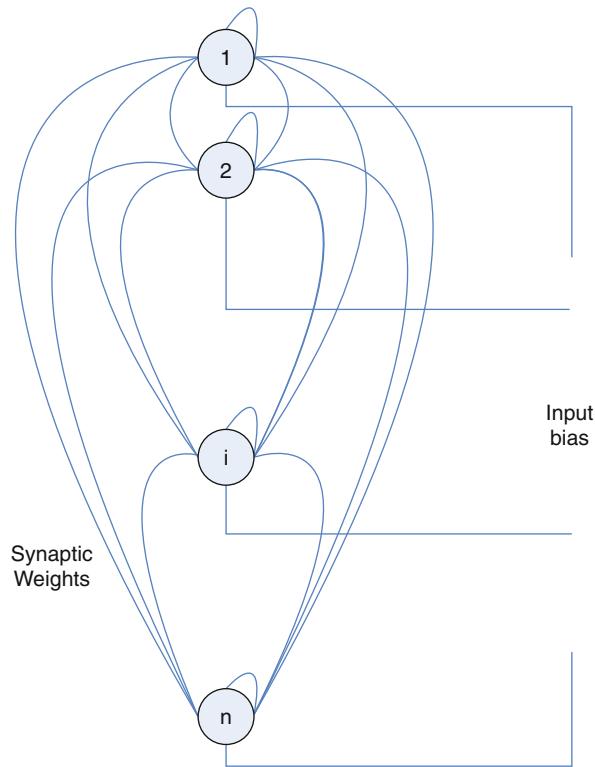


Fig. 5 Block representation of a neuron

These are the basic models proposed in 1980s–1990s as an alternate analog method to solve COPs. These models can be broadly classified as:

- Gradient-based methods
 - Hopfield and Tank's discrete and continuous models
- Self-organizing methods
 - Durbin-Willshaw's and Kohonen's models
- Projective gradient methods
 - Lagrange-based models

Fig. 6 Interconnected neurons



2.3 Hopfield and Tank (H-T) Models

In 1982, Hopfield [56] proposed a novel feedback type neural network, which is capable of performing computational tasks. Although feedback type models were proposed earlier by Anderson and Kohonen [10, 71], however, Hopfield [57] provided a complete mathematical analysis of feedback type neural networks. Thus, these feedback type neural networks were named as Hopfield networks. There are two different models proposed by Hopfield and Tank [58] for solving COPs. They are described below:

(a) *H-T's Discrete Model*

The discrete model consists of n interconnected neurons. The strength of the connection between the i th neuron and the j th neuron is represented by the weight w_{ij} . A presence of connection between any two neurons is indicated by a nonzero weight. Weights maybe positive or negative, representing the sense of connection (excitatory or inhibitory). In this model, each neuron has been assigned with the two states (external and internal states, like that in McCullough and Pitts [82] model) representing the embedded memory at each neuron. Moreover, the internal state of i th neuron I_i is continuous valued, whereas the external state x_i is binary valued. The internal and external state of a neuron are related as

$$\mathcal{I}_i(t+1) = \sum_{j=1}^n w_{ij} x_j(t) + \theta_i \quad (14)$$

$$x_i(t+1) = \Psi_d(\mathcal{I}_i) \begin{cases} 1 & \text{if } \mathcal{I}_i > U_i \\ 0 & \text{if } \mathcal{I}_i \leq U_i \end{cases} \quad (15)$$

where θ_i is a constant representing an external input bias to the i th neuron. $\Psi_d()$ represents a transfer function, which assign binary values to x_i . U_i represents the threshold value of the i th neuron. If the aggregate sum of the internal energies supplied to the i th neuron is greater than U_i , then this neuron will “fire,” that is, x_i will be set to 1. On the other hand, if the aggregate sum of the internal energies is less than the threshold, the neuron will be in a dormant or “not-firing” state. For solving the COP, w_{ij} ’s are calculated based on the objective function and the constraints of a given COP. The only decision variables of this model are x_i and \mathcal{I}_i . Unless or otherwise specified, the values of U_i ’s are taken as 0.

Initially, a random value is assigned to each external state, and the neurons update their states based on the updating rules. Usually, the neurons are selected randomly for the update (asynchronous updating). This updating rule is proven to converge to one of the stable states [83], which minimizes the energy function, provided the energy function is given by Eq. (16):

$$E_d = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j - \sum_{i=1}^n \theta_i x_i \quad (16)$$

Since one of the states is binary and the update sequence of neurons is random, this model is known as a discrete model with stochastic updates. With the above details, **Algorithm 1** summarizes the mechanism of this model.

This model can be used to compute the local minimum of any quadratic function. Hopfield and Tank [58, 121] showed that the way energy function is constructed will always result in decrease of energy and the algorithm leads to one of the stable states. The algorithm proceeds as the gradient descent method, converging to a local minimum.

The main step in solving a COP using H-T’s discrete model is to convert the constrained COP into an unconstrained COP using a penalty function method (or Lagrange method). Once the unconstrained penalty objective function is obtained, it is compared with the energy function given in Eq. (16).

(b) H-T’s Continuous Model

In the subsequent research [57], H-T proposed another model in which both the states of the neuron are continuous. In addition, in this model, two additional properties have been assigned to the neurons, namely, the input capacitance C_i and the transmembrane resistance R_i . Moreover, a finite impedance R_{ij} between the

Algorithm 1 Hopfield Discrete Model

```

Randomly assign initial values to  $x_i$ 
termination = false
while (termination) do
    Select one neuron randomly
    Update the states  $\mathcal{I}_i$  and  $x_i$ :
     $\mathcal{I}_i(t+1) = \sum_{j=1}^n w_{ij} x_j(t) + \theta_i$ 
    if  $\mathcal{I}_i > 0$  then
         $x_i(t+1) = \Psi_d(\mathcal{I}_i) = 1$ 
    else
         $x_i(t+1) = \Psi_d(\mathcal{I}_i) = 0$ 
    end if
    Calculate the energy of the system  $E_d$ :
     $E_d = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j - \sum_{i=1}^n \theta_i x_i$ 
    if termination criteria is met then
        termination = true
    end if
end while

```

i th neuron and the output x_j from j th neuron is assigned on the link ij . The main difference between the continuous model and the discrete model is in the continuous model, the external states of the neurons are continuous valued between 0 and 1. The equations of motion for this model are given as

$$\frac{d\mathcal{I}_i}{dt} = \sum_{j=1}^n w_{ij} x_j(t) - \frac{\mathcal{I}_i}{\tau} + \theta_i \quad (17)$$

$$x_i = \Psi_c(\mathcal{I}_i) \quad (18)$$

where $\tau = R_i C_i$ and usually, τ is assigned a value of 1, if the time step of any discrete time simulation is less than unity. A widely used transfer function is continuous sigmoidal function which is of the form, $\Psi_c(\mathcal{I}_i) = \frac{1}{2}(1 + \tanh(\frac{\mathcal{I}_i}{T}))$. The slope of the transfer function is controlled by parameter T . Similar to the previous model, this model will converge to a stable state which will minimize the energy function, given by Eq. (19):

$$E_c = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j - \sum_{i=1}^n \theta_i x_i + \int_0^{x_i} \Psi_c^{-1}(a) da \quad (19)$$

The only decision variables in this model are x_i and \mathcal{I}_i . **Algorithm 2** describes the mechanism of this model. From this illustration, it can be seen that a continuous ANN can be applied to solve discrete optimization problems, just by tuning the parameters of the transfer function. In specific, if the value of the slope, T , in the transfer function is set to a very high value, then this model will approximately behave as the discrete model.

Algorithm 2 Hopfield Continuous Model

```

Randomly assign initial states  $x_i$ 
termination = false
while (termination) do
    Select randomly one neuron
    Update the states  $\mathcal{I}_i$  and  $x_i$ :
    
$$\frac{d\mathcal{I}_i}{dt} = \sum_{j=1}^n w_{ij} x_j(t) - \frac{\mathcal{I}_i}{\tau} + \theta_i$$

    
$$x_i(t+1) = \Psi_c(\mathcal{I}_i) = \frac{1}{2}(1 + \tanh(\frac{\mathcal{I}_i}{T}))$$

    Calculate the energy of the system  $E_d$ :
    
$$E_c = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i x_j - \sum_{i=1}^n \theta_i x_i + \int_0^{x_i} \Psi_c^{-1}(a) da$$

    if termination criteria is met then
        termination = true
    end if
end while

```

The first attempt to solve the COP using ANN was demonstrated by Hopfield and Tank [58] in 1985. They illustrated the usage of ANN by solving sufficiently large instances of the famous traveling salesman problem (TSP). After this illustration by Hopfield and Tank, numerous approaches and extensions were proposed to solve various COPs using ANNs. Most of the proposed extensions were in the direction of improving H-T's ANN model. In the following subsection, the alternative approaches which are significantly different from H-T's ANN model are presented.

2.4 Durbin-Willshaw (D-W)'s Model

In 1987, Durbin and Willshaw [34] proposed a fundamentally different approach to the H-T's approach for solving the geometric COPs. The proposed method was named as elastic nets and often referred as deformable templates. The origin of this method is the seminal work of Willshaw and Malsburg [130]. In this model, the neurons have no memory or states. Moreover, the network is not arbitrarily connected based on the values of the weights. Instead, it is connected in the form of a ring (elastic ring), where the neurons are placed on the circumference of the ring. Since this model is used to solve the geometric COPs, we will describe the model with respect to TSP.

In this model, there are M neurons on the circumference of the ring, where M is greater than N , the number of cities. The first step is to find the center of gravity of the network. This point is taken as the center for the ring, and iteratively neurons on the rings are pulled apart. In every iteration, neurons are pulled in such a way that the distance between the neurons and the closest city is minimized. However, there is another force from the ring that tries to keep the length of the ring as small as possible. Thus, in the end, when each city has been close enough to a corresponding neuron, a Hamiltonian tour is obtained. The equation describing the movement of j th neuron is given as

$$\Delta y_j = \alpha \sum_{i=1}^N w_{ij} (a_i - y_i) + \beta K (y_{j+1} + y_{j-1} - 2y_j) \quad (20)$$

where a_i is the coordinate of the i th vertex of the TSP; α , β , and K are the scaling parameters:

$$w_{ij} = \frac{\phi(d_{a_i y_j}, K)}{\sum_{k=1}^M \phi(d_{a_i y_k}, K)} \quad \forall i, j \quad (21)$$

$d_{a_i y_j}$ is the Euclidean distance between the i th city and the j th neuron; and

$$\phi(d_{a_i y_j}, K) = e^{-d_{a_i y_j}^2 / 2K^2} \quad (22)$$

α represents the scaling factor that drives the neurons toward the cities whereas β represents the scaling factor for the force that keeps the neighboring neurons on the ring closer. Parameter K is gradually decreased, so that the neurons get closer to the cities. This parameter acts like temperature in simulated annealing and requires a cooling schedule. The energy of this system is given by Eq. (23):

$$E_{dw} = -\alpha K \sum_{i=1}^N \ln \sum_{j=1}^M \phi(d_{a_i y_j}, K) + \frac{\beta}{2} \sum_{j=1}^M d_{y_j y_{j+1}}^2 \quad (23)$$

The mechanism of this model can be explained by Algorithm 3.

2.5 Kohonen Model

Another fundamentally different approach to the H-T's approach was proposed by Kohonen [72] for solving geometric COPs. This model utilizes self-organizing maps (SOMs) [70, 71] which fall under the category of the competitive neural networks. In this model, there are two layers of neurons (input and output). The input layer of the neurons are fully connected to the output layer of the neurons. In general, the neurons at the inner layer represent the input from a high-dimensional space.

Algorithm 3 Durbin-Willshaw model

Find the center of gravity of the system, and assign it as the center of the circle with M equispaced ring points.

```

termination = false
while (termination) do
    Update the coordinates  $y_i$  of ring points using equation Eq. (20)
    if ( $\min_{1,\dots,M} \{d_{a_i,y_i}\} \leq \varepsilon$ ) or ( $K < K_{min}$ ) then
        termination = true
    end if
     $K = \alpha K$   $\alpha \in (0, 1)$ 
end while

```

However, the neurons at the output layer represent output to a low-dimensional space. Therefore, the whole model act as a mapping from a higher dimensional space to a lower dimensional space. The property of self-organizing maps is to map the neurons close in the input space to the neurons that are close in the output space. The output layer of the neurons is arranged according to a topological structure, which is based on the geometry of the problem under consideration. For example: for the TSP, the output layer is arranged in a ring structure. Let $a^i = (a_1^i, a_2^i, \dots, a_n^i)^T \quad \forall i = 1, \dots, N$ be the coordinates of i th city to be visited. Let $w^j = (w_{1j}, w_{2j}, \dots, w_{Nj})^T, \quad \forall j = 1, \dots, M, \quad M \geq N$ represent the weight vector of the j th output neuron. Unlike the previous models, in this model, the weights are the decision variables. They are changed over the time using the following equation:

$$w^j = w^j + \mu f_{j,j^*}(a^i - w^j), \quad \forall j = 1, 2, \dots, M, \quad 0 < \mu < 1 \quad (24)$$

where μ is called learning rate, j^* is called winning neuron, defined as $j^* = \operatorname{argmin}_{\forall j} \{d_{a^i w_j}\}$ for a given city i , and the function $f : (j, j^*) \mapsto [0, 1]$ is a decreasing function and represents the lateral distance between output units j and j^* . This function acts like a weight of attraction between two points. Typically, the function is modified as the algorithm proceed to gradually reduce the magnitude of the weight of attraction. This method can be described by [Algorithm 4](#).

Thus, this iterative approach at the end produces a Hamiltonian tour. Since both Durbin-Willshaw's and Kohonen's models are for geometrical problems, these models have not been widely used in the application of ANNs in solving COPs.

2.6 Lagrangian Model

Another approach which is based on the Lagrange programming is presented by Zhang and Constantinides [139]. The main difference in this method when compared to the Hopfield and Tank [58] approach is that this method is not similar to the gradient descent method. In this method, the constraints and objective function are not coupled using the penalty function. Instead, two types of neurons are used

Algorithm 4 Kohonen Model

Randomly assign initial weights w_j , set $i = 0$.

termination = false

while (*termination*) **do**

 Select neuron $i, i = (i + 1) \bmod (N + 1)$

 Calculate $d_j = d_{a^i w_j} \quad \forall j = 1, \dots, M$

 Assign $j^* = \operatorname{argmin}_j \{d_j\}$ and $d_j = d_{j^*}$

 Update the weights as: $w^j = w^j + \mu f(j, j^*)(a^i - w^j), \quad \forall j = 1, \dots, M, \quad I \& 0 < \mu < 1$

if $\min_j \{d_{a^i w_j}\} \leq \varepsilon, \forall i = 1, \dots, N$ **then**

termination = true

end if

end while

(variable type, x , and Lagrangian type, λ). One type of neurons (Lagrangian) looks for a feasible solution, and the other type of neurons (variable) looks for an optimal solution.

The main advantage of this model over the previous model is that the objective function can be free from the Lyapunov function criterion and can be different from a quadratic function. That is, any type of function can be optimized, unlike the quadratic function requirements of H-T's model [58]. For example, consider any general optimization problem defined as

minimize :

$$f(\mathbf{x})$$

subject to :

$$\begin{aligned} h_i(\mathbf{x}) &= 0 \quad \forall i = 1, \dots, m \\ \mathbf{x} &\in \mathbb{R}^n \end{aligned} \tag{25}$$

where $f(\mathbf{x}), h_i(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ are any continuous and twice differentiable functions of \mathbf{x} . The Lagrange for the problem Eq. (25) is written as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x}) \tag{26}$$

The equations of motion that describe this model are

$$\nabla_t \mathbf{x} = -\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) \tag{27}$$

$$\nabla_t \boldsymbol{\lambda} = -\nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda}) \tag{28}$$

where

$$\begin{aligned} \nabla_t \mathbf{x} &= \left(\frac{dx_1}{dt}, \frac{dx_2}{dt}, \dots, \frac{dx_n}{dt} \right)^T, \quad x \in \mathbb{R}^n \\ \nabla_t \boldsymbol{\lambda} &= \left(\frac{d\lambda_1}{dt}, \frac{d\lambda_2}{dt}, \dots, \frac{d\lambda_m}{dt} \right)^T, \quad \lambda \in \mathbb{R}^m \\ \nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) &= \left(\frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_1}, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_2}, \dots, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_n} \right)^T \end{aligned}$$

and

$$\nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_1}, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_2}, \dots, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_m} \right)^T$$

The above model is iterative as described in [Algorithm 5](#), similar to that of H-T's continuous model. Both the types of neuron are the decision variables.

The mathematical properties and proofs related to convergence of ANN models will be discussed in [Sect. 3](#) of this chapter.

Algorithm 5 Lagrangian Model

Randomly assign initial states x_i and λ_i
 $\text{termination} = \text{false}$
while (termination) **do**
 Select randomly one neuron (either variable or lagrangian)
 if *ordinary variable is selected* **then**
 Update the states x_i as:
 $\nabla_x \mathbf{x} = -\nabla_x L(\mathbf{x}, \boldsymbol{\lambda})$
 else
 Update the states λ_i as:
 $\nabla_{\boldsymbol{\lambda}} \boldsymbol{\lambda} = -\nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda})$
 end if
 Calculate the energy of the system $L(\mathbf{x}, \boldsymbol{\lambda})$:
 $L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x})$
 if *termination criteria is met* **then**
 $\text{termination} = \text{true}$
 end if
end while

2.7 General Methodology

In the previous subsections, a brief picture of different models that can be used to solve a COP is presented with their algorithmic structure. In the following part of this subsection, general guidelines for mapping COPs onto the ANNs models will be presented.

Guidelines for H-T's Model (Discrete and Continuous)

- The output state of the neurons represents the decision variables of the COP.
- Combine the objective function and constraints using the penalty functions, such that the overall modified objective function is quadratic in nature, and it is called as the energy function of the system.
- Obtain the weights and the bias for each neuron. This is the main step that maps the COP onto ANN. This can be done by any one of the following methods:
 - Compare the coefficients of terms in energy function with the coefficients in E_d or E_c functions. This comparison will determine the weights on the links and will determine the input bias of a neuron.
 - Compare the coefficients of degree 1 in the differential update rule to the weights. Similarly, compare the constant term of differential update rule to the bias.
- Assign a random initial state to all the neurons, and proceed as described in [Algorithm 1](#) or [2](#).

Note: Only models with linear and/or quadratic objective function can be solved using the classical H-T's model. For solving the general COPs, various extensions are proposed, like the Lagrange function method.

Guidelines for Durbin-Willshaw's Model

- The decision variable of the COP is represented by the Euclidean position of the neurons.
- Find the center of gravity of the given geometry and mark it as the ring's center.
- Initially, all the neurons are placed uniformly on the circumference of the ring.
- Update the position of the neurons as described in [Algorithm 3](#).

Guidelines for Kohonen's Model

- The decision variables of the COP are represented by the weight vector.
- Initially, all the cities of the input layer are connected with all the neurons of output layer.
- Update the weights of the neurons as described in [Algorithm 4](#).

Note that only COPs which have a low-dimensional geometrical structure can be solved using the Durbin-Willshaw's model or the Kohonen's model. However, there has been extension proposed for solving other COPs, like self-organizing neural networks (SONNs) [108]. SONNs exploit the fact that solutions to many optimization problems can be seen as finding the best arrangement in the permutation matrix. The best arrangement is the one which is both feasible and optimal to the given COP.

Guidelines for Lagrangian Model

- The output state of the neurons (both Lagrangian and variable neurons) represents the decision variable of the COP.
- Combine the objective function and constraints using the Lagrange method. This overall Lagrange function is called the energy function of the system.
- Obtain the weights and the bias. This is the main step that maps COP onto ANN. This can be done by any one of the following methods:
 - Compare the coefficients of terms in energy function with the coefficients in the $L(\mathbf{x}, \lambda)$ function. This comparison will determine the weights on the links and will determine the input bias of a neuron.
 - Compare the coefficients of degree 1 in the differential update rule to the weights. Similarly, compare the constant term of differential update rule to the bias.
- Assign a random initial state to all neurons and proceed as described in [Algorithm 5](#).

In the following subsection, a detail mapping of a COP using the H-T's model will be presented.

2.8 Example: Mapping

In this subsection, an illustration of mapping a COP to the H-T's model will be presented. A symmetric traveling salesman problem (TSP) is selected for the illustration because this problem has been considered as a standard representative test problem for COPs.

Hopfield Mapping

Consider the TSP formulated as

minimize :

$$\sum_{i=1}^N \sum_{k=1, k \neq i}^N \sum_{j=1}^N d_{ik} x_{ij} (x_{k,i+1} + x_{k,i-1}) \quad (29)$$

subject to :

$$\sum_{i=1}^N x_{ij} = 1 \quad \forall j = 1, \dots, N \quad (30)$$

$$\sum_{j=1}^N x_{ij} = 1 \quad \forall i = 1, \dots, N \quad (31)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j = 1, \dots, N \quad (32)$$

where

$$x_{ij} = \begin{cases} 1 & \text{if city } i \text{ is in the position } j \\ 0 & \text{otherwise} \end{cases}$$

and d_{ik} represents the distance between the cities i and j . The first step in mapping TSP to a H-T's model will be converting the given problem into an unconstrained quadratic programming problem, using the penalty functions. Following transformations are used by Hopfield and Tank to map the problem:

- The function in Eq. (30) is transformed into

$$\sum_{i=1}^N \sum_{k=1, k \neq i}^N x_{ij} x_{kj}$$

- The function in Eq. (31) is transformed into

$$\sum_{j=1}^N \sum_{k=1, k \neq j}^N x_{ij} x_{ik}$$

- In order to preserve the equality of 1 in Eqs. (30) and (31), additional penalty is added as

$$\left(\sum_{i=1}^N \sum_{j=1}^N x_{ij} - N \right)^2$$

Thus, the penalized objective function can be written as

$$E_{\text{tsp}} = \frac{A}{2} \sum_{j=1}^N \sum_{i=1}^N \sum_{k=1, k \neq i}^N x_{ij} x_{kj} + \frac{B}{2} \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1, k \neq j}^N x_{ij} x_{ik} \\ + \frac{C}{2} \left(\sum_{i=1}^N \sum_{j=1}^N x_{ij} - N \right)^2 + \frac{D}{2} \sum_{i=1}^N \sum_{k=1, k \neq i}^N \sum_{j=1}^N d_{ik} x_{ij} (x_{k,i+1} + x_{k,i-1}) \quad (33)$$

Comparing the Eq. (33) with (1), we have the following values for the synaptic weights and the input bias:

$$w_{ijkl} = -A\delta_{ik}(1 - \delta_{jl}) - B\delta_{jl}(1 - \delta_{ik}) - C - D\delta_{ik}(\delta_{l(j+1)} + \delta_{l(j-1)}) \quad (34)$$

$$\theta_{ij} = CN \quad (35)$$

where δ_{ik} is the Kronecker Delta.⁹ With the values of weight and bias, H-T's energy function can be written as

$$E_{\text{tsp}} = -\frac{1}{2} \sum_{i,j,k,l=1}^N w_{ijkl} x_{ij} x_{kl} - \sum_{i,j=1}^N \theta_{ij} x_{ij} \quad (36)$$

Thus, the system represented by Eqs. (34)–(36) can be implemented via an analog circuit or stepwise simulation on a digital computer. Although this mapping is not the best mapping (see [119] for a better mapping) for TSP, however, the purpose of this mapping was to illustrate the reader about various changes that are needed for mapping a simple COP onto a given ANN.

In this section, some of the classical models of the ANNs applied to solve the COPs were reviewed. However, the conditions under which the ANNs will converge to the optimal solution are not addressed in this section. In Sect. 3, conditions that guarantee the stability and the convergence ability of ANNs while solving the COPs will be presented.

3 Optimality Conditions

In the previous sections, it was stated that the ANNs will perform a gradient descent method. However, there were no mathematical analysis or proofs provided to support the claim. In this section, the stability and convergence analysis of ANNs will be presented. The objective of the such analysis is to prevent the network from oscillation or chaos and to guarantee its convergence to a local minimum [83, 106]. That is, to analyze that under what conditions from any arbitrary initial point, the ANN will converge to a local stable point. Although the stability term is used frequently from the electrical engineering point of view but from the view

⁹ $\delta_{ik} = 0$ if $i \neq k$; $\delta_{ik} = 1$ if $i = k$.

of optimization, these are more likely to be called as the optimality conditions. Therefore, in the following subsections, the necessary and sufficient conditions that lead the ANNs to converge at the local minima will be analyzed.

3.1 System Dynamics

The method of selecting neurons for the update in a given ANN is called its system dynamics. In the previous subsection, a random method to select a neuron for update is mentioned. However, it should be a valid question to ask, why to select a neuron randomly for the update? Why not select them in a synchronous, parallel, or consecutive way? This subsection investigates the best methods of selecting neurons. Before analyzing different update rules, in this subsection, different dynamics that can be applied to an ANN are described for a discrete state discrete dynamics system.

Consider a general form of discrete time ANN with the following dynamic equations:

$$\mathcal{I}_i(t+1) = \sum_{j=1}^n w_{ij} x_j(t) + \theta_i \quad (37)$$

$$x_i(t+1) = \Psi_d(\mathcal{I}_i) = \begin{cases} 1 & \text{if } \mathcal{I}_i > U_i \\ -1 & \text{if } \mathcal{I}_i \leq U_i \end{cases} \quad (38)$$

where x_i and \mathcal{I}_i represent the external state and internal state of an i th neuron and w_{ij} represents the weight between an i th neuron and a j th neuron. θ_i is a constant representing an external input to the i th neuron. Function $\Psi_d()$ is called as the transfer function. It assigns binary values to x_i . It should be noted that function $\Psi_d()$ is slightly different from the conventional *sign* or *signum* function. This is due to the simple observation that a single dormant neuron (i.e., not-firing neuron) will never excite itself (since $w_{ii} = 0$). U_i represents the threshold value of the i th neuron. If the aggregate sum of internal energies supplied to the i th neuron is greater than U_i , then this neuron will “fire,” that is, x_i will be set to 1. On the other hand, if the aggregate sum of internal energies is less than the threshold, the neuron will be in a dormant or “not-firing” state. Note that, here the external state of the neurons can take values of -1 or 1 . The dynamic equation of the discrete state system can be written in a compact form as

$$x_i(t+1) = \Psi_d \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i \right) \quad (39)$$

The three main types of dynamics that can be implemented in any ANN are described as:

- Synchronous Dynamics

$$x_i(t+1) = \Psi_d \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i \right) \quad \forall i = 1, \dots, n \quad (40)$$

- Parallel Dynamics

$$x_i(t+1) = \begin{cases} \Psi_d \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i \right) & \forall i \in S(t) \\ x_i(t) & \text{otherwise} \end{cases} \quad (41)$$

- Consecutive Dynamics

$$x_i(t+1) = \begin{cases} \Psi_d \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i \right) & i = K \\ x_i(t) & \text{otherwise} \end{cases} \quad (42)$$

where $S(t)$ ¹⁰ represents the set of selected neurons at time t and K ¹¹ represents the single selected vertex at time t . The above equations were defined for discrete state discrete dynamic system. That is, if the system is updated in discrete time, then the system is called as the discrete time system. Similarly, if the state (external) of the system is discrete, it is called as the discrete state system. On the other hand, if the system is updated in continuous time, it is called as the continuous time system. Moreover, if it has continuous state, it is called as the continuous state system. Thus, in general, all the ANN can be classified as discrete states discrete dynamics (DSDD), continuous state discrete dynamics (CSDD), continuous state continuous dynamics (CSCD), and discrete state continuous dynamics (DSCD). In the following sub sections, stability and optimality analysis of different systems will be presented.

3.2 Lyapunov Energy Function

When Hopfield introduced the notion of feedback type ANN, Lyapunov function was used to analyze the convergence characteristics of the H-T's model. This is

¹⁰They can be selected randomly; however, the best way to select them is considering an independent set of neurons. This will be discussed in the following subsections.

¹¹There are many ways to select the k based on the distribution function defined by:

- The gradient rule [50]
- The probabilistic-gradient rule
- The Hopfield rule [56]

However, these distribution functions are independent and have the same mean value (n) [83]. Thus, they convey the idea of random selection of the K th neuron.

done in order to prove that H-T's models are stable in nature and are capable of finding solutions to the optimization problem. Keeping the importance of Lyapunov analysis of ANNs in COPs, some of the main results from the literature will be presented in this subsection. Before proceeding for the Lyapunov analysis, some of the definitions are to be stated.

Basic Definitions:

Consider a dynamic system of the following form:

$$\nabla \mathbf{x}(t) = E(\mathbf{x}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \in \mathbb{R}^n \quad (43)$$

Definition 1 A point \mathbf{x}^* is called the equilibrium point (or critical point, or steady state point) of the system described by Eq. (43) if $E(\mathbf{x}^*) = 0$.

Definition 2 A function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is said to be a Lyapunov function if it holds the following properties:

- Function f is continuous and differentiable.
- The partial derivatives of function f w.r.t. any element of \mathbf{x} are continuous.
- Function f is nonnegative function over its entire domain.
- The derivative of function f w.r.t. time is nonpositive.

Definition 3 For the system described by Eq.(43), an equilibrium point \mathbf{x}^* is said to be Lyapunov stable (or stable in the sense of Lyapunov) if for any positive scalar ϵ , there exists a positive scalar δ such that

$$\|\mathbf{x}(t_0) - \mathbf{x}^*\| < \delta \implies \|\mathbf{x}(t) - \mathbf{x}^*\| < \epsilon \quad \forall t \geq t_0$$

Definition 4 An equilibrium point \mathbf{x}^* is said to be asymptotically stable (or asymptotically stable in the sense of Lyapunov) if the following conditions are satisfied:

- \mathbf{x}^* is stable.
- $\lim_{t \rightarrow \infty} \mathbf{x}(t) = \mathbf{x}^*$.

For the detailed explanation of the above definitions, see [76].

Analysis of DSDD Systems

Now, let us begin with systems of type DSDD. The two widely known Lyapunov functions for the DSDD systems can be written as

- Lyapunav 1

$$E_{\text{DDI}}(t+1) = -\langle \mathbf{x}(t+1), W\mathbf{x}(t) \rangle + \langle \boldsymbol{\theta}, (\mathbf{x}(t+1) + \mathbf{x}(t)) \rangle \quad (44)$$

or

$$E_{\text{DDI}}(t+1) = -\sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1) x_j(t)$$

$$+ \sum_{i=1}^n \theta_i (x_i(t+1) + x_i(t)) \quad (45)$$

- Lyapunov 2

$$E_{DD2}(t+1) = -\frac{1}{2} \langle \mathbf{x}(t+1), W\mathbf{x}(t+1) \rangle + \langle \boldsymbol{\theta}, (\mathbf{x}(t+1)) \rangle \quad (46)$$

or

$$E_{DD2}(t+1) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1)x_j(t+1) + \sum_{i=1}^n \theta_i (x_i(t+1)) \quad (47)$$

Theorem 1 If W is symmetric and update rule is synchronous, then a discrete system defined by Eq. (45) will converge to either a local minima or to a two-edge cycle.

Proof The change brought to the energy function given by Eq. (45) in one iteration is

$$\begin{aligned} \Delta E_{DD1}(t+1) &= (E_{DD1}(t+1)) - (E_{DD1}(t)) \\ &= \left(-\sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1)x_j(t) + \sum_{i=1}^n \theta_i (x_i(t+1) + x_i(t)) \right) \\ &\quad - \left(-\sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t)x_j(t-1) + \sum_{i=1}^n \theta_i (x_i(t) + x_i(t-1)) \right) \end{aligned} \quad (48)$$

solving Eq. (48) will lead to the following:

$$\begin{aligned} \Delta E_{DD1}(t+1) &= \left(\sum_{i=1}^n \theta_i (x_i(t+1) - x_i(t-1)) \right) \\ &\quad + \left(-\sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1)x_j(t) + \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t)x_j(t-1) \right) \end{aligned} \quad (49)$$

Changing index in the second term and using symmetry of W matrix, following simplification is obtained:

$$\begin{aligned}\Delta E_{DD1}(t+1) = & \left(\sum_{i=1}^n \theta_i (x_i(t+1) - x_i(t-1)) \right) \\ & + \left(- \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_j(t) (x_i(t+1) - x_i(t-1)) \right) \quad (50)\end{aligned}$$

Rewriting the above equation,

$$\Delta E_{DD1}(t+1) = - \sum_{i=1}^n (x_i(t+1) - x_i(t-1)) \left(\sum_{j=1}^n w_{ij} x_j(t) - \theta_i \right) \quad (51)$$

For each i th neuron, the contribution to the change in the energy function will be given by

$$(\Delta E_{DD1}(t+1))_i = -(x_i(t+1) - x_i(t-1)) \left(\sum_{j=1}^n w_{ij} x_j(t) - \theta_i \right) \quad (52)$$

or

$$(\Delta E_{DD1}(t+1))_i = (x_i(t-1) - x_i(t+1)) \mathcal{I}_i(t) \quad (53)$$

Thus, for the Lyapunov energy function given by Eq. (45), the change in energy of i th neuron is given by Eq. (3.2).

Since the update rule is synchronous, at every iteration any of the following scenarios may arise:

- If $\mathcal{I}_i(t) > 0 \Rightarrow x_i(t+1) = 1 \Rightarrow (\Delta E_{DD1}(t+1))_i \leq 0 \quad \forall i \in N.$
- If $\mathcal{I}_i(t) \leq 0 \Rightarrow x_i(t+1) = 0 \Rightarrow (\Delta E_{DD1}(t+1))_i \leq 0 \quad \forall i \in N.$

From the above implication, for every i th neuron, $(\Delta E_{DD1}(t+1))_i \leq 0$. Thus, the system is converging. At the converging limit, $(\Delta E_{DD1}(t+1))_i = 0$. This implies the following cases:

- $\mathcal{I}_i(t) = 0 \Rightarrow x_i(t) = x_i(t+1) = 0 \quad \forall i \in N$, a trivial case and can be discarded
- $x_i(t-1) - x_i(t+1) = 0 \quad \forall i \in N$.
 - $\Rightarrow x_i(t-1) = x_i(t+1) \neq x_i(t) \quad \forall i \in N \Rightarrow$ two-edge cycles.
 - $\Rightarrow x_i(t-1) = x_i(t+1) = x_i(t) \quad \forall i \in N \Rightarrow$ local minima.

Thus, the system will converge to a local minimum or to a two-edge cycle. \square

The next case to consider will be DSDD system with parallel dynamics. If a similar energy function is used, then following a similar argument, it is observed for the parallel dynamics that if $i \in S(t-1)$ and $i \notin S(t)$, then the sign of Eq. (3.2) is undetermined. Thus, there could be an increase in the energy of the system, if

parallel dynamics is followed. However, this case will not arise, if the set of neurons for parallel update is selected in a specified way and if a different energy function is used. The following theorem will give the *necessary and sufficient* condition for convergence of parallel dynamics.

Theorem 2 *If W is a symmetric matrix with zero diagonals and if the update rule is parallel, then a discrete system defined by Eq. (47) will converge to a local minima if and only if $S(t)$ contains an independent set of neurons for each iteration t . Where $S(t)$ is said to contain independent set of neurons, if $i, j \in S(t)$, then $w_{i,j} = 0$*

Proof Consider a generalized Lyapunov energy function given by Eq. (47). Let $S(t)$ be the independent set of neurons. The change in energy of the system is given by

$$\begin{aligned}\Delta E_{DD2}(t+1) &= (E_{DD2}(t+1)) - (E_{DD2}(t)) \\ &= \frac{1}{2} \left(- \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1)x_j(t+1) + \sum_{i=1}^n \theta_i(x_i(t+1)) \right) \\ &\quad - \frac{1}{2} \left(- \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t)x_j(t) + \sum_{i=1}^n \theta_i(x_i(t)) \right)\end{aligned}\quad (54)$$

Splitting the above equation for the cases $i, j \in S(t)$ and using symmetric property of W , we get

$$\begin{aligned}\Delta E_{DD2}(t+1) &= \frac{1}{2} \left(- \sum_{i \in S(t)} \sum_{j \notin S(t)} w_{ij} x_i(t+1)x_j(t+1) + \sum_{i=1}^n \theta_i(x_i(t+1)) \right) \\ &\quad - \frac{1}{2} \left(- \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t)x_j(t) + \sum_{i=1}^n \theta_i(x_i(t)) \right)\end{aligned}\quad (55)$$

Now, based on the above equation, the following two scenarios may happen:

- If $\mathcal{I}_k(t) > 0 \Rightarrow x_k(t+1) = 1 \Rightarrow \Delta E_{DD2}(t+1) \leq 0$
- If $\mathcal{I}_k(t) \leq 0 \Rightarrow x_k(t+1) = 0 \Rightarrow \Delta E_{DD2}(t+1) \leq 0$

Thus, the above system is converging. At the converging limit, $\Delta E_{DD2}(t+1) = 0$, which implies the following cases:

- $\mathcal{I}_k(t) = 0 \Rightarrow x_k(t) = x_k(t+1) = 0 \quad \forall k \in N$, a trivial case and can be discarded.
- $x_k(t) - x_k(t+1) = 0 \quad \forall k \in N \Rightarrow$ local minima.

Thus, the system will converge to a local minimum. For the proof of the converse, see [83]. \square

Theorem 3 If W is symmetric, the update rule is consecutive, then a discrete system defined by Eq. (47) will converge to a local minimum.

Proof Consider the Lyapunov energy function given by Eq. (47). The change in energy is given by

$$\begin{aligned} \Delta E_{\text{DDC}}(t+1) &= (E_{\text{DDC}}(t+1)) - (E_{\text{DDC}}(t)) \\ &= \frac{1}{2} \left(- \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1) x_j(t+1) + \sum_{i=1}^n \theta_i(x_i(t+1)) \right) \\ &\quad - \frac{1}{2} \left(- \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t) x_j(t) + \sum_{i=1}^n \theta_i(x_i(t)) \right) \end{aligned} \quad (56)$$

Splitting the above equation for cases when $i = K$ and when $i \neq K$ and using the property that W is symmetric, we get

$$\Delta E_{\text{DDC}}(t+1) = (x_k(t) - x_k(t+1)) \mathcal{I}_k(t) \quad (57)$$

The following two scenarios may happen:

- If $\mathcal{I}_k(t) > 0 \Rightarrow x_k(t+1) = 1 \Rightarrow \Delta E_{\text{DDC}}(t+1) \leq 0$
- If $\mathcal{I}_k(t) \leq 0 \Rightarrow x_k(t+1) = 0 \Rightarrow \Delta E_{\text{DDC}}(t+1) \leq 0$

Thus, the above system is converging. At the converging limit, $\Delta E_{\text{DDC}}(t+1) = 0$, which implies the following cases:

- $\mathcal{I}_k(t) = 0 \Rightarrow x_k(t) = x_k(t+1) = 0 \quad \forall k \in N$, a trivial case and can be discarded.
- $x_k(t) - x_k(t+1) = 0 \quad \forall k \in N \Rightarrow$ local minimum.

Thus, the system will converge to a local minimum. \square

Theorem 4 If W is symmetric positive semidefinite, then a discrete system defined by Eq. (47) will converge to a local minimum.

Proof See [83]. \square

The above analysis is performed for DSDD system. Another important class of system is CSDD system. Similar to the discrete state, three main types of dynamics can be applied for the continuous state system; they can be described as

- Synchronous Dynamics

$$x_i(t+1) = \Psi_c \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i \right) \quad \forall i = 1, \dots, n \quad (58)$$

- Parallel Dynamics

$$x_i(t+1) = \begin{cases} \Psi_c \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i \right) & \forall i \in S(t) \\ x_i(t) & \text{otherwise} \end{cases} \quad (59)$$

- Consecutive Dynamics

$$x_i(t+1) = \begin{cases} \Psi_c \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i \right) & i = K \\ x_i(t) & \text{otherwise} \end{cases} \quad (60)$$

where $S(t)$ and K have the same meaning as in the DSDD case. The main difference between the above equations and the discrete state equations is the definition of the transfer function. Here the transfer function is not similar to *sign* function. However, here the transfer can be a general real valued monotonically increasing continuous function with the following properties:

$$\begin{aligned} &\text{if } a \rightarrow \infty, \Psi_c(a) \rightarrow 1 \quad \forall a \in \mathbb{R} \\ &\text{if } a \rightarrow -\infty, \Psi_c(a) \rightarrow -1 \quad \forall a \in \mathbb{R} \end{aligned} \quad (61)$$

Apart from the above properties, the following property is also incorporated to maintain an easy shift from a discrete state to a continuous state neuron. This is one of the important property of the transfer function, which adds the flexibility of using a continuous state model for solving a discrete COP.

$$\Psi_c(\mu a) \rightarrow \Psi_d(a) \quad \text{as } \mu \rightarrow \infty \quad (62)$$

Even the monotonic increasing criteria is not necessary. To be specific, any monotonic nondecreasing continuous function with the above properties may be used as the transfer function. However, the monotonic increasing continuous function is used only for the sake of simplicity and clearness [83].

There are many standard mathematical functions that appropriately fit the above restriction for the transfer function. The following are some of the widely used transfer functions:

1. $\Psi_c(a) = \tan h(a)$
2. $\Psi_c(a) = \frac{2}{\pi} \tan^{-1}(a)$
3. $\Psi_c(a) = \frac{1-e^{-a}}{1+e^{-a}}$
4. $\Psi_c(a) = \coth(a) - \frac{1}{a}$

In the following part of this section, some more theorems without proof for the CSDD systems are presented. Interested readers are directed to [83] for the detailed proofs of the following theorems.

Theorem 5 *If W is symmetric, $w_{ii} \geq 0$, and the update rule is synchronous, then a discrete system will converge to a local minima or a two-edge cycle, if the following energy function is used.*

$$\begin{aligned}
E_{\text{CDS}}(t+1) = & - \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1) x_j(t) + \\
& \sum_{i=1}^n \theta_i \left(x_i(t+1) + x_i(t) + \sum_{i=1}^n \left(\int_0^{x_i(t+1)} \Psi_{c_i}^{-1}(a) da + \int_0^{x_i(t)} \Psi_{c_i}^{-1}(a) da \right) \right)
\end{aligned} \tag{63}$$

Theorem 6 If W is symmetric and positive semidefinite, then a discrete system will converge to a local minima (irrespective of the type of dynamic used), if the following energy function is used.

$$\begin{aligned}
E_{\text{CD}}(t+1) = & -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1) x_j(t) \\
& + \sum_{i=1}^n \theta_i(x_i(t+1)) + \sum_{i=1}^n \int_0^{x_i(t+1)} \Psi_{c_i}^{-1}(a) da
\end{aligned} \tag{64}$$

The restriction of positive semidefinite W can be relaxed if Ψ_c and Ψ_c^{-1} are differentiable. Let ζ_i be a scalar, defined as

$$\Psi_{c_i}' \leq \zeta_i \quad \text{and} \quad [\Psi_c^{-1}]' \geq \frac{1}{\zeta_i} \tag{65}$$

If such ζ_i 's exists, then the following theorem holds:

Theorem 7 Let $\widehat{w}_{ij} = w_{ij} + \frac{\delta_{ij}}{\zeta_j}$, where $\delta_{i,j}$ is Kronecker delta. Let the energy function be defined as

$$\begin{aligned}
E_{\text{CD}}(t+1) = & -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \widehat{w}_{ij} x_i(t+1) x_j(t) \\
& + \sum_{i=1}^n \theta_i(x_i(t+1)) + \sum_{i=1}^n \int_0^{x_i(t+1)} \Psi_{c_i}^{-1}(a) da
\end{aligned} \tag{66}$$

If \widehat{W} is positive definite, then a discrete system will converge to a local minimum (irrespective of the type of dynamic used).

Furthermore, for the specific case of consecutive dynamics, a more relaxed optimality condition can be used.

Theorem 8 Let the energy function be defined as

$$\begin{aligned}
E_{CD}(t+1) = & -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} x_i(t+1) x_j(t) \\
& + \sum_{i=1}^n \theta_i(x_i(t+1)) + \sum_{i=1}^n \int_0^{x_i(t+1)} \Psi_c^{-1}(a) da
\end{aligned} \quad (67)$$

If $w_{ii} \geq 0$, $w_{ij} + \frac{1}{\zeta_j} > 0$, the update rule is consecutive, and Ψ_c & Ψ_c^{-1} are differentiable, then a discrete system will converge to a local minimum.

A similar analysis can be done for continuous dynamics. Interested readers are referred to [83] for detailed analysis. The above theorems can be easily extended to $\{0, 1\}$ systems. Continuous dynamics are usually hard to simulate on the digital computers and are mostly used in designing the analog circuits. The typical way of implementing continuous dynamics on the digital computer is to discretize the time into small steps. Nevertheless, the objective of presenting the proofs for discrete state systems is to highlight the reader that the convergence of ANNs is a critical issue. Moreover, the convergence of ANNs depends not only on the energy function but also on the systems dynamics. However, it can also be seen from the above proofs that a careful design of energy function (based on various properties of the weight matrix, W) will converge the system to a local minimum, irrespective of the dynamics.

Although Lyapunov function analysis is enough to understand that only for specific scenarios ANNs will converge to local optima. Lyapunov analysis is useful for those optimization problems whose constraints and objective function can be converted into a quadratic form. Moreover, the constraints are to be incorporated into the Lyapunov function using an appropriate penalty function. In general, it is hard to obtain a Lyapunov energy function for a given system. This does not mean that ANNs cannot be applied to these systems. Therefore, for such systems, different types of energy functions are designed.

The primary approach that replaced the usage of Lyapunov energy function analysis is the usage of a general penalty function analysis (by incorporating constraints of a given optimization problem [77]). In the following subsections, the convergence and stability criteria for penalty- and Lagrange-based ANNs will be presented.

3.3 Penalty-Based Energy Functions

Consider the following constrained optimization problem:

minimize :

$$f(\mathbf{x})$$

subject to :

$$\begin{aligned} g_i(\mathbf{x}) &\geq 0 \quad \forall i = 1, \dots, m \\ \mathbf{x} &\in \mathbb{R}^n \end{aligned} \tag{68}$$

where, $f, g_i : \mathbb{R}^n \mapsto \mathbb{R}$. It is assumed that both f, g_i are continuously differentiable functions. The above problem can be converted to an unconstrained optimization problem as

minimize :

$$\begin{aligned} E_{\text{pen}}(\mathbf{x}) \\ \text{subject to :} \\ \mathbf{x} \in \mathbb{R}^n \end{aligned} \tag{69}$$

where

$$E_{\text{pen}}(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^m c_i P_i(\mathbf{x}) \tag{70}$$

and $P_i(\mathbf{x})$ is the penalty function. Now, the convergence of ANNs to solve the above problem depends upon the structure of the penalty function. In this work, convergence analysis based on the penalty function of the form $P_i(\mathbf{x}) = \frac{1}{2}[\min\{0, g_i(\mathbf{x})\}]^2 \forall i = 1, \dots, m$ will be presented. One of the properties of this penalty function is that it can be converted to a continuous differentiable penalty function [77]. Consider the following transformation:

Let

$$v_i = \begin{cases} c_i g_i(\mathbf{x}) & \text{if } g_i(\mathbf{x}) < 0 \\ 0 & \text{if } g_i(\mathbf{x}) \geq 0 \end{cases} \quad \forall i = 1, \dots, m \tag{71}$$

then,

$$c_i P_i(\mathbf{x}) = \frac{c_i}{2} [\min\{0, g_i(\mathbf{x})\}]^2 = \frac{1}{2} v_i g_i(\mathbf{x}) \quad \forall i = 1, \dots, m \tag{72}$$

since $g_i, \forall i = 1, \dots, m$, is assumed to be differentiable, the above penalty function is differentiable.

The dynamics of this system¹² is defined as

$$\frac{dx_i}{dt} = -\frac{1}{C_i} \frac{\partial E(\mathbf{x})}{\partial x_i} \quad \forall i = 1, \dots, n \tag{73}$$

Thus, these systems are also called as *gradient-based systems*. These systems [77] have some of the useful properties that can be used to solve the COPs. These properties are described in the following theorems:

¹²Using Kirchoffs law, see [77]

Theorem 9 Let the system be described by Eq. (73). The isolated equilibrium point of the system is asymptotically stable if and only if it is a strict local minimizer of function E_{pen} .

Theorem 10 Let the system be described by Eq. (73). If the isolated equilibrium point of the system is stable, then it is a local minimizer of function E_{pen} .

For the gradient-based systems, assuming Lipschitz continuity, Han et al. [51] proposed many other convergence properties.

One of the difficulties in applying the penalty based method in the ANNs, is the selection of various penalty parameters. Due to the physical limits on the analog circuits, c_i cannot be set to a large value. In order to overcome the difficulties of the penalty method, and to generalize the use of ANNs, Lagrange based energy function was proposed [139]. Following subsection discuss some of the results for the Lagrange-based ANNs.

3.4 Lagrange Energy Functions

Lagrange based neural networks are inspired by the saddle point theory and the duality analysis. The focus of these networks is to search for a point that satisfies K.K.T's first order necessary conditions of optimality [12]. Therefore, analogous to the gradient descent systems, these network are also known as *projective gradient systems*.

In the following subsections, necessary and sufficient conditions related to Lagrange type energy functions will be presented. Consider the following optimization problem:

minimize :

$$f(\mathbf{x})$$

subject to :

$$h_i(\mathbf{x}) = 0 \quad \forall i = 1, \dots, m \quad (74)$$

$$\mathbf{x} \in \mathbb{R}^n$$

where, $f, h_i : \mathbb{R}^n \mapsto \mathbb{R}$. It is assumed that both, f, h_i are continuously differentiable functions. The above problem can be converted to an unconstrained optimization problem using the Lagrange function, as:

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x}) \quad (75)$$

In this subsection, some of the basic optimality condition in the form of theorems are stated without proof. Interested readers may refer to [12] for the complete discussion, with proofs.

Theorem 11 Let us assume that \mathbf{x}^* is a regular point (one that satisfies constraint qualification). If \mathbf{x}^* is the local minimum of problem Eq. (74), then there exists $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that:

$$\nabla_x L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0 \quad (76)$$

and

$$\mathbf{d}^T \nabla_{xx}^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{d} \geq 0, \quad \forall \mathbf{d} \in \{\mathbf{d} : \nabla_x h(\mathbf{x}^*)^T \mathbf{d} = 0\} \quad (77)$$

Theorem 12 Let us assume that $h(\mathbf{x}^*) = 0$. If $\exists \boldsymbol{\lambda}^* \in \mathbb{R}^m$ such that:

$$\nabla_x L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = 0 \quad (78)$$

and

$$\mathbf{d}^T \nabla_{xx}^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*) \mathbf{d} > 0, \quad \forall \mathbf{d} \in \{\mathbf{d} : \mathbf{d} \neq 0; \nabla_x h(\mathbf{x}^*)^T \mathbf{d} = 0\} \quad (79)$$

then \mathbf{x}^* is the local minimum of problem Eq. (74).

Similar to the case of gradient-based dynamics, Lagrange systems has the following dynamic equations:

$$\nabla_t \mathbf{x} = -\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) \quad (80)$$

$$\nabla_t \boldsymbol{\lambda} = -\nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda}) \quad (81)$$

Although, the above equations are similar to the gradient-based systems, due to the difference in type of variables (ordinary variables and Lagrange variables), the system is named as *projective gradient systems*. In the presence of different types of the variables, the Lagrange will decrease w.r.t. one type of the variables and increase w.r.t. the other. This leads to the saddle point theory. For example, it is very easy to see that

$$\frac{dL(\mathbf{x}, \boldsymbol{\lambda})}{dt} \Big|_{\boldsymbol{\lambda}=\text{constant}} = \sum_{i=1}^n \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_i} \frac{dx_i}{dt} = - \sum_{i=1}^n \left(\frac{dx_i}{dt} \right)^2 \leq 0 \quad (82)$$

$$\frac{dL(\mathbf{x}, \boldsymbol{\lambda})}{dt} \Big|_{\mathbf{x}=\text{constant}} = \sum_{i=1}^m \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_i} \frac{d\lambda_i}{dt} = \sum_{i=1}^m \left(\frac{d\lambda_i}{dt} \right)^2 \geq 0 \quad (83)$$

Thus, $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ act as a saddle point of the system.

Theorem 13 If $\nabla_{xx}^2 L(\mathbf{x}, \boldsymbol{\lambda})$ is positive definite $\forall \mathbf{x} \in \mathbb{R}^n$ and $\forall \boldsymbol{\lambda} \in \mathbb{R}^m$, then the proposed ANN is Lyapunov stable.

The proof of this theorem is presented by Zhang and Constantinides in [139]. From the above theorem, following corollaries can be obtained:

Corollary 1 \mathbf{x} tends to a limit \mathbf{x}^* at which $\nabla_x L(\mathbf{x}^*, \boldsymbol{\lambda}) = 0$.

Corollary 2 If $\nabla h_i(\mathbf{x}^*) \quad \forall i = 1, \dots, m$ are linearly independent, then $\boldsymbol{\lambda}$ converges to $\boldsymbol{\lambda}^*$.

Theorem 14 Let the network be described by Eqs. (75), (80), and (81). If the network is physically stable, then the ANN converges to a local minimum. The point of convergence is a Lagrange solution to problem Eq. (74).

The above restriction of the global convexity can be relaxed to a local convexity, and the following theorem can be stated:

Theorem 15 *Let the network be described by Eqs. (75), (80), and (81). If following conditions hold:*

1. $\exists(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ such that $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ is a stationary point
2. $\nabla_{xx}^2 L(\mathbf{x}^*, \boldsymbol{\lambda}^*) > 0$
3. \mathbf{x}^* is a regular point of problem Eq. (74)

the initial point is in proximity of $(\mathbf{x}^, \boldsymbol{\lambda}^*)$, then the ANN will asymptotically converge to $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$.*

In this section, different ways to study convergence and stability performance of ANNs were illustrated. Various theorems were presented to understand the critical role of the energy functions in the convergence behavior of ANNs. However, the convergence will guarantee local optimality provided suitable energy function, and dynamics are used to update the states of neurons. For obtaining global optimality, there are numerous extensions proposed in the literature. In the following section, a discussion on the methods that can be used to escape from local minima, in the hope of finding the global minimum, is presented.

4 Escaping Local Minima

The very thought of the gradient descent method will kindle the question regarding global optimality. Classical ANNs perform gradient descent method and will hardly converge to the global minimum. In order to overcome such difficulties, various extensions have been proposed. The extension methods that may converge ANNs to global minima by escaping local minima can be broadly classified as:

- Stochastic extensions
- Chaotic extensions
- Convexification
- Hybridization

Clearly, any extension that can be applied to H-T models can be easily applied to the general penalty or the Lagrange-based ANNs. Thus, in most of the cases, the extensions for the H-T's model will be reviewed. In the following part of this section, a brief discussion of the above extension methods will be presented.

4.1 Stochastic Extensions

There are basically four ways to include stochasticity in the H-T's model [107]. They are:

1. Introducing stochasticity in the transfer function
2. Introducing noise in the synaptic weights

3. Introducing noise in the input bias
4. Any combination of the above three

The prominent stochastic extensions that can be applied to the classical ANNs are discussed in the following parts of this subsection.

4.1.1 Boltzmann Machines

This is the first extension of H-T's model, which proposes a way to escape from the local minima [1, 2, 54, 55]. This is done by replacing the deterministic transfer function with a probabilistic transfer function. Let the change in energy of the i th neuron of the H-T's model be represented by ∇E_i . Then, irrespective of the previous state, the i th neuron will fire with probability p_i , which is defined as

$$p_i = \frac{1}{(1 + \exp -\nabla E_i / T)} \quad (84)$$

and

$$x_i = \Psi_{\text{Blz}}(\mathcal{I}_i) = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } (1 - p_i) \end{cases} \quad (85)$$

where T is the parameter similar to the concept of temperature in simulated annealing.

4.1.2 Gaussian Machines

These are the extensions to the Boltzmann machines, where noise is introduced in the inputs [7]. The updating dynamics of an i th neuron in the H-T's model will be modified as

$$\mathcal{I}_i = \sum_{j=1}^n w_{ij} x_j + \theta_i + \varepsilon_i \quad (86)$$

where ε_i is the term representing the noise, which follows Gaussian distribution with mean zero and variance σ^2 . The deviation of σ depends upon the parameter T , the temperature of the Gaussian network. In addition to the noise, activation level a_i is being assigned to every neuron, which has the following dynamics:

$$\frac{da_i}{dt} = -\frac{a_i}{\tau} + \mathcal{I}_i \quad (87)$$

The external state of the i th neuron is determined by the following equation:

$$x_i = \Psi_{\text{Gss}}(a_i) = \frac{1}{2} \left(\tanh \left(\frac{a_i}{a_0} \right) + 1 \right) \quad (88)$$

where a_0 is the additional parameter added to the Gaussian network.

4.1.3 Cauchy Machines

This system is an extension of the Gaussian machines, where instead of a Gaussian noise, a Cauchy noise is added to the system [114, 116]. The reason of replacing the Gaussian noise with the Cauchy noise is based on the observation that the Cauchy noise produces both global random flights and local random flights, whereas Gaussian noise produces only local random noise. Thus, the system has a better chance of convergence to the global minimum. The probability that a neuron will fire is given as

$$p_i = \frac{1}{2} + \frac{1}{\pi} \left(\arctan\left(\frac{\mathcal{I}_i}{T}\right) \right) \quad (89)$$

and

$$x_i = \Psi_{\text{Cau}}(\mathcal{I}_i) = \begin{cases} 1 & \text{with probability } p_i \\ 0 & \text{with probability } (1 - p_i) \end{cases} \quad (90)$$

where T is the temperature of the system which has the similar meaning as in the Gaussian or Boltzmann machines. In [63], comparison of lower bounds for Boltzmann and Cauchy machines is presented.

4.2 Chaotic Extensions

Let E_{Hop} represent the general H-T's energy function. Then chaotic extensions for the H-T's model can be obtained by adding an additional function to E_{Hop} [73, 111], which can be described as

$$E_{\text{CNN}} = E_{\text{Hop}} + H(\mathbf{x}, W, \theta) \quad (91)$$

Different definitions for the function $H()$ lead to different chaotic extensions. Some of the chaotic extension methods for the H-T's model will be presented in the following subsection.

4.2.1 Chen and Ahira's Model

Chen and Ahira [20, 21] proposed a chaotic addition to the energy function, which is given as

$$H_{\text{CA}} = \frac{\lambda(t)}{2} \sum_i x_i(x_i - 1) \quad (92)$$

where $\lambda(t)$ is the decay parameter. If this term is added to H-T's energy function, then the system's dynamics is defined by the following equation:

$$\frac{d\mathcal{I}_i}{dt} = -\frac{\partial E_{\text{CNN}}}{\partial x_i} \quad (93)$$

The update rule for the internal energy of this model is given as

$$\mathcal{I}_i(t+1) = k\mathcal{I}_i(t) + \alpha \left(\sum_{j=1, j \neq i}^n w_{ij} x_j(t) + \theta_i \right) - z(t)(x_i(t) - \theta_0) \quad (94)$$

$$z(t+1) = (1 - \beta)z(t) \quad (95)$$

where $\alpha = \nabla t$, $z(t)$ is the self-feedback connection weight $z(t) = \lambda(t)\nabla t$, β is the damping factor, $\theta_0 = 1/2$, and $k = 1 - \nabla t/\tau$.

Initially, z is very high to create strong self-coupling, thereby leading to the chaotic dynamics. Chaotic dynamics acts like an exploration stage in the simulated annealing, searching for a global minima. In the later stages, z is decayed so that the system starts to converge to a stable fixed point (similar to exploitation stage in the simulated annealing).

4.2.2 Wang and Smith's Model

Wang and Smith [124] proposed the following chaotic addition to the H-T's energy function, which is given as

$$H_{WS} = (\beta^T - 1)E_c \quad (96)$$

where E_c is the original H-T's energy function and β is the parameter which takes the values between $[0, 1]$. If this term is added to H-T's energy function, then the system is defined as

$$\frac{d\mathcal{I}_i}{dt} = -\frac{\partial E_{CNN}}{\partial x_i} \quad (97)$$

The update rule of the internal energy for this model is given as

$$\mathcal{I}_i(t+1) = \left(1 - \frac{\beta^T \nabla t(0)}{\tau}\right) \mathcal{I}_i(t) + \beta^T \nabla t(0) \left(\sum_{j=1, j \neq i}^n w_{ij} x_j(t) + \theta_i \right) \quad (98)$$

$$\beta^T \nabla t(0) = \nabla t(t) \quad (99)$$

Initially, ∇t is sufficiently large to trigger chaotic search. As $t \rightarrow \infty$, ∇t gradually decreases.

4.2.3 Chaotic Noise Model

Recently, Wang et al. [125] proposed the following chaotic addition to the H-T's energy function, which is given as

$$H_\eta = - \sum_i \eta_i(t) x_i(t) \quad (100)$$

where η_i is the noise term. In general, η_i is a normalized time series. Initially, the neuron is set with the chaotic time series. If this term is added to H-T's energy function, then the system is defined as

$$\frac{d\mathcal{I}_i}{dt} = -\frac{\partial E_{\text{CNN}}}{\partial x_i} \quad (101)$$

The update rule of the internal state for this model is given as

$$\mathcal{I}_i(t+1) = \left(1 - \frac{\nabla t}{\tau}\right)\mathcal{I}_i(t) + \nabla t \left(\sum_{j=1}^n w_{ij} x_j(t) + \theta_i + \eta_i(t)\right) \quad (102)$$

This model has a linear form of function $H()$, when compared to the quadratic forms presented in the earlier models. This may be the reason for its superior optimization performance among the chaotic extension models.

4.3 Convexification

One of the important properties of a convex optimization problem is that the local minimum is the global minimum. Thus, convex reformulation of the problem and finding a local minimum of the reformulation will give the global minimum. If the actual model is nonconvex, then convexification leads to different approximations (or relaxations), which in turn may be useful in studying various bounds (lower and/or upper bound) of a given COP. Both penalty-based ANNs and Lagrange-based ANNs can be convexified. Convexification of COPs using penalty function depends upon the type of penalty function. Thus, it will not be discussed in this subsection. Nevertheless, convexification of Lagrange-based ANNs will be discussed. From the [Theorem 15](#), it can be seen that the local convexity of Lagrange systems is hard to obtain in practice. However, an augmented Lagrange method can be used to convexify any Lagrange-based ANN. In this subsection, one of the convexification methods that can be used for Lagrange-based ANN is presented. Let us assume that problem [Eq.\(74\)](#) is nonconvex, then its augmented Lagrange can be written as

$$L_{\text{Aug}}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x}) + \frac{1}{2}c|h(\mathbf{x})|^2 \quad (103)$$

Using [Eq.\(103\)](#), different ANNs can be designed which are known as Augmented Lagrange Neural Networks (ALNN). The state equations can be obtained as

$$\frac{dx_i}{dt} = -\frac{\partial f}{\partial x_i} - \sum_{j=1}^m (\lambda_j + ch_j) \frac{\partial h_j}{\partial x_i} \quad \forall i = 1, \dots, n \quad (104)$$

$$\frac{d\lambda_j}{dt} = h_j, \quad \forall j = 1, \dots, m \quad (105)$$

By selecting a sufficiently large value of c , the local convexity for any Lagrange neural networks can be obtained under some restrictions (see [13]). However, due to analog circuit and implementing restrictions, the selection of c will become a critical issue [139].

4.4 Hybridization

ANNs have been coupled with metaheuristics like simulated annealing [103, 123–125], tabu search [43], and evolutionary algorithm [102, 126] for solving COPs. The main purpose of these hybridizations is to find the global optimal solution, by adding flexibility to the classical H-T's model. The usual method of coupling H-T's model with a metaheuristic is to incorporate the constraint handling technique of the ANNs in the global search technique of the metaheuristic.

Moreover, there has been research to hybridize H-T's model with SONNs (see [108]). One of the successful implementations is illustrated by Smith et al. [109]. The experimental result from the illustration indicates that a careful and appropriate hybridization of H-T's model with SONNs will result in a better solution methodology, which can be competitive to the state-of-the art optimization solvers.

5 General Optimization Problems

In this section, a discussion on applying ANNs to some of general optimization problems will be presented. For the sake of simplicity and convenience, following general optimization problem will be considered:

- Linear programming problems
- Convex programming problems
- Quadratic programming problems
- Nonlinear programming problems
- Complementarity problems
- Mixed integer programming problem

The objective of selecting mixed integer programming problem is to highlight the reader, the method of transformation between discrete and continuous variables, when applying ANNs.

5.1 Linear Programming

Linear programming (LP) is the most widely used optimization technique for solving numerous optimization problems. Dantzig [30], the father of linear programming, proposed the famous and widely used simplex method to solve LP problems. However, simplex method has an exponential complexity. A polynomial time algorithm (ellipsoid method) for solving LP problems was presented by Khachiyan [68]. Although the ellipsoid method is polynomial time algorithm, in practice simplex

outperforms ellipsoid method. Later, in 1984, Karmarkar [66] proposed a more efficient algorithm than ellipsoid method, and it outperforms simplex method in few complicated problems like scheduling, routing, and planning. Based on the solution methodology, the simplex method is categorized as exterior point method, whereas Karmarkar's algorithm is categorized as interior point method. Further studies [122] on the classification of these two fundamentally different ideas lead to a conclusion that LP problems are finite in nature. Moreover, simplex method is a finite algorithm suitable to solve LP problems. However, it was also noted that sometimes, infinite algorithms like interior point method may outperform exterior point algorithms. Thus, there has been always a curiosity among researchers to investigate the infinite algorithms. ANNs approach in solving COPs can be seen as an infinite approach.

From the demonstration of Hopfield and Tank [58], ANNs were seen as an alternative solution approaches to solve the COPs. Many studies have been conducted to solve LP problems using ANNs [25–28, 35, 67, 100, 136]. Most of these models were extensions of H-T's continuous model. As noted in Sect. 2.7, these models will have an energy function-based model for minimization. Before presenting the models, consider the following notations for LP problem in canonical LP_c and standard LP_s form:

LP_c

minimize :

$$c^T x$$

subject to :

$$g_i(x) \geq 0 \quad \forall i = 1, \dots, m$$

LP_s

minimize :

$$\tilde{c}^T x$$

subject to :

$$\tilde{g}_i(x) = 0 \quad \forall i = 1, \dots, m$$

Following are the prominent models from the literature.

5.1.1 Kennedy and Chua's Model

Kennedy and Chua's [67] model has neurons similar to the H-T's continuous model. The external states of this model are represented by x . The equations of updates for states and energy function are given as

$$\begin{aligned} \nabla x &= C^{-1}(-c - \nabla g(x) \phi(g(x))) \\ E(x(t)) &= c^T x + \sum_{j=1}^m \int_0^{g_j(x)} \phi_j(s) ds \end{aligned}$$

where C is a matrix of capacitance and is usually taken as I for most of the cases. Kennedy and Chua showed that Tank and Hopfield's model is a special case of the proposed model. Furthermore, they proved that their model will converge to a steady state, under the assumption that E is bounded from below. Selection of $\phi()$ is similar to that of a penalty function. In [80], it was shown that $\phi()$ is indeed

an implicit form of the penalty function. The most widely used representation of $\phi()$ is $s \cdot g_i^-(\mathbf{x})$, where $s > 0$ is the penalty weight. Moreover, $g_i^-(\mathbf{x})$ is same as $-\min\{g_i(\mathbf{x}), 0\}$. Based on the idea of applying penalty function, there were many extension of neural networks for LP. Some of the prominent extensions can be seen in [100, 138]. These prominent extensions will be presented in the following part of this section.

5.1.2 Rodriquez-Vazquez et al. Model

Rodriquez-Vazquez et al. [100] proposed another model for LP. This model is described as

$$\nabla \mathbf{x} = -u(\mathbf{x})\mathbf{c} - s(1 - u(\mathbf{x}))\nabla \mathbf{g}(\mathbf{x})\mathbf{v}(\mathbf{x})$$

where $\mathbf{v}(\mathbf{x}) = [v_1, \dots, v_m]^T$,

$$u(\mathbf{x}) = \begin{cases} 1 & \text{if } g_j(\mathbf{x}) \geq 0 \quad \forall j = 1, \dots, m, \\ 0 & \text{otherwise} \end{cases}$$

$$v_j(\mathbf{x}) = \begin{cases} 1 & \text{if } g_j(\mathbf{x}) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

and $s > 0$. This model can be viewed as a dynamic system, which can start from any arbitrary point. Initially, the model will move from an infeasible point to a feasible point. Once a feasible point is obtained, the model will try to move to an optimal point. In [45], the convergence analysis of Rodriquez-Vazquez et al. [100] model was performed. It was shown that if the problem is convex, network will converge. However, Zak et al. [138] proposed that discrete time implementation of Rodriquez-Vazquez et al. model with an adaptive step size may have problem in reaching feasible region. In the following paragraph, the model proposed by Zak et al. will be presented.

5.1.3 Zak et al. Model

This model differs from the earlier ANNs for the LP, as Zak et al. used the exact penalty function (non-differentiable) to formulate the energy function. This selection of penalty function eliminates the selection of very high penalty weight. Apart from that, all the earlier models were proposed for the canonical representation of LP. However, this model is applicable for both standard and canonical representation of LP. Although, in the theory, there is no difference in solving the problem in any representation. But in the case of neural networks, changing from one representation to other will increase in the number of neurons (due to slack or artificial variables) and may increase in the complexity of the ANN model. The proposed model can be described as

$$\nabla \mathbf{x} = -\nabla \mathbf{E}_{p,\rho,\gamma}(\mathbf{x})$$

$$\nabla \mathbf{E}_{p,\rho,\gamma}(\mathbf{x}) = \tilde{\mathbf{c}}^T \mathbf{x} + \rho \nabla \|\tilde{\mathbf{g}}(\mathbf{x})\|_p + \gamma \nabla \left(\sum_{i=1}^m x_i^- \right)$$

where $x_i^- = -\min\{x_i, 0\}$, $\rho > 0$, $\gamma > 0$, $1 \leq p \leq \infty$. Although the model uses exact penalty function, this penalty function is also non-differentiable and hence has few drawbacks. In [137], convergence analysis of Zak et al.'s model is performed. It was shown that the system trajectory will converge to the solution of LP under some positivity restriction on ρ , γ , p . In fact, it is shown that equilibrium points of the ANN are solutions to the corresponding LP problem [137].

These are some of the prominent models of ANNs applied to solve LP problems. Similar to the duality theory in LP, there were attempts to solve LP using dual ANNs. Interested reader may refer to [29, 81].

5.2 Convex Programming

Consider the following convex problem:

minimize :

$$f(\mathbf{x})$$

subject to :

$$g_i(\mathbf{x}) = 0 \quad \forall i = 1 \dots m$$

$$\mathbf{x} \in \mathbf{X} \tag{106}$$

where $f(\mathbf{x}), g_i(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ are any convex functions of \mathbf{x} . In order to apply ANNs to solve this problem, the problem is reformulated into unconstrained optimization problem using penalty function. Let $\Phi()$ be the penalty function which has the following properties:

$$\Phi(\mathbf{x}) \begin{cases} = 0 & \text{if } \mathbf{x} \text{ is feasible} \\ > 0 & \text{otherwise} \end{cases} \tag{107}$$

Then, consider the following transformed convex problem:

minimize :

$$\mathbf{E}(\mathbf{x}, s)$$

subject to :

$$\mathbf{x} \in \mathbf{X}$$

where $E(\mathbf{x}, s) = f(\mathbf{x}) + s \sum_{i=1}^m \Phi(g_i(\mathbf{x}))$.

The update equation is

$$\nabla \mathbf{x} = -\eta \nabla \mathbf{E}(\mathbf{x}, s) \quad (108)$$

$$\nabla \mathbf{E}(\mathbf{x}, s) = \nabla f(\mathbf{x}) + s \sum_{i=1}^m \phi(g_i(\mathbf{x})) \nabla g_i(\mathbf{x})$$

The stability and convergence analysis of this model has been performed in [22]. It has been shown that dynamics given by Eq. (108) leads to an equilibrium point.

5.3 Quadratic Programming

Similar to the linear programming, quadratic programming problems can be modeled using ANNs. Consider the following quadratic programming problem:

minimize :

$$\mathbf{x}^T H \mathbf{x} + c^T \mathbf{x}$$

subject to :

$$A \mathbf{x} = \mathbf{b}$$

$$\mathbf{x} \in \mathbf{X} \quad (109)$$

where $\mathbf{x} \in \mathbb{R}^n$, $A, H \in \mathbb{R}^{n \times n}$. This program can be converted to an unconstrained problem using a penalty function method or a Lagrange method. A Lagrange method will be presented in the following part of this section. The Lagrange of problem Eq. (109) is written as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{x}^T H \mathbf{x} + c^T \mathbf{x} + \boldsymbol{\lambda}^T (A \mathbf{x} - \mathbf{b}) \quad (110)$$

The equations of motion that describe this model are

$$\nabla_t \mathbf{x} = -\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) \quad (111)$$

$$\nabla_t \boldsymbol{\lambda} = -\nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda}) \quad (112)$$

where

$$\nabla_t \mathbf{x} = \left(\frac{dx_1}{dt}, \frac{dx_2}{dt}, \dots, \frac{dx_n}{dt} \right)^T, \quad x \in \mathbb{R}^n$$

$$\nabla_t \boldsymbol{\lambda} = \left(\frac{d\lambda_1}{dt}, \frac{d\lambda_2}{dt}, \dots, \frac{d\lambda_m}{dt} \right)^T, \quad \lambda \in \mathbb{R}^m$$

$$\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_1}, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_2}, \dots, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_n} \right)^T$$

and

$$\nabla_{\lambda} L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_1}, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_2}, \dots, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_m} \right)^T$$

The above model is iterative Lagrangian neural network. The convergence analysis of this model is presented in Sect. 3 and in [139]. Apart from that, quadratic programming problems with ANNs have been exhaustively studied in [39, 80, 134, 135].

5.4 Nonlinear Programming

Similar to the quadratic and convex programming, general nonlinear programming problems can be solved using ANNs. One of the ways is to use a penalty method approach [67]. Another approach is based on the Lagrange method, presented by Zhang and Constantinides [139]. The main differences with penalty function method and the advantages of Lagrange methods are discussed in Sect. 3 of this chapter. In the following part of this subsection, the method of writing a Lagrange for any generalized nonlinear problem will be presented.

minimize :

$$f(\mathbf{x})$$

subject to :

$$h_i(\mathbf{x}) = 0 \quad \forall i = 1 \dots m$$

$$\mathbf{x} \in \mathbb{R}^n \tag{113}$$

where $f(\mathbf{x}), h_i(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ are any continuous and twice differentiable functions of \mathbf{x} . The Lagrange of the above COP is written as

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T h(\mathbf{x}) \tag{114}$$

The equations of motion that describes this model are

$$\nabla_t \mathbf{x} = -\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) \tag{115}$$

$$\nabla_t \boldsymbol{\lambda} = -\nabla_{\lambda} L(\mathbf{x}, \boldsymbol{\lambda}) \tag{116}$$

where

$$\nabla_t \mathbf{x} = \left(\frac{dx_1}{dt}, \frac{dx_2}{dt}, \dots, \frac{dx_n}{dt} \right)^T, \mathbf{x} \in \mathbb{R}^n$$

$$\nabla_t \boldsymbol{\lambda} = \left(\frac{d\lambda_1}{dt}, \frac{d\lambda_2}{dt}, \dots, \frac{d\lambda_m}{dt} \right)^T, \lambda \in \mathbb{R}^m$$

$$\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_1}, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_2}, \dots, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial x_n} \right)^T$$

and

$$\nabla_{\boldsymbol{\lambda}} L(\mathbf{x}, \boldsymbol{\lambda}) = \left(\frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_1}, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_2}, \dots, \frac{\partial L(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_m} \right)^T$$

The above model is iterative, similar to that of H-T's continuous model. Both the types of neuron are decision variables. Lagrangian variables tend to take the solution to a feasible space, whereas the ordinary variables will take the solution to an optimal value. The mathematical properties and proofs related to convergence of this model have been discussed in [Sect. 3](#).

5.5 Complementarity Problem

5.5.1 Linear Complementarity Problem (LCP)

Given a matrix $M \in \mathbb{R}^{n \times N}$ and a vector $\mathbf{q} \in \mathbb{R}^n$, the LCP can be stated as:

find :

\mathbf{x}

subject to :

$$M\mathbf{x} + \mathbf{q} \geq 0$$

$$x^T(M\mathbf{x} + \mathbf{q}) = 0$$

$$\mathbf{x} \geq 0 \quad (117)$$

where $x \in \mathbb{R}^n$. This problem can be solved by ANNs, by converting it to a nonlinear problem by using Fischer function. Let $\phi(a, b)$ be the Fischer function, defined as $\phi(a, b) = \sqrt{a^2 + b^2} - a - b$. The purpose of using this function is that it has an important property, which can be stated as

$$\phi(a, b) = 0 \Leftrightarrow a \geq 0, b \geq 0 \text{ and } ab = 0 \quad (118)$$

Thus, this property is exploited while solving LCP using ANN. The above property mimics the LCP, which can be represented as

$$\phi_i(x_i, F_i(\mathbf{x})) = 0 \quad \forall i \quad (119)$$

where $F_i(\mathbf{x}) = x_i(M\mathbf{x} + \mathbf{q})_i$. Let $\Phi(\mathbf{x})$ represent a vector, where the i th element is $\phi_i(x_i, F_i(\mathbf{x}))$. Then, solving LCP is same as solving the following problem:

$$\begin{aligned} & \text{minimize :} \\ & F(\mathbf{x}) \end{aligned} \tag{120}$$

where

$$F(\mathbf{x}) = \frac{1}{2} \|\Phi(\mathbf{x})\|^2 \tag{121}$$

Now, the above problem can be solved by ANNs as a quadratic programming problem, using either a H-T's model or Lagrange model.

5.5.2 Nonlinear Complementarity Problem (NCP)

Given a set of functions $F_i : \mathbb{R}^n \mapsto \mathbb{R}^n$, $\forall i = 1, \dots, n$, the NCP can be stated as

find :

\mathbf{x}

subject to :

$$F_i(\mathbf{x}) \geq 0$$

$$x_i F(\mathbf{x}) = 0$$

$$\mathbf{x} \geq 0$$

(122)

where $x \in \mathbb{R}^n$ and F_i is assumed to be continuously differentiable function. ANNs can be used to solve NCP by using the idea of Fischer function [75]. Similar to the LCP case, define a vector $\Phi(\mathbf{x})$, where each i th element is represented by

$$\phi_i(x_i, F_i(\mathbf{x})) = 0 \quad \forall i \tag{123}$$

Thus, solving NCP is same as solving the following problem:

$$\begin{aligned} & \text{minimize :} \\ & E(\mathbf{x}) \end{aligned} \tag{124}$$

where

$$E(\mathbf{x}) = \frac{1}{2} \|\Phi(\mathbf{x})\|^2 \tag{125}$$

Now, this above problem can be solved by ANNs using a Lagrange model.

5.6 Mixed Integer Programming Problems (MIPs)

The difficulty in solving any optimization problems is not due to the nonlinear (or discrete) representation of the problem. The criteria that separate

difficult and easy problems are the convexity of the problem. A convex problem in a linear or nonlinear representation is tractable, whereas a nonconvex problem in any representation is difficult to solve. By presenting the example of a general MIPs, we would like to highlight the thin line that separates a continuous optimization problem from a discrete optimization problem while applying ANNs.

Consider the following MIPs defined as

minimize :

$$f(\mathbf{x}, \mathbf{y})$$

subject to :

$$g_i(\mathbf{x}, \mathbf{y}) \leq 0 \quad \forall i = 1, \dots, m$$

$$h_j(\mathbf{x}, \mathbf{y}) = 0 \quad \forall j = 1, \dots, p$$

$$l_k \leq x_k \leq u_k \quad \forall k = 1, \dots, n$$

$$y_r \in \{0, 1\} \quad \forall r = 1, \dots, q \quad (126)$$

Similar to any previous approaches, this problem can be converted to unconstrained optimization problem using either a penalty function or a Lagrange function. For the sake of completeness, both approaches are illustrated in the following part of this subsection.

5.6.1 Penalty Function Approach

The modified (penalized) energy function for problem Eq. (126) can be written as

$$\begin{aligned} E_{\text{mip}}(\mathbf{x}, \mathbf{y}) &= C_1 f(\mathbf{x}, \mathbf{y}) + C_2 \sum_{i=1}^m \Phi[g_i(\mathbf{x}, \mathbf{y})] \\ &\quad + C_3 \sum_{j=1}^p h_j^2(\mathbf{x}, \mathbf{y}) + C_4 \sum_{r=1}^q y_r(1 - y_r) \end{aligned} \quad (127)$$

where C_1 , C_2 , C_3 , and C_4 are scaling constants, used as the penalty parameters. Φ is the penalty function. The dynamics of the network defined by Eq. (127) can be described as

$$\frac{d\mathcal{I}_k^x}{dt} = -\eta_x \frac{\partial E_{\text{mip}}}{\partial x_k} \quad \forall k = 1, \dots, n \quad (128)$$

$$\frac{d\mathcal{I}_r^y}{dt} = -\eta_y \frac{\partial E_{\text{mip}}}{\partial y_r} \quad \forall r = 1, \dots, q \quad (129)$$

$$x_k = \Psi_c(\mathcal{I}_k^x) \quad \forall k = 1, \dots, n \quad (130)$$

$$y_r = \Psi_d(\mathcal{I}_r^y) \quad \forall r = 1, \dots, q \quad (131)$$

where η_x and η_y are positive scaling coefficients for the dynamics of the system. In addition to that, Ψ_d can be replaced by the tuned Ψ_c that results in the discrete values of y_r .

5.6.2 Lagrange Function Approach

The Lagrange of problem Eq. (126) can be written as

$$\begin{aligned} L_{\text{mip}}(x, y, \mu, v^1, v^2) = & f(x, y) + \sum_{i=1}^m \mu_i g_i(x, y) \\ & + \sum_{j=1}^p v_j^1 h_j(x, y) + \sum_{r=1}^q v_r^2 y_r(1 - y_r) \end{aligned} \quad (132)$$

where $\mu_i \geq 0 \quad \forall i = 1 \dots m$ and $v_j^1, v_r^2 \in \mathbb{R} \quad \forall j = 1 \dots p$ and $\forall r = 1 \dots q$. Similar to penalty function approach, the system dynamics can be described as

$$\nabla_t \mathbf{x} = -\nabla_x L_{\text{mip}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}, \mathbf{v}^1, \mathbf{v}^2) \quad (133)$$

$$\nabla_t \mathbf{y} = -\nabla_y L_{\text{mip}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}, \mathbf{v}^1, \mathbf{v}^2) \quad (134)$$

$$\nabla_t v^1 = \nabla_{v^1} L_{\text{mip}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}, \mathbf{v}^1, \mathbf{v}^2) \quad (135)$$

$$\nabla_t v^2 = \nabla_{v^2} L_{\text{mip}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}, \mathbf{v}^1, \mathbf{v}^2) \quad (136)$$

$$\frac{d\mu_i}{dt} = \begin{cases} 0 & \text{for } \mu_i = 0 \text{ \& } \frac{\partial L_{\text{mip}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}, \mathbf{v}^1, \mathbf{v}^2)}{\partial \mu_i} < 0 \\ \frac{\partial L_{\text{mip}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\mu}, \mathbf{v}^1, \mathbf{v}^2)}{\partial \mu_i} & \text{otherwise} \end{cases} \quad \forall i \quad (137)$$

In [127], penalty-based approach is applied to solve factory allocation problem, and to solve the unit commitment problem. Direct use of inequality constraints in Lagrange programming ANNs have been introduced in [61, 79]. In Eq. (137), a simple method of update for inequality constrained is illustrated from [79]. A primitive way to deal with the inequality constraints is to transform them into equality constraints. Another novel approach for direct usage of inequality constraints is to use the square of the Lagrange multiplier for the inequality constraints [61].

From this section, it should be clear that ANNs are not limited for solving either a linear or a quadratic programming problem. With development of penalty- and Lagrange-based methods for the ANNs, they can be applied to any typical optimization problem. With the ease of transformation from continuous to discrete variable space, they can be appropriately designed for any COPs. In the following section, some of the applications of ANNs in solving discrete optimization problems will be surveyed.

6 Discrete Optimization Problems

Discrete optimization problems find their application in the problems related to transportation, scheduling, sorting, networking, etc. In this section, various methods of mapping the discrete optimization problems to the ANNs will be presented.

6.1 Graph Problems

Graph problems are one of the extensively studied and applied problems in operations research. The goal of this subsection is to discuss these problems and their mapping onto ANNs [96].

6.1.1 Graph Partitioning Problem (GPP)

Problem Statement: Given a graph $G = (V, E)$, where V represents the vertices, $|V| = n$ and E represents the edges, $|E| = m$. Let w_i be the weight on each vertex. Let e_{ij} be the weight on the link connecting i th and j th vertices. The problem is to partition the graph into two partitions of nearly equal weights, such that the cut-size is minimized. In other words, the problem is to bisect the graph, such that the sum of weights of the vertices assigned to the partitions is nearly equal, while the sum of weights on the links connecting the partitions is minimum.

Solution Methodology: A continuous H-T's model was used to solve this Problem [9]. Let x_i represent the external state of the neuron, which can take any values between 0 and 1. All the neurons that have the value of 0 are assigned to the first partition. The others neurons, which have the value of 1, were assigned to the second partition. The objective function for GPP is given as

$$E_{\text{GPP}} = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n e_{ij} (x_i + x_j - 2x_i x_j) + \sum_{i=1}^n \sum_{j=1}^n w_i w_j (1 - x_i - x_j + 2x_i x_j) \quad (138)$$

The first term of the objective function aims at minimizing the weighted sum of the edges which belong to the cut, whereas the second term of the objective function aims at balancing the weights in both the partitions. When the E_{GPP} is compared with E_c , the following values for the synaptic weights and the input bias are obtained:

$$W_{ij} = 2e_{ij} - 4w_i w_j \quad \text{and} \quad \theta_i = - \sum_{j=1}^n e_{ij} + 2w_i \sum_{j=1}^n w_j \quad (139)$$

Thus, the H-T's model is complete. The above system is solved to get the solution of GPP.

6.1.2 Graph K-Partitioning Problem (GKP)

Problem Statement: Given a graph $G = (V, E)$, where V represents the vertices, $|V| = n$ and E represents the edges, $|E| = m$. Let w_i be the weight on each vertex. Let e_{ij} be the weight on the link connecting i th and j th vertices. This problem is a generalized version of GPP, where the objective is to partition given graph with similar description as GPP into K partitions. In general, K is not the multiple of 2.

Solution Methodology: This problem can be formulated as an assignment problem, where each node is assigned to only one partition. Since there are K partitions and n nodes, there will be $N = nK$ number of variables. Since, in H-T's model, each variable represents a neuron, there will be altogether N neurons. Let x_{ij} be the external state of the neuron, which can take value between 0 and 1.

The objective function with penalty parameters is written as

$$\begin{aligned} E_{\text{GKP}} = & \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^K \sum_{k=1, k \neq j}^K x_{ij} x_{jk} + \frac{1}{2} \left(\sum_{i=1}^n \sum_{j=1}^K x_{ij} - n \right)^2 \\ & + \frac{1}{2} \sum_{i,j=1}^n \sum_{k,l=1}^K e_{ij} (x_{ik} + x_{jl} - 2x_{ik}x_{jl}) \\ & + \sum_{k=1}^K \sum_{i=1}^n \sum_{j=1}^n x_{ik} x_{jk} w_i w_j \end{aligned} \quad (140)$$

When the E_{GKP} is compared with E_d , the following values for the weights and input bias are obtained:

$$W_{ij,kl} = -\delta_{ij}(1 - \delta_{ij}) - 1 + 2e_{ij} - 2w_i w_j \quad \text{and} \quad \theta_i = +n - \sum_{j=1}^n e_{ij} \quad (141)$$

where $W_{ij,kl}$ is the synaptic weight between neuron ij ¹³ and neuron kl . Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 1](#) to get the solution of GKP. A higher order ANNs for solving GKP was presented in [40].

6.1.3 Vertex Cover Problem (VCP)

Problem Definition: Given a graph $G = (V, E)$, where V represents the vertices, $|V| = n$ and E represents the edges, $|E| = m$. Let A be the adjacency matrix, where $a_{ij} = 1$ if there is an edge between i th and j th vertices, otherwise $a_{ij} = 0$. A subset C of the vertices V of the graph G is called the *vertex cover* if all the edges

¹³neuron ij corresponds to variable x_{ij}

of G are adjacent to at least one vertex in the set C . The VCP is to find the *vertex cover*, such that the cardinality of set C is minimum.

Solution Methodology: Let

$$x_i = \begin{cases} 1 & \text{if vertex } i \in C \\ 0 & \text{otherwise} \end{cases} \quad (142)$$

Since there are $|V| = n$ vertices in G , there will be n neurons used to design the model. The modified objective function designed for H-T's model will be

$$E_{\text{VCP}} = C_1 \left(\sum_{i=1}^n \sum_{j=1}^n x_i x_j \right) - \frac{C_2}{2} \left(\sum_{i=1}^n \sum_{j=1}^n (1 - x_i)(1 - x_j) a_{ij} \right) \quad (143)$$

where C_1 and C_2 are penalty terms whose values decide the relative importance of the constraints and the objective function. The first term in E_{VCP} ensures the minimality of the cover. The second term (is the penalty term) will be zero if the nodes which are not in the *vertex cover* have no edges between them. By comparing the coefficients of E_{VCP} and E_d , the synaptic weights and the input bias are obtained as

$$W_{ij} = -2C_1 - C_2 a_{ij} \quad \text{and} \quad \theta_i = C_2 \sum_{j=1}^n a_{ij} \quad (144)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 1](#) to get the solution of GKP.

6.1.4 Maximum Independent Set Problem (MISP)

Problem Definition: Given a graph $G = (V, E)$, where V represents the vertices, $|V| = n$ and E represents the edges, $|E| = m$. Let A represent the adjacency matrix, where $a_{ij} = 1$ if there is an edge between i th and j th vertices, otherwise $a_{ij} = 0$. A subset I of the vertices V of the graph G is called the *independent set* if there is no edge between any pair of vertices which belong to I . The MISP is to find an *independent set* with the maximum cardinality.

Solution Methodology: Let

$$x_i = \begin{cases} 1 & \text{if vertex } i \in I \\ 0 & \text{otherwise} \end{cases} \quad (145)$$

Since there are $|V| = n$ vertices in G , there will be n neurons used to design the model. The modified objective function designed for H-T's model will be

$$E_{\text{MISP}} = -C_1 \left(\sum_{i=1}^n \sum_{j=1}^n x_i x_j \right) + \frac{C_2}{2} \left(\sum_{i=1}^n \sum_{j=1}^n x_i x_j a_{ij} \right) \quad (146)$$

where C_1 and C_2 are penalty terms whose values decide the relative importance of the constraints and the objective function. The first term in E_{MISP} minimizes cardinality of the set I . The second term will ensure that the set I forms an *independent set*. By comparing the coefficients of E_{MISP} and E_d , the synaptic weights and the input bias are obtained as

$$W_{ij} = 2C_1 - C_2 a_{ij} \quad \text{and} \quad \theta_i = 0 \quad (147)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 1](#) to get the solution of MISP.

6.1.5 Maximum Clique Set Problem (MCSP)

Problem Definition: Given a graph $G = (V, E)$, where V represents the vertices, $|V| = n$ and E represents the edges, $|E| = m$. Let A represent the adjacency matrix, where $a_{ij} = 1$ if there is an edge between i th and j th vertices, otherwise $a_{ij} = 0$. A subset of the vertices V of the graph G is called *clique* if every pair of vertices in the subset is connected by an edge. The MCSP is to find a *clique* with the maximum cardinality.

Solution Methodology: Let

$$x_i = \begin{cases} 1 & \text{if vertex } i \in \text{clique} \\ 0 & \text{otherwise} \end{cases} \quad (148)$$

Since there are $|V| = n$ vertices in G , there will be n neurons used to design the model. Let $a_{ij}^c = 1 - a_{ij}$, then with this modification, MCSP will be same as MISP. The modified objective function designed for H-T's model will be

$$E_{\text{MCSP}} = -C_1 \left(\sum_{i=1}^n \sum_{j=1}^n x_i x_j \right) + \frac{C_2}{2} \left(\sum_{i=1}^n \sum_{j=1}^n x_i x_j a_{ij}^c \right) \quad (149)$$

where C_1 and C_2 are penalty terms whose values decide the relative importance of the constraints and the objective function. The first term in E_{MCSP} minimizes cardinality of the clique set. The second term will ensure that the selected set forms a clique. By comparing the coefficients of E_{MCSP} and E_d , the synaptic weights and the input bias are obtained as

$$W_{ij} = 2C_1 - C_2 a_{ij}^c \quad \text{and} \quad \theta_i = 0 \quad (150)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 1](#) to get the solution of MCSP.

6.1.6 Graph Coloring Problem (GCP)

Problem Definition: Given a graph $G = (V, E)$, where V represents the vertices, $|V| = n$ and E represents the edges, $|E| = m$. Let A represent the adjacency matrix, where $a_{ij} = 1$ if there is an edge between i th and j th vertices, otherwise $a_{ij} = 0$. The GCP is to find the minimum number of subsets of V such that no two adjacent vertices (two vertices connected by an edge) belong to the same subset. In other words, find the minimum number of different colors needed to color the graph such that no two adjacent vertices are of the same color.

Solution Methodology: Let

$$x_{ij} = \begin{cases} 1 & \text{if vertex } i \text{ is colored in the color } j \\ 0 & \text{otherwise} \end{cases} \quad (151)$$

Since there are $|V| = n$ vertices in G , from the graph theory, the maximum number of different colors needed to color the graph will be one plus the maximum degree of any vertex. Let v represent the maximum number of different colors. Thus, there will be nv neurons used to design the model. The modified objective function designed for H-T's model will be

$$E_{\text{GCP}} = \frac{1}{2} \left(\sum_{k=1}^v \sum_{i=1}^n \sum_{j=1}^n x_{ik} x_{jk} a_{ij} \right) - \left(\sum_{k=1}^v \sum_{i=1}^n \sum_{j=1}^n x_{ik} x_{jk} \right) \quad (152)$$

By comparing the coefficients of E_{GCP} and E_d , the synaptic weights and the input bias are obtained as

$$W_{ij} = 1 - a_{ij}/2 \quad \text{and} \quad \theta_i = 0 \quad (153)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 1](#) to get the solution of GCP.

6.1.7 Graph Matching Problem (GMP)

Problem Definition: Given a graph $G = (V, E)$, where V represents the vertices, $|V| = n$ and E represents the edges, $|E| = m$. Let B be the incidence matrix, where $b_{ij} = 1$ if j th edge is incident on i th vertex, otherwise $b_{ij} = 0$. A subset M of the edges E of the graph G is called *matching* if there are no two edges in M that share a common node. The GMP is to find a *matching* with the maximum cardinality.

Solution Methodology: Let

$$x_i = \begin{cases} 1 & \text{if edge } i \in M \\ 0 & \text{otherwise} \end{cases} \quad (154)$$

Since there are $|E| = m$ vertices in G , there will be m neurons used to design the model. The modified objective function designed for H-T's model will be

$$E_{\text{GMP}} = - \left(\sum_{i=1}^m x_i \right) + \left(\sum_{i=1}^m \sum_{j=1}^m x_i x_j \sum_{k=1}^n b_{ki} b_{kj} \right) \quad (155)$$

By comparing the coefficients of E_{GMP} and E_d , the synaptic weights and the input bias are obtained as

$$W_{ij} = -2 \sum_{k=1}^n b_{ki} b_{kj} \quad \text{and} \quad \theta_i = 1 \quad (156)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 1](#) to get the solution of GMP.

6.2 Shortest Path Problems

Problem Description: Given a network $G = (V, E)$, where V represents the vertices or cities, $|V| = n$ and E represents the edges or paths, $|E| = m$. Let the distance between the cities be represented by a distance matrix D , where d_{ij} represents the distance between the i th city and the j th city. The problem is to find the shortest path between two given pair of cities. Let s be the origin and e be the destination.

Solution Methodology: Let

$$x_{ij} = \begin{cases} 1 & \text{if edge } (i, j) \text{ is in the path} \\ 0 & \text{otherwise} \end{cases} \quad (157)$$

Since there are n cities, there will be n neurons used to design the model. The modified objective function designed for H-T's model will be

$$E_{\text{SPP}}(t) = C_1 \left(\sum_{i=1}^n \sum_{j \neq i} d_{ij} \exp^{-t/\tau} x_{ij} \right) + \frac{C_2}{2} \sum_{i=1}^n \left(\sum_{k \neq i} x_{ik} - \sum_{l \neq i} x_{li} - \delta_{is} + \delta_{ie} \right) \quad (158)$$

By comparing the coefficients of E_{SPP} and E_c , the synaptic weights and input bias are obtained as

$$W_{ij} = 2 * C_2 \quad \text{and} \quad \theta_i = -2 * C_2 \quad (159)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 2](#) to get the solution of SPP.

6.3 Number Partitioning Problems (NPP)

Problem Definition: Given a set of n numbers $a_1, a_2, \dots, a_n \in S$, partition the set S into two sets S_1 and S_2 such that the following cost function is minimized:

$$C(S_1, S_2) = \left| \sum_{i \in S_1} a_i - \sum_{i \in S_2} a_i \right| \quad (160)$$

Solution Methodology: Let

$$x_i = \begin{cases} 1 & \text{if edge } i \in S_2 \\ 0 & \text{if edge } i \in S_1 \end{cases} \quad (161)$$

Since there are n numbers, there will be n neurons used to design the model. The modified objective function designed for H-T's model will be

$$E_{\text{NPP}} = \left(\sum_{i=1}^n \sum_{j=1}^n (1 + 2x_i x_j - x_i - x_j) a_i a_j \right) \quad (162)$$

The term $(1 + 2x_i x_j - x_i - x_j)$ ensures that only numbers which belong to the same partition will contribute to the sum. By comparing the coefficients of E_{GMP} and E_d , the synaptic weights and input bias are obtained as

$$W_{ij} = -4a_i a_j \quad \text{and} \quad \theta_i = \sum_{i=1}^n a_i \quad (163)$$

Thus, with the above mapping, the Hopfield model is complete. The above system is solved by using [Algorithm 1](#) to get the solution of GMP.

6.4 Assignment Problems

Problem Definition: Given n entities, n cells, and the cost c_{ij} of assigning i th entity to j th cell. The linear assignment problem (LAP) is to assign each entity to one cell, such that the assignment cost is minimized.

Solution Methodology: Since there are n entities and n cells, there will be altogether n^2 neurons used for designing ANN. Let x_{ij} represent the external state of H-T's model. Let \mathcal{I}_{ij} represent the internal state of H-T's neuron, such that

$$x_{ij} = \begin{cases} 1 & \text{if entity } i \text{ is assigned to cell } j \\ 0 & \text{otherwise} \end{cases} \quad (164)$$

The dynamics of this system is given as

$$\frac{d\mathcal{I}_{ij}(t)}{dt} = -C_1 \sum_{k=1}^n v_{kj}(t) - C_1 \sum_{k=1}^n v_{ik}(t) + 2C_1 - C_2 c_{ij} \exp(-t/\tau) \quad (165)$$

$$x_{ij}(t) = f_{ij}(\mathcal{I}_{ij}(t)) \quad (166)$$

From the above equations, the weights and input bias of H-T's model are defined as

$$w_{ij} = \delta_{ij} 2C_1 + (1 - \delta_{ij})C_1 \quad \text{and} \quad \theta_{ij} = 2C_1 \quad (167)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 2](#) to get the solution of LAP.

6.5 Sorting Problems

Problem Definition: Given n real numbers a_1, a_2, \dots, a_n , the problem is to order them in ascending, descending, or biotonic order (where both the ends have higher numbers).

This problem can be formulated as an assignment problem, that is, given n numbers that are to be assigned to n cells. The cost c_{ij} of assigning i th number to j th cell is given by $c_{ij} = a_i c_j$, where c_j is an arbitrary sequence of real numbers, which are selected based on the sense of sorting (ascending, descending, or biotonic).

Solution Methodology: Since there are n numbers and n cells, there will be altogether n^2 neurons used for designing ANN. Let x_{ij} represent the external state of H-T's continuous model. Let \mathcal{I}_{ij} represent the internal state of H-T's model such that

$$x_{ij} = \begin{cases} 1 & \text{if entity } i \text{ is assigned to cell } j \\ 0 & \text{otherwise} \end{cases} \quad (168)$$

The energy function and dynamics of this system are given as

$$\begin{aligned} E_{SP}(x(t), t) &= C_1 \left(\sum_{i=1}^n \sum_{j=1}^n a_i c_j \exp(-t/\tau) x_{ij} \right) \\ &\quad + \sum_{i=1}^n \frac{C_2^i}{2} \left(\sum_{j=1}^n x_{ij} - 1 \right)^2 \end{aligned}$$

$$+ \sum_{j=1}^n \frac{C_2^j}{2} \left(\sum_{i=1}^n x_{ij} - 1 \right)^2 \quad (169)$$

$$\begin{aligned} \frac{d\mathcal{I}_{ij}(t)}{dt} = & -C_2^j \sum_{k=1}^n v_{kj}(t) - C_2^i \sum_{k=1}^n v_{ik}(t) \\ & + C_2^j + C_2^i - C_1 a_i c_j \exp(-t/\tau) \end{aligned} \quad (170)$$

$$x_{ij}(t) = f_{ij}(\mathcal{I}_{ij}(t)) \quad (171)$$

From the above equations, the weights of H-T's model can be obtained as

$$w_{ij} = -\delta_{ij}(C_2^j + C_2^i) - (1 - \delta_{ij})C_2^i \quad \text{and} \quad \theta_{ij} = C_2^j + C_2^i \quad (172)$$

Thus, with the above mapping, the H-T's model is complete. The above system is solved by using [Algorithm 2](#) to get the solution of SPP.

6.6 Traveling Salesman Problems (TSP)

Problem Definition: Given n cities with the Euclidean distance between them, d_{ij} . A path that starts from a given city, passing through all the cities (without visiting a city twice), and returning to the first city is called a *tour*. The TSP problem can be defined as finding the shortest tour for the given cities.

Solution Methodology: In [Sect. 2](#), a detailed mapping of TSP onto H-T's model was shown. However, in this subsection, one of the improved TSP mappings [[119](#)] will be presented. This is the modification of H-T's approach, which can be described as

$$E_{\text{tsp}}^{\text{mod}} = E_{\text{tsp}} + E_{\text{cns}} + E_{\text{drv}} \quad (173)$$

where E_{tsp} is same as defined in [Eq. \(33\)](#),

$$E_{\text{cns}} = \frac{C1}{2} \sum_i \left(\sum_j x_{ij} - 1 \right)^2 + \frac{C2}{2} \sum_j \left(\sum_i x_{ij} - 1 \right)^2 \quad (174)$$

$$E_{\text{drv}} = \frac{C3}{2} \sum_i \sum_j x_{ij} (1 - x_{ij}) + \frac{C4}{2} \left(\frac{N^2}{4} - \sum_j \sum_i \left(x_{ij} - \frac{1}{2} \right)^2 \right) \quad (175)$$

and $C1$, $C2$, $C3$, and $C4$ are new penalty parameters that are used for scaling the respective penalty terms. E_{cns} represents a better way of mapping the TSP constraints, whereas E_{drv} represents the penalty that deflects the external state of the neurons to take the value of 0 or 1.

There are numerous COPs in the literature, and presenting ANNs model for each of them will be a cumbersome task. However, a similar approach (like the above models) can be followed for any discrete optimization problem. Moreover, the above mappings were based on H-T's model with penalty approach. As seen in Sect. 3, the capability of ANNs to solve COPs can be extended by using the Lagrange approach. Thus, any discrete optimization problem (linear or nonlinear) can be modeled using the techniques given in Sect. 5. In the following part of this section, a survey of some of the recent models in ANNs is cited.

A recent survey that presents the usage of ANNs in mathematical programming problems is conducted in [128]. Apart from the theoretical COPs, there has been many practical problems that have been mapped onto ANNs. For example, some of the recent applications of ANNs within the past 2 years are

- Resource allocation in wireless communication [17, 78]
- Scheduling in wireless sensors [105]
- Scheduling in packet radio networks [113]
- Scheduling via H-T's model [31]
- Scheduling in water pumps [49]
- Scheduling to minimize earliness and tardiness [8]
- Small-world nature of H-T's model [141]
- Constrained least absolute deviation problem [60]
- Hybrid approach with genetic algorithms [5]
- Hybrid approach with particle swarm optimization [33]
- Hybrid approach with ant colony optimization [129]
- Parameter setting for the GCP [118]
- Placement of electronic circuit problem [36]
- Edge detection in wood logs [95]
- Economic load dispatching problem [48]
- Hybrid routing algorithm [11]
- Data envelopment analysis [59]
- Water system management problem [120]
- Graphs isomorphism discernment problem [140]
- Nonlinear vector encoding problem [41]

7 Criticism

After the eye-catching implementation of applying ANNs in solving TSP, H-T's model caught the attention of many researchers [92]. However, this method had two drawbacks at that time. It has a lot of parameters to select, and it performs a gradient descent method while solving COPs. In [132], it was illustrated that H-T's model does not perform well. In fact, the results in [132] raised serious questions about the validity of this method. Thus, it drifted many researchers away from the use of ANNs in solving COPs. However, some of the researchers were inclined to improve the method proposed by Hopfield and Tank. One of the direction of improvements was to develop a method of optimally selecting the parameters in Hopfield and

Tank's energy function [37, 52, 65, 74, 117]. Recently, in [119], a systematic way of selecting optimal parameters for various TSP is presented. Apart from that, an enhanced method of mapping TSP onto an ANN is illustrated, and convergence and stability analysis is presented. From the results shown in [119], it can be concluded that by proper construction of energy function and by proper selection of parameters (penalty parameters), there can be a significant difference in the performance of ANNs while solving COPs.

Another direction of improvement in Hopfield and Tank's ANN was to free the algorithm from gradient descend method, that is, to avoid falling into the local minima. One of the methods, that was adopted by many researchers, was to modify Hopfield and Tank's objective function [4, 16]. However, some of the successful methods were to use hybrid techniques like convexification, improve choice of penalty function, and use hill-climbing techniques [123]. In [109], a novel approach that combines self-organizing neural networks and H-T's model is presented. Convergence of such networks is analyzed. The algorithm was tested on car-sequencing problem. From the results, it was concluded that hybrid method of ANN is better than MINOS¹⁴ solver. The results from this study showed that proper application of ANNs to solve COPs will prove better than the conventional solution methods of COPs.

Moreover, there were many stochastic and metaheuristic methods that were incorporated in H-T's model. Some of the studies showed that global convergence can be obtained with ANNs if they are properly coupled with metaheuristic methods like simulated annealing [91]. Apart from that, sophisticated penalty methods and Lagrange methods made the use of ANNs more general (i.e., not only applied to linear and quadratic but can be applied to any nonlinear optimization problems). Among the use of penalty methods, in [6, 42], subspace and hyperplane approaches were used. These were the pioneer approaches that modified the use of penalty function, by incorporating the feasibility issues of the system into a single penalty parameter. In [43], the capability of ANNs were demonstrated based on polyhedral combinatorics. It was one of the novel methods in ANNs that proposed the use of memory (like tabu search) to overcome the entrapment of algorithm at the polyhedral vertices. However, not many studies have actually tried to compare performance of ANNs with other conventional or heuristic methods for solving COPs. Thus, it is very hard to conclude if ANNs perform better or worse than other methods. Apart from that, most of the implementations were simulations of ANNs on the digital computers. Thus, the computing limitations of digital computer may hinder the performance of ANNs. Nevertheless, it should be noted that ANNs can overcome the limitations of digital computers since they can be implemented on the analog circuits.

¹⁴A standard nonlinear programming problem solver

8 Conclusion

In this chapter, a simple and brief introduction of ANNs is presented, which is followed by addressing the usage of ANNs in solving COPs. Our focus throughout this chapter was not only to highlight the application of ANNs in solving COPs but also to present the theoretical aspects of ANNs in solving COPs. At the beginning of this chapter, some of the classical ANN models that were used in solving COPs were illustrated. Along with these illustration, a methodology of implementing an LP to analog circuit is depicted. Moreover, a detail method of mapping TSP on one of the classical ANNs is presented. In addition to that, some of the stability and convergence analysis of ANNs is studied. Lastly, examples from literature on solving general optimization problems and discrete optimization problems using ANNs were presented.

It can be seen that unlike other metaheuristics, based on appropriate conditions, ANNs do converge to local and/or global optima. Simple methods that are used to define such optimality conditions were shown with proofs in Sect. 3 of this chapter. The aim in presenting these proofs was to highlight the importance of selecting proper energy function and the type of dynamics (i.e., consecutive, parallel, or synchronous). Apart from that, methods to extend the classical models (which can solve only linear or quadratic problems) by using penalty functions, Lagrange functions, or hybrid approaches were presented. Through various illustration, it was shown that a continuous model can be used to solve discrete problem very easily. It was underlined that the only change in solving a discrete problem and continuous problem is the adjustment in the transfer function and the adjustment in the energy function for the discrete variables.

Since the first implementation of ANN in 1980s until the present, there are handful analog implementations of ANNs in solving COPs. Due to the limitations of performance comparison of ANNs with other solution procedures, a concrete conclusion regarding the advantage or disadvantage of ANNs in solving COPs cannot be presented at this point. However, proper implementation of the analog circuit and systematic study in deployment of these circuits to solve real world problems will demonstrate the true potential of ANN.

Cross-References

- ▶ [Binary Unconstrained Quadratic Optimization Problem](#)
- ▶ [Graph Searching and Related Problems](#)
- ▶ [Quadratic Assignment Problems](#)

Recommended Reading

1. E. Aarts, J. Korst, *Simulated Annealing and Boltzmann Machines* (Wiley, Chichester, 1988)

2. E.H.L. Aarts, J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing* (Wiley, New York, 1989)
3. E.H.L. Aarts, P.H.P. Stehouwer, J. Wessels, P.J. Zwietering, Neural networks for combinatorial optimization. 1994
4. S. Abe, Global convergence and suppression of spurious states of the Hopfield neural networks. IEEE Trans Circuits Syst.-I Fundam. Theory Appl. **40**(4), 246–257 (1993)
5. A. Agarwal, S. Colak, J. Deane, NeuroGenetic approach for combinatorial optimization: an exploratory analysis. Ann. Oper. Res. **174**(1), 185–199 (2010)
6. S.V.B. Aiyer, M. Niranjan, F. Fallside, A theoretical investigation into the performance of the Hopfield model. IEEE Trans. Neural Netw. **1**(2), 204–215 (1990)
7. Y. Akiyama, A. Yamashita, M. Kajiura, H. Aiso, Combinatorial optimization with Gaussian machines, in *International Joint Conference on Neural Networks, 1989. IJCNN*, Washington, DC (IEEE, New York, 2002), pp. 533–540
8. D.E. Akyol, G.M. Bayhan, Multi-machine earliness and tardiness scheduling problem: an interconnected neural network approach. Int. J. Adv. Manuf. Technol. **37**(5), 576–588 (2008)
9. J.R. Anderson, Neural networks and NP-complete optimization problems; a performance study on the graph bisection problem. Complex Syst. **2**, 59–89 (1988)
10. J.A. Anderson, J. Davis, *An Introduction to Neural Networks* (MIT, Cambridge, 1995)
11. C.J.A. Bastos-Filho, W.H. Schuler, A.L.I. Oliveira, A fast and reliable routing algorithm based on Hopfield neural networks optimized by particle swarm optimization, in *IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence)* (IEEE, Piscataway, 2008), pp. 2777–2783
12. M.S. Bazaraa, H.D. Sherali, C.M. Shetty, *Nonlinear Programming: Theory and Algorithms* (Wiley, Hoboken, 2006)
13. D.P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods* (Athena Scientific, Belmont, 1996)
14. C.M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4 (Springer, New York, 2006)
15. C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput. Surv. (CSUR) **35**(3), 268–308 (2003)
16. R.D. Brandt, W. Yao, A.J. Laub, S.K. Mitra, Alternative networks for solving the traveling salesman problem and the list-matching problem, in *IEEE International Conference on Neural Networks*, San Diego (IEEE, Piscataway, 1988), pp. 333–340
17. D. Calabuig, J.F. Monserrat, D. Gómez-Barquero, N. Cardona, A delay-centric dynamic resource allocation algorithm for wireless communication systems based on HNN. IEEE Trans. Veh. Technol. **57**(6), 3653–3665 (2008)
18. G.A. Carpenter, Neural network models for pattern recognition and associative memory. Neural Netw. **2**(4), 243–257 (1989)
19. G.A. Carpenter, S. Grossberg, *Pattern Recognition by Self-organizing Neural Networks* (MIT, Cambridge, 1991)
20. L. Chen, K. Aihara, Chaotic simulated annealing by a neural network model with transient chaos. Neural Netw. **8**(6), 915–930 (1995)
21. L. Chen, K. Aihara, Global searching ability of chaotic neural networks. IEEE Trans. Circuits and Syst.-I Fundam. Theory Appl. **46**(8), 974–993 (1999)
22. Y.H. Chen, S.C. Fang, Solving convex programming problems with equality constraints by neural networks. Comput. Math. Appl. **36**(7), 41–68 (1998)
23. L.O. Chua, G.N. Lin, Non-linear optimization with constraints: a cook-book approach. Int. J. Circuit Theory Appl. **11**(2), 141–159 (1983)
24. L.O. Chua, G.N. Lin, Nonlinear programming without computation. IEEE Trans. Circuits Syst. **31**(2), 182–188 (1984)
25. A. Cichocki, R. Unbehauen, Neural networks for solving systems of linear equations and related problems. IEEE Trans. Circuits Syst. **39**(2), 124–138 (1992)
26. A. Cichocki, R. Unbehauen, Neural networks for solving systems of linear equations and related problems. IEEE Trans. Circuits Syst. 1 Fundam. Theory Appl. **39**(2), 124–138 (1992)

27. A. Cichocki, R. Unbehauen, K. Weinzierl, R. Hözel, A new neural network for solving linear programming problems. *Eur. J. Oper. Res.* **93**(2), 244–256 (1996)
28. J.C. Culoli, V. Protopopescu, C. Britton, N. Ericson, Neural network models for linear programming. Technical report, Oak Ridge National Lab., TN (1989)
29. T. Da-Gang, Duality and neural networks for solving linear programming. *Acta Autom. Sin.* **29**(2), 219–226 (2003)
30. G.B. Dantzig, M.N. Thapa, *Linear Programming: Theory and Extensions* (Springer, New York, 2003)
31. V. David, S. Rajasekaran, Retracted chapter: solving scheduling problems with competitive Hopfield neural networks, in *Pattern Recognition Using Neural and Functional Networks* (Springer, Berlin, 2009), pp. 115–122
32. J.B. Dennis, *Mathematical Programming and Electrical Networks*, (MIT, Cambridge, 1959)
33. Y.H. Duan, Y.L. Gao, PSO algorithm connected with neural network for solving a class of 0/1 optimization problems. *Jisuanji Yingyong/J. Comput. Appl.* **28**(6), 1559–1562 (2008)
34. R. Durbin, D. Willshaw, An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, **326**(6114), 689–691 (1987)
35. S. Effati, A.R. Nazemi, Neural network models and its application for solving linear and quadratic programming problems. *Appl. Math. Comput.* **172**(1), 305–331 (2006)
36. M. Ettouail, K. Elmoutaouakil, Y. Ghanou, The continuous Hopfield networks (CHN) for the placement of the electronic circuits problem. *WSEAS Trans. Comput.* **8**(12), 1865–1874 (2009)
37. G. Feng, C. Douligeris, Using Hopfield networks to solve traveling salesman problems based on stable state analysis technique, in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 2000. IJCNN 2000*, Como, vol. 6 (IEEE, New York, 2002), pp. 521–526
38. T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures. *J. Glob. Optim.* **6**(2), 109–133 (1995)
39. M. Forti, A. Tesi, New conditions for global stability of neural networks with application to linear and quadratic programming problems. *IEEE Trans. Circuits Syst. I Fundam. Theory Appl.* **42**(7), 354–366 (2002)
40. G.C. Fox, W. Furmanski, Load balancing loosely synchronous problems with a neural network, in *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications: Architecture, Software, Computer systems, and General Issues-Volume 1* (ACM, New York, 1988), pp. 241–278
41. V. Gardaševic, R.R. Muller, G.E. Øien, Hopfield neural networks for vector precoding, in *International Zurich Seminar on Communications*, Zurich, 2010, p. 66
42. A.H. Gee, Problem solving with optimization networks (1993)
43. A.H. Gee, R.W. Prager, Polyhedral combinatorics and neural networks. *Neural Comput.* **6**(1), 161–180 (1994)
44. M. Gendreau, J.Y. Potvin, Metaheuristics in combinatorial optimization. *Ann. Oper. Res.* **140**(1), 189–213 (2005)
45. M.P. Glazos, S. Hui, S.H. Zak, On solving constrained optimization problems with neural networks, in *IEEE International Conference on Neural Networks, 1994. IEEE World Congress on Computational Intelligence, 1994*, Orlando, vol. 7 (IEEE, New York, 2002), pp. 4547–4552
46. F. Glover, G.A. Kochenberger, *Handbook of Metaheuristics* (Springer, 2003)
47. R.E. Gomory, Outline of an algorithm for integer solutions to linear programs. *Bull. Am. Math. Soc.* **64**(5), 275–278 (1958)
48. D. Gupta, S.K. Jain, Solving combinatorial optimization problem of economic load dispatch using modified Hopfield neural network, in *Joint International Conference on Power System Technology and IEEE Power India Conference, 2008. POWERCON 2008* (IEEE, Piscataway, 2009), pp. 1–6
49. M. Hajji, A. Fares, F. Glover, O. Driss, Water pump scheduling system using scatter search, tabu search and neural networks the case of bouregreg water system in morocco. *ASCE* (2010)

50. A. Haken, M. Luby, Steepest descent can take exponential time for symmetric connection networks. *Complex Syst.* **2**(2), 191–196 (1988)
51. Q. Han, L.Z. Liao, H. Qi, L. Qi, Stability analysis of gradient-based neural networks for optimization problems. *J. Glob. Optim.* **19**(4), 363–381 (2001)
52. S.U. Hedge, J.L. Sweet, W.B. Levy, Determination of parameters in a Hopfield/Tank computational network, in *IEEE International Conference on Neural Networks, 1988*, San Diego (IEEE, San Diego, 2002), pp. 291–298
53. A. Hertz, M. Widmer, Guidelines for the use of meta-heuristics in combinatorial optimization. *Eur. J. Oper. Res.* **151**(2), 247–252 (2003)
54. G.E. Hinton, T.J. Sejnowski, Learning and relearning in Boltzmann machines, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, ed. by D.E. Rumelhart, J.L. McClelland MIT, Cambridge, 1986, pp. 283–317
55. G.E. Hinton, T.J. Sejnowski, D.H. Ackley, *Boltzmann Machines: Constraint Satisfaction Networks that Learn* (Carnegie-Mellon University, Department of Computer Science, Pittsburgh, 1984)
56. J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **79**(8), 2554 (1982)
57. J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA* **81**(10), 3088 (1984)
58. J.J. Hopfield, D.W. Tank, Neural computation of decisions in optimization problems. *Biol. Cybern.* **52**(3), 141–152 (1985)
59. S.C. Hu, Y.K. Chung, Y.S. Chen, Using Hopfield neural networks to solve DEA problems, in *IEEE Conference on Cybernetics and Intelligent Systems, 2008* (IEEE, Piscataway, 2008), pp. 606–611
60. X. Hu, C. Sun, B. Zhang, Design of recurrent neural networks for solving constrained least absolute deviation problems. *IEEE Trans. Neural Netw.* **21**(7), 1073–1086 (2010)
61. Y. Huang, Lagrange-type neural networks for nonlinear programming problems with inequality constraints, in *44th IEEE Conference on Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05*, Sevilla (IEEE, Sevilla, 2006), pp. 4129–4133
62. J.L. Huertas, A. Rueda, A. Rodriguez-Vazquez, L.O. Chua, Canonical non-linear programming circuits. *Int. J. Circuit Theory Appl.* **15**(1), 71–77 (1987)
63. H. Jeong, J.H. Park, Lower bounds of annealing schedule for Boltzmann and Cauchy machines, in *International Joint Conference on Neural Networks, 1989. IJCNN*, Washington, DC (IEEE, 2002), pp. 581–586
64. G. Joya, M.A. Atencia, F. Sandoval, Hopfield neural networks for optimization: study of the different dynamics. *Neurocomputing* **43**(1–4), 219–237 (2002)
65. B. Kamgar-Parsi, Dynamical stability and parameter selection in neural optimization, in *International Joint Conference on Neural Networks, 1992. IJCNN*, Baltimore, vol. 4 (IEEE, 2002), pp. 566–571
66. N. Karmarkar, A new polynomial-time algorithm for linear programming, in *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, Washington, DC (ACM, New York, 1984), p. 311
67. M.P. Kennedy, L.O. Chua, Neural networks for nonlinear programming. *IEEE Trans. Circuits Syst.* **35**(5), 554–562 (1988)
68. L.G. Khachiyan, Polynomial algorithms in linear programming. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* **20**, 51–68 (1980)
69. V. Klee, G.J. Minty, How good is the simplex algorithm (1970)
70. T. Kohonen, *Self-Organization and Associative Memories* (Springer, Heidelberg, 1982)
71. T. Kohonen, Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **43**(1), 59–69 (1982)
72. T. Kohonen, *Self-Organization and Associative Memory* (Springer, Berlin/New York, 1989); S. Ridella, S. Rovetta, R. Zunino, Plastic algorithm for adaptive vector quantization. *Neural Comput. Appl.* **1**, 152–159 (1998)

73. T. Kwok, K.A. Smith, A unified framework for chaotic neural-network approaches to combinatorial optimization. *IEEE Trans. Neural Netw.* **10**(4), 978–981 (2002)
74. W.K. Lai, G.G. Coghill Genetic breeding of control parameters for the Hopfield/Tank neural net, in *International Joint Conference on Neural Networks, 1992. IJCNN*, Baltimore, vol. 4 (IEEE, 2002), pp. 618–623
75. L.Z. Liao, H. Qi, L. Qi, Solving nonlinear complementarity problems with neural networks: a reformulation method approach*. *1. J. Comput. Appl. Math.* **131**(1–2), 343–359 (2001)
76. A.M. Liapounoff, *Probleme Général de la Stabilité du Mouvement* (Princeton University Press, Princeton, 1947)
77. W.E. Lillo, M.H. Loh, S. Hui, S.H. Zak, On solving constrained optimization problems with neural networks: a penalty method approach. *IEEE Trans. Neural Netw.* **4**(6), 931–940 (2002)
78. Y. Liu, M. Jiang, D. Yuan, Cross-layer resource allocation optimization by Hopfield neural networks in OFDMA-based wireless mesh networks, in 2009 Fifth International Conference on Natural Computation (IEEE, Los Alamitos, 2009), pp. 119–123
79. P.B. Luh, X. Zhao, Y. Wang, L.S. Thakur, Lagrangian relaxation neural networks for job shop scheduling. *IEEE Trans. Robot. Autom.* **16**(1), 78–88 (2002)
80. C.Y. Maa, M.A. Shanblatt, Linear and quadratic programming neural network analysis. *IEEE Trans. Neural Netw./A Publication of the IEEE Neural Networks Council* **3**(4), 580 (1992)
81. A. Malek, A. Yari, Primal-dual solution for the linear programming problems using neural networks. *Appl. Math. Comput.* **167**(1), 198–211 (2005)
82. W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biol.* **5**(4), 115–133 (1943)
83. I.I. Melamed, Neural networks and combinatorial optimization. *Autom. Remote Control* **55**(11), 1553 (1994) Translated from Avtomatika i Telemekhanika, No. 11, pp. 3–40, 1994.
84. A.N. Michel, J.A. Farrell, Associative memories via artificial neural networks. *IEEE Control Syst. Mag.* **10**(3), 6–17 (2002)
85. M.L. Minsky, S.A. Papert, *Perceptrons*, Expanded edn. (MIT, Cambridge, 1988)
86. K.S. Narendra, K. Parthasarathy, Gradient methods for the optimization of dynamical systems containing neural networks. *IEEE Trans. Neural Netw.* **2**(2), 252–262 (2002)
87. I.H. Osman, J.P. Kelly, Meta-heuristics: an overview, in *Meta-Heuristics: Theory and Applications* (Kluwer, Boston, 1996)
88. A.S. Pandya, R.B. Macy, *Pattern Recognition with Neural Networks in C++* (CRC, Boca Raton, 1996)
89. Y. Pao, *Adaptive Pattern Recognition and Neural Networks* (Addison-Wesley, Reading, 1989)
90. M. Papadrakakis, N.D. Lagaros, Reliability-based structural optimization using neural networks and Monte Carlo simulation. *Comput. Methods Appl. Mech. Eng.* **191**(32), 3491–3507 (2002)
91. M. Peng, N.K. Gupta, A.F. Armitage, An investigation into the improvement of local minima of the Hopfield network. *Neural Netw.* **9**(7), 1241–1253 (1996)
92. J.Y. Potvin, *The Traveling Salesman Problem: A Neural Network Perspective* (Centre for Research on Transportation, Montréal, 1992)
93. D. Psaltis, N. Farhat, Optical information processing based on an associative-memory model of neural nets with thresholding and feedback. *Opt. Lett.* **10**(2), 98–100 (1985)
94. J. Puchinger, G.R. Raidl, Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification, in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach*, ed. by J. Mira (Springer, Berlin, 2005), pp. 41–53
95. D. Qi, P. Zhang, X. Jin, X. Zhang, Study on wood image edge detection based on Hopfield neural network, in *IEEE International Conference on Information and Automation (ICIA), 2010* (IEEE, Piscataway, 2010), pp. 1942–1946
96. J. Ramanujam, P. Sadayappan, Mapping combinatorial optimization problems onto neural networks. *Inf. Sci.* **82**(3), 239–255 (1995)
97. J. Ramanujam, P. Sadayappan, Optimization by neural networks, in *IEEE International Conference on Neural Networks, 1988*, San Diego (IEEE, San Diego, 2002), pp. 325–332

98. M. Resendel, C. Ribeiro, GRASP with path-relinking: recent advances and applications, in *Metaheuristics: Progress as Real Problem Solvers*, ed. by T. Ibaraki, K. Nonobe, M. Yagiura (Springer, New York, 2005), pp. 29–63
99. M.G.C. Resende, K.G. Ramakrishnan, Z. Drezner, Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. *Oper. Res.* **43**(5), 781–791 (1995)
100. A. Rodrigues-vazquez, R. Dominguez-castro, A. Rueda, J.L. Huertas, E. Sanchez-sinencio, Nonlinear switched-capacitor Neural networks for optimization problems. *IEEE Trans. Circuits Syst.* **37**(3), 384–398 (1990)
101. F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **65**(6), 386–408 (1958)
102. S. Salcedo-Sanz, C. Bousño-Calzón, A.R. Figueiras-Vidal, A mixed neural-genetic algorithm for the broadcast scheduling problem. *IEEE Trans. Wirel. Commun.* **2**(2), 277–283 (2003)
103. S. Salcedo-Sanz, R. Santiago-Mozos, C. Bousono-Calzon, A hybrid Hopfield network-simulated annealing approach for frequency assignment in satellite communications systems. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* **34**(2), 1108–1116 (2004)
104. R. Sharda, Neural networks for the MS/OR analyst: an application bibliography. *Interfaces* **24**(2), 116–130 (1994)
105. Y.J. Shen, M.S. Wang, Broadcast scheduling in wireless sensor networks using fuzzy Hopfield neural network. *Expert Syst. Appl.* **34**(2), 900–907 (2008)
106. Y. Shrivastava, S. Dasgupta, S.M. Reddy, Guaranteed convergence in a class of Hopfield networks. *IEEE Trans. Neural Netw.* **3**(6), 951–961 (2002)
107. K.A. Smith, Neural networks for combinatorial optimization: a review of more than a decade of research. *INFORMS J. Comput.* **11**, 15–34 (1999)
108. K. Smith, Solving the generalised quadratic assignment problem using a self-organising process, in *IEEE International Conference on Neural Networks, 1995. Proceedings*, Perth, vol. 4 (IEEE, Piscataway, 2002), pp. 1876–1879
109. K. Smith, M. Palaniswami, M. Krishnamoorthy, A hybrid neural approach to combinatorial optimization. *Comput. Oper. Res.* **23**(6), 597–610 (1996)
110. K. Smith, M. Palaniswami, M. Krishnamoorthy, Neural techniques for combinatorial optimization with applications. *IEEE Trans. Neural Netw.* **9**(6), 1301–1318 (2002)
111. K.A. Smith, J.Y. Potvin, T. Kwok, Neural network models for combinatorial optimization: a survey of deterministic, stochastic and chaotic approaches. *Control Cybern.* **31**(2), 183–216 (2002)
112. T.E. Stern, *Theory of Nonlinear Networks and Systems* (Addison-Wesley, Reading, 1965)
113. M. Sun, L. Zhao, W. Cao, Y. Xu, X. Dai, X. Wang, Novel hysteretic noisy chaotic neural network for broadcast scheduling problems in packet radio networks. *IEEE Trans. Neural Netw.* **21**(9), 1422–1433 (2010)
114. H. Szu, R. Hartley, Fast simulated annealing*. *1. Phys. Lett. A* **122**(3–4), 157–162 (1987)
115. E.D. Taillard, L.M. Gambardella, M. Gendreau, J.Y. Potvin, Adaptive memory programming: A unified view of metaheuristics. *Eur. J. Oper. Res.* **135**(1), 1–16 (2001)
116. Y. Takefuji, H. Szu, Design of parallel distributed cauchy machines, in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, 1989. IJCNN*, Washington, DC, vol. 1 (IEEE, New York, 1989), pp. 529–532
117. P.M. Talaván, J. Yáñez, Parameter setting of the Hopfield network applied to TSP. *Neural Netw.* **15**(3), 363–373 (2002)
118. P.M. Talaván, J. Yáñez, The graph coloring problem: a neuronal network approach. *Eur. J. Oper. Res.* **191**(1), 100–111 (2008)
119. K.C. Tan, H. Tang, S.S. Ge, On parameter settings of Hopfield networks applied to traveling salesman problems. *IEEE Trans. Circuits Syst. I Regul. Pap.* **52**(5), 994–1002 (2005)
120. B.B. Tan, S.L. Shen, Z.H. Wu, L.P. Wang, Application of neural networks in project optimization of water system management in mining subsidence. *J. Anhui Univ. Sci. Technology (Nat. Sci.)* **29**(2), 13–16 (2009)

121. D.W. Tank, J.J. Hopfield, Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Trans. Circuits Syst.* **33**(5), 533–541 (1986)
122. L.N. Trefethen, *The Definition of Numerical Analysis* (Cornell University, Ithaca, 1992)
123. D.E. Van den Bout, T.K. Miller, Improving the performance of the Hopfield-Tank neural network through normalization and annealing. *Biol. Cybern.* **62**(2), 129–139 (1989)
124. L. Wang, K. Smith, On chaotic simulated annealing. *IEEE Trans. Neural Netw.* **9**(4), 716–718 (2002)
125. L. Wang, S. Li, F. Tian, X. Fu, A noisy chaotic neural network for solving combinatorial optimization problems: Stochastic chaotic simulated annealing. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **34**(5), 2119–2125 (2004)
126. Y. Watanabe, N. Mizuguchi, Y. Fujii, Solving optimization problems by using a Hopfield neural network and genetic algorithm combination. *Syst. Comput. Jpn* **29**(10), 68–74 (1998)
127. P.B. Watta, M.H. Hassoun, A coupled gradient network approach for static and temporal mixed-integer optimization. *IEEE Trans. Neural Netw.* **7**(3), 578–593 (2002)
128. U.P. Wen, K.M. Lan, H.S. Shih, A review of Hopfield neural networks for solving mathematical programming problems. *Eur. J. Oper. Res.* **198**(3), 675–687 (2009)
129. Y.L. Weng, C.Y. Lee, Z.J. Lee, Incorporating Hopfield neural networks into ant colony system for traveling salesman problem, in *New Advances in Intelligent Decision Technologies* (Springer, Berlin/Heidelberg), pp. 287–294 (2009)
130. D.J. Willshaw, C. Von Der Malsburg, A marker induction mechanism for the establishment of ordered neural mappings: its application to the retinotectal problem. *Philos. Trans. R. Soc. Lond. Ser. B Biol. Sci.* **287**(1021), 203–243 (1979)
131. G. Wilson, Quadratic programming analogs. *IEEE Trans. Circuits Syst.* **33**(9), 907–911 (1986)
132. G.V. Wilson, G.S. Pawley, On the stability of the travelling salesman problem algorithm of Hopfield and Tank. *Biol. Cybern.* **58**(1), 63–70 (1988)
133. S.J. Wright, *Primal-Dual Interior-Point Methods* (Society for Industrial Mathematics, Philadelphia, 1997)
134. X.Y. Wu, Y.S. Xia, J. Li, W.K. Chen, A high-performance neural network for solving linear and quadratic programming problems. *IEEE Trans. Neural Netw.* **7**(3), 643–651 (2002)
135. Y. Xia, A new neural network for solving linear and quadratic programming problems. *IEEE Trans. Neural Netw.* **7**(6), 1544–1548 (2002)
136. Y. Xia, J. Wang, A recurrent neural network for solving linear projection equations. *Neural Netw.* **13**(3), 337–350 (2000)
137. S.H. Zak, V. Upatising, W.E. Lillo, S. Hui, A dynamical systems approach to solving linear programming problems. *Lect. Notes Pure Appl. Math.* **152**, 913–913 (1993)
138. S.H. Zak, V. Upatising, S. Hui, Solving linear programming problems with neural networks: a comparative study. *IEEE Trans. Neural Netw.* **6**(1), 94–104 (1995)
139. S. Zhang, A.G. Constantinides, Lagrange programming neural networks. *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.* **39**(7), 441–452 (1992)
140. M. Zhang, N.B. Liao, C. Zhou, Neural networks model for optimization an study on isomorphism discernment. *Adv. Mater. Res.* **156**, 492–495 (2011)
141. P. Zheng, W. Tang, J. Zhang, A simple method for designing efficient small-world neural networks. *Neural Netw.* **23**(2), 155–159 (2010)

On Coloring Problems

Weifan Wang and Yuehua Bu

Contents

1	Introduction	2096
1.1	Definitions	2097
1.2	Planar Graphs and Basic Known Facts	2098
2	Classical Vertex Coloring	2099
2.1	General Upper Bounds	2099
2.2	Hajós Conjecture and Hadwiger Conjecture	2102
2.3	Coloring Graphs Embedded in a Surface	2103
2.4	Coloring Planar Graphs	2106
3	Acyclic Coloring	2110
3.1	Acyclic Vertex Coloring	2110
3.2	Acyclic List Coloring	2116
3.3	Acyclic Edge Coloring	2118
3.4	Related Coloring Problems	2125
4	Vertex-Distinguishing Coloring	2132
4.1	Proper Edge-Weighting	2133
4.2	Non-proper Edge-Weighting	2139
4.3	Some Related Topics	2145
5	$L(p, q)$ -Labeling of Graphs	2148
5.1	$L(2, 1)$ -Labeling	2149
5.2	Coloring the Square of Graphs	2151
5.3	Backbone Coloring	2156
5.4	Injective Coloring	2164
5.5	$(d, 1)$ -Total Coloring	2168
6	Conclusion	2173
	Cross-References	2174
	Recommended Reading	2174

W. Wang • Y. Bu

Department of Mathematics, Zhejiang Normal University, Jinhua, Jinhua, People's Republic of China

Abstract

Graph coloring is an important branch in graph theory, since it has come from the famous Four-Color Problem and is of many applications in time tabling, sequencing, scheduling, coding, frequency channel assignment, and other practical problems. Since this area includes quite a number of subjects and there exist too many interesting problems, only several interested subjects are selected in this chapter. An introduction of concepts and symbols on graph coloring is given in Sect. 1. Section 2 is devoted to discussing the classical vertex-coloring problem, involving the general upper bound of the vertex chromatic number, and the colorability and choosability of planar graphs and graphs embeddable in a surface. Section 3 investigates the acyclic vertex coloring and acyclic edge coloring of graphs as well as various generalizations such as star coloring, linear coloring, and acyclic improper coloring. Section 4 focus on some recent progresses on vertex-distinguishing edge-weighting problems. Two types of edge-weighting, that is, proper edge-weighting and non-proper edge-weighting, are mentioned respectively. The $L(p, q)$ -labeling problem of graphs will be finally investigated in Sect. 5, including the $L(2, 1)$ -labeling, the coloring of the square of a graph, the injective coloring, the backbone coloring, and the $(d, 1)$ -total labeling. In each section, a somewhat detailed survey on the recent advance of the related direction and some open problems are provided.

1 Introduction

Many real-world situations can conveniently be described by means of a diagram consisting of a set of points together with lines connecting certain pairs of these points. For example, the points may represent people, with lines connecting pairs of friends, or the points may be communication centers, with lines representing communication links. Notice that in such diagrams, one is mainly interested in whether two given points are connected by a line; the manner in which they are connected is immaterial. A mathematical abstraction of situations of this type gives rise to the graph concept. The basic concepts of graph theory are simple and can be used to express problems from many different subjects.

Graph coloring is an important branch in graph theory. It has a central position in discrete mathematics and is of interest for its applications. Graph coloring deals with the fundamental problem of partitioning a set of objects into classes, according to certain rules. Time tabling, sequencing, and scheduling problems, in their many forms, are basically of this nature.

Most of graph coloring problems come from the famous Four-Color Problem which states that any map in a plane can be colored using four colors in such a way that regions sharing a common boundary (other than a single point) have distinct colors. This was a question which Francis Guthrie asked his brother Frederick Guthrie, who was a student of De Morgan in mathematics. In 1878, the Four-Color

Problem was formally declared by Cayley in Mathematics Conference in London, and thus, it received much attention. A proposed solution by Kempe [224] in 1879 stood for more than a decade until it was refuted by Heawood [187] in 1890 in his first paper. In 1976, the Four-Color Problem was proved by Appel and Haken [27] using computer. The proof was based on the same basic idea as Kempe's proof, that is, to find a set of unavoidable and reducible configurations. Since then, the Four-Color Problem has been changed into the Four-Color Theorem.

In this chapter, some coloring problems of graphs are studied. The organization is as follows. Firstly, an introduction of graph coloring and planar graphs in the rest of this section is given. Then, Sect. 2 is devoted to presenting some general upper bounds on chromatic number of graphs. In particular, the coloring problems of graphs that are embeddable in a surface are also considered in this section. Section 3 is concerned with the study of acyclic coloring and acyclic edge coloring of graphs. One interesting problem is to prove or disprove the Acyclic Edge-Coloring Conjecture, which has attracted considerable much attention in recent years. Some positive evidences about this challenging conjecture are collected. Moreover, some related colorings such as star coloring, linear coloring, frugal coloring, and improper coloring are also discussed in this section. Section 4 focuses on some recent progresses on vertex-distinguishing edge-weighting problems. Two types of edge-weightings, that is, proper edge-weighting and non-proper edge-weighting, are the main topics in this section. Finally, $L(p, q)$ -labeling problem of graphs is discussed in Sect. 5. Several well-known conjectures such as Griggs and Yeh's Conjecture, Wegner's Conjecture, and Havet and Yu's Conjecture are mentioned.

Simple graphs with n vertices, m edges, maximum degree Δ , minimum degree δ , clique number $\omega(G)$, and girth $g(G)$ (i.e., the length of a shortest cycle in G) are considered throughout this chapter. For all graph theoretic concepts, definitions, symbols, and graph classes inclusions not given in this chapter are referred either to [126] or the related references.

1.1 Definitions

A *vertex k -coloring* of a graph G is an assignment π of integers $1, 2, \dots, k$ (as colors) to the vertices of G . The coloring π is said to be *proper* if $\pi(u) \neq \pi(v)$ for any pair of adjacent vertices u and v of G . Obviously, in a proper vertex coloring, each color class induces an *independent* set of vertices; that is, no two of them are joined by an edge. The *chromatic number*, denoted by $\chi(G)$, is the least integer k such that G has a proper k -coloring. The graph G is called *k -chromatic* if $\chi(G) = k$.

A *t -improper k -coloring* of a graph G (sometimes called generalized, defective, or relaxed coloring) or simply a $(k, t)^*$ -coloring is a k -coloring of G in which every color class has a bounded degree t . The *t -improper chromatic number* of G is therefore defined as the smallest integer k such that G is $(k, t)^*$ -colorable. Notice that 0-improper coloring corresponds to the ordinary notion of proper coloring.

So a $(k, 0)^*$ -coloring of G is a proper k -coloring of G , and the 0-improper chromatic number of G is the chromatic number of G .

The smallest integer k , such that the edges of a graph G can be colored with k colors in such a way that any two adjacent edges have distinct colors, is called the *chromatic index* or *edge chromatic number* of G , denoted by $\chi'(G)$. Such a coloring is called a *proper edge-coloring* of G . Note that $\chi'(G) = \chi(L(G))$, where $L(G)$ denotes the line graph of a graph G .

L is called an *assignment* for the graph G if it assigns a list $L(v)$ of possible colors to each vertex v of G . If G has a proper coloring π such that $\pi(v) \in L(v)$ for all vertices v , then G is *L -colorable* or π is called an *L -coloring* of G . The graph G is *k -choosable* (or *list k -colorable*) if it is L -colorable for every assignment L satisfying $|L(v)| \geq k$ for all vertices v . The *list chromatic number* of G , denoted $\chi^l(G)$, is the smallest integer k such that G is k -choosable.

The concept of list coloring was introduced by both Vizing [341] in 1976 and Erdős et al. [140] in 1979. Note that $\chi^l(G) \geq \chi(G)$, and $\chi^l(G)$ can be arbitrarily larger than $\chi(G)$.

1.2 Planar Graphs and Basic Known Facts

A graph G is *planar* if it has a drawing without crossings in the plane. Such a drawing is a *planar embedding* of G . A *plane graph* is a particular embedding of a planar graph. A planar embedding of a graph G cuts the plane into a number of arcwise-connected open sets. These sets are called the *faces* of G , denoted by $F(G)$. Each plane graph has exactly one unbounded face, called the *outer face* and others called *internal faces*. The *dual graph* G^* of a plane graph G is a plane graph whose vertices correspond to the faces of G . Two vertices in G^* are adjacent if the corresponding faces in G are adjacent.

The next lemma may be regarded as a dual version of the Handshake Lemma.

Lemma 1 *If G is a plane graph with m edges, then*

$$\sum_{f \in F(G)} d(f) = 2m.$$

There is a celebrated formula relating the numbers of vertices, edges, and faces in a connected plane graph. It was first established for polyhedral graphs by Euler in 1752 and is known as Euler's formula, which plays a key role in the proof of results on planar graphs that use the powerful discharging method.

Theorem 1 (Euler's Formula) *Let G be a connected plane graph with n vertices, m edges, and f faces. Then*

$$n - m + f = 2. \quad (1)$$

Corollary 1 can be easily obtained by the Euler's formula.

Corollary 1 *If G is a planar graph with girth $g \geq 3$ and order $n \geq 3$, then*

$$m \leq \frac{g}{g-2}(n-2). \quad (2)$$

In particular, if G is a simple connected planar graph, then

$$m \leq 3n - 6. \quad (3)$$

The most elegant result about planar graphs should be the Kuratowski's Theorem [242] which characterizes completely planar graphs in terms of forbidden subgraphs. The first relatively simple proof on this theorem was published in 1954 by Dirac and Schuster [130]. Other alternative proofs can be seen in [266, 326, 327].

A *subdivision* of a graph G is obtained by replacing each edge of G with a path. A k -*subdivision* of G is such that any path replacing an edge of G has at most k internal vertices.

Theorem 2 (Kuratowski's Theorem) *A graph is planar if and only if it does not contain a subdivision of K_5 or $K_{3,3}$ as a subgraph.*

2 Classical Vertex Coloring

The problem of the classical vertex coloring of graphs is considered in this section. Some important results and recent progresses in this direction will be surveyed. Usually, the coloring considered in this section is proper unless otherwise stated.

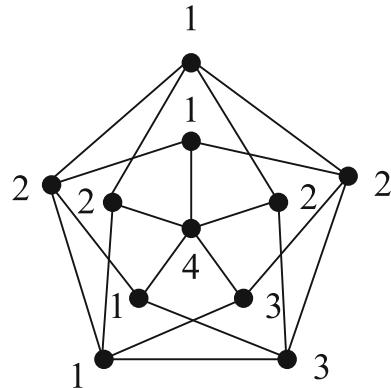
2.1 General Upper Bounds

Classical graph coloring or, more precisely, the task of finding an optimal proper vertex coloring, is an NP-hard problem [159]. Indeed, this type of problem still remains NP-hard even in spite of strong restrictions imposed on the class of graphs which are taken into consideration and on the number of colors used. Recently, a proof of the NP-hardness of 4-coloring of tripartite graphs was presented in [225]. Furthermore, the problem of determining whether a given planar graph with a degree not exceeding 4 is 3-colorable is also NP-hard [160]. In order to determine a simple bound on the graph chromatic number in the general case, it is sufficient to observe that an empty graph is 1-colorable, while the complete graph K_n has the chromatic number n . A well-known result is stated as follows:

Theorem 3 *A graph has chromatic number at most 2, that is, is bipartite, if and only if it contains no odd cycles.*

On the one hand, the proof of Theorem 3 yields an efficient (in fact, linear time) algorithm for determining if G is bipartite. On the other hand, it is NP-complete

Fig. 1 The Grötzsch graph:
a 4-chromatic and
triangle-free graph



to determine if G is of the chromatic number 3. Hence, determining the chromatic number of a graph G precisely seems difficult. However, there are a number of results yield simple bounds on the chromatic number of G . Among of them, a trivial lower bound is easy to obtain.

Recall that a *clique* is a set of pairwise adjacent vertices. Let $\omega(G)$ denote the *clique number* of G , that is, the number of vertices in the largest clique of G . It is still unknown if there exists an algorithm for determining $\omega(G)$ in the general case, because this problem is NP-hard.

Theorem 4 *For any graph G , $\chi(G) \geq \omega(G)$.*

Notice that this bound is best possible for complete graphs, bipartite graphs, and interval graphs. However, there are also many graphs whose chromatic number exceeds their clique number such as the Petersen graph and the odd cycles of length 5 or more. As it is about to see, the chromatic number of a graph can be considerably larger than its clique number.

For instance, the graph G , depicted in Fig. 1, is triangle-free (and so $\omega(G) = 2$) but $\chi(G) = 4$. So $\chi(G)$ exceeds $\omega(G)$ by 2 in this case. This graph is the famous Grötzsch graph. It is known to be the unique smallest graph (in terms of order) that is 4-colorable and triangle-free. The fact that a graph can be triangle-free and yet have a large chromatic number has been established by a number of mathematicians, including Descartes [125], Kelly and Kelly [223], Mycielski [289], and Zykov [402].

Theorem 5 *For every positive integer k , there exists a triangle-free k -chromatic graph.*

Most upper bounds on the chromatic number come from algorithms that produce colorings. For example, assigning distinct colors to the vertices yields $\chi(G) \leq |G|$, where $|G|$ usually denotes the order of G . This bound is tight, since $\chi(K_n) = n$, but it holds with equality only for complete graphs. The following upper bound can be directly derived from the definition of the chromatic number:

Proposition 1 Every graph G with m edges satisfies

$$\chi(G) \leq \frac{1}{2} + \sqrt{2m + \frac{1}{4}}.$$

Clearly, $\chi(G) \leq |G|$ uses nothing about the structure of G . Yet, it can do better by coloring the vertices in some order and always using the *least available* color.

The *greedy coloring* [379] relative to a vertex ordering v_1, v_2, \dots, v_n of $V(G)$ is obtained by coloring vertices in the order v_1, v_2, \dots, v_n , assigning to v_i the smallest indexed color not already used on its lower-indexed neighbors.

Theorem 6 $\chi(G) \leq \Delta + 1$ for every graph G .

Proof First, give a random ordering of vertices. Clearly, each vertex has at most Δ vertices as (early) neighbors. So the greedy coloring cannot be forced to use more than $\Delta + 1$ colors. \square

Theorem 7 If a graph G has degree sequence d_1, d_2, \dots, d_n , then $\chi(G) \leq 1 + \max_i \{\min\{d_i, i - 1\}\}$.

Observe that the upper bound in Theorem 7 is always at most $\Delta + 1$. So this is at least as good as Theorem 6. On the other hand, Szekeres and Wilf [324] in 1968 showed a relationship between chromatic number and minimum degree.

Theorem 8 If G is a graph, then $\chi(G) \leq 1 + \max_{H \subseteq G} \{\delta(H)\}$.

The bound $\chi(G) \leq \Delta + 1$ holds with equality for complete graphs and odd cycles. By choosing the vertex ordering more carefully, it can be shown that there are essentially the only such graphs. To avoid unimportant complications, it suffices to give the statement for connected graphs. It extends to all graphs because the chromatic number of a graph is the maximum chromatic number of its components.

Theorem 9 (Brooks' Theorem) If G is a connected graph other than a complete graph or an odd cycle, then $\chi(G) \leq \Delta$.

For the proof of Brooks' Theorem, one may refer to [258]. Notice that for some special class of graphs, $\chi(G)$ is far from Δ . Such a situation occurs for stars $K_{1,m}$. Obviously, $K_{1,m}$ has chromatic number exactly 2, while the discussed bound implies that $\chi(K_{1,m}) \leq m$. But the bound Δ can be improved when G has no large clique. Brooks' Theorem states that $\chi(G) \leq \Delta$ whenever $3 \leq \omega(G) \leq \Delta$. Borodin and Kostochka [75] conjectured that $\omega(G) < \Delta$ implies $\chi(G) < \Delta$ if $\Delta \geq 9$ (an example shows that the condition $\Delta \geq 9$ is needed). Reed [301] proved that this conjecture holds when $\Delta \geq 10^{14}$. Moreover, Reed [300] proved the following:

Theorem 10 If G is a graph, then $\chi(G) \leq \lceil \frac{\Delta+1+\omega(G)}{2} \rceil$.

2.2 Hajós Conjecture and Hadwiger Conjecture

Notice that the clique number $\omega(G)$ of a graph G is a lower bound for $\chi(G)$. **Theorem 5** states that for every integer $k \geq 3$, there is a graph G such that $\chi(G) = k$ and $\omega(G) = 2$. Indeed, every odd cycle of order at least 5 is 3-chromatic but none of these graphs contains K_3 as a subgraph. But all of these do contain a subdivision of K_3 . Since every 3-chromatic graph contains an odd cycle, it follows that if G is a graph with $\chi(G) \geq 3$, then G must contain a subdivision of K_3 . In 1952, Dirac [128] showed that the corresponding result is true as well for graphs having chromatic number of at least 4.

Theorem 11 *If G is a graph with $\chi(G) \geq 4$, then G contains a subdivision of K_4 .*

Therefore, for $2 \leq k \leq 4$, every k -chromatic graph contains a subdivision of K_k . In 1961, Hajós [176] conjectured that this is true for every integer $k \geq 2$.

Conjecture 1 (Hajós Conjecture) *If G is a k -chromatic graph, where $k \geq 2$, then G contains a subdivision of K_k .*

In 1979, Catlin [95] constructed a family of graphs that showed that Hajós Conjecture is false for every integer $k \geq 7$. A graph G is called *perfect* if $\chi(H) = \omega(H)$ for every induced subgraph H of G . Then a class of perfect graphs can be obtained from a given perfect graph. Let G be a graph with $v \in V(G)$. Then the *replication graph* $R_v(G)$ of G (with respect to v) is the graph obtained from G by adding a new vertex v' to G and joining v' to the vertices in the closed neighborhood $N[v]$ of v .

In 1972, Lovász [257] obtained the following result.

Theorem 12 *Let G be a graph where $v \in V(G)$. If G is perfect, then $R_v(G)$ is perfect.*

Recently, Thomassen [333] showed that there is a connection between perfect graphs and Hajós' Conjecture.

Theorem 13 *A graph G is perfect if and only if $R_v(G)$ satisfies Hajós Conjecture.*

A graph H is a *minor* of a graph G if H can be obtained from G by a sequence of edge contractions, edge deletions, and vertex deletions (in any order). Since Hajós Conjecture does not hold for the case $k \geq 7$, a k -chromatic graph need not contain a subdivision of K_k . But this fact does not imply that a k -chromatic graph need not contain K_k as a minor. Before Hajós Conjecture was made, in 1942, Hadwiger posed the following conjecture:

Conjecture 2 (Hadwiger Conjecture) *Every k -chromatic graph contains K_k as a minor.*

Hadwiger Conjecture holds trivially for $k \leq 1$. Dirac [128] proved the conjecture for $k = 3$. The case $k = 4$ is significantly more challenging. The Four-Color

Conjecture is clearly implied by Hadwiger Conjecture for the case $k = 4$, as it is well known that any graph with a K_5 -minor is non-planar. In fact, Wagner [345] showed that in this case Hadwiger Conjecture is equivalent to the Four-Color Conjecture. In 1980, Albertson and Hutchinson [15] showed that Hadwiger Conjecture is true if $k = 6$ and the graph is the Klein bottle. Recently, by the Four-Color Theorem, Robertson, Seymour, and Thomas [305] confirmed Hadwiger Conjecture for the case $k = 6$. Hence, Hadwiger Conjecture remains open for all larger values of k .

One may check that the complete graph is a perfect graph, since every subgraph of a complete graph is also complete. Moreover, it was proved in [275] that the bipartite graph and the complement of bipartite graph are both perfect graphs. It means that if a graph G is either a complete graph or a bipartite graph, then G and \overline{G} are both perfect. Indeed, in 1961, Berge made the following conjecture.

Conjecture 3 (Perfect Graph Conjecture) *A graph is perfect if and only if its complement is perfect.*

In 1972, Lovász [257] showed that this conjecture is, in fact, true.

Theorem 14 (Perfect Graph Theorem) *A graph is perfect if and only if its complement is perfect.*

It is easy to observe that the perfect graph cannot contain a subgraph H which is an induced odd cycle with more than five vertices. If that is indeed the case, then $\chi(H) = 3$ and $\omega(H) = 2$. Denote $H = C_{2k+1}$, where $k \geq 2$. One may easily check that $\chi(\overline{H}) = k+1$ and $\omega(\overline{H}) = k$. Basing on these observations, Berge [48] proposed the following famous conjecture:

Conjecture 4 (Strong Perfect Graph Conjecture) *A graph G is perfect if and only if neither G nor \overline{G} contains an induced odd cycle of length 5 or more.*

This challenging conjecture was recently solved by Chudnovsky, Robertson, Seymour, and Thomas [114] by a long and technical proof.

Theorem 15 (Strong Perfect Graph Theorem) *A graph G is perfect if and only if neither G nor \overline{G} contains an induced odd cycle of length 5 or more.*

2.3 Coloring Graphs Embedded in a Surface

This section is concerned with the vertex coloring of graphs embedded in a given (closed) surface. Motivated by the Four-Color Problem, coloring problems on graphs embedded in various surfaces have been studied extensively in the literature. Surfaces can be classified according to their genus and colorability (choosability). The *orientable surfaces* are the sphere with g handles S_g , where $g \geq 0$. For example, cylinder and torus are both orientable surfaces.

The *non-orientable surfaces* are the surfaces Π_h ($h \geq 1$) obtained by taking the sphere with h holes and attaching h Möbius bands along their boundary to the boundaries of the holes. For example, Π_1 is the projective plane and Π_2 is the Klein bottle. An embedding of a graph G in a surface S is called a *2-cell embedding* if each face of G is homeomorphic to an open unit disk. The *Euler characteristic* $\varepsilon(S) = 2 - 2g$ of a surface S is equal to $|V(G)| - |E(G)| + |F(G)|$ for any multigraph G that is 2-cell embedded in S . When $\varepsilon(S) = 2$, S denotes the Euclidean plane and G is a plane graph. When $\varepsilon(S) = 0$, S denotes the torus or the Klein bottle.

Among all those graphs embedded in the surfaces, the most interested one is the *toroidal graph*. A graph G is *toroidal* if it can be embedded in the torus. In other words, the graph's vertices can be placed in a torus such that no edges cross. Usually, it is assumed that G is also non-planar. For example, the Heawood graph, the complete graph K_7 (and hence K_5 and K_6), the Petersen graph, and all Möbius ladders are toroidal. More generally, any graph with crossing number 1 is toroidal. But some graph with greater crossing numbers is also toroidal, that is, the Möbius-Kantor graph has crossing number 4 and is also toroidal.

The well-known Four-Color Theorem [27] asserts that every plane graph G is 4-colorable. In 1890, Heawood [187] established an upper bound for the vertex chromatic number of graphs embedded in a surface S of genus $g \geq 1$. This upper bound was called the Heawood number $h(S) = \lfloor \frac{7 + \sqrt{49 - 24\varepsilon(S)}}{2} \rfloor$.

Theorem 16 *If G is a graph embedded in a surface S of genus $g \geq 1$, then $\chi(G) \leq h(S)$.*

Almost a century later, Ringel [302] and Ringel and Youngs [303] proved that the Heawood number $h(S)$ is in fact the maximum chromatic number as well as the maximum clique number of graphs embedded in a surface of genus $g \geq 1$ besides the Klein bottle, which is 6-colorable (showing by Franklin [156]) but has $h(S) = 7$.

In 1956, Dirac [129] refined this by showing that the upper bound $h(S)$ is attained only if a graph G contains $K_{h(S)}$ as a subgraph except for three surfaces. Albertson and Hutchinson [14] settled these excluded cases in 1979. This result is nowadays known as Dirac's Map-Color Theorem. Later, Böhme, Mohar, and Stiebitz [53] extended Dirac's Map-Color Theorem to the case of choosability by showing that G is $(h(S) - 1)$ -choosable unless G contains $K_{h(S)}$ as a subgraph for genera $g \geq 1$ and $g \neq 3$. For the left case $g = 3$, it was then solved by Král and Škrekovski [238].

By Theorem 16, if G is a graph embedding in a surface with $\varepsilon(S) = 0$, then $\chi(G) \leq 7$. It follows that every toroidal graph is 7-colorable. This result is best possible, since the complete graph K_7 has chromatic number 7. Dirac [128] showed that every 7-chromatic toroidal graph contains K_7 as a subgraph. In 1999, Böhme, Mohar, and Stiebitz [53] further generalized these two results to the list-coloring situation:

Theorem 17 *Every toroidal graph G is 7-choosable and $\chi^l(G) = 7$ if and only if $K_7 \subseteq G$.*

Characterizing k -colorable toroidal graphs with $k = 3, 4, 5$ turns out to be an interesting problem. It has received much attention in the past years, even before 1990. A cycle C of a graph G embedded in a surface S is called *contractible* if, when viewed as a closed curve on S , it is homotopic to a point; otherwise, it is called *noncontractible*. By considering noncontractible cycles with given length, Albertson and Stromquist [16] proved the following result.

Theorem 18 *If G can be embedded in the torus such that noncontractible cycles have length at least 8, then $\chi(G) \leq 5$.*

Let G_1 and G_2 be graphs. Their *union* $G_1 \cup G_2$ is the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$. The *join* of G_1 and G_2 is the graph $G_1 \vee G_2$ obtained from $G_1 \cup G_2$ by adding an edge v_1v_2 for each vertex v_1 of G_1 and each vertex v_2 of G_2 . A graph is called *Hajós- n -constructible* if it can be obtained from complete graphs K_n by repeated application of the following two operations:

- (a) Let G_1 and G_2 be already obtained disjoint graphs with edges x_1y_1 and x_2y_2 . Remove x_1y_1 and x_2y_2 , identify x_1 and x_2 , and join y_1 and y_2 by a new edge.
- (b) Identify independent vertices (i.e., apply a homomorphism).

Let H_7 be the graph with seven vertices obtained by applying Hajós' construction to two copies of K_4 (only operation (a)). Let T_{11} be a special triangulation with 11 vertices of the torus. In 1994, Thomassen [330] gave a new sufficient condition for toroidal graphs to be 5-colorable.

Theorem 19 *Every toroidal graph is 5-colorable unless it contains one of the following four subgraphs: (1) K_6 , (2) T_{11} , (3) $C_3 \vee C_5$, and (4) $K_2 \vee H_7$.*

If a toroidal graph G is 6-regular, then the above theorem implies the following:

Corollary 2 *Every 6-regular toroidal graph is 5-colorable unless it contains either K_6 or T_{11} .*

Recently, the 4-colorable 6-regular toroidal graphs have been studied by many authors. Combining their results in [22, 118, 389], all 4-colorable 6-regular toroidal graphs have been completely characterized.

As an effort to understand graphs embedded in orientable surfaces of nonnegative characteristic, one has studied k -colorable toroidal graphs without some specific short cycles or with given girth. In 1972, Kronk and White [240] proved the following:

Theorem 20 *Let G be a toroidal graph with girth g :*

- (1) *If $g \geq 4$, then $\chi(G) \leq 4$.*
- (2) *If $g \geq 6$, then $\chi(G) \leq 3$.*

The second conclusion in **Theorem 20** has been strengthened in two aspects. On the one hand, the girth requirement was reduced to 5 by Thomassen [329]. On the other hand, Cai et al. [91] proved that every toroidal graph G of girth at least 6 is 3-choosable. Moreover, they proposed the following conjecture:

Conjecture 5 *Every toroidal graph of girth at least 5 is 3-choosable.*

In a slightly different approach, Cai et al. [91] considered the coloring of toroidal graphs without specific cycles. Suppose that G is a toroidal graph without k -cycles. Then the following result was proved in [91]:

$$\chi^l(G) \leq \begin{cases} 4, & \text{if } k \in \{3, 4, 5\}; \\ 5, & \text{if } k = 6; \\ 6, & \text{if } k = 7. \end{cases}$$

Observe that toroidal graphs without 6-cycles (e.g., K_5) may not be 4-choosable. However, K_5 seems to be the only obstacle for such graphs to be 4-choosable. Basing on these observations, Cai et al. [91] proposed the following problem:

Conjecture 6 *Let G be a toroidal graph without 6-cycles. If G is K_5 -free, then $\chi^l(G) \leq 4$.*

For other 3-colorable graphs embedded in surfaces, one classical result of Heawood and Kempe states that every Eulerian triangulation of the plane is 3-colorable. Note that the condition that the graph be embedded in the plane is essential. For example, K_7 can be embedded as an Eulerian triangulation of the torus but has chromatic number 7. On the other hand, it is recently proved by Hutchinson et al. [208] that every Eulerian triangulation G of an orientable surface is 4-colorable if every noncontractible cycle of G is sufficiently large.

2.4 Coloring Planar Graphs

As it is mentioned in the previous section, Vizing [341] and Erdős et al. [140] independently introduced the concepts of list coloring and choosability. By the Four-Color Theorem [27], every planar graph G has a partition (V_1, V_2, V_3, V_4) such that each V_i induces an independent set. However, Wegner [377] showed that there exists a planar graph which cannot be partitioned into (V_1, V_2, V_3) such that V_1 and V_2 are independent sets and V_3 is a forest, and even earlier, Chartrand and Kronk [101] showed that there exists a planar graph which cannot be partitioned into two forests. On the other hand, Voigt [342] and Mirzakhani [268], independently, proved that not all planar graphs are 4-choosable. All of these facts imply that it is impossible to strengthen the Four-Color Theorem.

However, it is evident that every planar graph is 6-choosable, which follows from the fact that every planar graph contains a vertex of degree at most 5. In 1979, Erdős et al. [140] conjectured that there exists a planar graph G such that $\chi^l(G) = 5$ and that there is no planar graph G such that $\chi^l(G) = 6$. The second conjecture was confirmed by Thomassen [328] in 1994.

Theorem 21 *Every planar graph is 5-choosable.*

The bound in [Theorem 21](#) is best possible, since Voigt [342] and Mirzakhani [268] independently proved the following theorem:

Theorem 22 *There exists a planar graph G such that $\chi^l(G) = 5$.*

The graph presented in the original proof given by Voigt [342] has 238 vertices. In [343], she further constructed a planar graph G of order 130 with $\chi(G) = 3$ and $\chi^l(G) = 5$. This gives a negative answer to the question whether $\chi^l(G) - \chi(G) \leq 1$ for every planar graph G . The smallest known so far planar graph G with $\chi^l(G) = 5$ was constructed by Mirzakhani [268]. This graph has 63 vertices and chromatic number 3.

If all vertices of degree 1 are removed recursively from a graph G , then the final graph has no vertices of degree 1 and is called the *core* of G . A graph is called a $Q_{2,2,p}$ -graph if it consists of two vertices x and y and three internally disjoint paths of lengths 2, 2 and p joining x and y . For example, $\theta_{2,2,2}$ is isomorphic to $K_{2,3}$. Using these two concepts, Erdős et al. [140] established the following characterization for the 2-choosability of a graph.

Theorem 23 *A connected graph is 2-choosable if and only if the core of G is isomorphic to K_1 , C_{2m+2} , or $\theta_{2,2,2m}$ for some $m \geq 1$.*

So characterizing 3- or 4-choosable planar graphs becomes one of the most interesting problems in investigating the choosability of planar graphs. However, Gutner [173] proved that both these problems are NP-complete.

2.4.1 4-Choosable Planar Graphs

A graph G is k -degenerate if every subgraph H of G has a vertex of degree at most k in H . It is well known that every k -degenerate graph is $(k + 1)$ -choosable. It is easy to prove that every planar triangle-free graph is 3-degenerate by Euler's formula. Wang and Lih [359] proved that planar graphs without 5-cycles are 3-degenerate, while Fijavž et al. [155] showed that planar graphs without 6-cycles are also 3-degenerate. The lack of 4-cycles does not imply the 3-degeneracy of a planar graph, for example, the line graph of a dodecahedron. However, Lam et al. [244] proved that planar graphs without 4-cycles are 4-choosable. Recently, Farzad [144] proved the conjecture proposed by Fijavž et al. [155] and independently by Wang and Lih [358] that planar graphs without 7-cycles are 4-choosable.

These facts together summarize the following theorem:

Theorem 24 *If G is a planar graph without i -cycles for some fixed $i \in \{3, 4, 5, 6, 7\}$, then G is 4-choosable.*

Borodin and Ivanova [66] improved the above-mentioned result in [244] by showing that every planar graph without 4-cycles adjacent to 3-cycles is 4-choosable. By considering the distance of two triangles, Wang and Lih [360] proved the following

Theorem 25 *Every planar graph without intersecting triangles is 4-choosable.*

Recently, Luo [261] extends the result to all graphs embeddable in an orientable surface of non-negative Euler's characteristic by showing the following.

Theorem 26 *Every toroidal graph without intersecting triangles is 4-choosable.*

Two triangles, that is, two 3-cycles, are *intersecting* if they share at least one common vertex. Two triangles are *adjacent* if they share at least one common edge.

The following challenging conjecture was proposed in [360]:

Conjecture 7 *Every planar graph without adjacent triangles is 4-choosable.*

This conjecture remains open. However, a $(4, 1)^*$ -choosability of planar graphs without adjacent triangles is indeed the case by Xu and Zhang [384]. A graph G is called $(k, d)^*$ -*choosable* if, for every list assignment L satisfying $|L(v)| = k$ for all $v \in V(G)$, there is an L -coloring of G such that each vertex of G has at most d neighbors colored with the same color as itself. The notion of list improper coloring was introduced independently by Škrekovski [320] and by Eaton and Hull [137]. They proved that every planar graph is $(3, 2)^*$ -choosable and every outerplanar graph is $(2, 2)^*$ -choosable.

In fact, Xu and Zhang [384] extended such a list coloring to any toroidal graph. Their main result is the following:

Theorem 27 *Every toroidal graph without adjacent triangles is $(4, 1)^*$ -choosable.*

Note that this result is best possible in the sense that K_7 is a non- $(3, 1)^*$ -choosable toroidal graph.

2.4.2 Steinberg's Conjecture

In 1958, Grötzsch [170] proved that planar graphs without 3-cycles are 3-colorable. But not every triangle-free planar graph is 3-choosable. The first example of such graphs was provided by Voigt [343]. However, Škrekovski [319] proved that every planar graph without 3-cycles is $(3, 1)^*$ -choosable. Alon and Tarsi [20] proved that every planar bipartite graph is 3-choosable. Moreover, Thomassen [331] proved that planar graphs with girth at least 5 are 3-choosable. As early as 1969, Havel [180] asked if there existed a constant C such that each planar graph with the minimal distance between triangles at least C is 3-colorable. In 1976, Aksionov and Mel'nikov [11] and, independently, Steinberg [322] proved that $d \geq 4$. Moreover, Steinberg [322] proposed the following conjecture:

Conjecture 8 *Every planar graph without 4-cycles and 5-cycles is 3-colorable.*

This challenging conjecture still remains unsolved. In 1990, Erdős suggested the following relaxation of Steinberg's conjecture:

Question 1 *What is the smallest integer k such that every planar graph without i -cycles for $4 \leq i \leq k$ is 3-colorable?*

In [1], Abbott and Zhou showed that such k exists and $k \leq 11$. This result was improved to $k \leq 9$ by Borodin [57] and independently by Sanders and Zhao [310], and then it was further reduced to $k \leq 7$ by Borodin et al. [65]. Recently, Borodin et al. [79], and independently Wang et al. [364], strengthened this result by showing that every planar graph without adjacent cycles of lengths at most 7 is 3-colorable.

2.4.3 3-Choosable Planar Graphs

To answer Erdős's question, many authors considered some restricted planar graphs, that is, without some cycles of four lengths; see [108, 263, 314, 354, 398, 399]. Strengthening results of these types, Wang and Chen [353] proved that every planar graph without 4-, 6-, and 8-cycles is 3-colorable; Lu et al. [259] proved that every planar graph without 4-, 7-, and 9-cycles is 3-colorable; Borodin et al. [64] proved that every planar graph without 5-cycles, 7-cycles, and adjacent 3-cycles is 3-colorable, which implies that every planar graph without 4-, 5-, and 7-cycles is 3-colorable.

Naturally, a similar question on choosability of planar graphs is stated as follows:

Question 2 *What is the smallest integer k^* such that every planar graph without j -cycles for $4 \leq j \leq k^*$ is 3-choosable?*

Notice that it is impossible to extend Steinberg's conjecture to list coloring by the example given by Voigt [344] and independently, by Montassier [277]. Hence $k^* \geq 6$. The best known upper bound is $k^* \leq 9$ obtained by Borodin [57] in 1996, that is, every planar graph without $\{4, 5, \dots, 9\}$ -cycles is 3-choosable.

These results have also been improved by showing that forbidding four cycles of certain size would lead to 3-choosability of planar graphs. These results are summarized in the following theorem:

Theorem 28 *A planar graph is 3-choosable if it has no*

- (Zhang and Wu [398]) 4-, 5-, 7-, and 9-cycles
- (Zhang and Wu [399]) 4-, 5-, 6-, and 9-cycles
- (Chen et al. [105]) 4-, 6-, 7-, and 9-cycles
- (Shen and Wang [314]) 4-, 6-, 8-, and 9-cycles
- (Wang et al. [375]) 4-, 7-, 8-, and 9-cycles
- (Wang et al. [362]) 4-, 5-, 8-, and 9-cycles

As it can be seen, it is really not easy to reduce the value of k to answer Question 2. Lih et al. [249] gave the following result on improper choosability of planar graphs without some short cycles, which may be regarded as a solution to a weakened form of Steinberg's conjecture:

Theorem 29 *Every planar graph without 4-cycles and l -cycles for some $l \in \{5, 6, 7\}$ is $(3, 1)^*$ -choosable.*

Since every triangle-free planar graph is 3-colorable, it seems to be interesting to discuss the 3-choosability of such planar graph which contains no other short cycles.

There are many possible combinations of cycles one may try to forbid. These results are collected as follows:

Theorem 30 *A triangle-free planar graph is 3-choosable if it has no*

- (Lam et al. [243]) 5- and 6-cycles
- (Zhang et al. [401]) 8- and 9-cycles
- (Dvořák et al. [135]) 6- and 7-cycles
- (Dvořák et al. [136]) 7- and 8-cycles

By investigating planar graphs with sparse triangles and without short cycles, Montassier et al. [283] proved the following two results.

Theorem 31 *Every planar graph without 4- and 5-cycles, and without two triangles at distance less than 4 is 3-choosable.*

Theorem 32 *Every planar graph without 4-, 5-, and 6-cycles and without two triangles at distance less than 3 is 3-choosable.*

Moreover, they constructed a non-3-choosable planar graph which contains no 4-cycles, 5-cycles and intersecting triangles. So the following problem may be of interest.

Problem 1 *Is a planar graph without 4- and 5-cycles and without triangles at distance less than k ($1 \leq k \leq 3$) 3-choosable?*

3 Acyclic Coloring

This section is devoted to dealing with the acyclic coloring of graphs. This direction includes many interesting results and open problems.

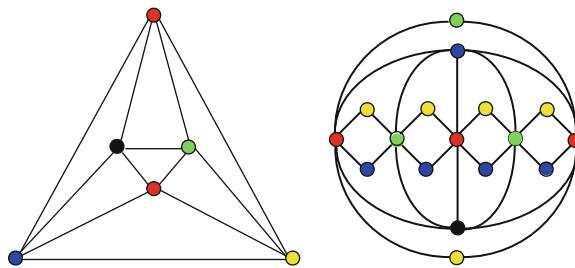
3.1 Acyclic Vertex Coloring

A proper vertex coloring of a graph G is *acyclic* if there is no bicolored cycle in G . Namely, the union of any two color classes induces a forest. The *acyclic chromatic number*, denoted by $\chi_a(G)$, of a graph G is the smallest integer k such that G has an acyclic k -coloring. It is obvious that $\chi(G) \leq \chi_a(G)$ for any graph G .

The notion of acyclic coloring of graphs was introduced by Grünbaum [171] in 1973. It is not difficult to show that if G is a graph with $\Delta \leq 1$, then $\chi_a(G) \leq 2$; if G is a graph with $\Delta \leq 2$, then $\chi_a(G) \leq 3$ and the upper bound value 3 is tight. Moreover, using a greedy coloring procedure, it is easy to verify that for any graph G , $\chi_a(G) \leq \Delta^2 + 1$.

Obviously, determining $\chi_a(G)$ is a hard problem. Actually, Kostochka [231] proved that it is an NP-complete problem to decide for a given graph G whether $\chi_a(G) \leq 3$, and Coleman and Cai [117] showed that it is NP-hard when restricted to bipartite graphs.

Fig. 2 Example of Grünbaum and example of Kostochka and Mel'nikov



3.1.1 Planar Graphs

Grünbaum [171] proved that every planar graph is acyclically 9-colorable and conjectured that 5 is sufficient. This conjecture was studied by many people, for example, Mitchem [269], Albertson, and Berman [12], and Kostochka [229]. In 1979, Borodin [55] settled eventually this conjecture:

Theorem 33 *Every planar graph is acyclically 5-colorable.*

The bound in [Theorem 33](#) is sharp. In 1973, Grünbaum [171] gave an example of a 4-regular planar graph which is not acyclically 4-colorable; furthermore, bipartite planar graphs which are not acyclically 4-colorable were constructed in [233]; see [Fig. 2](#).

Borodin, Kostochka, and Woodall [78] reduced this upper bound 5 by considering planar graphs with large girth.

Theorem 34 *Let G be a planar graph with girth g :*

1. *If $g \geq 5$, then $\chi_a(G) \leq 4$.*
2. *If $g \geq 7$, then $\chi_a(G) \leq 3$.*

More recently, Angelini and Frati [25] investigated the acyclic 3-colorability of some subclasses of planar graphs. They showed the following three results:

Theorem 35 *There exist infinitely many cubic and subcubic planar graphs that admit no acyclic 3-coloring.*

Recall that the 1-subdivision of a graph G is the graph obtained by replacing each edge of G with a path of length at most 2.

Theorem 36 *Every planar graph has a 1-subdivision that admits an acyclic 3-coloring. Such a coloring can be constructed in linear time.*

A graph G is called a *series-parallel* graph if G can be obtained from K_2 by applying a sequence of operations, where each operation is either to duplicate an edge (i.e., replace an edge with two parallel edges) or to subdivide an edge (i.e., replace an edge with a path of length 2).

Theorem 37 *There exists a linear-time algorithm for deciding whether every 3-coloring of a series-parallel graph is acyclic.*

A planar graph is *1-planar* if it can be drawn on the plane in such a way that every edge crosses at most one other edge. Borodin et al. [76] showed the following:

Theorem 38 *Every 1-planar graph is acyclically 20-colorable.*

3.1.2 Graphs Embedded in a Surface

Alon, Mohar, and Sanders [18] proved that every projective planar graph is acyclically 7-colorable. Borodin et al. (see [212]) conjectured that the acyclic chromatic number of any surface but the plane is the same as its chromatic number. Unfortunately, this conjecture is false. An easy counterexample may be referred to the Klein bottle which is 6-colorable [156] but has the acyclic chromatic number at least 7 [18].

Alon et al. [18] presented a more general upper bound of any graph embedded in a surface.

Theorem 39 *The acyclic chromatic number of an arbitrary surface with Euler characteristic $\chi = -\gamma$ is at most $O(\gamma^{\frac{4}{7}})$.*

Notice that this upper bound is nearly tight, since for every $\gamma > 0$, there are graphs embeddable in surfaces of Euler characteristic $-\gamma$ whose acyclic chromatic number is at least $\Omega(\gamma^{\frac{4}{7}}/(\log \gamma)^{\frac{1}{7}})$.

There are two widely recognized notions of *local planarity*, that is, large edge width and large face width (see [275]). For coloring problems, the natural one is the large edge-width condition, meaning that all noncontractible cycles of an embedded graph are large. Recently, acyclic colorings of locally planar graphs have been studied by Mohar [272]. The *nonseparating edge width*, denoted $\text{new}(G)$, is the length of a shortest surface-nonseparating cycle in an embedded graph G . The following theorem was established by Mohar [272].

Theorem 40 *For every surface S , there is a constant c such that every graph G embedded in S with $\text{new}(G) \geq c$ is acyclically 8-colorable.*

Note that the constant c is depending on the genus g of the surface. It is shown to be of order $O(g^3 2^g)$.

More recently, Kawarabayashi and Mohar [222] proved that every graph embedded in a fixed surface with sufficiently large edge width is acyclically 7-colorable. Furthermore, it was shown in [222] that for every surface S , there exists an integer $w = w(S)$ such that every graph embedded in S with edge width at least w is acyclically 7-colorable.

3.1.3 Graphs with a Fixed Maximum Degree

Concerning graphs with a fixed maximum degree, certain results come from Alon, McDiarmid, and Reed [17] using a probabilistic method and the Lovász Local Lemma [45]. Two results in the following were established in [17]:

Theorem 41 *If G has maximum degree Δ , then $\chi_a(G) = O(\Delta^{4/3})$ as $\Delta \rightarrow \infty$.*

Theorem 42 *There exists a graph G such that $\chi_a(G) = \Omega\left(\frac{\Delta^{4/3}}{(\log \Delta)^{1/3}}\right)$ as $\Delta \rightarrow \infty$.*

Notice that [Theorem 41](#) settled a conjecture, due to Erdős, which says that $\chi_a(G) = O(\Delta^2)$ for any graph G of maximum degree Δ . These results show that there are graphs G with maximum degree Δ whose acyclic chromatic numbers are significantly larger than Δ . It turns out that such graphs must contain certain complete bipartite graphs. The proofs of these results rely heavily on probabilistic arguments.

Alon et al. [17] presented a polynomial-time greedy algorithm to acyclically color any graph G with maximum degree Δ using $\Delta^2 + 1$ colors. This was later improved by Albertson et al. [13] to that $\chi_a(G) \leq \Delta(\Delta - 1) + 2$.

For the graphs with a smaller maximum degree Δ , it was shown that every subcubic graph G is acyclically 4-colorable by Grünbaum [171] and independently by Skulrattanakulchai [321]. Observe that four colors are necessary because of K_4 . Moreover, Skulrattanakulchai [321] constructed a linear-time algorithm for acyclically coloring any subcubic graphs:

Theorem 43 *Every subcubic graph G is acyclically 4-colorable; moreover, the vertices of G can be acyclically colored using four colors in $O(n)$ time.*

The acyclic chromatic number of graphs with maximum degree at most 4 has been studied by Buršteín [90] in 1979. He proved that every graph of maximum degree 4 is acyclically 5-colorable (this bound is sharp because of K_5). Recently, Yadava et al. [385] obtained the same result as in [90], by presenting a linear-time algorithm for acyclically coloring any graph of maximum degree 4:

Theorem 44 *The vertices of any graph G of degree at most 4 can be acyclically colored using five colors in $O(n)$ time, where n is the number of vertices.*

The upper bound $\Delta(\Delta - 1) + 2$ [13] for any graph G seems to be not tight. In fact, this bound was later improved to $\frac{\Delta(\Delta-1)}{2}$ by Fertin and Raspaud [148] if $\Delta \geq 5$. They also constructed an $O(n\Delta^2)$ algorithm to acyclically color any graph of maximum degree Δ . More specifically, they proved the following two results:

Theorem 45 *There exists an $O(n\Delta^2)$ time algorithm to acyclically color any graph of order n and maximum degree Δ . This algorithm uses at most:*

1. $\frac{\Delta(\Delta-1)}{2} + 1$ colors when Δ is even and $\Delta \geq 4$.
2. $\frac{\Delta(\Delta-1)}{2}$ colors when Δ is odd and $\Delta \geq 5$.

Theorem 46 *There exists an $O(n\Delta^2)$ time algorithm to acyclically color any graph of order n and having $\Delta \geq 6$. This algorithm uses at most:*

1. $\frac{\Delta(\Delta-1)}{2}$ colors when Δ is even.
2. $\frac{\Delta(\Delta-1)}{2} - 1$ colors when Δ is odd.

By a deep study for the $\Delta = 5$ case, Fertin and Raspaud also showed that every graph G of maximum degree 5 has acyclic chromatic number at most 9, which has been recently improved to 8 by Hocquard and Montassier [193] and independently

by Yadava et al. [386]. Moreover, in [386], the authors gave a linear-time algorithm to achieve this bound as follows:

Theorem 47 *Every $\Delta = 5$ graph G with n vertices can be acyclically colored using 8 colors in $O(n)$ time.*

In 2009, Varagani et al. [337] proved that every graph G with $\Delta = 6$ can be acyclically colored using at most 12 colors in $O(n)$ time, which improves the result of [148] by reducing 15–12.

3.1.4 Products of Graphs

Given two graphs G_1 and G_2 , the *Cartesian product* $G_1 \times G_2$ of G_1 and G_2 is defined to be the graph G with $V(G) = V(G_1) \times V(G_2)$, and $E(G)$ contains the edge joining (u_1, u_2) and (v_1, v_2) if and only if either $u_1 = v_1$ and $(u_2, v_2) \in E(G_2)$ or $u_2 = v_2$ and $(u_1, v_1) \in E(G_1)$.

Let $d \in N$ and $(n_1, n_2, \dots, n_d) \in N^d$, with $n_i \geq 2$ for any $1 \leq i \leq d$. The d -dimensional grid of lengths n_1, n_2, \dots, n_d , denoted by $G_d(n_1, n_2, \dots, n_d)$, is the following graph:

- $V(G_d(n_1, n_2, \dots, n_d)) = [1, n_1] \times [1, n_2] \times \dots \times [1, n_d]$.
- $E(G_d(n_1, n_2, \dots, n_d)) = \{(u, v) | u = (u_1, u_2, \dots, u_d), v = (v_1, v_2, \dots, v_d), \text{ and there exists } i_0 \text{ such that } |u_{i_0} - v_{i_0}| = 1\}$.

Fertin, Godard, and Raspaud [147] discussed the acyclic coloring of d -dimensional grids $G_d(n_1, n_2, \dots, n_d)$, with each $n_i \geq 2$. The following theorem gives one of their main results:

Theorem 48 *Let $(n_1, n_2, \dots, n_d) \in N^d$ with $n_i \geq 2$ for any $1 \leq i \leq d$. For any grid $G_d(n_1, n_2, \dots, n_d)$ of dimension d , $\chi_a(G_d(n_1, n_2, \dots, n_d)) \leq d$.*

Jamison et al. [211] considered the Cartesian product of trees by showing the following:

Theorem 49 *Let $G = T_1 \times T_2 \times \dots \times T_d$ be a product of trees T_1, T_2, \dots, T_d . Then*

$$\left\lceil \frac{d+3}{2} \right\rceil \leq \chi_a(G) \leq d+1.$$

For an arbitrary graph, Jamison and Matthews [209] gave a simple but very useful lower bound on the acyclic chromatic number.

Theorem 50 *For any graph G with n vertices and m edges, $\chi_a(G) > \frac{m}{n} + 1$.*

Proof Assume that φ is an acyclic coloring of G using k colors. Denote the color classes of φ by C_1, C_2, \dots, C_k , and set $|C_i| = n_i$. Since G has no bichromatic cycles, the subgraph of G induced by any two color classes C_i and C_j is a forest T_{ij} . Clearly, $|E(T_{ij})| \leq n_i + n_j - 1$. Thus,

$$\begin{aligned} m &= \sum_{1 \leq i < j \leq k} |E(T_{ij})| \leq \sum_{1 \leq i < j \leq k} n_i + n_j - 1 = (k-1) \sum_{i=1}^k n_i - \binom{k}{2} \\ &= (k-1)n - \binom{k}{2}. \end{aligned}$$

So $\frac{m}{n} \leq (k-1) - \frac{\binom{k}{2}}{n}$, and hence, $k \geq \frac{m}{n} + 1 + \frac{\binom{k}{2}}{n}$. Since $\frac{\binom{k}{2}}{n} > 0$, it follows that $\chi_a(G) > \frac{m}{n} + 1$. \square

The following corollary is an immediate consequence of [Theorem 50](#).

Corollary 3 *Let G_1, G_2, \dots, G_d be graphs. Then*

$$\chi_a(G_1 \times G_2 \times \dots \times G_d) > \sum_{i=1}^d \frac{|E(G_i)|}{|V(G_i)|} + 1.$$

By applying [Corollary 3](#), the following result is easy to obtain.

Corollary 4 *Let P_m be a path on m vertices and C_n be a cycle on n vertices. Then $\chi_a(P_m \times C_n) \geq 3$ and $\chi_a(C_m \times C_n) \geq 4$.*

It is easy to see that $P_m \times C_n$ is a cylinder graph and $C_m \times C_n$ is a toroidal graph. In fact, in [209], the acyclic chromatic number of these two types of graphs was completely characterized:

Theorem 51

$$\chi_a(C_m \times C_n) = \begin{cases} 5, & \text{if } m = n = 3; \\ 4, & \text{otherwise.} \end{cases}$$

Theorem 52

$$\chi_a(P_m \times C_n) = \begin{cases} 4, & \text{if } n = 4; \\ 3, & \text{otherwise.} \end{cases}$$

Let $H(s_1, s_2, \dots, s_t)$ denote the Hamming graph, defined by $K_{s_1} \times K_{s_2} \times \dots \times K_{s_t}$ with the dimension t , and assume $2 \leq s_1 \leq s_2 \leq \dots \leq s_t$. For simplicity, write $\chi_a(s_1, s_2, \dots, s_t)$ for $\chi_a(H(s_1, s_2, \dots, s_t))$. Jamison and Matthew [210] considered the acyclic coloring of Hamming graphs by showing the following:

Theorem 53 *Suppose that $t \geq 3$ and define $B(s_1, s_2, \dots, s_t) = \sum_{i < j} s_i s_j - (t-2)(\sum_{i=1}^t s_i) + \frac{t(t-3)}{2}$. Then*

$$\chi_a(s_1, s_2, \dots, s_t) \leq B(s_1, s_2, \dots, s_t) \leq \binom{t}{2} s_t^2.$$

Moreover,

$$\chi_a(s, s, \dots, s) \leq \binom{t}{2} s^2 - t(t-2)(s - \frac{1}{2})$$

provided $t \geq 3$.

In the same paper [210], Jamison and Matthew investigated the t -dimensional hypercube, defined by $Q_t = K_2 \times K_2 \times \dots \times K_2$. Determining the acyclic chromatic

number of the hypercube is mentioned as an open problem in [147], where it is shown that $\lfloor \frac{t}{2} \rfloor + 2 \leq \chi_a(Q_t) \leq t + 1$ (see Theorem 4 in [147] or Theorem 2.1 in [211]). The authors in [147, 211] observed that the exact value may be equal to the lower bound. It was shown in [210] that this is indeed the case if $t + 3$ is a Fermat prime. In addition, they obtained an improved upper bound in a number of other cases.

Theorem 54 Assume that $a \leq b$ and $1 \leq c, d \leq 4$. Then

$$\chi_a(Q_{2^a+2^b-c-d}) = \begin{cases} 2^b, & \text{if } a < b; \\ \beta(2^b), & \text{if } a = b. \end{cases}$$

Next, by [Theorem 54](#), the exact acyclic chromatic number of hypercubes of dimensions related to the Fermat primes can be obtained.

Corollary 5 The acyclic chromatic number of the hypercube of dimension $t = 2^{r+1} - 2$ satisfies

$$2^r + 1 \leq \chi_a(Q_{2^{r+1}-2}) \leq NP(2^r).$$

In particular, if $2^r + 1$ is a Fermat prime, then $\chi_a(Q_{2^{r+1}-2}) = 2^r + 1$.

More recently, Joshi and Kothapalli [215] investigated the acyclic coloring of grid-like graphs. A graph is *chordal* if each of its cycles of four or more vertices has a chord, which is an edge joining two vertices that are not adjacent in the cycle. An equivalent definition is that any chordless cycles have at most three vertices. Gebremedhin et al. [162] showed that every coloring of a chordal graph is also an acyclic coloring. Since recognizing and optimally coloring chordal graphs can be done in linear time, this result immediately implies a linear-time algorithm for the acyclic coloring problem on chordal graphs.

3.2 Acyclic List Coloring

Given an assignment $L = \{L(v)|v \in V(G)\}$ of colors to the vertices of a graph G , G is *acyclically L -colorable* if there is an acyclic coloring π of the vertices such that $\pi(v) \in L(v)$ for every vertex v . If G is acyclically L -colorable for any list assignment L with $|L(v)| \geq k$ for all $v \in V$, then G is *acyclically k -choosable* or *acyclically list k -colorable*. The *acyclic list chromatic number* of G , denoted by $\chi_a^l(G)$, is the smallest integer k such that G is acyclically k -choosable.

3.2.1 Acyclic List 5-Coloring

In 2002, Borodin et al. [62] first investigated the acyclic list coloring of planar graphs. They proved that every planar graph is acyclically 7-choosable and put forward the following challenging conjecture:

Conjecture 9 *Every planar graph is acyclically 5-choosable.*

This conjecture has attracted much attention in recent years. If [Conjecture 9](#) is true, then it would strengthen the Borodin's Acyclic 5-Color Theorem [55] and the Thomassen's 5-Choosable Theorem [328] about planar graphs. Evidently, it is very hard to settle this conjecture. As yet, it has been verified only for several restricted classes of planar graphs: those of girth at least 5 [279], without 4- and 5-cycles or without 4- and 6-cycles [284], without 4-cycles and without triangles at distance less than 3 [113], and with neither 4-cycles nor chordal 6-cycles [400]. In particular, in [67], Borodin and Ivanova proved that a planar graph G is acyclically 5-choosable if G does not contain an i -cycle adjacent to a j -cycle where $3 \leq j \leq 5$ if $i = 3$ and $4 \leq j \leq 6$ if $i = 4$. This result absorbs most of the previous work in this direction.

Wang and Chen [355] proved that every planar graph without 4-cycles is acyclically 6-choosable. To attack [Conjecture 9](#), in [113], the authors proposed the following weaker version of [Conjecture 9](#):

Conjecture 10 *Every planar graph without 4-cycles is acyclically 5-choosable.*

Recently, Chen and Raspaud [106] proved that every planar graph with neither 4-cycles nor intersecting triangles is acyclically 5-choosable. This result partially confirms [Conjecture 10](#). Lately, Conjecture 10 has been verified by Borodin and Ivanova [68].

3.2.2 Acyclic List 4-Coloring

Some sufficient conditions for a planar graph to be acyclically 4-choosable (or colorable) have been obtained in the past decade. It was proved in [59] that $\chi_a(G) \leq 4$ if G contains no $\{4, 5\}$ -cycles. Moreover, $\chi_a^l(G) \leq 4$ was confirmed in each of the following cases: (i) if $g(G) \geq 5$ [278], which extends two results in [78, 279]; (ii) if G has no $\{4, 5, 6\}$ -cycles, or without $\{4, 5, 7\}$ -cycles, or without $\{4, 5\}$ -cycles and intersecting 3-cycles [282], or without $\{4, 7, 8\}$ -cycles [107], or with neither 4-cycles nor 6-cycles adjacent to a triangle [74]. In addition, in [282], Montassier et al. proposed the following conjecture which is still unsettled:

Conjecture 11 (Domaine de la Solitude 2000 Conjecture) *Every planar graph without 4-cycles is acyclically 4-choosable.*

Recently, Chen and Raspaud [111], independently Borodin and Ivanova [71], proved that every planar graph without 4- and 5-cycle is acyclically 4-choosable. This result is a new approach to [Conjecture 11](#) and is the best possible in the sense that there are planar graphs without 4- and 5-cycles that are not 3-choosable (see [344]).

3.2.3 Acyclic List 3-Coloring

In 1999, Borodin et al. [78] proved that every planar graph with girth at least 7 is acyclically 3-colorable. Ten years later, Borodin et al. [61] further generalized this result to the list-coloring version by showing that every planar graph with girth

at least 7 is acyclically 3-choosable. Recently, the similar question of Erdős was studied in the acyclic choosability case:

Question 3 *What is the smallest integer i such that every planar graph without cycles of lengths 4 to i is acyclically 3-choosable?*

Borodin [58], and independently Hocquard and Montassier [192], proved that such i exists and $i \leq 12$.

Let $d_\Delta(G)$ denote the minimum distance between triangles in a given graph G . Based on this concept, Hocquard et al. [194] established some new sufficient conditions for a planar graph to be acyclically 3-choosable. More precisely, they proved the following:

Theorem 55 *If a planar graph G satisfies one of the following conditions, then G is acyclically 3-choosable:*

1. *No cycles of length 4 to 10, and $d_\Delta(G) \geq 2$.*
2. *No cycles of length 4 to 9, and $d_\Delta(G) \geq 3$.*
3. *No cycles of length 4 to 8, and $d_\Delta(G) \geq 5$.*
4. *No cycles of length 4 to 7, and $d_\Delta(G) \geq 7$.*

The following three interesting questions were supplied in [194]:

Question 4 *Is every planar graph without 4-, 5-, 6-cycles acyclically 3-choosable?*

Question 5 *Is every planar graph with girth 6 acyclically 3-choosable?*

Question 6 *Does there exist a constant d such that every planar graph G without 4-, 5-, and 6-cycles and $d_\Delta(G) \geq d$ is acyclically 3-choosable?*

3.3 Acyclic Edge Coloring

An *acyclic edge-coloring* of a graph is a proper edge-coloring without bichromatic cycles. The *acyclic chromatic index* or *acyclic edge chromatic number* of a graph G , denoted by $\chi'_a(G)$, is the least integer k such that G admits an acyclic edge-coloring using k colors. The concept of acyclic edge-coloring is a very natural extension of acyclic vertex coloring, which was first investigated by Fiamčík [151].

Since every acyclic edge-coloring is proper, so $\chi'_a(G) \geq \chi'(G) \geq \Delta$ for any graph G . On the other hand, in 2001, Alon et al. [19] posed the following conjecture:

Conjecture 12 *For any graph G , $\chi'_a(G) \leq \Delta + 2$.*

This conjecture would be tight, if true, because there exist graphs G which need at least $\Delta + 2$ colors to form an acyclic edge-coloring. However, the only known so far such graphs are subgraphs of K_{2n} before 2001. Basing on this fact, Alon et al. [19] proposed another conjecture:

Conjecture 13 *If G is a Δ -regular graph, then $\chi'_a(G) = \Delta + 1$ unless $G = K_{2n}$.*

Using probabilistic method, Alon et al.[17] proved that $\chi'_a(G) \leq 64\Delta$ for any graph G . Molloy and Reed [273] improved this bound to that $\chi'_a(G) \leq 16\Delta$, which is the better known upper bound for $\chi'_a(G)$ in terms of Δ . By adding girth restriction, Muthu et al. [285, 286] proved that $\chi'_a(G) \leq 4.52\Delta$ if G is a graph with girth at least 220 and that if G has girth at least 9, then $\chi'_a(G) \leq 6\Delta$, whose proofs were based on probabilistic arguments as well as the following:

Theorem 56 *There are absolute constants $c_1, c_2 > 0$ such that, for any graph G with $g \geq c_1 \log \Delta$, $\chi'_a(G) \leq \Delta + 1 + \lceil c_2(\frac{\Delta \log \Delta}{g}) \rceil$.*

Though the best known upper bound for general case is far from the conjectured $\Delta + 2$, Conjecture 12 has been shown to be true for some special classes of graphs.

It is easy to inspect that $\chi'_a(G) \leq 3$ if $\Delta = 2$. Suppose that G is a subcubic graph, that is, $\Delta \leq 3$. Then the linear graph $L(G)$ of G is of maximum degree at most 4, thus is cyclically 5-colorable [90]. Since $\chi'_a(G) = \chi_a(L(G))$, then $\chi'_a(G) \leq 5 = \Delta + 2$. So Conjecture 12 is true for all subcubic graphs. Skurattankulchai [321] provided a polynomial-time algorithm to color a subcubic graph using five colors.

Theorem 57 *Every subcubic graph G has $\chi'_a(G) \leq 5$, and the edges of G can be acyclically colored using five colors in $O(n)$ time.*

Fiamčík [152], and independently Basavaraju and Chandran [35], showed the following result:

Theorem 58 *Every graph G with $\Delta = 3$ is acyclically edge 4-colorable unless either $G = K_4$ or $G = K_{3,3}$.*

Theorem 59 *If G is a non-regular connected graph with $\Delta = 3$, then $\chi'_a(G) \leq 4$.*

Alon et al. showed in [19] that Conjecture 12 holds for almost all Δ -regular graphs. This has been improved by Něsetřil and Wormald [292] by showing that $\chi'_a(G) \leq \Delta + 1$ for a random Δ -regular graph G . Moreover, if the girth of a graph G is sufficiently large, say $g(G) \geq c\Delta \log \Delta$, then $\chi'_a(G) \leq \Delta + 2$ [19]. More recently, Basavaraju and Chandran [37] proved that Conjecture 12 also holds for complete bipartite graphs $K_{p,p}$, where p is an old prime. More specifically, they proved the following theorems:

Theorem 60 $\chi'_a(K_{p,p}) \leq p + 2$, when p is an odd prime.

Theorem 61 *For a prime $p > 2$, if G is a graph obtained by removing just one edge from $K_{p,p}$, then $\chi'_a(G) = p + 1$.*

Notice that determining $\chi'_a(G)$ is a hard problem from both theoretical and algorithmic points of view. Even for the simple and highly structured classes, for instance complete graphs, the value of $\chi'_a(G)$ is still not determined exactly. It has also been shown by Alon and Zaks [21] that determining whether $\chi'_a(G) \leq 3$ is NP-complete for an arbitrary graph G . Meanwhile, the same authors proved the following:

Theorem 62 *The edges of a graph G can be colored acyclically in polynomial time using $\Delta + 2$ colors, provided that $g(G) \geq cd^3$, where c is an appropriate absolute constant.*

The rest part of this section is concerned with the acyclic edge-coloring of some special graphs, including planar graphs and 2-degenerate graphs.

3.3.1 Planar Graphs

The acyclic edge-coloring of planar graphs has been deeply studied in recent years. In 2008, Fiedorowicz et al. [154] gave a general upper bound $2\Delta + 29$ for the class of planar graphs. This result was gradually improved to $\max\{2\Delta - 2, \Delta + 22\}$ by Hou et al. [202] and then to $\Delta + 25$ by Cohen et al. [115]. Today, the better known upper bound for the class of planar graphs is $\Delta + 12$, due to Basavaraju and Chandran [39].

In 2008, Fiedorowicz et al. [154] considered triangle-free planar graphs by showing the following bound:

Theorem 63 *Every planar graph G without triangles has $\chi'_a(G) \leq \Delta + 6$.*

The upper bound $\Delta + 6$ has been reduced to $\Delta + 5$ by Dong and Xu [131] and then to $\Delta + 3$ by Basavaraju and Chandran [40]. More recently, Shu and Wang [318] obtained a better bound:

Theorem 64 *Every planar graph G without triangles has $\chi'_a(G) \leq \Delta + 2$.*

The value $\Delta + 2$ is the best known upper bound for the class of triangle-free planar graphs. A natural and interesting problem arises to consider the acyclic chromatic index of planar graphs with girth 5. In 2009, Hou et al. [202], and independently Borowiecki and Fiedorowicz [80], proved that if G is a planar graph of girth at least 5, then $\chi'_a(G) \leq \Delta + 2$. Obviously, this result is implied in **Theorem 64**. At the nearly same time, Yu et al. [390] improved this result by restricting the value of maximum degree:

Theorem 65 *If G is a planar graph with girth $g \geq 5$ and $\Delta \geq 11$, then $\chi'_a(G) \leq \Delta + 1$.*

Other sufficient conditions for planar graphs without some specific cycles to be acyclically edge $(\Delta + 1)$ -colorable or $(\Delta + 2)$ -colorable have been presented by some authors. At first, Sun and Wu [323] showed the following:

Theorem 66 *Every planar graph G has $\chi'_a(G) \leq \Delta + 2$ if one of the following conditions holds:*

1. *G contains no i -cycles for all $4 \leq i \leq 8$.*
2. *G contains no 4- and 5-cycles and no intersecting triangles.*

Next, Borowiecki and Fiedorowicz [80] obtained following result, which can be regarded, to some extent, as an improvement to **Theorem 66**.

Theorem 67 *If G is a planar graph without 4-, 6-, 8-, and 9-cycles, then $\chi'_a(G) \leq \Delta + 2$.*

In his Ph.D. Dissertation, Hou [201] also discussed the upper bound $\Delta + 2$ on the acyclic chromatic index of planar graphs without cycles of special lengths. He proved the following result:

Theorem 68 *If G is a planar graph without 4- and 5-cycles, or without 4- and 6-cycles, then $\chi'_a(G) \leq \Delta + 2$.*

Clearly, **Theorem 68** improved greatly **Theorem 66**, which requires only the lack of cycles of two lengths. More naturally, one needs to seek which planar graphs can be acyclically edge colored using $\Delta + 1$ colors. Gao and Yu [158] first obtained the following result:

Theorem 69 *A planar graph G has $\chi'_a(G) \leq \Delta + 1$, if one of the following conditions holds:*

1. $\Delta \geq 10$ and no cycles of lengths 4–11.
2. $\Delta \geq 9$ and no cycles of lengths 4–9.

In the study of planar graphs with acyclic chromatic index attaining lower bound Δ , Cohen et al. [115] made the following conjecture:

Conjecture 14 *There exists an integer Δ_0 such that every planar graph G with maximum degree $\Delta \geq \Delta_0$ has $\chi'_a(G) = \Delta$.*

Indeed, Cohen et al. [115] confirmed **Conjecture 14** for outerplanar graphs with $\Delta_0 = 9$ and for planar graphs G with $\Delta_0 = 19$ and $g(G) \geq 5$. To obtain this, they proved a stronger result by considering maximum average degree.

The *maximum average degree* $\text{mad}(G)$ of a graph G is defined by

$$\text{mad}(G) = \max_{H \subseteq G} \{2|E(H)|/|V(H)|\}.$$

The following is a folklore fact:

Proposition 2 *Let G be a planar graph of girth g . Then*

$$\text{mad}(G) < \frac{2g}{g-2}.$$

The maximum average degree of graphs is an important measure to determine the sparseness of an arbitrary graph (not necessarily planar). On the other hand, it is known that the maximum average degree of a graph can be computed in polynomial time by using the Matroid Partitioning Algorithm due to Edmonds [138].

Theorem 70 *If G is a planar graph with $\Delta \geq 19$ and $\text{mad}(G) < \frac{10}{3}$, then $\chi'_a(G) \leq \Delta$.*

Actually, Cohen et al. [115] established a more general result as follows:

Theorem 71 For any $\epsilon > 0$, there exists an integer Δ_ϵ such that if $\Delta \geq \Delta_\epsilon$ and $\text{mad}(G) \leq 4 - \epsilon$, then $\chi'_a(G) = \Delta$.

In view of the previous arguments, two parameters, that is, girth g and maximum degree Δ , play important roles in deciding an acyclically edge Δ -colorable planar graphs. Different combinations of these two parameters will yield different results on this topic. In 2009, Hou et al. [202] and Yu et al. [390] studied this problem and showed the following:

Theorem 72 Let G be a planar graph with $\Delta \geq 3$ and girth $g \geq 16$. Then $\chi'_a(G) = \Delta$.

Theorem 73 Let G be a planar graph with girth g . Then $\chi'_a(G) = \Delta$ if there is a pair $(k, m) \in \{(4, 12), (5, 10), (6, 8), (12, 7)\}$ such that G satisfies $\Delta \geq k$ and $g \geq m$.

Later, Dong and Xu [131], Wang et al. [317, 365], and Hudák et al. [206] obtained several similar results, respectively. Their results are summarized as follows:

Theorem 74 Let G be a planar graph with girth g . Then $\chi'_a(G) = \Delta$ if there is a pair $(k, m) \in \{(3, 14), (4, 10), (5, 9), (6, 8), (8, 7)\}$ such that G satisfies $\Delta \geq k$ and $g \geq m$.

Theorem 75 Let G be a planar graph with girth g . Then $\chi'_a(G) = \Delta$ if there is a pair $(k, m) \in \{(3, 11), (4, 8), (5, 7), (8, 6)\}$ such that G satisfies $\Delta \geq k$ and $g \geq m$.

Theorem 76 Let G be a planar graph with girth $g \geq 5$ and $\Delta \geq 12$. Then $\chi'_a(G) = \Delta$.

Theorem 77 Let G be a planar graph with girth g . Then $\chi'_a(G) = \Delta$ if there is a pair $(k, m) \in \{(3, 12), (4, 8), (5, 7), (6, 6), (10, 5)\}$ such that G satisfies $\Delta \geq k$ and $g \geq m$.

3.3.2 2-Degenerate Graphs

Recall that a graph G is k -degenerate if any induced subgraph H of G has a vertex of degree at most k in H . For example, planar graphs are 5-degenerate and forests are 1-degenerate. The class of 2-degenerate graphs includes series-parallel graphs, partial 2-trees, outerplanar graphs, non-regular subcubic graphs, planar graphs with girth, at least 6 and circle graphs with girth at least 5 as subclasses.

The earliest result on the acyclic edge-coloring of 2-degenerate graphs was obtained by Card and Roditty [94] who proved that $\chi'_a(G) \leq \Delta + k - 1$, where k is the maximum edge connectivity, defined by $k = \max_{u,v \in V(G)} \{\lambda(u, v)\}$, where $\lambda(u, v)$ is the edge connectivity of a pair of vertices u, v . Note that here k can be as large as Δ .

An *outerplanar graph* is a graph which can be embedded in the plane in such a way that every vertex lies on the boundary of the outer face. Muthu et al. [287] proved that [Conjecture 12](#) is true for outerplanar graphs. In fact, they gave a better bound, $\Delta + 1$, as follows:

Theorem 78 Every outerplanar graph G with n vertices is acyclically edge $(\Delta+1)$ -colorable. Moreover, an acyclic edge $(\Delta+1)$ -coloring can be obtained in $O(n \log \Delta)$ time.

In addition, they posed the following interesting question:

Question 7 Is every 2-degenerate graph acyclically edge $(\Delta+2)$ -colorable?

Let H_{2n} , $n \geq 2$ and denote the graph obtained by adding $n-1$ chords $x_2x_{2n}, x_3x_{2n-1}, \dots, x_nx_{n+2}$ to a $2n$ -cycle $x_1x_2 \dots x_{2n}x_1$. Let Q denote the graph obtained by adding four edges y_2y_7, y_2y_6, y_3y_5 , and y_3y_6 to a 7-cycle $y_1y_2 \dots y_7y_1$.

Hou et al. [203], and independently Wang and Wang [367], determined the acyclic edge chromatic number of outerplanar graphs G with $\Delta \neq 4$.

Theorem 79 Let G be an outerplanar graph with $\Delta \geq 3$. Then

1. $\chi'_a(G) = \Delta$ if $\Delta \geq 5$.
2. If $\Delta = 3$, then $3 \leq \chi'_a(G) \leq 4$; and $\chi'_a(G) = 4$ if and only if G contains a 2-connected block isomorphic to H_{2n} for some $n \geq 2$.

For the $\Delta = 4$ case, Hou et al. [203] also provided a characterization, which says that if G is an outerplanar graph with $\Delta = 4$, then $4 \leq \chi'_a(G) \leq 5$ and $\chi'_a(G) = 5$ if and only if G contains a subgraph isomorphic to the graph Q defined above. However, their characterization is incorrect, as Wang and Wang [367] constructed infinitely many outerplanar graphs G with $\Delta = 4$ and $\chi'_a(G) = 5$, which do not contain any configuration isomorphic to Q .

Recall that a graph G has a graph H as a *minor* if H can be obtained from a subgraph of G by contracting edges, and G is called H -*minor-free* if G does not have H as a minor. It is well-known that a graph G is an outerplanar graph if and only if G is K_4 -minor-free and $K_{2,3}$ -minor-free. A graph G is K_4 -minor-free if and only if each block of G is a series-parallel graph. Thus, the class of K_4 -minor-free graphs is a subclass of planar graphs that includes both outerplanar graphs and series-parallel graphs.

Hou et al. [202] and independently Muthu et al. [288] proved that $\Delta + 1$ colors are sufficient for an acyclic edge-coloring of a series-parallel graph G .

Theorem 80 Every series-parallel graph G is acyclically edge $(\Delta+1)$ -colorable.

Theorem 80 extends the result in [287], since every outerplanar graph is a series-parallel graph but the converse may be not true.

A *2-tree* is a graph obtained from the triangle K_3 , by a sequence of operations of adding a new vertex adjacent to the endpoints of an existing edge in the graph, and a *partial 2-tree* is any subgraph of a 2-tree. It is easy to observe that 2-trees are planar and 2-degenerate. Muthu et al. [288] confirmed Conjecture 12 for 2-trees and partial 2-trees.

Theorem 81 If G is a 2-tree or a partial 2-tree, then $\chi'_a(G) \leq \Delta + 1$.

Recently, Wang and Shu [366] further studied the acyclic chromatic indices of K_4 -minor-free graphs. They proved the following:

Theorem 82 *Let G be a K_4 -minor-free graph with $\Delta = 3$ or $\Delta \geq 5$. Then $\Delta \leq \chi'_a(G) \leq \Delta + 1$; and $\chi'_a(G) = \Delta + 1$ if and only if $\Delta = 3$ and G contains a block isomorphic to H_{2n} for some $n \geq 2$.*

Note that [Theorem 82](#) improves and extends the previous [Theorems 79](#) and [80](#). However, it remains open to determine the acyclic chromatic index of a K_4 -minor-free graph with $\Delta = 4$, even for an outerplanar graph with $\Delta = 4$. Solving this question seems to be very interesting.

As it is pointed out before, another two important subclasses of 2-degenerate graphs are planar graphs of girth 6 and circle graphs of girth 5. Nothing much is known about the acyclic edge chromatic number of circle graphs of girth 5. For planar graphs of girth at least 6, some existing conclusions were independently presented in [80, 202, 390].

More recently, Basavaraju and Chandran [38] answered positively [Question 7](#). In fact, they got a stronger result:

Theorem 83 *Every 2-degenerate graph is acyclically edge $(\Delta + 1)$ -colorable.*

Note that the upper bound $\Delta + 1$ in [Theorem 83](#) is sharp, since there exist infinitely many 2-degenerate graphs which require $\Delta + 1$ colors to admit an acyclic edge-coloring (e.g., odd cycles and non-regular subcubic graphs). Moreover, the constructive proof of [Theorem 83](#) yields actually an efficient polynomial-time algorithm with the complexity $O(\Delta|G|^2)$.

3.3.3 Graphs with a Fixed Maximum Degree

Determining $\chi'_a(G)$ for general graphs is a quite difficult problem, only few partial results have been known. For a graph G with $\Delta = 1$, it is trivial that $\chi'_a(G) = 1$. If G is a graph with $\Delta = 2$, then $\chi'_a(G) = 2$ when G is a linear forest and $\chi'_a(G) = 3$ when G contains a cycle. For the general upper bound, Fiamčík [151] proved the following:

Theorem 84 *Let G be a graph. Then $\chi'_a(G) \leq \Delta(\Delta - 1) + 1$; and $\chi'_a(G) \leq 9$ if $\Delta = 4$.*

By considering the relationship between the number of vertices and edges of a graph G , Guldan [172] obtained some helpful lower bounds:

Theorem 85 *Let k be a positive integer and let G be a graph with n vertices and m edges.*

1. *If $n = 2k$ and $m > (k - 1)(\Delta + 1) + 1$, then $\chi'_a(G) \geq \Delta + 2$.*
2. *If $n = 2k + 1$ and $m > k\Delta$, or if $n = 2k$ and $m > (k - 1)\Delta + 1$, then $\chi'_a(G) \geq \Delta + 1$.*

It follows immediately from [Theorem 85](#) that there exist infinitely many graphs with $\chi'_a(G) \geq \Delta + 2$. This disproves a conjecture introduced by Fiamčík at the

Czechoslovak Conference Graph Theory 1984, which states that if a connected graph G has maximum degree Δ and if $G \notin \{K_{\Delta+1}, K_{\Delta,\Delta}, (K_{\Delta+2} - F_1)\}$, then $\chi'_a(G) \leq \Delta + 1$.

The following useful fact is a direct consequence of [Theorem 85](#):

Corollary 6 *Let G be a Δ -regular graph with $\Delta > 1$. Then $\chi'_a(G) \geq \Delta + 1$.*

Alon et al. [19] suggested a possibility that complete graphs of even order are the only regular graphs which require $\Delta + 2$ colors to be acyclically edge colored. Něsetřil and Wormald [292] supported the statement by showing that the acyclic edge chromatic number of a random d -regular graph is asymptotically almost surely equal to $d + 1$ (when $d \geq 2$). However, Basavaraju et al. [41] showed that this is not true in general case by using a simple counting argument. More specifically, they proved the following three theorems:

Theorem 86 *Let G be a d -regular graph with $2n$ vertices and $d > n$. Then $\chi'_a(G) \geq d + 2$.*

Theorem 87 *For any d and n such that dn is even and $d \geq 5$, $n \geq 2d + 3$, there exists a connected d -regular graph that requires $d + 2$ colors to be acyclically edge colored.*

Theorem 88 $\chi'_a(K_{n,n}) \geq n + 2$, when n is odd.

In 2009, Basavaraju and Chandran [36] nearly settled [Conjecture 12](#) for connected graphs G with $\Delta = 4$ with an additional restriction:

Theorem 89 *Let G be a connected graph with n vertices and m edges. If $\Delta = 4$ and $m \leq 2n - 1$, then $\chi'_a(G) \leq 6$.*

Note that a graph G with order n and maximum degree 4 has at most $2n$ edges. Thus, [Theorem 89](#) implies that, to confirm [Conjecture 12](#) for $\Delta = 4$, it suffices to consider the left case where $m = 2n$.

Fiedorowicz [153] studied the acyclic edge-coloring of graphs with the number of edges linearly bounded by the number of vertices and gave the following theorem.

Theorem 90 *Let $t \geq 2$ be an integer. If G satisfies the condition that $|E(G')| \leq t|V(G')| - 1$ for each subgraph $G' \subseteq G$, then $\chi'_a(G) \leq (t - 1)\Delta + p$, where $p = 2t^3 - 3t + 2$.*

Based on this result, the authors gave a polynomial-time algorithm to compute such a coloring. Note that the class of graphs considered in [Theorem 90](#) is quite large, for example, including all t -degenerate graphs.

3.4 Related Coloring Problems

This section is devoted to introducing some concepts related to acyclic coloring and acyclic edge coloring, including acyclic improper colorings, star coloring, frugal coloring, and linear coloring.

3.4.1 Acyclic Improper Coloring

Boiron et al. [54] extended in a natural way the notion of acyclic coloring to the acyclic improper coloring as follows. An *acyclic t -improper k -coloring*, or simply an acyclic $(k, t)^*$ -coloring, of G is a $(k, t)^*$ -coloring which is acyclic, that is, G contains no bichromatic cycles. In other words, an acyclic t -improper coloring is an acyclic coloring such that each color class induces a graph with maximum degree at most t . The t -improper acyclic chromatic number $\chi_a^t(G)$ is the smallest k for which there exists an acyclic t -improper coloring.

The main motivation in the study of acyclic improper coloring is the link with oriented coloring. It was proved in [54] that every subcubic graph is acyclically $(3, 1)^*$ -colorable and conjectured that every subcubic graph is acyclically $(2, 2)^*$ -colorable. Moreover, they constructed subcubic graphs which are not acyclically $(2, 1)^*$ -colorable. They also proved that every outerplanar graph is acyclically $(2, 5)^*$ -colorable and constructed outerplanar graphs which are not acyclically $(2, 4)^*$ -colorable. Furthermore, they proved that for every $k \geq 0$, there exist planar graphs which are not acyclically $(4, k)^*$ -colorable.

Let $\chi_a^t(d) = \max\{\chi_a^t(G) : \Delta(G) \leq d\}$. Clearly, for any d , $\chi_a^0(d) \geq \chi_a^1(d) \geq \dots \geq \chi_a^d(d) \geq 1$, and it is easily seen that $\chi_a^t(d) = \Omega((d/t)^{4/3}/(\ln d)^{1/3})$.

Esperet and Pinlou [143] first introduced and studied the acyclic improper choosability of graphs. It is natural to define the acyclic improper choosability of a graph. A graph G is *acyclically t -improper L -colorable* if for a given list assignment $L = \{L(v) : v \in V(G)\}$, there exists an acyclic t -improper coloring f such that $f(v) \in L(v)$ for every v . If G is acyclically t -improper L -colorable for any list assignment L with $|L(v)| \geq l$ for every v , then G is called *acyclically t -improper l -choosable* or simply acyclically $(l, t)^*$ -choosable. The acyclic t -improper list chromatic number of G is therefore defined as the smallest integer l such that G is acyclically $(l, t)^*$ -choosable.

In [143], Esperet and Pinlou extended two results in [54] by showing the following:

Theorem 91 *Every subcubic graph is acyclically $(3, 1)^*$ -choosable.*

Theorem 92 *Every outerplanar graph is acyclically $(2, 5)^*$ -choosable.*

Furthermore, in [142, 143], the authors also proved that

Theorem 93 *There exists a constant $\eta > 0$ such that if $t < \eta(\frac{n}{\log n})^{3/4}$, then $\chi_a^t(d) = \Omega\left(\frac{d^{3/4}}{(\log d)^{1/3}}\right)$.*

Recently, Addario-Berry et al. [5] studied how $\chi_a^t(d)$ varies as a function of t and d and gave the following three theorems:

Theorem 94 *If $t \leq d - 10\sqrt{d \ln d}$, then $\chi_a^t(d) = \Omega((d-t)^{4/3}/(\ln d)^{1/3})$.*

In particular, if $t = (1 - \varepsilon)d$ for any fixed constant ε , $0 < \varepsilon < 1$, then they obtained the same asymptotic lower bound as Alon et al. [17].

Theorem 95 $\chi_a^t(d) = O(d \ln d + (d - t)d)$.

Theorem 96 $\chi_a^{d-1}(d) = \Omega(d^{2/3})$.

3.4.2 Star Coloring

While Grünbaum [171] introduced the acyclic coloring notion, he also noted that the condition that the union of any two color classes inducing a forest can be generalized to other bipartite graphs. Among other problems, he suggested requiring that the union of any pair of color classes induces a star forest, namely, a proper coloring avoiding 2-colored paths with four vertices. Precisely, a proper coloring of the vertices of a graph G is called a *star coloring* if the union of any two color classes induces a star forest. In other words, no path on four vertices is 2-colored. The *star chromatic number* of G , denoted by $\chi_s(G)$, is the smallest integer k for which G admits a star coloring with k colors.

Grünbaum noted (without proof) that bounding the acyclic chromatic number bounds the star chromatic number. Here the result is stated, a proof of which was given by Fertin et al. [149].

Theorem 97 If $\chi_a(G) = k$, then $\chi_s(G) \leq k \cdot 2^{k-1}$.

Both acyclic and star colorings have applications in the field of combinatorial scientific computing, where they model two different schemes for the evaluation of sparse Hessian matrices. The general idea behind the use of coloring in computing derivative matrices is the identification of entities that are essentially independent and thus may be computed concurrently; see [161] for a survey.

Let \mathcal{P} denote the family of planar graphs. By the Borodin's acyclically 5-colorable theorem and Theorem 97, it is easy to obtain that $\chi_s(\mathcal{P}) \leq 80$. In 2003, Nešetřil and Ossona de Mendez [291] made a big step by showing that $\chi_s(\mathcal{P}) \leq 30$. This result also implies that every triangle-free planar graph can be star colored using 18 colors, whereas Kierstead, Kündgen, and Timmons [226] gave an example of a bipartite planar graph that requires 8 colors to star color. One year later, Albertson et al. [13] further reduced the upper bound 30 to 20 and gave a lower bound by showing an example of a planar graph H using at least 10 colors to star color. It therefore follows that $10 \leq \chi_s(\mathcal{P}) \leq 20$. Moreover, they made an improvement to Theorem 97 as follows:

Theorem 98 If $\chi_a(G) = k$, then $\chi_s(G) \leq k(2k - 1)$.

For graphs with smaller maximum average degree, Jensen and Toft [213] showed that there is a polynomial-time algorithm for determining the maximum average degree of a given graph G . Moreover, as stated before, there exists a close relation between the maximum average degree and girth of a graph G .

In [85], Bu et al. proved the following result:

Theorem 99 Let G be a graph with girth g :

1. If $\text{mad}(G) < \frac{26}{11}$, then $\chi_s(G) \leq 4$.
2. If $\text{mad}(G) < \frac{18}{7}$ and $g \geq 6$, then $\chi_s(G) \leq 5$.
3. If $\text{mad}(G) < \frac{8}{3}$ and $g \geq 6$, then $\chi_s(G) \leq 6$.

From [Proposition 2](#) and [Theorem 99](#), it is immediate to derive that for a planar graph G with girth g , $\chi_s(G) \leq 4$ if $g \geq 13$, $\chi_s(G) \leq 5$ if $g \geq 9$, and $\chi_s(G) \leq 6$ if $g \geq 8$.

Other star coloring results related to planar graphs were provided in Timmons Master's Thesis [[334](#)].

A graph G is *star L -colorable* if for a given list assignment L , there is a star coloring c such that $c(v) \in L(v)$. If G is star L -colorable for any list assignment L with $|L(v)| \geq k$ for all $v \in V(G)$, then G is star k -choosable. The *star list chromatic number*, denoted by $\chi_s^l(G)$, of G is the smallest integer k such that G is star k -choosable.

The star list coloring of graphs has been recently investigated by many authors. Kierstead et al. [[226](#)] showed that bipartite planar graphs are star 14-choosable. In [[241](#)], Kündgen and Timmons proved a theorem about the dependence between the maximum average degree of graphs and their star list chromatic numbers. Their main result is the following:

Theorem 100 *Let G be a graph:*

1. *If $\text{mad}(G) < \frac{8}{3}$, then $\chi_s^l(G) \leq 6$.*
2. *If $\text{mad}(G) < \frac{14}{5}$, then $\chi_s^l(G) \leq 7$.*
3. *If G is planar and with girth at least 6, then $\chi_s^l(G) \leq 8$.*

It should be noticed that the conclusion (1) in [Theorem 100](#) is stronger than the third conclusion in [Theorem 99](#). It can be deduced from [Proposition 2](#) and (1) and (2) in [Theorem 100](#) that $\chi_s^l(G) \leq 6$ if $g \geq 8$, and $\chi_s^l(G) \leq 7$ if $g \geq 7$, where G is a planar graph with girth g .

By the definition, every star coloring is an acyclic coloring but star coloring a graph typically requires more colors than acyclic coloring the same graph. Moreover, determining the minimum (list) chromatic number of many families of graphs is proved to be a challenging problem. This is indeed the case for families as simple as subcubic graphs.

In 2001, Fertin et al. [[149](#)] gave the exact value of the star chromatic number of different families of graphs such as trees, cycles, complete bipartite graphs, outerplanar graphs, and two-dimensional grids. In the same paper, they also gave bounds for the star chromatic number of other families of graphs, such as hypercubes, tori, d -dimensional grids, graphs with bounded treewidth, and planar graphs. One interesting result may be that every subcubic graph is star 9-colorable, while there exists a subcubic graph, that is, the Wagner graph (see [Fig. 3](#)), has star chromatic number at least 6. Recently, Chen et al. [[112](#)] showed that 6 colors are enough to star color any subcubic graphs. Moreover, Albertson et al. [[13](#)] proved that every subcubic graph is star 7-choosable.

[Table 1](#) describes the currently best known bounds for the star chromatic number of planar graphs with girth g . The best known bound is given along with the corresponding reference.

Table 1 Best known bounds on star chromatic number of planar graphs

Girth g	Best known bounds	
	Lower bound	Upper bound
3	10 [13]	20 [13]
4	8 [226]	18 [291]
5	6 [334]	16 [13]
6	5 [334]	8 [241]
7	5 [335]	7 [334]
8	4 [13]	6 [334] [85]
9–13	4 [13]	5 [335]
14 ⁺	4 [13]	4 [335]

By Table 1, planar graphs with girth 4 are star 18-colorable. Recently, Kierstead et al. [226] showed that bipartite planar graphs are star 14-choosable. Since the girth of bipartite planar graphs is at least 4, it seems to be interesting to study the following problem.

Problem 2 *Can the upper bound 18 on star chromatic number of planar graphs of girth at least 4 be improved to 14?*

Besides, Table 1 also shows that planar graphs of girth at least 14 can be star colored with 4 colors and there is a planar graph with girth 7 that requires 5 colors to star color. Finally, this section is ended with the following problem:

Problem 3 *What is the smallest integer k such that planar graphs of girth at least k is star 4-colorable?*

3.4.3 Linear Coloring

A coloring (not necessarily proper) of G is t -frugal if no color appears more than t times in any neighborhood of any vertex. The t -frugal chromatic number $\varphi^t(G)$ denotes the least number of colors needed in a t -frugal coloring that is not necessarily proper. For $t \geq 1$, the superscript t will denote that the coloring considered is t -frugal. That is, $\chi^t(G)$ will denote the least number of colors used in a proper t -frugal coloring. The notion of frugal coloring was introduced in 1997 by Hind et al. [190]. They investigated proper t -frugal colorings as a way to improve bounds related to the Total Coloring Conjecture (cf. [191]). For the graphs of bounded maximum degree, the authors in [190] studied proper t -frugal colorings, and Yuster [391] studied acyclic proper 2-frugal colorings. Recently, Kang and Müller [220] expanded this study and set up some relations among parameters $\varphi^t(G)$, $\chi^t(G)$, $\chi_a^t(G)$, and $\chi_s^t(G)$.

A related concept about t -frugal coloring of graphs is the linear coloring. A proper vertex coloring of a graph G is linear if the graph induced by the vertices of any two color classes is the union of vertex-disjoint paths. The linear chromatic number $lc(G)$ of the graph G is the smallest number of colors in a linear coloring

of G . Thus, a linear coloring is just a 3-frugal coloring and is also a special case of acyclic coloring of G .

If G is linearly L -colorable for any list assignment L satisfying $|L(v)| \geq k$ for any $v \in V(G)$, then G is said to be *linearly k -choosable*. The smallest integer k such that G is linearly k -choosable is called the *linear choice number*, denoted by $\Lambda^l(G)$. By the definition, it is obvious that $lc(G) \leq \Lambda^l(G)$ for any graph G .

Yuster [391] first introduced and investigated the linear coloring of graphs. With a probabilistic method, he proved that $lc(G) = O(\Delta^{\frac{3}{2}})$ for a general graph G with maximum degree Δ and constructed graphs G such that $lc(G) = \Omega(\Delta^{\frac{3}{2}})$.

In 2005, Esperet et al. [141] generalized the linear coloring of graphs to a list-coloring version. They investigated linear choosability for some classical families of graphs such as trees, grids, bipartite complete graphs, planar graphs, outerplanar graphs, graphs with maximum degree 3 or 4, and graphs with given maximum average degree. In particular, the following result is implied in their paper:

Theorem 101 *Let G be a planar graph with girth g :*

1. *If $\Delta \geq 12$, then $\Lambda^l(G) \leq 2\Delta + 26$.*
2. *If $g \geq 16$ and $\Delta \geq 3$, then $\Lambda^l(G) = \lceil \frac{\Delta}{2} \rceil + 1$.*
3. *If $g \geq 10$, then $\Lambda^l(G) \leq \lceil \frac{\Delta}{2} \rceil + 2$.*
4. *If $g \geq 8$, then $\Lambda^l(G) \leq \lceil \frac{\Delta}{2} \rceil + 3$.*

It is easy to see by the definition of linear coloring that $lc(G) \geq \lceil \frac{\Delta}{2} \rceil + 1$ for any graph G . A natural question is which graphs G can attain the lower bound $\lceil \frac{\Delta}{2} \rceil + 1$? Raspaud and Wang [299] proved that planar graphs with high girth and large maximum degree possess such a property. More precisely, they proved the following two theorems:

Theorem 102 *Every planar graph G has $lc(G) = \lceil \frac{\Delta}{2} \rceil + 1$ if there is a pair $(k, g) \in \{(13, 7), (7, 9), (5, 11), (3, 13)\}$ such that G satisfies $\Delta \geq k$ and G has girth at least g .*

Note that the condition that $\Delta \geq 3$ in **Theorem 102** is essential, since every cycle C of length at least 3 satisfies $lc(G) = 3 > 2 = \lceil \frac{\Delta}{2} \rceil + 1$.

Theorem 103 *Let G be a planar graph with girth g :*

1. *If $g \geq 6$, then $lc(G) \leq \lceil \frac{\Delta}{2} \rceil + 4$.*
2. *If $g \geq 5$, then $lc(G) \leq \lceil \frac{\Delta}{2} \rceil + 14$.*

More recently, **Theorem 102** was extended by Wang and Li [357] by showing the following result.

Theorem 104 *Every graph G embeddable in a surface of nonnegative characteristic has $lc(G) = \lceil \frac{\Delta}{2} \rceil + 1$ if there is a pair $(k, g) \in \{(13, 7), (9, 8), (5, 10), (3, 13)\}$ such that G satisfies $\Delta \geq k$ and G has girth at least g .*

Dong, Xu, and Zhang [132] improved partially the results in **Theorem 103** by showing:

Theorem 105 Let G be a planar graph with girth g :

1. If $g \geq 8$, then $lc(G) \leq \lceil \frac{\Delta}{2} \rceil + 2$.
2. If $g \geq 6$, then $lc(G) \leq \lceil \frac{\Delta}{2} \rceil + 3$.
3. If $g \geq 5$, then $lc(G) \leq \lceil \frac{\Delta}{2} \rceil + 10$.

The latest results on linear chromatic number were established by Li et al. [247].

Theorem 106 Let G be a graph:

1. $lc(G) \leq \frac{1}{2}(\Delta^2 + \Delta)$.
2. If $\Delta \leq 4$, then $lc(G) \leq 8$.
3. If $\Delta \leq 5$, then $lc(G) \leq 14$.
4. If G is a planar graph with $\Delta \geq 52$, then $lc(G) \leq \lfloor 0.9\Delta \rfloor + 5$.

3.4.4 Generalized Acyclic Colorings

A generalization of the acyclic edge chromatic number was introduced by Gerke et al. in [163].

Let $r \geq 3$ be a positive integer. Given a graph G , let $\chi'_{a:r}(G)$ be the minimum number of colors required to color the edges of the graph such that every k -cycle has at least $\min\{k, r\}$ colors, for all $k \geq 3$. So $\chi'_a(G) = \chi'_{a:3}(G)$. The r -acyclic edge chromatic number $\chi'_{a:r}(G)$ is the minimum number of colors required to color the edges of the graph G such that every cycle C of G has at least $\min\{|C|, r\}$ colors.

Replacing the word *edges* with *vertices* gives the definition of the r -acyclic chromatic number $\chi_{a:r}(G)$ of G . A proper vertex (respectively, edge) coloring is r -acyclic if each k -cycle has at least $\min\{k, r\}$ colors, for $k \geq 3$. Obviously, a proper r -acyclic coloring of the line graph $L(G)$ of a graph G gives a proper r -acyclic edge coloring of G . Hence, $\chi'_{a:r}(G) \leq \chi_{a:r}(L(G))$.

Note that the complete bipartite graph $K_{d,d}$ satisfies $\chi_{a:4}(K_{d,d}) = d^2$, since every edge must receive a distinct color to avoid a 4-cycle that is colored with less than four colors. This is an example of a class of graphs where the 4-acyclic edge chromatic number of any member is quadratic in its maximum degree. For the case $r \geq 6$, Greenhill and Pikhurko [167] considered the generalized acyclic chromatic number and generalized acyclic edge chromatic number of graphs with maximum degree d and constructed Δ -regular graphs G with $\chi'_{a:r}(G) \geq c_r \Delta^{\lfloor r/2 \rfloor}$, where c_r depends on r but is a constant with respect to Δ . They also proved upper bounds on $\chi'_{a:r}(G)$ of graphs with order $O(\Delta)^{\lfloor r/2 \rfloor}$. Exactly, they proved the following theorem by using the Local Lemma while considering the Hamming cube graphs. The upper bounds and the lower bounds were given in [167].

Before stating the following theorem, some notation is first introduced. Let d be a positive integer. Let $\chi_{a:r}(\Delta)$ and $\chi'_{a:r}(\Delta)$, respectively, be the maximum of $\chi_{a:r}(G)$ and $\chi'_{a:r}(G)$, over all graphs G with maximum degree Δ . The following theorem was obtained in [167].

Theorem 107 Let $r \geq 4$ be fixed. There exist positive constants c, C, c', C' such that $c\Delta^{\lfloor r/2 \rfloor} \leq \chi_{a:r}(\Delta) \leq C\Delta^{\lfloor r/2 \rfloor}$, $c'\Delta^{\lfloor r/2 \rfloor} \leq \chi'_{a:r}(\Delta) \leq C'\Delta^{\lfloor r/2 \rfloor}$ for $\Delta \geq 3$.

In addition, the authors in [167] showed that some well-known algebraic constructions produce families of graphs which, unlike the Hamming cubes, have girth at least r but still provide fairly good lower bounds for the r -acyclic chromatic numbers when $r = 6, 8, 12$.

Theorem 108 Suppose that $d = q + 1$ where q is a prime or prime power. Then there is a graph G which is d -regular, has girth 6, and satisfies $\chi_{a:6}(G) = 2(d^2 - d + 1)$, $\chi'_{a:6}(G) = d(d^2 - d + 1)$.

Theorem 109 Suppose that $d = q + 1$ where q is a prime or prime power. Then there is a graph G which is d -regular, has girth 6, and satisfies $\chi_{a:8}(G) = 2d(d^2 - 2d + 2)$, $\chi'_{a:8}(G) = d^2(d^2 - 2d + 2)$.

Theorem 110 Suppose that $d = q + 1$ where q is a prime or prime power. Then there is a graph G which is d -regular, has girth 12, and satisfies $\chi_{a:12}(G) = 2d(d^4 - 4d^3 + 7d^2 - 6d + 3)$, $\chi'_{a:12}(G) = d^2(d^4 - 4d^3 + 7d^2 - 6d + 3)$.

In [163], it was shown that typical Δ -regular graphs have much smaller r -acyclic edge chromatic number. More precisely, it was shown that a random Δ -regular graph on n vertices has, with probability tending to 1 as n tends to infinity, an r -acyclic edge chromatic number at most $(r - 2)\Delta$.

A property holds *asymptotically almost surely* (a.a.s.) if the probability that it holds tends to 1 as n tends to infinity.

Theorem 111 ([163]) Let $\Delta \geq 2$ and $r \geq 4$ be fixed. The r -acyclic edge chromatic number of a uniformly chosen Δ -regular graph on n vertices is a.a.s. at most $(r - 2)\Delta$.

Therefore, a typical regular graph has an r -acyclic edge chromatic number that is linear in its maximum degree. A natural question is thus to ask which structures force a class of graphs to have an r -acyclic edge chromatic number that is quadratic in Δ . In 2006, Gerke and Raemy [164] showed that such a class has to contain graphs with small girth. More specifically, they showed the following theorem:

Theorem 112 Let $r \geq 4$ be an integer. For any graph G with maximum degree Δ and girth $g \geq 3(r - 1)\Delta$, it holds that $\chi'_{a:r}(G) \leq 6(r - 1)\Delta$.

Theorem 112 showed that for the class of graphs with large girth, the r -acyclic edge chromatic number is linear in Δ if r is fixed. The upper bound is at most a factor 12 away from the optimal bound as it is known [163] that every d -regular graph with girth $r \geq 3$ satisfies $\chi'_{a:r}(G) > (r - 1)d/2$.

4 Vertex-Distinguishing Coloring

This section is concerned with vertex-distinguish coloring problems. To start with this, a new concept is needed, namely, edge-weighting. Let G be a graph. A k -edge weighting of G is a mapping $\phi : E(G) \rightarrow \{1, 2, \dots, k\}$. Let $\phi(e)$

denote the weight of edge e . Usually, it is needed to seek a much smaller size of $\{1, 2, \dots, k\}$ such that there exists a mapping ϕ satisfying several given constraints. Given a graph G and an edge-weighting ϕ of G , for $v \in V(G)$, let $S_\phi(v) = \{\phi(e) | v \in e\}$ be the set of edge-weights appearing on edges incident to v under the edge-weighting ϕ .

Based on the Ph.D. dissertation [293], the class of edge-weighting problems can be roughly split into two parts. The one is the *proper edge-weightings* (also named as edge-coloring), that is, edge weightings of G where any two adjacent edges must be mapped to different weights by ϕ . The other is the *improper edge-weightings* of G , that is, edge-weightings of G where two adjacent edges may be mapped to the same weight by ϕ .

4.1 Proper Edge-Weighting

Firstly, proper edge weighting (namely, edge-coloring) is considered. It is evident that Δ is the lower bound for all the considered edge-coloring problems. Moreover, if G has two adjacent vertices of maximum degree, then $\Delta + 1$ is the corresponding lower bound.

4.1.1 Vertex-Distinguishing Edge-Colorings

An edge-coloring ϕ of G is called *vertex-distinguishing* (or *vde-coloring* for short), if any two vertices u, v have different weight sets, i.e., $S_\phi(u) \neq S_\phi(v)$. A vde-coloring is also called a *strong edge coloring*, or *point-distinguishing edge-coloring*. The *vertex-distinguishing chromatic index* of G , denoted by $\chi'_{vd}(G)$ (or $vd(G)$), is the smallest integer k such that there is a k -vde-coloring of G . This graph parameter is introduced by Burris and Schelp [88, 89] and, independently (as *observability* of a graph), by Černý et al. [96], and Horňák and Soták [195].

If G has an isolated edge uv , then $S(u) = S(v)$ always and G cannot have a vde-coloring. Similarly, if G has two isolated vertices u, v , then $S(u) = S(v) = \emptyset$. However, every graph without isolated edges and with at most one isolated vertex must have a vde-coloring: simply assign different colors to all edges of G . Such graphs are called as *nice graphs*.

In [88], the author conjectured two different upper bounds. The first one has been confirmed by Bazgan et al. [42].

Theorem 113 *If G is a nice graph on n vertices, then $\chi'_{vd}(G) \leq n + 1$.*

The sharpness of Theorem 113 can be confirmed by considering the complete graph on n vertices with n even.

Given a graph G , let n_d denote the number of vertices of degree d . It is easy to see that if there exists a k -vde-coloring of G , then $\binom{k}{d} \geq n_d$ for all $1 \leq d \leq \Delta$. The second conjecture of Burris and Schelp [88, 89] remains open, which is analogous to the Vizing's Theorem on edge coloring:

Conjecture 15 If G is a nice graph and k is the minimum integer such that $\binom{k}{d} \geq n_d$ for all $\delta \leq d \leq \Delta$, then $k \leq \chi'_{\text{vd}}(G) \leq k + 1$.

In the same papers [88, 89], Burris and Schelp gave an upper bound of $\chi'_{\text{vd}}(G)$ for a graph G .

Theorem 114 For a graph G , $m_1 \leq \chi'_{\text{vd}}(G) \leq (\Delta + 1)\lfloor 2m_2 + 5 \rfloor$, where

$$m_1 = \max_{1 \leq k \leq \Delta} \left\{ (k!n_k)^{1/k} + \frac{k-1}{2} \right\}, \text{ and } m_2 = \max_{1 \leq k \leq \Delta} n_k^{1/k}.$$

Corollary 7 For an r -regular graph G of order n , $\chi'_{\text{vd}}(G) \leq (r+1)\lfloor 2n^{1/r} + 5 \rfloor$.

Recently, there has been significant progress on this conjecture. Balister [28] considered the vde-coloring of random graphs by giving a strong bound below:

Theorem 115 If G is a random graph on n vertices with edge probability $p = p(n)$, and if $\frac{pn}{\log n}, \frac{(1-p)n}{\log n} \rightarrow \infty$ as $n \rightarrow \infty$, then the probability that $\chi'_{\text{vd}}(G) = \Delta$ goes to 1 as $n \rightarrow \infty$.

Balister et al. [31] proved that [Conjecture 15](#) is true for graphs with large maximum degree.

Theorem 116 If G is a graph with n vertices, $\Delta \geq \sqrt{2n} + 4$, $\delta \geq 5$, and k is the smallest positive integer such that $n_d \leq \binom{k}{d}$ for all $\delta \leq d \leq \Delta$, then $k \leq \chi'_{\text{vd}}(G) \leq k + 1$.

Balister et al. [29] confirmed [Conjecture 15](#) if G consists of just paths or of just cycles. Specially, they proved the following result.

- Theorem 117**
1. Let G be a vertex-disjoint union of cycles, and let $n_2 = |G| \leq \binom{k}{2}$, with k as small as possible. Then $\chi'_{\text{vd}}(G) = k$ or $k + 1$.
 2. Let G be a vertex-disjoint union of paths of length at least 2. Let $n_1 \leq k$ and $n_2 \leq \binom{k}{2}$, with k as small as possible. Then $\chi'_{\text{vd}}(G) = k$ or $k + 1$.
 3. Let G be any nice graph with $\Delta = 2$. Let $n_1 \leq k$ and $n_2 \leq \binom{k}{2}$, with k as small as possible. Then $k \leq \chi'_{\text{vd}}(G) \leq k + 5$.

[Theorem 117](#) implies that [Conjecture 15](#) holds for 2-regular graphs. Since the conjecture holds for graphs with large maximum degree [31], it is natural to ask whether the conjecture holds for small maximum degree, for example, 3-regular graphs. Many people tried to think about the bounds of $\chi'_{\text{vd}}(G)$ for some regular graphs G , such as Fibonacci and Lucas cubes [124]; Q_n [197]; nG , where G is a 3-regular graph with 1-factor on at most 12 vertices or $G \in \{K_{4,4}, K_{5,5}, K_{6,6}, K_{7,7}, K_6\}$ [307]; ladders and pK_4 [325]; and extended Fibonacci cubes [380].

The following result has been testified for d -regular graphs with small components by Balister et al. [32].

Theorem 118 Let $d \geq 3$ and assume that G is a d -regular graph which contains $d - 2$ disjoint 1-factors. Let $G = \bigcup_{i=1}^s G_i$, where G_1, G_2, \dots, G_s are components of G . If $|G| \leq \binom{k}{d}$ and $|G_i| \leq \frac{3(k-1)}{4(d-1)}$ for all i , then $\chi'_{\text{vd}}(G) \leq k + 1$.

Balister, Riordan, and Schelp [32] defined a few parameters for nice graphs. Here, \oplus is the symmetric difference operator, which produces the set of items which appear in exactly an odd number of its operands. Suppose that G is a nice graph. Let $k(G)$ be the minimum integer k such that $\binom{k}{j} \geq n_j$ for all j with $\delta \leq j \leq \Delta$. Recall for $v \in V(G)$, $S(v)$ denotes the set of colors used to color the edges incident to v . If a vde-coloring with k colors exists for G , then each color meets an even number of vertices. Thus, $\bigoplus_v S(v) = \emptyset$. Let $k'(G)$ be the minimum k such that there exist distinct sets $S(v) \subseteq \{1, 2, \dots, k\}$ with $|S(v)| = d(v)$ for all $v \in V(G)$, such that $\bigoplus_v S(v) = \emptyset$. Therefore, $k'(G) \geq k(G)$. Let $k''(G)$ be the smallest k such that for any set of vertices $X \subseteq V(G)$, there exist distinct sets $S(v) \subseteq \{1, 2, \dots, k\}$, $v \in X$, such that $|S(v)| = d(v)$ and $|\bigoplus_{v \in X} S(v)| \leq |E(X, X^c)|$, where $E(X, X^c)$ is the set of edges between X and $X^c = V(G) \setminus X$. Thus, $\chi'_{\text{vd}}(G) \geq k''(G) \geq k'(G) \geq k(G)$. Also, in [32], it is shown that $k''(G) \leq k(G) + 1$. Hence, they presented the following conjecture.

Conjecture 16 For all nice graphs G , $\chi'_{\text{vd}}(G) = k''(G)$.

Observe that this conjecture is consistent with the inequalities mentioned above. Also, this conjecture implies clearly Conjecture 15.

It is not even known whether there exists an absolute constant c such that $\chi'_{\text{vd}}(G) \leq k(G) + c$, when G is a nice graph. Yet, for graphs with large minimum degree, Bazgan, Harkat-Benhamdine, Li, and Woźniak [43] obtained the following result.

Theorem 119 Let G be a nice graph of order $n \geq 3$. If $\delta > \frac{n}{3}$, then $\chi'_{\text{vd}}(G) \leq \Delta + 5$.

For an integer k with $k \leq \alpha(G)$, where $\alpha(G)$ stands for the independence number of G , $\sigma_k(G)$ is defined by

$$\sigma_k(G) = \min \left\{ \sum_{i=1}^k d(v_i) \mid \{v_1, v_2, \dots, v_k\} \text{ is an independent set of } G \right\}.$$

If $\alpha(G) = 1$, then G is complete and the exact value of $\chi'_{\text{vd}}(G)$ is computed in [89], that is, $\chi'_{\text{vd}}(K_n) = n$ if n is odd and $\chi'_{\text{vd}}(G) = n + 1$ if n is even for $n \geq 4$. For $\alpha(G) \geq 2$, Liu and Liu [254] presented the following result.

Theorem 120 Let G be a connected graph of order $n \geq 3$. If $\sigma_2(G) \geq \frac{2n}{3}$, then $\chi'_{\text{vd}}(G) \leq \Delta + 5$.

It is easy to see that if $\delta > \frac{n}{3}$, then $\sigma_2(G) \geq \frac{2n}{3}$. Thus, Theorem 120 generalizes Theorem 119.

4.1.2 Adjacent Vertex-Distinguishing Edge-Coloring

Instead of requiring that any two vertices have different weight sets, it can be required that only adjacent vertices have different weight sets. An edge-coloring satisfying this condition is *adjacent vertex-distinguishing* (also called an *adjacent strong edge-coloring*, *1-strong edge-coloring*, or *neighbor-distinguishing edge-coloring* and denoted as *avde-coloring* for short). A k -*avde-coloring* is an avde-coloring using k colors. The smallest integer k such that there is a k -*avde-coloring* of G is called the *adjacent vertex-distinguishing chromatic index* of G , denoted by $\chi'_{\text{avd}}(G)$.

Obviously, when G contains an isolated edge, there is no any k -*avde-coloring* of G for any $k \geq 1$. Thus, only simple graphs without isolated edges are considered in the rest of this section.

This graph parameter was introduced by Zhang et al. [396]. They completely determined the adjacent vertex-distinguishing chromatic indices for paths, cycles, complete graphs, and complete bipartite graphs. In light of these examples, they proposed the following conjecture.

Conjecture 17 *If G is a connected graph and $G \neq C_5$, then $\chi'_{\text{avd}}(G) \leq \Delta + 2$.*

Note that $\chi'_a(C_5) = 5 = \Delta + 3$. It is unknown if C_5 is the unique exception graph. The conjecture has been confirmed by Balister et al. [30] for bipartite graphs and for graphs of maximum degree 3. They also gave a general upper bound on $\chi'_{\text{avd}}(G)$, depending on the maximum degree Δ and chromatic number $\chi(G)$:

Theorem 121 *If G is a graph, then $\chi'_{\text{avd}}(G) \leq \Delta + O(\log \chi(G))$.*

Akbari et al. [10] obtained an upper bound of $\chi'_{\text{avd}}(G)$, which was improved by Dai and Bu [123] by one.

Theorem 122 ([10]) *For any graph G , $\chi'_{\text{avd}}(G) \leq 3\Delta$.*

Theorem 123 ([123]) *For any graph G , $\chi'_{\text{avd}}(G) \leq 3\Delta - 1$.*

Later, Ghandehari and Hatami [165] improved these bounds by restricting the maximum degree of the graph:

Theorem 124 *If G is a graph with $\Delta \geq 10^6$, then $\chi'_{\text{avd}}(G) \leq \Delta + 27\sqrt{\Delta \ln \Delta}$.*

Hatami [179] gave an asymptotically stronger upper bound on $\chi'_{\text{avd}}(G)$ by using a random edge-weighting technique.

Theorem 125 *If G is a graph with $\Delta > 10^{20}$, then $\chi'_{\text{avd}}(G) \leq \Delta + 300$.*

Despite the large maximum degree requirement, this result is of importance as it shows that a simple additive constant is enough for an upper bound on $\chi'_{\text{avd}}(G)$.

For graphs with small maximum average degree, Wang and Wang [371] verified Conjecture 17:

Theorem 126 Let G be a graph:

1. If $\text{mad}(G) < 3$ and $\Delta \geq 3$, then $\chi'_{\text{avd}}(G) \leq \Delta + 2$.
2. If $\text{mad}(G) < \frac{5}{2}$ and $\Delta \geq 4$, or $\text{mad}(G) < \frac{7}{3}$ and $\Delta = 3$, then $\chi'_{\text{avd}}(G) \leq \Delta + 1$.
3. If $\text{mad}(G) < \frac{5}{2}$ and $\Delta \geq 5$, then $\chi'_{\text{avd}}(G) = \Delta + 1$ if and only if G contains adjacent vertices of maximum degree.

Recall that if G is a planar graph with girth g , then $\text{mad}(G) \leq \frac{2g}{g-2}$. This fact together with [Theorem 126](#) leads to the following corollary:

Corollary 8 Let G be a planar graph with girth g :

1. If $g \geq 6$ and $\Delta \geq 3$, then $\chi'_{\text{avd}}(G) \leq \Delta + 2$.
2. If $g \geq 10$ and $\Delta \geq 4$, or $g \geq 4$ and $\Delta = 3$, then $\chi'_{\text{avd}}(G) \leq \Delta + 1$.
3. If $g \geq 10$ and $\Delta \geq 5$, then $\chi'_{\text{avd}}(G) = \Delta + 1$ if and only if G contains adjacent vertices of maximum degree.

It should be pointed out that Bu et al. [86] also proved that [Conjecture 17](#) holds for planar graphs of girth at least six, that is, if G is a planar graph with girth $g \geq 6$, then $\chi'_{\text{avd}}(G) \leq \Delta + 2$.

Liu and Liu [255] considered the adjacent vertex-distinguishing chromatic index for graphs with dominating subgraphs. A subgraph H of a graph G is called a *dominating subgraph* of G if $V(G) - V(H)$ is an independent set.

The following three theorems appeared in [255]:

Theorem 127 If G is a Hamiltonian graph with $\chi(G) \leq 3$, then $\chi'_{\text{avd}}(G) \leq \Delta + 3$.

Theorem 128 If G is a graph with $\chi(G) \leq 3$ and G has a Hamiltonian path, then $\chi'_{\text{avd}}(G) \leq \Delta + 4$.

Theorem 129 If a graph G has a dominating cycle or a dominating path H such that $\chi(G[V(H)]) \leq 3$, then $\chi'_{\text{avd}}(G) \leq \Delta + 5$.

For regular graphs, Greenhill and Rucinski [168] used a probabilistic method to obtain the following better bound:

Theorem 130 Let $r \geq 4$. Then a random r -regular graph G a.a.s. has $\chi'_{\text{avd}}(G) \leq \lceil 3r/2 \rceil$.

When $r = 4$, it can be deduced from the above theorem that almost always all 4-regular graphs satisfy [Conjecture 17](#).

Edwards et al. [139] proved that $\chi'_{\text{avd}}(G) \leq \Delta + 1$ if G is bipartite, planar, and $\Delta \geq 12$. Chen and Guo [103] completely determined the adjacent vertex-distinguishing chromatic index of hypercubes. In precise, they showed that if Q_d is a hypercube of dimension $d \geq 3$, then $\chi'_{\text{avd}}(Q_d) = d + 1$. Liu et al. [256] proved that if G is a 2-connected outerplanar graph with $\Delta \geq 5$, then $\Delta \leq \chi'_{\text{avd}}(G) \leq \Delta + 1$, and $\chi'_{\text{avd}}(G) = \Delta + 1$ if and only if G contains adjacent vertices of maximum degree. Wang and Wang [372] extended this result to the following:

Theorem 131 Let G be a K_4 -minor-free graph with $\Delta \geq 4$. Then the following statements hold:

1. $\Delta \leq \chi'_a(G) \leq \Delta + 1$.
2. If $\Delta \geq 5$, then $\chi'_a(G) = \Delta + 1$ if and only if G contains adjacent vertices of maximum degree.

It is somewhat surprising that there exists an interesting relationship between $\chi'_{\text{avd}}(G)$ and $\chi''(G)$, the total chromatic number of a graph G . Zhang et al. [397] proposed the following conjectures:

Conjecture 18 For a graph G with no K_2 or C_5 component, $\chi'_{\text{avd}}(G) \leq \chi''(G)$.

Conjecture 19 For an r -regular graph G with no C_5 component ($r \geq 2$), $\chi'_{\text{avd}}(G) = \chi''(G)$.

In [397], Conjecture 19 was shown to be true for many classes of graphs, such as graphs with $\chi''(G) = \Delta + 1$; 2-regular, 3-regular, and $(|G| - 2)$ -regular graphs; bipartite regular graphs; balanced complete multipartite graphs; hypercubes; and joins of two matchings or cycles.

4.1.3 r -Strong Chromatic Index

Akbari et al. [10] introduced another graph parameter, called as r -strong edge coloring. A proper edge coloring of a graph G is called an r -strong edge-coloring if for any two distinct vertices $u, v \in V(G)$ with the distance $d(u, v) \leq r$, $S(u) \neq S(v)$. The r -strong chromatic index $\chi'_{si}(G, r)$ is the minimum number of colors required for an r -strong edge-coloring of the graph G . Thus, $\chi'_{si}(G, 1) = \chi'_{\text{avd}}(G)$, and for any connected graph G with n vertices, $\chi'_{si}(G, n) = \chi'_{\text{vd}}(G)$.

Akbari et al. [10] generalized a result in [396] about the parameter $\chi'_{\text{avd}}(T)$, where T is a tree as follows:

Theorem 132 Let T be a tree with at least three vertices:

1. $\chi'_{si}(T, 2) \leq \Delta + 1$. Furthermore, if for any two vertices u and v of maximum degree, $d(u, v) \geq 3$, then $\chi'_{si}(T, 2) = \Delta$.
2. If $\Delta \geq 3$, then $\chi'_{si}(T, 3) \leq 2\Delta - 1$.
3. For any integers $k \geq 4$ and $\Delta \geq 1$, there exists a tree T with maximum degree Δ such that $\chi'_{si}(T, k) \geq \Delta(\Delta - 1)^{\lfloor k/2 \rfloor - 1}$.

Ghandehari and Hatami [165] presented an upper bound of $\chi'_{si}(G, k)$ for regular graphs:

Theorem 133 For each r -regular graph G with $r > 100$, $\chi'_{si}(G, k) \leq r + (k + 2) \log_2 r$.

As a direct corollary of Theorem 133, the following holds:

Corollary 9 For an r -regular graph G with $r > 100$, $\chi'_{\text{avd}}(G) \leq r + 3 \log_2 r$.

4.2 Non-proper Edge-Weighting

In this section, non-proper edge-weighting is concerned. Unlike the case of proper edge-weighting, there is no lower bounds compared to $\chi'(G)$.

4.2.1 Point-Distinguishing Chromatic Index

Similar to the vertex-distinguishing edge-coloring problem, an edge-weighting (not needed to be proper) such that all vertices have different weight sets is considered here. This graph parameter is introduced by Harary and Plantholt [177] (as *point-distinguishing chromatic index*) and is denoted by $\chi_0(G)$. Note that if a graph G has at least two isolated vertices or an isolated edge, then point-distinguishing chromatic index cannot be defined very well. This means that once G admits the point-distinguishing chromatic index, then G must be a nice graph.

Harary and Plantholt [177] determined the value of this parameter for paths, cycles, complete graphs, and hypercubes. Furthermore, they presented a lower bound for complete bipartite graphs. Even for complete bipartite graphs, it seems that the problem of determining $\chi_0(K_{m,n})$ is not easy. In fact, it was shown in [177] that $\chi_0(K_{m,n}) \geq \lceil \log_2(m+n) \rceil$ and $\chi_0(K_{m,n}) \leq \lceil \log_2 n \rceil + 2$ if $\lceil \log_2 n \rceil + 1 \leq m \leq n$.

Zagaglia Salvi [393] proved that the sequence $\{\chi_0(K_{n,n})\}_{n=2}^{\infty}$ is non-decreasing with jumps of value 1. As a consequence, there is an increasing sequence $\{n_k\}_{k=2}^{\infty}$ of integers in which $n_2 = 1$ and for any $k \geq 3$, $\chi_0(K_{n,n}) = k$ if and only if $n_{k-1} + 1 \leq n \leq n_k$. Only few exact values of n_k are known, namely, $n_3 = 2$, $n_4 = 5$, $n_5 = 11$, $n_6 = 22$ [392], and $n_7 = 46$ [196]. According to Zagaglia Salvi [393], $n_{k+1} \geq 2n_k$ and so $\{\frac{n_k}{2^{k-1}}\}$ is convergent and $\frac{46}{64} \leq \lim \frac{n_k}{2^{k-1}} \leq 1$. Horňák and Soták [198] succeeded in showing that $\lim \frac{n_k}{2^{k-1}} \geq 3 - \sqrt{5}$.

Horňák and Zagaglia Salvi [200] asked the following question.

Problem 4 Is it true that $\lim \frac{n_k}{2^{k-1}} < 1$?

They also determined $\chi_0(K_{m,n})$ for all n sufficiently large with respect to m in [200]. Let k, l, m, n be integers with $2 \leq m \leq n$, $l = \lceil \log_2 m \rceil$, $k \geq \min\{l+1, 3\}$.

$$d_m^{(k)} = \begin{cases} 2m, & \text{if } k \leq m; \\ 2m-1, & \text{if } k = m+1; \\ m, & \text{if } k \geq m+2. \end{cases}$$

$$b_m^{(k)} = \sum_{i=0}^m \binom{k}{i} - d_m^{(k)},$$

$$I_m^{(k)} = 7[b_m^{(k-1)} + 1, b_m^{(k)}],$$

$$L_m^{(k)} = [b_m^{(k-1)} + 1, 2^{k-1} - m - l - 1],$$

$$R_m^{(k)} = [2^{(k-1)} - m - l, b_m^{(k)}],$$

where $[p, q]$ denotes the interval of all integers z with $p \leq z \leq q$ for fixed integers p, q .

The following several results and one problem appeared in [200]:

Theorem 134 If $n \in I_m^{(k)}$, then $\chi_0(K_{m,n}) \leq k$.

Theorem 135 $\chi_0(K_{m,n}) \geq k + 1$ follows from either of the following conditions:

1. $k \leq m + 1$ and $n \geq 2^k - m - l$.
2. $k \geq m + 2$ and $n \geq b_m^{(k)} + l$.

Theorem 136 The equality $\chi_0(K_{m,n}) = k \Leftrightarrow n \in I_m^{(k)}$ follows from any of the following conditions:

1. $k \leq 2l + 2$.
2. $m \in \{2, 3\}$.

Corollary 10 If $m \geq 7$ and $n \in [2 \cdot 4^l - 2m + 1, 2^{m+1} - 2m]$, then $\chi_0(K_{m,n}) = \lceil \log_2(n + 2m) \rceil$.

Theorem 137 The equality $\chi_0(K_{m,n}) = k$ follows from any of the following conditions:

1. $k \in [l + 2, 2l + 1]$ and $n \in R_m^{(k)}$.
2. $k = l + 1$ and $n \in I_m^{(k)}$.

Theorem 138 Suppose that $m \in [4, 10]$ and $n \in L_m^{(k)}$. Then $\chi_0(K_{m,n}) = k - 1$ if and only if $(k, m, n) = (6, 10, 13)$.

Problem 5 Decide whether there is a triple of integers (k, m, n) such that $m \geq 11$, $k \in [l + 3, 2l + 1]$, $n \in L_m^{(k)}$, and $\chi_0(K_{m,n}) = k - 1$.

4.2.2 General Neighbor-Distinguishing Index

Also, a generalized version of adjacent vertex-distinguishing edge-coloring may be considered, in which edge weighting may be not proper. The corresponding invariant is the *general neighbor-distinguishing index* of a graph G , denoted by $\text{gndi}(G)$. This graph parameter was first introduced and investigated by Győri et al. [174]. Clearly, G is a nice graph when considering this parameter.

First, four authors in [174] established an interesting relationship between $\text{gndi}(G)$ and $\chi(G)$:

Theorem 139 For any graph G , the following statements are equivalent:

1. $\text{gndi}(G) = 2$.
2. G is bipartite and there is a bipartition $\{X_1 \cup X_2, Y\}$ of $V(G)$ such that $X_1 \cap X_2 = \emptyset$ and any vertex of Y has at least one neighbor in both X_1 and X_2 .

Theorem 140 If G is a connected bipartite graph with $|E(G)| \geq 2$, then $\text{gndi}(G) \leq 3$.

Theorem 141 $\text{gndi}(G) \leq 2\lceil \log_2 \chi(G) \rceil + 1$ for any nice graph G .

Thus, when $\chi(G) \leq 2$, a good characterization on $\text{gndi}(G)$ has been established. If $\chi(G) \geq 3$, Horňák and Soták [199] improved the above bound:

Theorem 142 *If G is a connected graph with $\chi(G) \geq 3$, then $\lceil \log_2 \chi(G) \rceil + 1 \leq \text{gndi}(G) \leq \lfloor \log_2 \chi(G) \rfloor + 2$.*

Therefore, when $\log_2 \chi(G)$ is not an integer, then $\text{gndi}(G) = \lceil \log_2 \chi(G) \rceil + 1$. A natural question is as follows: what about the case where $\log_2 \chi(G)$ is an integer? This problem is recently solved by Győri and Palmer [175]:

Theorem 143 *If G is a connected graph with $\chi(G) \geq 3$, then $\text{gndi}G = \lceil \log_2 \chi(G) \rceil + 1$.*

4.2.3 Irregularity Strength

Instead of distinguishing the color set, the problem of distinguishing the total sum of weights can be considered. Chartrand et al. [100] defined the *irregularity strength* of a graph G , denoted by $s(G)$, to be the smallest integer k such that there is a k -edge-weighting ϕ of G and for any two vertices u, v ,

$$\sum_{e \ni u} \phi(e) \neq \sum_{e \ni v} \phi(e).$$

It is easy to see that for every nice graph G , there exists an irregular strength. If G is not a nice graph, set $s(G) = \infty$.

Most of the upper bounds which have been determined for special classes of graphs are based on the minimum and maximum degree. The proofs rely on two techniques, either using the congruence method to find special spanning graphs and then determining a labeling of this spanning graph explicitly or through employing the probabilistic method.

In [100], it was shown that for any graph G with $n \geq 3$ vertices, $s(G) \leq 2n - 3$. This bound was later improved by Nierhoff [290] by refining the idea of Aigner and Triesch [8] to $s(G) \leq n - 1$ for any graph G with $n \geq 4$ vertices.

For regular graphs, stronger upper bounds on $s(G)$ are known. Faudree and Lehel [145] showed that if G is d -regular ($d \geq 2$), then $\lceil \frac{n+d-1}{d} \rceil \leq s(G) \leq \lceil \frac{n}{2} \rceil + 9$. They also proposed the following conjecture.

Conjecture 20 *If G is a d -regular graph with $d \geq 2$, then $s(G) \leq \lceil \frac{n}{d} \rceil + c$ for some constant c .*

The question was motivated by the following facts:

- (i) For every d -regular graph of order n , $s(G) \geq \lceil (n-1)/d \rceil + 1$ (see [100]).
- (ii) In all those cases where the irregularity strength of a regular graph is known, its value is very close to $\lceil (n-1)/d \rceil + 1$. For example, Amar [23] proved that if G is an $(n-3)$ - or $(n-4)$ -regular graph of order n , then $s(G)$ is 3 (except if $G = K_{3,3}$). In light of this result, Amar [23] proposed the following conjecture:

Conjecture 21 *The irregular strength of an r -regular graph of order $n \leq 2r$ is 3, except if G is $K_{l,l}$ with l odd.*

Frieze, Gould, Karoński, and Pfender [157] used a probabilistic analysis to show that for a graph G with minimum degree δ and maximum degree Δ , $s(G) \leq c_1 n / \delta$ if $\Delta \leq n^{1/2}$ and $s(G) \leq c_2 (\log n) n / \delta$ if $\Delta > n^{1/2}$, where c_1 and c_2 are explicit constants. For regular graphs, they obtained the following results in the same paper.

Theorem 144 *Let G be a d -regular graph of order n with no isolated vertices or edges:*

1. *If $d \leq \lfloor (n / \ln n)^{1/4} \rfloor$, then $s(G) \leq 10n/d + 1$.*
2. *If $\lfloor (n / \ln n)^{1/4} \rfloor + 1 \leq d \leq \lfloor n^{1/2} \rfloor$, then $s(G) \leq 48n/d + 1$.*
3. *If $d \geq \lfloor n^{1/2} \rfloor + 1$, then $s(G) \leq 240(\log n)n/d + 1$.*

Cuckler and Lazebnik [122] further obtained the following result:

Theorem 145 *Let G be a graph of order n :*

1. *If $\delta \geq 10n^{3/4} \log^{1/4} n$, then $s(G) \leq 48(n/\delta) + 6$.*
2. *If G is d -regular with $d \geq 10^{4/3} n^{2/3} \log^{1/3} n$, then $s(G) \leq 48(n/d) + 6$.*

However, these results do not confirm even a weaker form of [Conjecture 20](#), namely, $s(G) \leq c_1 n / d + c_2$ holds for all d -regular graphs of order n , with c_1 and c_2 being absolute positive constants. A big step toward this direction has been achieved by Przybyło, who shows that actually constants c_1 and c_2 exist, both for regular graphs in [295] and general graphs in [296].

Theorem 146 *Let G be a d -regular graph of order n with no isolated vertices or edges. Then $s(G) < 16n/d + 6$.*

Theorem 147 *Let G be a graph of order n with no isolated vertices or edges. Then $s(G) < 112n/\delta + 28$.*

Recently, Kalkowski et al. [218] obtained a much better upper bound.

Theorem 148 *Let G be a graph of order n and of minimum degree δ . Then $s(G) \leq 6\lceil n/\delta \rceil$.*

Recall that n_i figures the number of vertices of degree i in a graph G . Then a simple counting argument shows that

$$s(G) \geq \beta(G) = \max_k \left\lceil \frac{1}{k} \sum_{i=1}^k n_i \right\rceil.$$

Lehel [245] felt that $\beta(G)$ with addition of an absolute constant was enough for $s(G)$. Namely,

Conjecture 22 *If G is a connected graph, then $s(G) \leq \beta(G) + c$ for some absolute constant c .*

Note that if G is r -regular with $r \geq 2$, then $\beta(G) = \lceil n/r \rceil$. Therefore, [Conjecture 20](#) is a special case of [Conjecture 22](#).

Baril et al. [33] studied the irregularity strength of circulant graphs of degree 4. Let $C_n(1, k)$ denote the circulant graph which consists of a cycle of length n plus chords joining vertices at distance k on the cycle.

Theorem 149 *For $n \geq 4k + 1$ and $k \geq 2$,*

$$s(C_n(1, k)) = \left\lceil \frac{n+3}{4} \right\rceil.$$

For grids, Dinitz et al. [127] proved that for every positive integer d , $s(X_{m,m}) = \beta(X_{m,n})$ or $\beta(X_{m,n}) + 1$ for all but a finite number of $m \times n$ grids $X_{m,n}$ where $n-m = d$. This finite number was computed in [127] to at most $d+13$. Togni [336] proved that the irregularity strength of the toroidal grid $C_m \times C_n$ is $\lfloor (mn+3)/4 \rfloor$.

Amar and Tongi [24] showed that $s(T) = \beta(T) = n_1$ for all trees T with $n_2 = 0$ and $n_1 \geq 3$. For general trees, it is not even the case that $s(T)$ is within an additive constant of n_1 . Bohman and Kravitz [52] presented an infinite sequence of trees with irregularity strength converging to $\frac{11-\sqrt{5}}{8}n_1 > n_1 > \frac{n_1+n_2}{2}$. Ferrara et al. [146] presented an iterative algorithm to show that $s(T) = \beta(T)$ for another class of trees, but this time, $n_1 < n_2$, that is, $s(T) = \lceil \frac{n_1+n_2}{2} \rceil$.

If the point-distinguishing chromatic index problem is modified so that for any two vertices $u, v \in V(G)$, the multiset of all edge weights appearing on edges incident to u is different from the multiset of all edge weights appearing on edges incident to v (where the multiplicity of an element is exactly how many times that edge weight appears on edges incident to the corresponding vertex). For a given graph G , the smallest k such that a mapping satisfying these conditions exists is denoted by $c(G)$. It is easy to see that $c(G) \leq s(G)$. Aigner, Triesch, and Tuza introduced this problem in [9]. They proved the following result for regular graphs:

Theorem 150 *If G is a regular graph, then there exist constants c_1, c_2 , such that $c_1 n^{1/r} \leq c(G) \leq c_2 n^{1/r}$.*

For other known results of parameter $s(G)$, see a nice survey paper of Lehel [245].

4.2.4 On 1, 2, 3-Conjecture

Motivated by the results on irregularity strength, Karoński et al. [221] studied the improper edge-weighting problem where it is only required that for adjacent vertices u, v ,

$$\sum_{e \ni u} \phi(e) \neq \sum_{e \ni v} \phi(e).$$

The smallest integer k such that there is a mapping ϕ satisfying the above condition is denoted by $\mu(G)$ and is called the *vertex-coloring k -edge-weighting*. Notice that $\mu(G) \leq s(G)$ for every nontrivial graph G . Karoński et al. [221] proved that $\mu(G) \leq 3$ for graphs G without an edge component and having chromatic

number $\chi(G) \leq 3$ and conjectured that $\mu(G) \leq 3$ for all graphs G without an edge component. This is sometimes referred to as 1,2,3-conjecture.

Conjecture 23 *If G is a graph without isolated edges, then $\mu(G) \leq 3$.*

Conjecture 23 has been attacked primarily in two ways. The first is to show it being true for smaller classes of graphs and the second is to find general upper bound on $\mu(G)$. Karoński et al. [221] first proved that for all graphs G without an edge component, $\mu(G) \leq 213$. Addario-Berry et al. [3] improved this upper bound to 30, which was improved later by Addario-Berry et al. [4] to 16 and by Wang and Yu [376] to 13. More recently, Kalkowski et al. [217] improved the bound to 5, based on a method developed in [216].

Note that 2 is not sufficient as seen for instance in complete graphs. Even if it is still far from providing a positive answer to the conjecture, actually $\mu(G) \leq 2$ for many graphs. In fact, Addario-Berry et al. [4] showed that $\mu(G) \leq 2$ for almost all graphs.

Theorem 151 *Let G be a random graph chosen from $G_{n,p}$ for constant $0 < p < 1$. Then there exists, a.a.s., a vertex-coloring 2-edge-weighting for G . In fact, there exists a 2-edge-weighting such that the colors of two adjacent vertices are distinct mod $2\chi(G)$.*

In [99], Chang et al. gave some sufficient conditions for bipartite graphs with $\mu(G) \leq 2$.

Theorem 152 *Every connected bipartite graph $G = (A, B; E)$ without isolated edges admits $\mu(G) \leq 2$ if one of following conditions holds:*

1. $|A|$ or $|B|$ is even.
2. $\delta(G) = 1$.
3. $\lfloor d(u)/2 \rfloor + 1 \neq d(v)$ for any edge of G .

Also, they proposed the following problem.

Problem 6 *Characterize 2-colorable graphs G (i.e., bipartite graphs) with $\mu(G) = 2$.*

Lu et al. [260] continued to consider this problem. They also obtained several sufficient conditions for graphs with $\mu(G) \leq 2$. In particular, they showed that 3-connected bipartite graphs admit $\mu(G) \leq 2$, and every r -regular nice bipartite graph ($r \geq 3$) has $\mu(G) \leq 2$. There exist infinitely many bipartite graphs (e.g., generalized θ -graphs) which are 2-connected and have a vertex-coloring 3-edge-weighting but not a vertex-coloring 2-edge-weighting.

Bartnicki et al. [34] considered the choosability version of edge weighting. A graph is said to be k -edge-weight choosable if the following holds: for any list assignment L which assigns to each edge e a set $L(e)$ of k real numbers, G has a proper edge-weighting f such that $f(e) \in L(e)$ for each edge e . They conjectured

that every graph without isolated edges is 3-edge weight choosable and verified the conjecture for complete graphs, complete bipartite graphs, and some other graphs.

Similar to the problem posed in [8], it can be considered a multiset version of vertex coloring k -edge-weighting. In this case, for any edge $uv \in E(G)$, the multiset of weights on edges incident to u is different from the multiset of weights on edges incident to v . A k -edge-weighting satisfying the above condition is called as *vertex-coloring k -edge partition*.

Karoński et al. [221] proved that if $\chi(G) \leq 3$, then three colors are enough for a graph. Formally, they proved the following result.

Theorem 153 *Let G be a connected graph with at least three vertices. If $\chi(G) \leq 3$, then G permits a vertex-coloring 3-edge partition.*

Addario-Berry et al. [2] proved the following two theorems concerning this version of edge-weighting.

Theorem 154 *If G is a graph without isolated edges, then G permits a vertex-coloring 4-edge partition.*

Theorem 155 *If G is a graph without isolated edges and with minimum degree 1,000, then G permits a vertex-coloring 3-edge partition.*

Recently, Bian et al. [49] improved their results.

Theorem 156 *If G is a graph without isolated edges and with minimum degree 188, then G permits a vertex-coloring 3-edge partition.*

It is unknown if there exists a graph G permits a vertex-coloring 4-edge partition. If such a graph exists, then by [49], $\chi(G) \geq 4$ and $\delta(G) < 188$.

4.3 Some Related Topics

All of the previous vertex-distinguishing edge-weighting problems can be generalized to the vertex-distinguishing total-weighting problems. Given a graph G , a k -*total-weighting* is a mapping $\phi : V(G) \cup E(G) \rightarrow \{1, 2, \dots, k\}$. Given a graph G and a total-weighting ϕ of G , for $v \in V(G)$, let $S_\phi(v) = \{\phi(e) | v \in e\} \cup \{\phi(v)\}$ be the set of weights appearing on the vertex v and the edges incident to v under the total-weighting ϕ . A k -total-weighting is *proper*, if $\phi(e) \neq \phi(f)$ for any pair of adjacent edges e, f ; $\phi(v) \neq \phi(e)$ for any vertex v and incident edge e ; and $\phi(u) \neq \phi(v)$ for any two adjacent vertices u, v .

4.3.1 Adjacent Vertex-Distinguishing Total Coloring

In [395], Zhang et al. defined the *adjacent vertex-distinguishing total chromatic number* of a graph G , denoted by $\chi''_{\text{avd}}(G)$, which is the smallest integer k such that there exists a proper total k -coloring ϕ of G such that for any edge $uv \in E(G)$, $S_\phi(u) \neq S_\phi(v)$. The authors [395] determined the adjacent vertex-distinguishing total chromatic numbers for some special classes of graphs including cycles and complete graphs. Based on these results, they made the following conjecture:

Conjecture 24 *If G is a connected graph with $n \geq 2$ vertices, $\chi''_{\text{avd}}(G) \leq \Delta + 3$.*

If Conjecture 24 is true, then the upper bound $\Delta + 3$ is tight, for instance, a complete graph on odd order can attain this bound. Chen [102] further constructed a class of graphs, that is, the joint graphs $sP_3 \vee K_t$, attaining the upper bound. Chen [102], and independently Wang [347], confirmed Conjecture 24 for graphs G with $\Delta \leq 3$. Hulgan [207] presented a more concise proof for this result and proposed the following question:

Question 8 *For a graph G with $\Delta = 3$, is the bound $\chi''_{\text{avd}}(G) \leq 6$ sharp?*

The following result follows immediately from the definitions of parameters under consideration.

Proposition 3 *For any graph G , $\chi''_{\text{avd}}(G) \leq \chi(G) + \chi'(G)$.*

By Proposition 3, Conjecture 24 holds for all bipartite graphs. For planar graphs, by the Four-Color Theorem and the Vizing's Theorem on edge coloring, $\chi''_{\text{avd}}(G) \leq \Delta + 5$. Moreover, if G is of Class 1, that is, $\chi'(G) = \Delta$, then $\chi''_{\text{avd}}(G) \leq \Delta + 4$.

Coker and Johannson [116] used a probabilistic method to give an upper bound for $\chi''_{\text{avd}}(G)$.

Theorem 157 *There exists a constant $c > 0$ such that for every graph G , $\chi''_{\text{avd}}(G) \leq \Delta + c$.*

For graphs of larger maximum degree, Liu et al. [252] showed the following result.

Theorem 158 *If G is a graph with Δ sufficiently large and $\delta \geq 32\sqrt{\Delta \ln \Delta}$, then $\chi''_{\text{avd}}(G) \leq \Delta + 10^{26} + 2\sqrt{\Delta \ln \Delta}$.*

For graphs with the smaller maximum average degree, Wang and Wang [370] verified Conjecture 24. More precisely, the following theorem and corollary were showed:

Theorem 159 *Let G be a graph:*

1. *If $\text{mad}(G) < 3$ and $\Delta \geq 5$, then $\Delta + 1 \leq \chi''_{\text{avd}}(G) \leq \Delta + 2$; and $\chi''_{\text{avd}}(G) = \Delta + 2$ if and only if G contains adjacent vertices of maximum degree.*
2. *If $\text{mad}(G) < 3$ and $\Delta = 4$, then $\chi''_{\text{avd}}(G) \leq 6$.*
3. *If $\text{mad}(G) < \frac{8}{3}$ and $\Delta = 3$, then $\chi''_{\text{avd}}(G) \leq 5$.*

As an immediate consequence of [Theorem 159](#), the following corollary holds:

Corollary 11 *Let G be a planar graph with girth g :*

1. *If $g \geq 6$ and $\Delta \geq 5$, then $\Delta + 1 \leq \chi''_{\text{avd}}(G) \leq \Delta + 2$; and $\chi''_{\text{avd}}(G) = \Delta + 2$ if and only if G contains adjacent vertices of maximum degree.*
2. *If $g \geq 6$ and $\Delta = 4$, then $\chi''_{\text{avd}}(G) \leq 6$.*
3. *If $g \geq 8$ and $\Delta = 3$, then $\chi''_{\text{avd}}(G) \leq 5$.*

Wang and Wang [373] also completely characterized the adjacent vertex-distinguishing total chromatic number of outerplanar graphs by showing the following:

Theorem 160 *Let G be an outerplanar graph with $\Delta \geq 3$. Then $\Delta + 1 \leq \chi''_{\text{avd}}(G) \leq \Delta + 2$; and $\chi''_{\text{avd}}(G) = \Delta + 2$ if and only if G contains adjacent vertices of maximum degree.*

In [368], [Theorem 160](#) was generalized to the following form:

Theorem 161 *Let G be a K_4 -minor-free graph with $\Delta \geq 3$. Then $\Delta + 1 \leq \chi''_{\text{avd}}(G) \leq \Delta + 2$; and $\chi''_{\text{avd}}(G) = \Delta + 2$ if and only if G contains adjacent vertices of maximum degree.*

When studying the adjacent vertex-distinguishing edge-coloring of hypercubes, Chen and Guo [103] considered also the adjacent vertex-distinguishing total coloring of this kind of graphs. Their result is stated below:

Theorem 162 *If Q_d is a hypercube of dimension $d \geq 2$, then $\chi''_{\text{avd}}(Q_d) = d + 2$.*

4.3.2 Total Vertex Irregularity Strength

Przybyło and Woźniak [297] introduced an improper k -total-weighting question in the light of the problem posed by Karoński et al. [221]. A k -total-weighting of a graph G is called *neighbor-distinguishing* (or *vertex-coloring*, see [4]) if for every edge uv ,

$$\phi(u) + \sum_{e \ni u} \phi(e) \neq \phi(v) + \sum_{e \ni v} \phi(e).$$

If it exists, then G permits a neighbor-distinguishing k -total-weighting. The smallest integer k for which G permits a neighbor-distinguishing k -total-weighting is called *total vertex irregularity strength*, denoted by $\tau(G)$.

Note that if a graph permits a vertex-coloring k -edge-weighting, then it also permits a neighbor-distinguishing k -total-weighting (it is enough to put ones at all vertices). Thus $\tau(G) \leq 5$ for all graphs. Przybyło and Woźniak [297] made the following conjecture.

Conjecture 25 *Every simple graph permits a neighbor-distinguishing 2-total-weighting.*

In [297], the authors gave a positive answer to [Conjecture 25](#) whenever G is 3-colorable, complete, or 4-regular. They also presented upper bounds 11 or

$\lfloor \frac{\chi(G)}{2} \rfloor + 1$ for $\tau(G)$. Przybyło [294] showed that $\tau(G) \leq 7$ for all regular graphs. Later, a big breakthrough is due to Kalkowski [216], who proved that $\tau(G) \leq 3$ for all graphs. Anholcer et al. [26] presented an upper bound of $\tau(G) \leq 3\lceil n/\delta \rceil + 1$ for each graph G of order n and with $\delta > 0$.

The values of $\tau(G)$ for some special graphs are computed. Kristiana and Slamin [239] computed $\tau(G)$ for wheels W_n , fans F_n , suns S_n , and friendship graphs f_n , namely, $\tau(W_n) = \lceil \frac{n+3}{4} \rceil$ for $n \geq 3$, $\tau(F_n) = \lceil \frac{n+2}{4} \rceil$ for $n > 3$, $\tau(S_n) = \lceil \frac{n+1}{2} \rceil$ for $n \geq 3$, and $\tau(f_n) = \lceil \frac{2n+2}{3} \rceil$ for all n .

The choosability version of total weighting was studied in [383] and in [298]. For positive integers k, k' , a (k, k') -total list assignment is a mapping L which assigns to each vertex v a set $L(v)$ of k real numbers as permissible weights and assigns to each edge e a set $L(e)$ of k' real numbers as permissible weights. A graph is called (k, k') -totally weight choosable if for any (k, k') -total list assignment L , G has a proper total-weighting f such that $f(y) \in L(y)$ for all $y \in V(G) \cup E(G)$. It was known in [383] that a graph is $(k, 1)$ -totally weight choosable if and only if it is k -choosable. The following conjectures were proposed in [383] by Wong and Zhu.

Conjecture 26 *Every graph is $(2, 2)$ -totally weight choosable.*

Conjecture 27 *Every graph with no isolated edges is $(1, 3)$ -totally weight choosable.*

However, it is unknown if there are constants k, k' such that every graph is (k, k') -totally weight choosable. It was shown in [383] that complete graphs, trees, cycles, and generalized θ -graphs are $(2, 2)$ -totally weight choosable, $K_{2,n}$ are $(1, 2)$ -totally weight choosable, and $K_{3,n}$ are $(2, 2)$ -totally weight choosable. Wong et al. [382] introduced a method, the max-min weighting method, for finding proper L -total weightings of graphs. Using this method, they proved that $K_{n,m,1,1,\dots,1}$ are $(2, 2)$ -totally weight choosable, and complete bipartite graphs other than K_2 are $(1, 2)$ -totally weight choosable.

5 $L(p, q)$ -Labeling of Graphs

Motivated by the frequency channel assignment problem, Griggs and Yeh [169] introduced the $L(2, 1)$ -labeling of graphs. This notion was subsequently generalized to the $L(p, q)$ -labeling problem of graphs. Let p and q be two nonnegative integers. An $L(p, q)$ -labeling of a graph G is a function f from its vertex set $V(G)$ to the label set $\{0, 1, \dots, k\}$ such that $|f(x) - f(y)| \geq p$ if x and y are adjacent and $|f(x) - f(y)| \geq q$ if x and y are at distance 2. The $L(p, q)$ -labeling number $\lambda_{p,q}(G)$ of G is the smallest integer k such that G has an $L(p, q)$ -labeling f with $\max\{f(v) | v \in V(G)\} = k$.

The $L(p, q)$ -labeling of graphs has been studied rather extensively in recent years [98, 219, 276, 313, 348]. In particular, the case $(p, q) = (p, 1)$ with $p \in \{0, 1, 2\}$ has received much more attention.

5.1 $L(2, 1)$ -Labeling

In 1991, Roberts [304] proposed a variation of the frequency assignment problem in which *close* transmitters must receive different frequencies and *very close* transmitters must receive frequencies that are at least two units apart. To translate the problem into the language of graph theory, the transmitters are represented by the vertices of a graph; two vertices are *very close* if they are adjacent and *close* if they are at distance 2 in the graph. This is indeed the case $L(2, 1)$ -labeling problem of graphs. Along this direction, Griggs and Yeh [169] investigated the $L(2, 1)$ -labeling problem. For a latest survey on this field, see [388].

There are a considerable number of papers studying $L(2, 1)$ -labelings (see [97, 98, 169, 214, 308]). Most of the papers considered the values of $\lambda_{2,1}(G)$ on particular classes of graphs G ; however, Griggs and Yeh [169] provided an upper bound $\Delta^2 + 2\Delta$ for general graphs with maximum degree Δ . Later, Chang and Kuo [98] improved it to $\Delta^2 + \Delta$. Recently, Král and Škrekovski [237] reduced the bound to $\Delta^2 + \Delta - 1$.

In general, Griggs and Yeh [169] suggested the following conjecture:

Conjecture 28 ($L(2, 1)$ -Labeling Conjecture) *For any graph G with $\Delta \geq 2$, $\lambda_{2,1}(G) \leq \Delta^2$.*

It is obvious that [Conjecture 28](#) cannot hold for $\Delta = 1$, as $\Delta(K_2) = 1$ but $\lambda_{2,1}(K_2) = 2$. However, if G is a diameter 2 graph, then $\lambda_{2,1}(G) \leq \Delta^2$. Besides, the upper bound is attainable by Moore graphs (diameter 2 graph with order $\Delta^2 + 1$). Moreover, in [169], the authors proved that determining $\lambda_{2,1}(G)$ of any graph G is NP-complete, and Bodlaender et al. [51] showed that the problem of finding the minimum number $\lambda_{2,1}$ for planar graphs, bipartite graphs, chordal graphs, and split graphs is also NP-complete. Fiala and Kratochvíl [150] even proved that for every integer $p \geq 3$, it is still NP-complete to decide whether a p -regular graph admits an $L(2, 1)$ -labeling of span (at most) $p + 2$.

For general graphs, in [166], Gonçalves constructed a labelling algorithm to get a little bit better upper bound:

Theorem 163 *For any graph G with $\Delta \geq 2$, $\lambda_{2,1}(G) \leq \Delta^2 + \Delta - 2$.*

Recently, Havet et al. [182] showed that [Conjecture 28](#) is true for any graph G with sufficiently large Δ , for example, $\Delta \geq 10^{69}$. This is the first paper addressing the solution of [Conjecture 28](#) for general graphs, although it does not completely solve it. More generally, they showed the following:

Theorem 164 For any positive integer d , there exists a constant Δ_d such that every graph with $\Delta \geq \Delta_d$ has an $L(d, 1)$ -labeling with span at most $\Delta^2 + 1$.

This yields that, for each positive integer d , there is an integer C_d such that every graph has an $L(d, 1)$ -labeling with span at most $\Delta^2 + C_d$.

In the following, it is concerned with the $L(2, 1)$ -labeling of some special graphs such as planar graphs, outerplanar graphs, K_4 -minor-free graphs, trees, and chordal graphs. Some recent results will be displayed.

5.1.1 Planar Graphs

For planar graphs, Bella et al. [47] showed that [Conjecture 28](#) holds for all planar graphs G with $\Delta \geq 4$. Kang [219] showed for every subcubic Hamiltonian graph G , $\lambda_{2,1}(G) \leq 9$, that is, [Conjecture 28](#) holds for this type of graphs. Heuvel and McGuinness [189] proved that $\lambda_{p,q}(G) \leq (4q - 2)\Delta + 10p + 38q - 23$ for every planar graph G . It implies that $\lambda_{2,1}(G) \leq 2\Delta + 35$ for a planar graph G . Molloy and Salavatipour [276] improved this result to $\lambda_{p,q}(G) \leq q\lceil\frac{5}{3}\Delta\rceil + 18p + 77q - 18$ for every planar graph G . It implies that $\lambda_{2,1}(G) \leq \lceil\frac{5}{3}\Delta\rceil + 95$ for a planar graph G . By considering planar graphs without short cycles, Wang and Cai [351] showed that every planar graph G without 4-cycles has $\lambda_{p,q}(G) \leq \min\{(8q - 4)\Delta + 8p - 6q - 1, (2q - 1)\Delta + 10p + 84q - 47\}$. It means that $\lambda_{2,1}(G) \leq \min\{4\Delta + 9, \Delta + 57\}$ for such a planar graph G .

5.1.2 Outerplanar Graphs

For any outerplanar graph G , Bodlaender et al. [50] observed that at most $\Delta + 9$ colors are sufficient to give an $L(2, 1)$ -labeling for outerplanar graph and they suspected this bound is not tight. Later, Calamoneri and Petreschi [93] proved that $\lambda_{2,1}(G) \leq \max\{10, \Delta + 2\}$ for any outerplanar graph G and $\lambda_{2,1}(G) \leq 8$ if $\Delta = 3$. Their results imply that $\lambda_{2,1}(G) \leq \Delta + 6$ for every outerplanar graph G and $\lambda_{2,1}(G) \leq \Delta + 2$ if $\Delta \geq 8$. By investigating the circular distance two labeling of graphs, Liu and Zhu [253] also proved independently that $\lambda_{2,1}(G) \leq \Delta + 6$ for an outerplanar graph G and $\lambda_{2,1}(G) \leq \Delta + 2$ if $\Delta \geq 15$. Wang and Luo [363] improved these results by showing the following: (1) $\lambda_{2,1}(G) \leq \Delta + 5$ for any outerplanar graph G and (2) $\lambda_{2,1}(G) \leq 7$ for an outerplanar graph G with $\Delta = 3$.

In [50], the authors conjectured that every outerplanar graph G has $\lambda_{2,1}(G) \leq \Delta + 2$. Calamoneri and Petreschi [93] disproved this conjecture by constructing an outerplanar graph G with $\Delta = 3$ and $\lambda_{2,1}(G) = 6$ and further conjectured that every outerplanar graph G with $\Delta \geq 4$ has $\lambda_{2,1}(G) \leq \Delta + 2$. Unfortunately, Wang and Luo [363] constructed another example of outerplanar graph G with $\Delta = 4$ and $\lambda_{2,1}(G) = 7$ to disprove their new conjecture. However, it is yet unknown if every outerplanar graph G with $\Delta \in \{5, 6, 7\}$ has $\lambda_{2,1}(G) \leq \Delta + 2$.

5.1.3 K_4 -Minor Free Graphs

Wang and Wang [369] first studied the $L(p, q)$ -labeling of K_4 -minor-free graphs. Their result is as follows:

Theorem 165 Let G be a K_4 -minor-free graph. If $p + q \geq 3$, then $\lambda_{p,q}(G) \leq 2(2p - 1) + (2q - 1)\lfloor \frac{3\Delta}{2} \rfloor - 2$.

Corollary 12 If G is a K_4 -minor-free graph, then $\lambda_{2,1}(G) \leq \lfloor \frac{3\Delta}{2} \rfloor + 4$.

It should be pointed out that [Theorem 165](#) is a generalization of results in [251]. [Corollary 12](#) implies that [Conjecture 28](#) holds for all K_4 -minor-free graphs. More recently, Král and Nejedly [236] showed that there exists Δ_0 such that for every K_4 -minor-free graph G with $\Delta \geq \Delta_0$, $\lambda_{p,q}(G) \leq q\lfloor \frac{3\Delta}{2} \rfloor$, which is best possible.

5.1.4 Trees and Chordal Graphs

Griggs and Yeh [169] proved that every tree T satisfies $\Delta + 1 \leq \lambda_{2,1}(T) \leq \Delta + 2$. Chang and Kuo [98] presented a polynomial algorithm for determining the precise value of $\lambda_{2,1}(T)$ for any tree T . Wang [348] provided a sufficient condition for a tree T to have $\lambda_{2,1}(T) = \Delta + 1$. More precisely, he showed the following:

Theorem 166 If a tree T contains no two vertices of maximum degree at distance either 1, 2, or 4, then $\lambda_{2,1}(T) = \Delta + 1$.

[Theorem 166](#) is the best possible in the sense that for each $\Delta \geq 3$, there exists a tree T of maximum degree Δ such that $\lambda_{2,1}(T) = \Delta + 2$ and the distance between any two vertices of maximum degree is at least 4. Such examples of trees were constructed in [348].

To attack [Conjecture 28](#), Sakai [308] considered the class of chordal graphs. She showed that $\lambda_{2,1}(G) \leq (\Delta + 3)^2/4$ for any chordal graph G . For a unit interval graph G , which is a very special chordal graph, she also proved that $2\chi(G) - 2 \leq \lambda_{2,1}(G) \leq 2\chi(G)$. In [235], Král showed that $\lambda_{2,1}(G) \leq \frac{\Delta^{3/2}}{\sqrt{6}} + O(\Delta)$ if G is a chordal graph. Besides, he conjectured the following:

Conjecture 29 For a chordal graph G , $\lambda_{2,1}(G) \leq \frac{2\Delta^{3/2}}{3\sqrt{3}} + O(\Delta) - 1$.

For a detailed survey on $L(2, 1)$ -labeling problem of graphs, see [92].

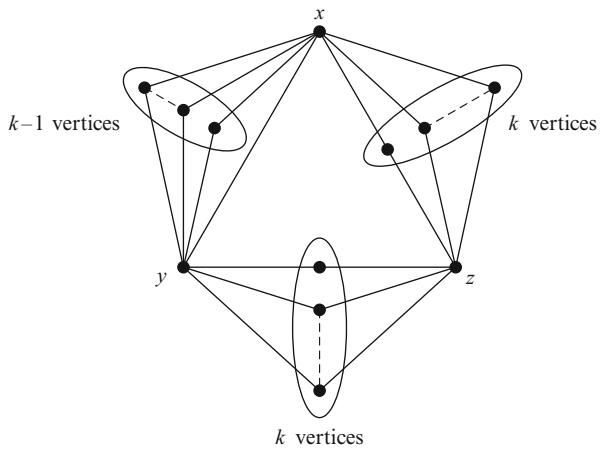
5.2 Coloring the Square of Graphs

The *square* G^2 of a graph G is the graph defined on the vertex set $V(G)$ such that two vertices u and v are adjacent in G^2 if and only if $1 \leq d(u, v) \leq 2$ in G . A mapping ϕ is a k -coloring of G^2 if and only if $\phi(u) \neq \phi(v)$ whenever $1 \leq d(u, v) \leq 2$ in G .

Since G^2 contains a clique of order at least $\Delta(G) + 1$, $\chi(G^2) \geq \Delta(G) + 1$ for any graph G . If G is a tree, then $\chi(G^2) = \Delta(G) + 1$. On the other hand, $\Delta(G^2) \leq \Delta^2(G)$. Hence, $\chi(G^2) \leq \Delta^2(G) + 1$ for any graph G . The 5-cycle and Petersen graph G satisfy $\chi(G^2) = \Delta^2(G) + 1$. By the definition of $L(p, q)$ -labeling, $\lambda_{1,1}(G) = \chi(G^2) - 1$.

In 1977, Wegner [378] first investigated the chromatic number of the square of a planar graph. He proved that $\chi(G^2) \leq 8$ for every planar graph G with $\Delta = 3$ and

Fig. 3 A planar graph with $\Delta = 2k$ that needs at least $3k + 1 = \frac{3}{2}\Delta + 1$ colors



conjectured that the upper bound could be reduced to 7, which has been confirmed by Thomassen [332]. Moreover, Montassier and Raspaud [281] proved that 7 colors are necessary by giving an example. For planar graphs of maximum degree $\Delta \geq 4$, Wegner [378] proposed the following conjecture.

Conjecture 30 *Let G be a planar graph with maximum degree Δ . Then*

$$\chi(G^2) \leq \begin{cases} \Delta + 5, & \text{if } 4 \leq \Delta \leq 7; \\ \lfloor \frac{3}{2}\Delta \rfloor + 1, & \text{if } \Delta \geq 8. \end{cases}$$

Wegner also gave some bounds for the case $\Delta \leq 7$ and constructed planar graphs to attain the conjectured bounds. For $\Delta \geq 8$ even, these examples are sketched below. The graph in Fig. 3 has maximum degree $2k$, and yet all vertices except z are adjacent in its square. Hence, to color these $3k + 1$ vertices, at least $3k + 1 = \frac{3}{2}\Delta + 1$ colors is required.

Many upper bounds on $\chi(G^2)$ for planar graphs in terms of Δ have been obtained in the past 15 years. In [6], Agnarsson and Halldorsson showed the following:

Theorem 167 *For every planar graph G with maximum degree $\Delta \geq 749$, $\chi(G^2) \leq \lfloor \frac{9}{5}\Delta \rfloor + 2$.*

A similar result was obtained by Borodin et al. [60].

Theorem 168 *For every planar graph G with maximum degree $\Delta \geq 47$, $\chi(G^2) \leq \lceil \frac{9}{5}\Delta \rceil + 1$.*

Nearly at the same time, Heuvel and McGuinness [189] gave a general upper bound without maximum degree restriction.

Theorem 169 For every planar graph G with maximum degree Δ , $\chi(G^2) \leq 2\Delta + 25$.

The currently best known upper bound on this parameter for planar graphs was obtained by Molloy and Salavatipour [276].

Theorem 170 For every planar graph G with maximum degree Δ , $\chi(G^2) \leq \lceil \frac{5}{3}\Delta \rceil + 78$.

Also, Molloy and Salavatipour [276] showed that if G is a planar graph with maximum degree $\Delta \geq 241$, then $\chi(G^2) \leq \lceil \frac{5}{3}\Delta \rceil + 25$. Lih, Wang, and Zhu [251] studied the coloring of the squares of K_4 -minor-free graphs. More precisely, they proved the following:

Theorem 171 Let G be a K_4 -minor-free graph with maximum degree Δ . Then

$$\chi(G^2) \leq \begin{cases} \Delta + 3, & \text{if } 2 \leq \Delta \leq 3; \\ \lfloor \frac{3}{2}\Delta \rfloor + 1, & \text{if } \Delta \geq 4. \end{cases}$$

The upper bound in [Theorem 171](#) is tight. For $\Delta = 2$, it is easy to obtain that $\Delta(C_5) = 2$ and $\chi(C_5^2) = 5$. For $\Delta = 3$, let G be the graph consisting of three internally disjoint paths joining two vertices x and y , where two of the paths are of length 2 and the third one is of length 3. Then $\Delta = 3$ and $\chi(G^2) = \chi(K_6) = 6$. For $\Delta = 2k \geq 4$, let G_{2k} be the graph consisting of k internally disjoint paths joining x and y , k internally disjoint paths joining x and z , and k internally disjoint paths joining y and z . All these paths are of length 2, except one path joining x and y is of length 1, and one path joining x and z is of length 1. Then $\Delta(G_{2k}) = 2k$ and $\chi(G_{2k}^2) = \chi(K_{3k+1}) = 3k + 1$. For $\Delta = 2k + 1 \geq 5$, let G_{2k+1} be obtained from G_{2k} by adding a new path of length 2 joining y and z . Then $\Delta(G_{2k+1}) = 2k + 1$ and $\chi(G_{2k+1}^2) = \chi(G_{3k+2}) = 3k + 2$.

In 2001, Kostochka and Woodall [234] investigated the list coloring of square of graphs and raised the following conjecture:

Conjecture 31 For every graph G , $\chi(G^2) = \chi^l(G^2)$.

If true, this conjecture (together with Thomassen's result [332]) implies that every planar graph G with $\Delta = 3$ satisfies $\chi^l(G^2) \leq 7$. Recently, Cranston and Kim [119] proved that every connected graph (not necessarily planar) with $\Delta = 3$ other than the Petersen graph satisfies $\chi^l(G^2) \leq 8$ (and this is best possible by the Wagner graph).

In addition, Cranston and Kim [119] showed that if G is a planar graph with $\Delta = 3$ and girth $g \geq 7$, then $\chi^l(G^2) \leq 7$. Dvořák, Škrekovski, and Tancer showed that if G is a planar graph with $\Delta = 3$ and $g \geq 10$, then $\chi^l(G^2) \leq 6$, which has been improved to $g \geq 9$ in [119].

For planar graphs G with $\Delta \geq 8$, [Conjecture 31](#) leads directly to the following conjecture:

Conjecture 32 For every planar graph G with $\Delta \geq 8$, $\chi^l(G^2) \leq \lfloor \frac{3\Delta}{2} \rfloor + 1$.

Havet et al. [181] announced the following theorem.

Theorem 172 The square of every planar graph G of maximum degree Δ has list chromatic number at most $(1 + o(1))\frac{3}{2}\Delta$. Moreover, given lists of this size, there is a proper coloring in which the colors on every pair of adjacent vertices of G differ by at least $\Delta^{\frac{1}{4}}$.

The $o(1)$ term here is as $\Delta \rightarrow \infty$. The first-order term $\frac{3}{2}\Delta$ is essentially best possible; see the example described above. On the other hand, the term $\Delta^{\frac{1}{4}}$ is probably far from best possible; it was chosen to keep the proof simple.

In what follows, it is going to study the coloring of squares of some special graphs such as hypercubes, outerplanar graphs, chordal graphs, and planar graphs with high girth.

5.2.1 Planar Graphs with High Girth

By investigating the $L(p, q)$ -labeling of planar graphs, Wang and Lih [360] obtained a general result of planar graphs with high girth.

Theorem 173 Let G be a planar graph with maximum degree Δ and girth g . Then

$$\lambda_{p,q}(G) \leq \begin{cases} (2q - 1)\Delta + 4p + 4q - 4, & \text{if } g \geq 7; \\ (2q - 1)\Delta + 6p + 12q - 9, & \text{if } g \geq 6; \\ (2q - 1)\Delta + 6p + 24q - 15, & \text{if } g \geq 5. \end{cases}$$

As it is mentioned at the beginning of this section, every proper coloring of the square of a graph is exactly an $L(1, 1)$ -labeling of the graph. So [Theorem 173](#) implies the following:

Corollary 13 Let G be a planar graph with maximum degree Δ and girth g . Then

$$\chi(G^2) \leq \begin{cases} \Delta + 5, & \text{if } g \geq 7; \\ \Delta + 10, & \text{if } g \geq 6; \\ \Delta + 16, & \text{if } g \geq 5. \end{cases}$$

$$\lambda_{2,1}(G) \leq \begin{cases} \Delta + 8, & \text{if } g \geq 7; \\ \Delta + 15, & \text{if } g \geq 6; \\ \Delta + 21, & \text{if } g \geq 5. \end{cases}$$

Note that $\lfloor \frac{3\Delta}{2} \rfloor + 1 \geq \Delta + 5$ when $\Delta \geq 8$, $\lfloor \frac{3\Delta}{2} \rfloor + 1 \geq \Delta + 10$ when $\Delta \geq 18$, and $\lfloor \frac{3\Delta}{2} \rfloor + 1 \geq \Delta + 16$ when $\Delta \geq 30$. Thus, the following result is an immediate consequence of [Corollary 13](#).

Corollary 14 [Conjecture 30](#) holds for planar graphs G with $g \geq 7$, or $g = 6$ and $\Delta \geq 18$, or $g = 5$ and $\Delta \geq 30$.

Borodin et al. [63] proved that if G is a planar graph with maximum degree Δ and girth at least 7, then $\chi(G^2) = \Delta + 1$ whenever $\Delta \geq 30$. On the other hand, there are planar graphs G with girth at most 6, arbitrarily large maximum degree Δ , and $\chi(G^2) > \Delta + 1$. Recently, Dvořák et al. [134] discussed the coloring of the square of a planar graph with girth 6. Their result is as follows:

Theorem 174 *Let G be a planar graph with maximum degree $\Delta \geq 8821$ and girth $g \geq 6$. Then $\chi(G^2) \leq \Delta + 2$.*

This result has been improved and extended by Borodin and Ivanova [69, 70].

Theorem 175 ([69]) *Let G be a planar graph with maximum degree $\Delta \geq 18$ and girth $g \geq 6$. Then $\chi(G^2) \leq \Delta + 2$.*

Theorem 176 ([70]) *Let G be a planar graph with maximum degree $\Delta \geq 36$ and girth $g \geq 6$. Then $\chi^l(G^2) \leq \Delta + 2$.*

Borodin et al. [73] gave a sufficient condition for planar graphs with given maximum degree and girth with respect to the list coloring.

Theorem 177 *For every planar graph G with maximum degree Δ and girth g , $\chi^l(G^2) = \Delta + 1$ in each of the following cases:*

- (1) $\Delta = 3$ and $g \geq 24$.
- (2) $\Delta = 4$ and $g \geq 15$.
- (3) $\Delta = 5$ and $g \geq 13$.
- (4) $\Delta = 6$ and $g \geq 12$.
- (5) $\Delta \geq 7$ and $g \geq 11$.
- (6) $\Delta \geq 9$ and $g = 10$.
- (7) $\Delta \geq 15$ and $g = 8$.
- (8) $\Delta \geq 30$ and $g = 7$.

5.2.2 Outerplanar Graphs

For the class of outerplanar graphs, Lih and Wang [250] proved the following result:

Theorem 178 *Let G be an outerplanar graph with maximum degree Δ . Then*

1. $\chi(G^2) \leq \Delta + 2$ if $\Delta \geq 3$.
2. $\chi(G^2) = \Delta + 1$ if $\Delta \geq 7$.

An alternative proof of Theorem 178 was given in [361]. Moreover, Calamoneri and Petreschi [93] also obtained the same result through an algorithmic aspect.

Wang and Luo [363] extended the result of Theorem 178 to the case $\Delta = 6$. That is, they proved the following:

Theorem 179 *If G is an outerplanar graph with $\Delta = 6$, then $\chi(G^2) = 7$.*

Theorem 179 was independently obtained by Agnarsson and Halldórsson [7]. Moreover, it was shown in [7] that there exist infinitely many biconnected outerplanar graphs G with maximum degree $\Delta = 5$ and $\chi(G^2) = 7 = \Delta + 2$. This shows that Theorems 178 and 179 are the best possible with respect to the requirement of maximum degree.

An outerplanar graph G is said to be *maximal* if $G + xy$ is not outerplanar for any two nonadjacent vertices x and y . Let Q denote a maximal outerplanar graph obtained by adding three chords y_1y_3, y_3y_5, y_5y_1 to a 6-cycle $y_1y_2 \dots y_6y_1$.

Luo [262] characterized completely the chromatic number of the square of a maximal outerplanar graph. More precisely, she showed the following:

Theorem 180 *Let G be a maximal outerplanar graph with maximum degree Δ . Then $\Delta + 1 \leq \chi(G^2) \leq \Delta + 2$; and $\chi(G^2) = \Delta + 2$ if and only if G is isomorphic to Q .*

5.2.3 Other Graphs

The coloring of the square of hypercube Q_d has been studied by Wan in [346]. Such coloring problem is known to be related to the conflict-free channel set assignment for the hypercube cluster interconnection topology. According to the definition of hypercube, the vertices with at most one coordinate equal to form a clique of size $d + 1$ in Q_d^2 ; hence, $\chi(Q_d^2) \geq d + 2$. An independent set in Q_d^2 is precisely an error-correcting code. So, a proper coloring of this graph may be viewed as a covering of the hypercube with codes. Wan [346] constructed a coloring of Q_d^2 using $2^{\lfloor \log_2 d \rfloor + 1}$ colors. Furthermore, he conjectured that this is optimal. Note that Q_d^2 contains a subgraph isomorphic to Q_{d-1}^2 . Thus, it suffices to prove the conjecture when d is a power of 2. The cases $d = 1, 2, 4$ are rather easy to verify. However, this conjecture is false, since it was proved by Royle that $13 \leq \chi(Q_8^2) \leq 14$, see [212].

For a chordal graph G with maximum degree Δ , Král [235] showed that the k th power G^k of G is $O(\sqrt{k}\Delta^{(k+1)/2})$ -degenerate for even values of k and $O(\Delta^{(k+1)/2})$ -degenerate for odd values. In particular, this bounds the chromatic number $\chi(G^k)$ of the k th power of G . The bound proven for odd values of k is the best possible. More generally, the author in [235] showed that $\lambda_{p,q}(G) \leq \lfloor \frac{(\Delta+1)^{3/2}}{\sqrt{6}} \rfloor (2q - 1) + \Delta(2p - 1)$, which implies that $\chi(G^2) \leq \lfloor \frac{(\Delta+1)^{3/2}}{\sqrt{6}} \rfloor + \Delta + 1$. On the other hand, a construction of such graphs with $\lambda_{p,q}(G) \geq \Omega(\Delta^{3/2}q + \Delta p)$ was found.

5.3 Backbone Coloring

Let G be a graph and H a spanning subgraph of G . For an integer $\lambda \geq 2$, a λ -backbone- k -coloring of (G, H) is a mapping $f: V(G) \rightarrow \{1, 2, \dots, k\}$ such that $|f(u) - f(v)| \geq \lambda$ if $uv \in E(H)$ and $|f(u) - f(v)| \geq 1$ if $uv \in E(G) \setminus E(H)$. The λ -backbone chromatic number, denoted by $\chi_\lambda(G, H)$ or $\text{BBC}_\lambda(G, H)$, of (G, H) is the smallest integer k such that (G, H) has a λ -backbone- k -coloring. In particular, when $\lambda = 2$, it is called, for short, the *backbone coloring* and the *backbone chromatic number*, denoted by $\chi_b(G, H)$ or $\text{BBC}(G, H)$.

The backbone coloring of graphs was introduced in 2003 by Broersma et al. [81] and later was investigated by some authors; see [82, 83, 270, 271, 350].

The following result is an easy, useful observation:

Theorem 181 If H is a spanning subgraph of G , then $\chi_b(G, H) \leq 2\chi(G) - 1$.

Some better upper bounds for $\chi_b(G, H)$ can be derived from [Theorem 181](#):

- (i) If G is a planar graph, then $\chi(G) \leq 4$ by the Four-Color Theorem, and hence, $\chi_b(G, H) \leq 7$.
- (ii) If G is a bipartite graph, then $\chi(G) \leq 2$, and hence, $\chi_b(G, H) \leq 3$.
- (iii) If G is a triangle-free planar graph, then $\chi(G) \leq 3$ by [170], and hence, $\chi_b(G, H) \leq 5$.
- (iv) If G is a k -degenerate graph, then $\chi(G) \leq k+1$, and thus, $\chi_b(G, H) \leq 2k+1$.

In particular, if G is a K_4 -minor-free graph (especially, outerplanar graph and series-parallel graph), then G is 2-degenerate, and therefore, $\chi_b(G, H) \leq 5$.

It is easy to verify that if G is a connected non-bipartite graph and H is a connected spanning subgraph of G , then $\chi_b(G, H) \geq 4$. Moreover, if G is a connected graph with at least two vertices and H is a connected spanning subgraph of G , then $\chi_b(G, H) = 3$ if and only if G is a bipartite graph.

Now, consider the relation between the backbone chromatic number and ordinary chromatic number of a graph. How far away from $\chi(G)$ to $\chi_\lambda(G, H)$ in the worst case? To answer this question, some symbols are needed. Suppose that k is a positive integer. Set

$$T_\lambda(k) = \max\{\chi_\lambda(G, T) : G \text{ is a graph with a spanning tree } T, \text{ and } \chi(G) = k\}.$$

$$P_\lambda(k) = \max\{\chi_\lambda(G, P) : G \text{ is a graph with a Hamiltonian path } P, \text{ and } \chi(G) = k\}.$$

$$M_\lambda(k) = \max\{\chi_\lambda(G, M) : G \text{ is a graph with a perfect matching } M, \text{ and } \chi(G) = k\}.$$

$$S_\lambda(k) = \max\{\chi_\lambda(G, S) : G \text{ is a graph with disjoint spanning stars } S, \text{ and } \chi(G) = k\}.$$

If $\lambda = 2$, simply write $T(k)$ for $T_2(k)$, $P(k)$ for $P_2(k)$, $M(k)$ for $M_2(k)$, and $S(k)$ for $S_2(k)$.

Broersma et al. [81, 82] determined the exact values of the first three parameters $T(k)$, $P(k)$, and $M(k)$. More precisely, their results are described in [Theorems 182–184](#).

Theorem 182 $T(k) = 2k - 1$ for all $k \geq 1$.

Theorem 183 For $k \geq 1$, $P(k)$ takes the following values:

- (a) If $1 \leq k \leq 4$, then $P(k) = 2k - 1$.
- (b) $P(5) = 8$ and $P(6) = 10$.
- (c) If $k \geq 7$ and $k = 4t$, then $P(4t) = 6t$.
- (d) If $k \geq 7$ and $k = 4t + 1$, then $P(4t + 1) = 6t + 1$.
- (e) If $k \geq 7$ and $k = 4t + 2$, then $P(4t + 2) = 6t + 3$.
- (f) If $k \geq 7$ and $k = 4t + 3$, then $P(4t + 3) = 6t + 5$.

Theorem 184 For $k \geq 1$, $M(k)$ takes the following values:

- (a) $M(4) = 6$.
- (b) If $k = 3t$, then $M(3t) = 4t$.

- (c) If $k = 4$ and $k = 3t + 1$, then $M(3t + 1) = 4t + 1$.
 (d) If $k = 3t + 2$, then $M(3t + 2) = 4t + 3$.

For a general problem, that is, λ may be equal to or greater than 2, Broersma et al. [83] established the following two results:

Theorem 185 For $\lambda \geq 2$, $M_\lambda(k)$ takes the following values:

$$M_\lambda(k) = \begin{cases} \lambda + k - 1, & \text{if } 2 \leq k \leq \lambda; \\ 2k - 2, & \text{if } \lambda + 1 \leq k \leq 2\lambda; \\ 2k - 3, & \text{if } k = 2\lambda + 1; \\ 2t\lambda, & \text{if } k = t(\lambda + 1) \text{ with } t \geq 2; \\ 2t\lambda + 2c - 1, & \text{if } k = t(\lambda + 1) + c \text{ with } t \geq 2 \text{ and } 1 \leq c \leq \frac{\lambda+3}{2}; \\ 2t\lambda + 2c - 2, & \text{if } k = t(\lambda + 1) + c \text{ with } t \geq 2 \text{ and } \frac{\lambda+3}{2} \leq c \leq \lambda. \end{cases}$$

Theorem 186 For $\lambda \geq 2$, $S_\lambda(k)$ takes the following values:

- (a) $S_\lambda(2) = \lambda + 1$, $S_2(3) = 5$, and $S_2(4) = 6$.
 (b) If $3 \leq k \leq 2\lambda - 3$, then $S_\lambda(k) = \lceil \frac{3k}{2} \rceil + \lambda - 2$.
 (c) If $2\lambda - 2 \leq k \leq 2\lambda - 1$ with $\lambda \geq 3$, then $S_\lambda(k) = k + 2\lambda - 2$.
 (d) If $k = 2\lambda$ with $\lambda \geq 3$, then $S_\lambda(k) = 2k - 1$.
 (e) If $k \geq 2\lambda + 1$, then $S_\lambda(k) = 2k - \lceil \frac{k}{\lambda} \rceil$.

Concerning the computational complexity of backbone coloring, Broersma et al. [81, 83] gave the following two theorems:

Theorem 187 (a) The following problem is polynomially solvable for any $l \leq 4$: Given a graph G and a spanning tree T , decide whether $\chi_b(G, T) \leq l$.
 (b) The following problem is NP-complete for all $l \geq 5$: Given a graph G and a Hamiltonian path P , decide whether $\chi_b(G, T) \leq l$.

Theorem 188 (a) The following problem is polynomially solvable for any $l \leq \lambda + 1$: Given a graph G and a star backbone S , decide whether $\chi_\lambda(G, S) \leq l$.
 (b) The following problem is NP-complete for all $l \geq \lambda + 2$: Given a graph G and a star backbone S , decide whether $\chi_\lambda(G, S) \leq l$.

5.3.1 Split Graphs

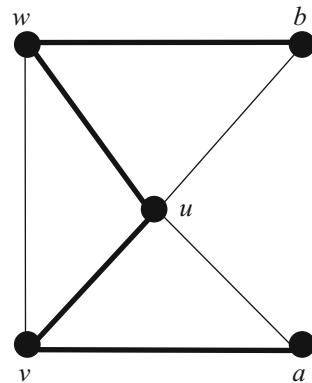
A *split graph* is a graph whose vertex set can be partitioned into a clique and an independent set, with possibly edges in between. Recall that the *size* of a largest clique in G is denoted by $\omega(G)$. It is known that a split graph G is perfect, and hence, $\chi(G) = \omega(G)$. In [81], the authors proved the following:

Theorem 189 Let G be a split graph:

- (a) For every spanning tree T in G , $\chi_b(G, T) \leq \chi(G) + 2$.
 (b) If $\omega(G) \neq 3$, then for every Hamiltonian path P in G , $\chi_b(G, P) \leq \chi(G) + 1$.

Furthermore, they showed that both bounds in **Theorem 189** are tight. Explain here why for split graphs with clique number 3 the statement in **Theorem 189** does

Fig. 4 A split graph G with a Hamiltonian path P such that $\omega(G) = 3$ and $\chi_b(G, P) = 5$



not work. Consider the split graph G on five vertices from Fig. 4. Vertices v, u, w form a clique, vertex a is adjacent to v, u , and vertex b is adjacent to u, w . The clique number (and hence the chromatic number) of this graph is equal to 3. Let $P = avuwba$ be a Hamiltonian path. Then $\chi_b(G, P) > \chi(G) + 1 = 4$. Assume to the contrary that (G, P) has a backbone coloring with colors 1, 2, 3, and 4. It is easy to see that u cannot be colored with color 2 or 3; otherwise, v and w are forced to be colored with the same color, an obvious contradiction. Now suppose that u is colored with color 1 (the case when u is colored with color 4 is similar). Then one of its neighbors in P must have color 3 and the other one color 4. Without loss of generality, assume that v has color 3. Vertex a is adjacent in P to v , so the colors 2, 3, and 4 are forbidden for a and the only valid color for a is 1. But a is adjacent in G to u which has color 1 as well. This contradiction completes the proof of the claim.

In 2006, Salman [309] investigated the λ -backbone colorings of split graphs with a spanning tree. He obtained the following three theorems, where all the upper bounds are tight.

Theorem 190 *Let $\lambda \geq 2$ and let G be a split graph. For every tree backbone T of G ,*

$$\chi_\lambda(G, T) \leq \begin{cases} 1, & \text{if } \chi(G) = 1; \\ 1 + \lambda, & \text{if } \chi(G) = 2; \\ \chi(G) + \lambda, & \text{if } \chi(G) \geq 3. \end{cases}$$

Theorem 191 *Let $\lambda \geq 2$ and let G be a split graph with $\chi(G) = k \geq 2$. For every matching backbone M of G ,*

$$\chi_\lambda(G, M) \leq \begin{cases} \lambda + 1, & \text{if } k = 2; \\ k + 1, & \text{if } k \geq 4 \text{ and } \lambda \leq \min\{\frac{k}{2}, \frac{k+5}{3}\}; \\ k + 2, & \text{if } k = 9 \text{ or } k \geq 11 \text{ and } \frac{k+6}{3} \leq \lambda \leq \lceil \frac{k}{2} \rceil; \\ \lceil \frac{k}{2} \rceil + \lambda, & \text{if } k = 3, 5, 7 \text{ and } \lambda \geq \lceil \frac{k}{2} \rceil; \\ \lceil \frac{k}{2} \rceil + \lambda + 1, & \text{if } k = 4, 6 \text{ or } k \geq 8 \text{ and } \lambda \geq \lceil \frac{k}{2} \rceil + 1. \end{cases}$$

Theorem 192 Let $\lambda \geq 2$ and let G be a split graph with $\chi(G) = k \geq 2$. For every star backbone S of G ,

$$\chi_\lambda(G, S) \leq \begin{cases} k + \lambda, & \text{if either } k = 3 \text{ and } \lambda \geq 2 \text{ or } k \geq 4 \text{ and } \lambda = 2; \\ k + \lambda + 1, & \text{otherwise.} \end{cases}$$

5.3.2 Halin Graphs

Suppose that T is a tree with no vertex of degree 2 and at least one vertex of degree 3 or more. A *Halin graph* is a plane graph $G = T \cup C$, where C is a cycle connecting the leaves, the vertices of degree 1, of T in the cyclic order determined by the drawing of T . Vertices of C are called *outer vertices* of G , and vertices in $V(G) \setminus V(C)$ are called *inner vertices* of G . A Halin graph G is called a *wheel* if G contains only one inner vertex. When $\Delta = 3$, G is 3-regular. In particular, K_4 is a 3-regular Halin graph. It is easy to see that every Halin graph is 3-connected.

It is easy to show that for a Halin graph G , $\chi(G) = 4$ if G is a wheel of order even and $\chi(G) = 3$ otherwise. Wang et al. [350] discussed the backbone coloring of Halin graphs and got the following result.

Theorem 193 Let $G = T \cup C$ be a Halin graph with $X \cup Y$ as a bipartition of T . Then

1. $4 \leq \chi_b(G, T) \leq 5$.
2. $\chi_b(G, T) = 5$ if and only if $|M(T)|$ is odd, and $M(T) \subseteq X$ or $M(T) \subseteq Y$, where $M(T)$ denotes the set of all leaves of T .

An *almost binary tree* is a tree T with $\Delta = 3$ and $|T| \geq 4$ that contains no 2 vertices. Let Γ denote the set of almost binary trees T with the properties: (1) $|M(T)|$ is odd and (2) $M(T) \subseteq X$ or $M(T) \subseteq Y$, where $X \cup Y$ is a bipartition of T .

Let A denote the graph obtained from a path $uxrys$ by gluing a leaf v at x and a leaf t at y . The vertex r is called the *root* of A . Given an almost binary tree T with a leaf z , A is *attached* to T at z if z is identical to the root r of A . And A is an *end subgraph* of T if $A \subseteq T$ and u, v, s, t are leaves of T and r, x, y are of degree 3 in T . Removing an end subgraph A from T means that six vertices x, y, u, v, s, t are removed from T .

Let Λ' denote the set of graphs G_0, G_1, G_2, \dots , where $G_0 = K_{1,3}$, and G_i is obtained by consecutively attaching two copies of A to the graph G_{i-1} for $i = 1, 2, \dots$

In [350], the authors first showed the equality that $\Lambda' = \Lambda$ and afterward gave the following interesting characterization:

Theorem 194 Let $G = T \cup C$ be a 3-regular Halin graph. Then $4 \leq \chi_b(G, T) \leq 5$; and $\chi_b(G, T) = 5$ if and only if $T \in \Lambda'$, that is, T can be obtained by consecutively attaching $2k$, $k \geq 0$, copies of the graph A to the graph $K_{1,3}$.

Notice that the backbone spanning tree T taken in Theorems 193 and 194 is just the tree in the definition of Halin graph $G = C \cup T$. What happens when picking

up other spanning tree as backbone in a Halin graph? The authors in [350] posed the following problem:

Problem 7 *Given a spanning tree T^* , different from T , in a Halin graph $G = T \cup C$, determine $\chi_b(G, T^*)$.*

It seems very difficult to settle [Problem 7](#). The following is a partial solution by considering a class of special spanning trees [350].

For a Halin graph $G = T \cup C$, take $e' = u'v' \in E(T)$ with $u' \in M(T)$ and $e'' = x''y'' \in E(C)$. So, $x'', y'' \in M(T)$. Let $T' = T - M(T)$. Define a spanning tree T^* of G as follows:

$$T^* = T' \cup (C \setminus \{e''\}) \cup \{e'\}.$$

Theorem 195 *Let $G = T \cup C$ be a Halin graph. Let T^* be a spanning tree of G defined as above. If G satisfies one of the following conditions, then $\chi_b(G, T^*) = 4$:*

1. $|M(T)|$ is even.
2. e' is adjacent to e'' in G .

5.3.3 Other Related Results

In [81], the authors asked the following: Is there a constant c such that $\chi_b(G, T) \leq \chi(G) + c$ for every triangle-free graph G with T as a backbone tree? Miškuf et al. [270] answered negatively this question. More precisely, they proved the following:

Theorem 196 *For every $n \in N$, there exist a (finite) triangle-free graph R_n and its spanning tree T_n such that $\chi_b(R_n, T_n) = 2\chi(R_n) - 1 = 2n - 1$.*

Wang et al. [350] considered the backbone coloring of graphs with small maximum average degree. Their result is stated as follows:

Theorem 197 *If G is a connected graph with $\text{mad}(G) < \frac{5}{2}$, then there exists a spanning tree T of G such that $\chi_b(G, T) \leq 4$.*

Corollary 15 *If G is a connected non-bipartite graph with $\text{mad}(G) < \frac{5}{2}$, then there is a spanning tree T such that $\chi_b(G, T) = 4$.*

Note that every planar graph G with girth $g \geq 10$ satisfies $\text{mad}(G) < \frac{5}{2}$ by [Proposition 2](#). This fact, together with [Corollary 15](#), gives the following fact:

Corollary 16 *If G is a connected non-bipartite planar graph with girth at least 10, then there is a spanning tree T such that $\chi_b(G, T) = 4$.*

Let μ denote the smallest integer k such that for every non-bipartite planar graph G with girth at least k , there exists a spanning tree T of G such that $\chi_b(G, T) = 4$. By virtue of [Corollary 16](#), $\mu \leq 10$. What is the exact value of μ ? Bu and Zhang [87] partially answered this question:

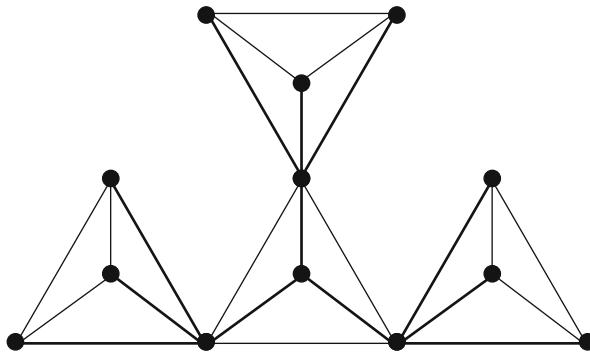


Fig. 5 A planar graph G^* with a spanning tree T^* such that $\chi_b(G^*, T^*) = 6$

Theorem 198 *If G is a connected non-bipartite planar graph without 4-cycles, then there exists a spanning tree T of G such that $\chi_b(G, T) = 4$.*

The following result was given in [350]:

Theorem 199 *There exist infinitely many graphs G with $\Delta \leq 3$ that contain a Hamiltonian path P such that $\chi_b(G, P) = 5$.*

5.3.4 Some Open Problems

The Four-Color Theorem implies that $\chi_b(G, T) \leq 7$ for any planar graph G with an arbitrary spanning tree T . However, this bound 7 is probably not best possible. Can it be improved to 6? The planar graph G^* in Fig. 5, see [82], demonstrates that this bound cannot be improved to 5. Note that the graph G^* consists of four copies of K_4 , each having a $K_{1,3}$ as a spanning tree. In any backbone coloring of such a K_4 with only five colors, its central vertex must receive color 1 or 5. With this observation, it is easy to see that $\chi_b(G^*, T^*) \geq 6$.

Problem 8 *Is $\chi_b(G, T) \leq 6$ for any planar graph G with a spanning tree T ?*

How about the Hamiltonian path case? In this case, it is natural to ask whether the bound 7 is best possible. It can be shown that one cannot improve this bound to 5. Let G_2 be the planar graph from Fig. 6 in which the Hamiltonian path P is indicated by the bold edges. To prove that $\chi_b(G_2, P) \geq 6$, the following claim is needed. Let G_1 be the subgraph of G_2 with the indicated Hamiltonian path P_1 as shown in Fig. 6.

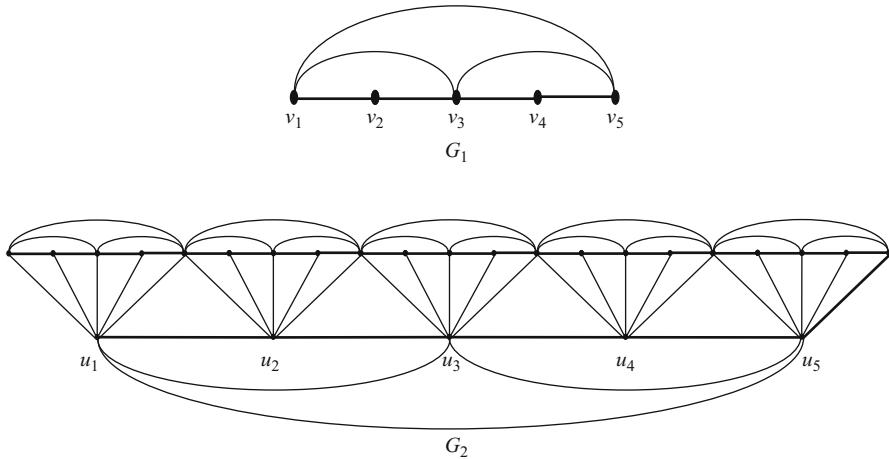


Fig. 6 A planar graph \$G_2\$ with a Hamiltonian path \$P\$ such that \$\chi_b(G_2, P) = 6\$

Claim 1 In any backbone coloring of \$G_1\$ with colors \$\{1, 2, 3, 4, 5\}\$, at least one vertex has color 3.

Proof Suppose that there exists a backbone coloring \$c\$ for \$(G_1, P_1)\$ with colors 1, 2, 4, and 5. By symmetry, assume that \$c(v_1) \in \{1, 2\}\$. Then, it is obvious that \$c(v_2) \in \{4, 5\}\$, \$c(v_3) \in \{1, 2\}\$, and \$c(v_4) \in \{4, 5\}\$. Hence, it follows that \$c(v_4), c(v_5) \in \{4, 5\}\$, contradicting the fact that \$v_1, v_3\$, and \$v_5\$ induce a \$K_3\$ in \$G_1\$. \$\square\$

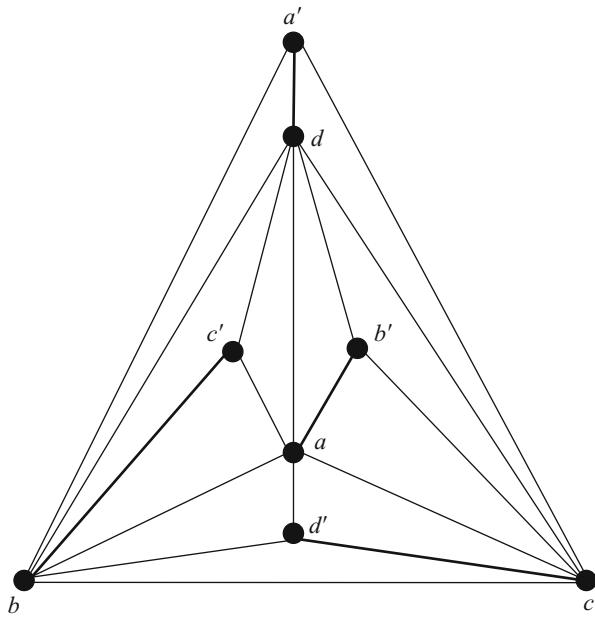
Now suppose that there exists a backbone coloring for \$(G_2, P)\$ with colors \$\{1, 2, 3, 4, 5\}\$. Then **Claim 1** implies that there is a vertex with color 3 among the neighbors of \$u_1\$, hence, \$u_1\$ cannot receive color 3. Similarly, **Claim 1** implies that \$u_2, u_3, u_4\$, and \$u_5\$ cannot receive color 3. But the graph induced by \$\{u_1, u_2, u_3, u_4, u_5\}\$ is isomorphic to \$G_1\$, contradicting **Claim 1**.

Problem 9 Is \$\chi_b(G, P) \leq 6\$ for any planar graph \$G\$ with a Hamiltonian path \$P\$?

What about the perfect matching case? It was shown in [50] that \$\chi_b(G, M) \leq 6\$ for any planar graph \$G\$ with a perfect matching \$M\$. This bound 6 seems to be not best possible. In order to show that this bound cannot be reduced to 4, the authors in [50] constructed an example of planar graph \$G_3\$, as shown in Fig. 7, with an indicated perfect matching \$M = \{ab', bc', cd', da'\}\$. The detailed argument is here omitted.

Problem 10 Is \$\chi_b(G, M) \leq 5\$ for any planar graph \$G\$ with a perfect matching \$M\$?

Fig. 7 A graph G_3 with a perfect matching M such that $\chi_b(G_3, M) = 5$



5.4 Injective Coloring

The injective coloring of a graph was first introduced by Hahn et al. in [188]. A vertex coloring of a graph G is called *injective* if any two vertices having a common neighbor get different colors. Let $\chi_i(G)$ be the minimum number of colors in injective colorings of G . The injective coloring is originated in complexity theory and is used in coding theory [188]. Note that an injective coloring is not necessarily a proper coloring and vice versa.

It is easy to see that if G is a triangle-free graph, then the injective coloring is just the $L(0, 1)$ -labeling of the graph G .

The *common neighbor graph* G_{cn} of a graph G is the graph defined by $V(G_{\text{cn}}) = V(G)$ and

$$E(G_{\text{cn}}) = \{uv \mid \text{there is a path of length 2 in } G \text{ joining } u \text{ and } v\}.$$

By the definition of $L(p, q)$ -labeling, the following relations among several parameters, that is, $\chi_i(G)$, $\chi(G^2)$, and $\chi(G_{\text{cn}})$, can be easily deduced:

- (i) $\chi_i(G) = \chi(G_{\text{cn}})$.
- (ii) If G is a triangle-free graph, then $\lambda_{0,1}(G) = \chi_i(G) - 1$.
- (iii) $\Delta \leq \chi_i(G) \leq \chi(G^2) \leq |G|$.

Let H denote the *prism*, which is a graph obtained by joining three chords u_1u_5, u_2u_4, u_3u_6 to a 6-cycle $u_1u_2 \dots u_6u_1$. It is easy to inspect that H^2 is K_6 , and H_{cn} is $K_6 - M$, where M is a perfect matching of H .

By (iii), $\chi_i(H) \geq \Delta = 3$. On the other hand, an injective 3-coloring c is easily constructed as follows: $c(u_1) = c(u_2) = 1$, $c(u_3) = c(u_6) = 2$, and $c(u_4) = c(u_5) = 3$. Thus, $\chi_i(H) = 3$. Moreover, it is easy to derive that $\chi_i(H) = \chi(H_{\text{cn}}) = 3 < 6 = \chi(H^2)$. This example shows that there exists a graph G such that $\chi(G^2) = 2\chi_i(G)$. In fact, it was proved in [227] that $\chi(G^2) \leq 2\chi_i(G)$ for any graph G .

The authors in [188] considered the injective chromatic number of some special graphs such as complete graphs, paths, cycles, and stars. Moreover, they gave the following results:

Proposition 4 *Let G be a graph with maximum degree Δ :*

1. *If G is d -regular and if $\chi_i(G) = d$, then d divides $|G|$.*
2. *If G is connected and not K_2 , then $\chi(G) \leq \chi_i(G)$.*
3. *If G has the diameter 2 and the independence number α , then $\chi_i(G) \geq \alpha$.*
4. $\chi_i(G) \leq \Delta(\Delta - 1) + 1$.

Theorem 200 *Let Q_n be the hypercube of dimension n :*

1. $\chi_i(Q_n) = n$ if and only if n is a power of 2.
2. $\chi_i(Q_n) \leq 2n - 2$.
3. $\chi_i(Q_{2n+1}) \leq 2\chi_i(Q_{n+1})$.
4. $\chi_i(Q_{2^m-j}) = 2^m$ for $0 \leq j \leq 3$.

The algorithmic aspect of the injective coloring for some special graphs was also considered in [188]:

Theorem 201 *It is NP-complete for a given split (and hence chordal) graph G and an integer k , to decide whether the injective chromatic number of G is at most k .*

Theorem 202 *It is NP-complete to decide, for a given split (and hence chordal) graph G , whether $\chi_i(G) = \chi(G^2) - 1$.*

Theorem 203 *The injective chromatic number of a power chordal graph can be computed in polynomial time.*

Bu et al. [84] studied the injective chromatic number of planar graphs having large maximum degree and girth.

Theorem 204 *Let G be a planar graph with maximum degree Δ and girth g :*

1. $\chi_i(G) = \Delta$ if either $g \geq 20$ and $\Delta \geq 3$ or $g \geq 7$ and $\Delta \geq 71$.
2. $\chi_i(G) \leq \Delta + 1$ if $g \geq 11$.
3. $\chi_i(G) \leq \Delta + 2$ if $g \geq 8$.

Lužar et al. [264] improved some results in [84] by showing the following:

Theorem 205 *Let G be a planar graph with maximum degree Δ and girth g :*

1. $\chi_i(G) \leq \Delta$ if $g \geq 19$ and $\Delta \geq 3$.
2. $\chi_i(G) \leq \Delta + 1$ if $g \geq 10$.
3. $\chi_i(G) \leq \Delta + 4$ if $g \geq 5$ and $\Delta \geq 3$.

Instead of studying planar graphs with high girth, Doyon et al. [133] considered the injective chromatic number of graphs with small maximum average degree. Their result is the following theorem:

Theorem 206 *Let G be a graph with maximum degree Δ :*

1. *If $\text{mad}(G) < \frac{14}{5}$, then $\chi_i(G) \leq \Delta + 3$.*
2. *If $\text{mad}(G) < 3$, then $\chi_i(G) \leq \Delta + 4$.*
3. *If $\text{mad}(G) < \frac{10}{3}$, then $\chi_i(G) \leq \Delta + 8$.*

Again, applying the relation $\text{mad}(G) < \frac{2g}{g-2}$ for a planar graph G with girth g , the following results hold: If G is a planar graph, then $\chi_i(G) \leq \Delta + 3$ if $g \geq 7$, $\chi_i(G) \leq \Delta + 4$ if $g \geq 6$, and $\chi_i(G) \leq \Delta + 8$ if $g \geq 5$.

In [120, 121], Cranston et al. improved the upper bounds given in [133, 264] in certain cases. More specifically, they studied sufficient conditions to imply $\chi_i(G) = \Delta$ and $\chi_i(G) \leq \Delta + 1$.

Theorem 207 ([121]) *Let G be a graph with maximum degree Δ :*

1. *If $\text{mad}(G) < \frac{14}{5}$ and $\Delta \geq 4$, then $\chi_i(G) \leq \Delta + 2$.*
2. *If $\text{mad}(G) < \frac{36}{13}$ and $\Delta = 3$, then $\chi_i(G) \leq 5$.*

Furthermore, an example to show that the maximum average degree in the second conclusion of **Theorem 207** is sharp was constructed in [121].

Chen et al. [104] obtained some upper bounds of the injective chromatic numbers for some graphs, that is, planar graphs, K_4 -minor-free graphs, etc. The following three theorems were established in [104].

Theorem 208 *Let G be a K_4 -minor-free graph with $\Delta \geq 1$. Then $\chi_i(G) \leq \lceil \frac{3}{2}\Delta \rceil$.*

Theorem 208 is best possible in the sense that there exist K_4 -minor-free graphs G such that $\chi_i(G) = \lceil \frac{3}{2}\Delta \rceil$, where Δ is even. Such an example was constructed in [104]. However, it is unknown if, for each given odd integer $k \geq 3$, there exists a K_4 -minor-free graph G with $\Delta = k$ and $\chi_i(G) = \frac{3k+1}{2}$.

Theorem 209 *If G is a planar graph with $\Delta \geq 3$, then $\chi_i(G) \leq \Delta^2 - \Delta$.*

It follows from **Theorem 209** that every planar graph G with $\Delta \leq 3$ has $\chi_i(G) \leq 6$. Some examples of planar subcubic graphs G with $\chi_i(G) = 5$ have been constructed by different authors (in private communications). **Figure 8** is such a planar cubic graph example, which has ten vertices.

Theorem 210 *Let G be a connected graph with $\Delta \geq 3$. Then $\chi_i(G) = \Delta^2 - \Delta + 1$ if and only if $G_{\text{cn}} = 2K_{\Delta(\Delta-1)+1}$.*

By **Theorem 210**, the following consequences hold automatically:

- (i) *If G is a graph with $\Delta \leq 4$, then $\chi_i(G) \leq 13$. Moreover, $\chi_i(G) = 13$ if and only if $G_{\text{cn}} = 2K_{13}$.*
- (ii) *If G is a graph with $\Delta \leq 3$, then $\chi_i(G) \leq 7$. Moreover, $\chi_i(G) = 7$ if and only if G is isomorphic to the Heawood graph.*

Fig. 8 A planar graph G with $\Delta = 3$ and $\chi_i(G) = 5$

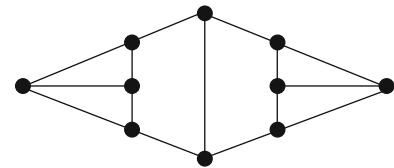
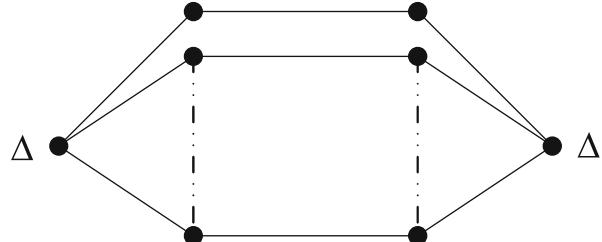


Fig. 9 Graph G' with $g(G') = 6$ and $\chi_i(G') = \Delta + 1$ for $\Delta \geq 2$



In the end of the paper [265], the authors put forward the challenging problem:

Conjecture 33 *Let G be a planar graph with maximum degree Δ . Then*

- (a) $\chi_i(G) \leq 5$, if $\Delta = 3$.
- (b) $\chi_i(G) \leq \Delta + 5$, if $4 \leq \Delta \leq 7$.
- (c) $\chi_i(G) \leq \lfloor \frac{3}{2}\Delta \rfloor + 1$, if $\Delta \geq 8$.

In 2011, Borodin and Ivanova [72] first investigated the list injective colorings of planar graphs. A graph G is said to be *injectively k -choosable* if any list L of size k allows an injective coloring φ such that $\varphi(v) \in L(v)$ whenever $v \in V(G)$. The *injective choice number* of G , denoted by $\chi_i^l(G)$, is the least integer k such that G is injectively k -choosable. Clearly, $\chi_i^l(G) \geq \chi_i(G) \geq \Delta$ for every graph G .

Theorem 211 ([72]) *If G is a planar graph with maximum degree Δ and girth g , then $\chi_i^l(G) = \chi_i(G) = \Delta$ in each of the following cases:*

1. $\Delta \geq 16$ and $g = 7$.
2. $\Delta \geq 10$ and $8 \leq g \leq 9$.
3. $\Delta \geq 6$ and $10 \leq g \leq 11$.
4. $\Delta = 5$ and $g \geq 12$.

Observe that a cycle C_{4n-1} has $\chi_i(C_{4n-1}) = \Delta(C_{4n-1}) + 1 = 3$, whereas the graph G' in Fig. 9 has girth $g = 6$ and $\chi_i(G) = \Delta + 1$ for arbitrarily large Δ . It means that the assumption on $g = 7$ of Theorem 211 (1) is tight.

Theorem 212 ([72]) *Every planar graph G with $\Delta \geq 24$ and $g \geq 6$ has $\chi_i^l(G) \leq \Delta + 1$.*

Furthermore, Borodin and Ivanova [72] posed the following problem.

Problem 11 *Find a precise upper bound for the injective choice number of planar graphs with given girth and maximum degree.*

5.5 $(d, 1)$ -Total Coloring

This section is devoted to the $(d, 1)$ -total coloring problem of graphs. Recall that a *total k -coloring* of a graph G is a coloring of $V(G) \cup E(G)$ using k colors such that no two adjacent or incident elements in $V(G) \cup E(G)$ receive the same color. The *total chromatic number* $\chi''(G)$ of G is the smallest integer k such that G has a total k -coloring. By the definition, $\chi''(G) \geq \Delta + 1$ for any graph G .

Behzad [46] and Vizing [340] posed independently the famous conjecture, named as the Total Coloring Conjecture:

Conjecture 34 (Total Coloring Conjecture) *For any simple graph G , $\chi''(G) \leq \Delta + 2$.*

Conjecture 34 remains still open. For the case $\Delta = 3$, it was verified by Rosenfeld [306] and independently by Vijayaditya [338]. Later, Kostochka [230–232] gradually settled the case $4 \leq \Delta \leq 5$. For planar graphs G , Conjecture 34 was confirmed by Borodin [56] for $\Delta \geq 9$, then by Yap [387] for $\Delta = 8$, and later by Sanders and Zhao for $\Delta = 7$ [311]. However, it remains open for planar graphs with $\Delta = 6$.

Recently, the total coloring of planar graphs with certain restrictions, that is, without some special short cycles or with given maximum degree, attracts much attention and has been deeply investigated in literatures [77, 204, 315, 316, 349, 374].

Whittlesey et al. [381] investigated the $L(2, 1)$ -labeling of incidence graphs. The *incidence graph*, denoted by G_I , of a graph G is the graph obtained from G by replacing each edge by a path of length 2. The $L(2, 1)$ -labeling of the incidence graph of G is equivalent to an assignment of integers to each element of $V(G) \cup E(G)$ such that adjacent vertices have different labels, adjacent edges have different labels, and incident vertex and edge have the difference of labels by at least 2.

Let d be a positive integer and G be a simple graph. A $(d, 1)$ -*total labeling* of G is an integer-valued function f defined on the set $V(G) \cup E(G)$ such that

$$|f(x) - f(y)| \geq \begin{cases} 1, & \text{if verticse } x \text{ and } y \text{ are adjacent in } G; \\ 1, & \text{if edges } x \text{ and } y \text{ are adjacent in } G; \\ d, & \text{if vertex } x \text{ and edge } y \text{ are incident.} \end{cases}$$

Notice that $|f(x) - f(y)|$ for adjacent elements x and y may be required to be greater than or equal to p , instead of 1, in the above, defining inequality for some given positive integer p to obtain a more general notion of (d, p) -total labeling. A $(d, 1)$ -*total labeling* taking values in the set $\{0, 1, \dots, k\}$ is called a $(d, 1)$ - k -*total labeling*. The *span* of a $(d, 1)$ -total labeling is the maximum difference between two labels. The minimum span, that is, the minimum k , among all $(d, 1)$ - k -*total labelings* of G , denoted by $\lambda_d^T(G)$, is called the $(d, 1)$ -*total number* of G .

It is easy to observe that a $(1, 1)$ -total labeling is a total coloring of G and that $\lambda_1^T(G) = \chi''(G) - 1$. Moreover, $\lambda_d^T(G) = \lambda_{d,1}(G_I)$, where G_I is the incidence graph of G .

The concept of $(d, 1)$ -total labeling was firstly introduced and investigated by Havet and Yu [184, 185]. They proposed the following conjecture:

Conjecture 35 (($d, 1$)-Total Labeling Conjecture) *For any graph G and any integer $d \geq 1$, $\lambda_d^T(G) \leq \min\{\Delta + 2d - 1, 2\Delta + d - 1\}$.*

Notice that when $d = 1$, Conjecture 35 is equivalent to Conjecture 34. By generalizing a result of [191], Havet and Yu in [185] showed that $\lambda_d^T(G) \leq 2\Delta - 2\log(\Delta + 2) + 2\log(16d - 8) + d - 1$ for any graph G . Moreover, Havet and Thomassé [183] showed that determining $(d, 1)$ -total number of graphs is very difficult, even for bipartite graphs. For instance, if $\Delta \geq 2d \geq 4$, or if $2d - 1 \geq \Delta \geq d + 2 \geq 5$, then the Δ -bipartite $(d, 1)$ -total labeling problem is NP-complete; if $d \geq 3$, then the $(d + 1)$ -bipartite $(d, 1)$ -total labeling problem is NP-complete.

5.5.1 General Bounds

Observing the label of a vertex with maximum degree and its incident edges, it is easy to see that $\lambda_d^T(G) \geq \Delta + d - 1$. This lower bound may be increased in some cases. In [185], the authors proved that if G is Δ -regular, or if $d \geq \Delta$, then $\lambda_d^T(G) \geq \Delta + d$. For upper bound, the following obvious fact was indicated in [185]:

Proposition 5 *Let G be a graph. Then*

1. $\lambda_d^T(G) \leq \chi(G) + \chi'(G) + d - 2$.
2. $\lambda_d^T(G) \leq 2\Delta + d - 1$.

Since $\chi'(G) = \Delta$ for a bipartite graph G (see [228]), it follows from Proposition 5(1) that every bipartite graph G satisfies $\Delta + d - 1 \leq \lambda_d^T(G) \leq \Delta + d$. In particular, if $d = 2$, then $\Delta + 1 \leq \lambda_d^T(G) \leq \Delta + 2$.

In [267], McDiarmid and Reed proved that if G is a graph with n vertices and k is an integer such that $k! > n$, then $\chi''(G) \leq \chi'(G) + k + 1$. A slight modification of the proof can be applied to the following extension [185]:

Theorem 213 *If G is a graph with n vertices and k is an integer with $\frac{k!}{(2p-1)^k} > n$, then $\lambda_d^T(G) \leq \chi'(G) + k + 3d - 3$. Hence, as $n \rightarrow \infty$, $\lambda_d^T(G) \leq \chi'(G) + O\left(\frac{\log n}{\log(\log n)}\right)$.*

It was proved in [191] that if Δ is large enough, then $\chi''(G) \leq \Delta + O(\log^{10} \Delta)$. This result was also extended to the $(d, 1)$ -total labeling situation [185]:

Theorem 214 *There exists a Δ_0 such that for each graph G with $\Delta \geq \Delta_0$, $\lambda_d^T(G) \leq \Delta + 2\log^{10} \Delta + 3d - 2$.*

Molloy and Reed [274] affirmed that there is a constant c so that the total chromatic number of a graph G is at most $\Delta + c$ as long as Δ is sufficiently large, where $c \leq 10^{26}$. It is possible that a similar proof leads to an analogous theorem for $(d, 1)$ -total labeling but with a larger constant.

The following bound, which slightly improves the known upper bound $2\Delta + d - 1$, also appeared in [185].

Theorem 215 *For any graph G and an integer $d \geq 1$, $\lambda_d^T(G) \leq 2\Delta - 2\log(\Delta + 2) + 2\log(16d - 8) + d - 1$.*

Employing a probabilistic discussion, Havet and Yu [185] showed the following result:

Theorem 216 *If G is a graph with n vertices and k is an integer with $\frac{k!}{(2d-1)^k} > n$, then $\lambda_d^T(G) \leq \chi' + k + 3d - 3$.*

It was shown in [185] that $\lambda_2^T(G) \leq 2\Delta$ for any graph G with $\Delta \geq 2$, and $\lambda_2^T(G) \leq 2\Delta - 1$ if $\Delta \geq 5$ is odd. This implies that $\lambda_2^T(G) \leq 6$ if $\Delta \leq 3$. Also it was conjectured [185] that $\lambda_2^T(G) \leq 5$ if G is a graph with $\Delta \leq 3$ and $G \neq K_4$. Moreover, it was affirmed that $\lambda_3^T(G) \leq 7$ if $\Delta \leq 3$.

Lih et al. [248] also discussed the general upper bounds of $(d, 1)$ -total number of graphs. They gave the following two results and one problem:

Theorem 217 *For any graph G with the list chromatic index k , $\lambda_d^T(G) \leq k + 4d - 3$.*

Theorem 218 *For any graph G , $\lambda_d^T(G) \leq \lfloor \frac{3}{2}\Delta \rfloor + 4d - 3$.*

Conjecture 36 *Let G be a graph with $\chi(G) > \max\{2, d\}$. Then $\lambda_d^T(G) \leq \chi(G) + \chi'(G) + d - 3$.*

5.5.2 Special Graphs

The $(d, 1)$ -total labeling for some kind of special graphs have been studied, for example, complete graphs [185], complete bipartite graphs [248], planar graphs [44], graphs with a given maximum average degree [280], trees for $d = 2$ [205, 352], and outerplanar graphs for $d = 2$ [110].

Complete graphs In their papers, Havet and Yu [185, 186] occupied nearly five pages to study the $(d, 1)$ -total labeling of the complete graphs K_n , according to the various values of n and d . In particular, they determined completely the exact value of $\lambda_2^T(K_n)$ for any $n \geq 1$. Here, only several main results are listed on this topic. For more details, readers may refer to [185].

Theorem 219 *Let $n, d \geq 1$ be integers:*

1. *If $d \geq n$, then $\lambda_d^T(K_n) = 2n + d - 3$.*
2. *If $d \leq n - 1$, then $\lambda_d^T(K_n) \leq n + 2d - 2$.*
3. *If $n \geq d$, then $\lambda_d^T(K_n) \geq n + 2d - 3$.*

4. If n is odd, then $\lambda_d^T(K_n) \leq n + 2d - 3$.
5. If n is even and $n > 6d^2 - 10d + 4$, then $\lambda_d^T(K_n) = n + 2d - 2$.
6. If $n \geq 4$ is even and $d \geq n - 3$, then $\lambda_d^T(K_n) \leq n + 2p - 3$.

Lih et al. [248] studied the $(d, 1)$ -total number of complete bipartite graphs $K_{m,n}$. In particular, they achieved the exact values of $\lambda_d^T(K_{m,n})$ for $d = 1, 2, 3$. Three main results are stated as follows:

Theorem 220 *If $m < n + d$, then $\lambda_d^T(K_{m,n}) = m + d$.*

Theorem 221 *Let $1 \leq n \leq m$. Then*

$$\lambda_2^T(K_{m,n}) = \begin{cases} m + 2, & \text{if } m \leq n + 1, \text{ or } m = n + 2 \text{ and } n \geq 3; \\ m + 1, & \text{otherwise.} \end{cases}$$

Theorem 222 *Let $1 \leq n \leq m$. Then*

$$\lambda_3^T(K_{m,n}) = \begin{cases} m + 3, & \text{if } m \leq n + 2, \text{ or } m = n + 3 \text{ and } n \geq 4, \\ & \quad \text{or } m = n + 4 \text{ and } n = 5, 9, 10, 13, 14, 15; \\ m + 2, & \text{otherwise.} \end{cases}$$

Planar graphs Suppose that G is a planar graph with maximum degree Δ . By the Four-Color Theorem, $\chi(G) \leq 4$. By the Vizing's Theorem [339], $\chi'(G) \leq \Delta + 1$. Thus, by Proposition 5(1), it is immediate to conclude that $\lambda_d^T(G) \leq \Delta + d + 3$. Further, if $\Delta \geq 7$, then $\chi'(G) = \Delta$ by [312] or [394], and therefore, $\lambda_d^T(G) \leq \Delta + d + 2$. If G is triangle-free and $\Delta \geq 5$, then $\chi(G) \leq 3$ [170] and $\chi'(G) = \Delta$ [246]; therefore, $\lambda_d^T(G) \leq \Delta + d + 1$.

In [44], Bazzaro et al. proved that $\lambda_d^T(G) \leq \Delta + 2d - 2$ for a planar graph G with high girth and large maximum degree Δ . Precisely, their main result is stated below:

Theorem 223 *Let G be a planar graph with maximum degree Δ and girth g . Then $\lambda_d^T(G) \leq \Delta + 2d - 2$ with $d \geq 2$ in each of the following cases:*

1. $\Delta \geq 2d + 1$ and $g \geq 11$.
2. $\Delta \geq 2d + 2$ and $g \geq 6$.
3. $\Delta \geq 2d + 3$ and $g \geq 5$.
4. $\Delta \geq 8d + 2$.

As pointed out above, the case $d = 1$ corresponds just to the total coloring of graphs. So, Theorem 223 extends actually the partial results obtained by Borodin et al. [77] regarding the total coloring of planar graphs with large girth. It should be emphasized that Bazzaro et al. [44] constructed a planar graph G which needs at least $\Delta + 2d - 1$ colors for a $(d, 1)$ -total labeling when $d = 2$. Furthermore, they posed the following risky conjecture:

Conjecture 37 *For any triangle-free planar graph G with $\Delta \geq 3$, $\lambda_d^T(G) \leq \Delta + d$.*

In [280], Montassier and Raspaud proved the following:

Theorem 224 *Let G be a graph with maximum degree Δ . Then $\lambda_d^T(G) \leq \Delta + 2d - 2$ with $d \geq 2$ in each of the following cases:*

1. $\Delta \geq 2d + 1$ and $\text{mad}(G) < \frac{5}{2}$.
2. $\Delta \geq 2d + 2$ and $\text{mad}(G) < 3$.
3. $\Delta \geq 2d + 3$ and $\text{mad}(G) < \frac{10}{3}$.

Moreover, Montassier and Raspaud [280] gave examples to show that **Theorem 224** is optimal for $d = 2$.

Let G be an outerplanar graph with $\Delta \geq 3$. It is known that $\chi(G) \leq 3$ and $\chi'(G) = \Delta$. Thus, it follows from [Proposition 5\(1\)](#) that $\lambda_d^T(G) \leq \Delta + d + 1$. In particular, when $d = 2$, then $\lambda_d^T(G) \leq \Delta + 3$. The upper bound $\Delta + 3$ is not the best possible. In 2007, Chen and Wang [110] obtained the following result:

Theorem 225 *Let G be an outerplanar graph:*

1. *If $\Delta \geq 5$, then $\lambda_2^T(G) \leq \Delta + 2$.*
2. *If G is 2-connected and $\Delta = 3$, then $\lambda_2^T(G) \leq 5$.*
3. *If $\Delta = 4$ and G does not contain intersecting triangles, then $\lambda_2^T(G) \leq 6$.*

Recently, Hasunumaa et al. [178] improved the results (2) and (3) of **Theorem 225**. That is, they showed that every outerplanar graph G with $3 \leq \Delta \leq 4$ also satisfies $\lambda_2^T(G) \leq \Delta + 2$.

Other graphs For a tree T , it is easy to prove that $\Delta + 1 \leq \lambda_2^T(T) \leq \Delta + 2$. In [352], Wang and Chen gave a complete characterization for a tree T with $\Delta = 3$ to have $\lambda_2^T(T) = 4$ or $\lambda_2^T(T) = 5$.

Suppose that T is a tree. A path $P_{k+2} = x_0x_1 \dots x_kx_{k+1}$ of length $k + 1$ of T is called a *k-chain* if $d(x_0) > 2$, $d(x_{k+1}) > 2$, and $d(x_i) = 2$ for all $i = 1, 2, \dots, k$. Note that a 0-chain is exactly an edge of T with two ends of degree more than 2. Let $V_i(T)$ denote the set of vertices of degree i in T .

Assume that $\Delta(T) = 3$. A subtree T^* of T is called *bad* if it satisfies the following conditions (1)–(4):

1. $V_3(T^*) \neq \emptyset$.
2. $V_1(T^*) \cup V_3(T^*) \subseteq V_3(T)$ and $V_2(T^*) \subseteq V_2(T)$.
3. Every vertex $x \in V_1(T^*)$ is connected to some vertex $y \in V_3(T^*)$ by a 1-chain.
4. Every two closest vertices $x, y \in V_3(T^*)$ are connected by a 2-chain or 1-chain.

A tree T is called *good* if it contains neither bad subtrees nor 3 vertices adjacent to two other 3 vertices.

Theorem 226 ([352]) *If T is a tree with $\Delta = 3$, then $\lambda_2^T(T) = 4$ if and only if T is good.*

Wang and Chen [352] also proposed the following interesting problem:

Problem 12 *Give a complete classification for trees T with $\Delta \geq 4$ according to their (2, 1)-total numbers.*

It seems hard to settle the above problem. Huang et al. [205] gave a partial solution.

For a tree T , let $D_\Delta(T)$ denote the set of integers k for which there exist two distinct vertices of maximum degree of distance at k in T . Based on this definition, the authors in [205] established the following sufficient condition:

Theorem 227 *Let T be a tree with $\Delta \geq 4$. If $1 \notin D_\Delta(T)$ or $2 \notin D_\Delta(T)$, then $\lambda_2^T(T) = \Delta + 1$.*

This result is best possible in the sense that for any fixed integer $k \geq 3$, there exist infinitely many trees T with $\Delta \geq 4$ and $k \notin D_\Delta(T)$ such that $\lambda_2^T(T) = \Delta + 2$.

Chen and Wang [109] determined completely the $(2, 1)$ -total number of the Cartesian product of cycles or paths.

Theorem 228 *If $m, n \geq 3$, then $\lambda_2^T(C_m \times C_n) = 6$.*

Theorem 229 *Let $m \geq n \geq 2$. Then*

$$\lambda_2^T(P_m \times P_n) = \begin{cases} 4, & \text{if } m = 2 \text{ and } n = 2, 3; \\ 5, & \text{if } m = 2 \text{ and } n \geq 4, \text{ or } m = 3 \text{ and } 3 \leq n \leq 6, \text{ or } m = n = 4; \\ 6, & \text{otherwise.} \end{cases}$$

Recall that the join $G \vee H$ of two vertex-disjoint graphs G and H is the graph obtained by joining each vertex of G to each vertex of H .

Wang et al. [356] provided a complete characterization for the $(2, 1)$ -total number of the join of cycles or paths.

Theorem 230 *Let $n \geq m \geq 3$. Then*

$$\lambda_2^T(C_m \vee C_n) = \begin{cases} n + 3, & \text{if either } n \geq m + 2 \text{ and } m \text{ is even,} \\ & \quad \text{or } n = m + 1 \text{ and } m \equiv 2, 4 \pmod{12}; \\ n + 4, & \text{otherwise.} \end{cases}$$

Theorem 231 *Let $n \geq m \geq 1$. Then*

$$\lambda_2^T(P_m \vee P_n) = \begin{cases} n + 1, & \text{if } m = 1 \text{ and } n \geq 4; \\ n + 2, & \text{if } m = 1 \text{ and } 1 \leq n \leq 3, \text{ or } m = 2 \text{ and } n \geq 4; \\ n + 3, & \text{if } m = 2 \text{ and } n = 3, \text{ or } m \geq 3 \text{ and } n \geq m + 1; \\ n + 4, & \text{if } m = n \geq 2. \end{cases}$$

6 Conclusion

Graph coloring is an important branch in graph theory. From the viewpoint of historical development, graph theory came first from the graph coloring problems, for example, the Four-Coloring Conjecture. Since the famous Four-Coloring Conjecture

was proposed, many variants, generalizations, techniques, and applications on graph coloring problem have emerged. There have existed a large amount of literature regarding this area.

In this chapter, some subjects of graph coloring problems are selected to investigate, including classical chromatic number, choice number, acyclic chromatic number, acyclic chromatic index, vertex-distinguishing edge-weighting (especially, adjacent vertex-distinguishing edge coloring and total coloring), and $L(p, q)$ -labeling problems. So far there have been many well-known conjectures and open problems on these selected topics, such as Hajós Conjecture, Hadwiger Conjecture, Steinberg's Conjecture, Acyclic Edge-Coloring Conjecture, Griggs and Yeh's Conjecture, Wegner's Conjecture, and Havet and Yu's Conjecture. Some main results and progresses on each of these problems are surveyed, without proofs provided. Of course, methods and tools to solve the related problems are rarely mentioned, because of the limited pages. For the interested reader, they can refer to a mass of literatures listed below.

Acknowledgements The authors deeply thank their postgraduate students M. Chen, D. Huang, Q. Shu, S. Zhang, W. Gao, and Y. Wang for their careful search for a number of literatures and patience inspection for this manuscript.

Cross-References

- ▶ [Equitable Coloring of Graphs](#)
- ▶ [Graph Searching and Related Problems](#)
- ▶ [Graph Theoretic Clique Relaxations and Applications](#)
- ▶ [Max-Coloring](#)
- ▶ [Maximum Flow Problems and an NP-Complete Variant on Edge-Labeled Graphs](#)
- ▶ [Online and Semi-online Scheduling](#)
- ▶ [Optimal Partitions](#)
- ▶ [Partition in High Dimensional Spaces](#)

Recommended Reading

1. H.L. Abbott, B. Zhou, On small faces in 4-critical graphs. *Ars Combin.* **32**, 203–207 (1991)
2. L. Addario-Berry, R.E.L. Aldred, K. Dalal, B.A. Reed, Vertex colouring edge partitions. *J. Combin. Theory Ser. B* **94**, 237–244 (2005)
3. L. Addario-Berry, K. Dalal, C. McDiarmid, B.A. Reed, A. Thomason, Vertex-colouring edge-weightings. *Combinatorica* **27**, 1–12 (2007)
4. L. Addario-Berry, K. Dalal, B.A. Reed, Degree constrained subgraphs. *Discrete Appl. Math.* **156**, 1168–1174 (2008)
5. L. Addario-Berry, L. Esperet, R.J. Kang, C.J.H. McDiarmid, Acyclic improper colourings of graphs with bounded maximum degree. *Discrete Math.* **310**, 223–229 (2010)
6. G. Agnarsson, M.M. Halldórsson, Coloring powers of planar graphs. *SIAM J. Discrete Math.* **16**, 651–662 (2003)

7. G. Agnarsson, M.M. Halldórsson, Vertex coloring the square of outerplanar graphs of lower degree. *Discuss. Math. Graph Theory* **30**, 619–636 (2010)
8. M. Aigner, E. Triesch, Irregular assignments of trees and forests. *SIAM J. Discrete Math.* **3**, 439–449 (1990)
9. M. Aigner, E. Triesch, Z. Tuza, Irregular assignments and vertex distinguishing edge-colorings of graphs. *Ann. Discrete Math.* **52**, 1–9 (1992)
10. S. Akbari, H. Bidkhori, N. Nosrati, r -Strong edge colorings of graphs. *Discrete Math.* **306**, 3005–3010 (2006)
11. V.A. Aksionov, L.S. Mel'nikov, Some counterexamples associated with the Three Color Problem. *J. Combin. Theory Ser. B* **28**, 1–9 (1980)
12. M.O. Albertson, D.M. Berman, Every planar graph has an acyclic 7-coloring. *Israel J. Math.* **28**, 169–174 (1977)
13. M.O. Albertson, G.G. Chappell, H.A. Kierstead, A. Kündgen, R. Ramamurthy, Coloring with no 2-colored P_4 's. *Electron. J. Combin.* **11**(1), #R26 (2004)
14. M.O. Albertson, J.P. Hutchinson, The three excluded cases of Dirac's map-color theorem. *Ann. N Y Acad. Sci.* **319**, 7–17 (1979)
15. M.O. Albertson, J.P. Hutchinson, Hadwiger's conjecture for graphs on the Klein bottle. *Discrete Math.* **29**, 1–11 (1980)
16. M.O. Albertson, W. Stromquist, Locally planar toroidal graphs are 5-colorable. *Proc. Am. Math. Soc.* **84**, 449–456 (1982)
17. N. Alon, C. McDiarmid, B. Reed, Acyclic colourings of graphs. *Random Struct. Algorithms* **2**, 277–288 (1990)
18. N. Alon, B. Mohar, D.P. Sanders, On acyclic colorings of graphs on surfaces. *Israel J. Math.* **94**, 273–283 (1996)
19. N. Alon, B. Sudakov, A. Zaks, Acyclic edge colorings of graphs. *J. Graph Theory* **37**, 157–167 (2001)
20. N. Alon, M. Tarsi, Colorings and orientations of graphs. *Combinatorica* **12**, 125–134 (1992)
21. N. Alon, A. Zaks, Algorithmic aspects of acyclic edge coloring. *Algorithmica* **32**, 611–614 (2002)
22. A. Altshuler, Construction and enumeration of regular maps on the torus. *Discrete Math.* **4**, 201–217 (1973)
23. D. Amar, Irregularity strength of regular graphs of large degree. *Discrete Math.* **114**, 9–17 (1993)
24. D. Amar, O. Togni, Irregularity strength of trees. *Discrete Math.* **190**, 15–38 (1998)
25. P. Angelini, F. Frati, Acyclically 3-colorable planar graphs. *J. Comb. Optim.* (2011). doi:10.1007/s10878-011-9385-3
26. M. Anholcer, M. Kalkowski, J. Przybyło, A new upper bound for the total vertex irregularity strength of graphs. *Discrete Math.* **309**, 6316–6317 (2009)
27. K. Appel, W. Haken, The existence of unavoidable sets of geographically good configurations. *Illinois J. Math.* **20**, 218–297 (1976)
28. P.N. Balister, Vertex-distinguishing edge colorings of random graphs. *Random Struct. Algorithms* **20**, 89–97 (2001)
29. P.N. Balister, B. Bollobás, R.H. Schelp, Vertex distinguishing colorings of graphs with $\Delta(G) = 2$. *Discrete Math.* **252**, 17–29 (2002)
30. P.N. Balister, E. Győri, J. Lehel, R.H. Schelp, Adjacent vertex distinguishing edge-colorings. *SIAM J. Discrete Math.* **21**, 237–250 (2007)
31. P.N. Balister, A. Kostochka, H. Li, R.H. Schelp, Balanced edge colorings. *J. Combin. Theory Ser. B* **90**, 3–20 (2004)
32. P.N. Balister, O.M. Riordan, R.H. Schelp, Vertex-distinguishing edge colorings of graphs. *J. Graph Theory* **42**, 95–109 (2003)
33. J.-L. Baril, H. Kheddouci, O. Togni, The irregularity strength of circulant graphs. *Discrete Math.* **304**, 1–10 (2005)
34. T. Bartnicki, J. Grytczuk, S. Niwczyk, Weight choosability of graphs. *J. Graph Theory* **60**, 242–256 (2009)

35. M. Basavaraju, L.S. Chandran, Acyclic edge coloring of subcubic graphs. *Discrete Math.* **308**, 6650–6653 (2008)
36. M. Basavaraju, L.S. Chandran, Acyclic edge coloring of graphs with maximum degree 4. *J. Graph Theory* **61**, 192–209 (2009)
37. M. Basavaraju, L.S. Chandran, A note on acyclic edge coloring of complete bipartite graphs. *Discrete Math.* **309**, 4646–4648 (2009)
38. M. Basavaraju, L.S. Chandran, Acyclic edge coloring of 2-degenerate graphs. *J. Graph Theory* **69**, 1–27 (2012)
39. M. Basavaraju, L.S. Chandran, N. Cohen, F. Havet, T. Müller, Acyclic edge-coloring of planar graphs. *SIAM J. Discrete Math.* **25**, 463–478 (2011)
40. M. Basavaraju, L.S. Chandran, Acyclic edge coloring of triangle free planar graphs. [arXiv.org/abs/1007.2282v1](https://arxiv.org/abs/1007.2282v1).
41. M. Basavaraju, L.S. Chandran, M. Kummini, d -Regular graphs of acyclic chromatic index at least $d + 2$. *J. Graph Theory* **63**, 226–230 (2010)
42. C. Bazgan, A. Harkat-Benhamdine, H. Li, M. Woźniak, On the vertex-distinguishing proper edge-colorings of graphs. *J. Combin. Theory Ser. B* **75**, 288–301 (1999)
43. C. Bazgan, A. Harkat-Benhamdine, H. Li, M. Woźniak, A note on the vertex-distinguishing proper colorings of graphs with large minimum degree. *Discrete Math.* **236**, 37–42 (2001)
44. F. Bazzaro, M. Montassier, A. Raspaud, $(d, 1)$ -Total labelling of planar graphs with large girth and high maximum degree. *Discrete Math.* **307**, 2141–2151 (2007)
45. J. Beck, An algorithmic approach to the Lovász local lemma. *Random Struct. Algorithms* **2**, 343–365 (1991)
46. M. Behzad, Graphs and their chromatic number. Doctoral Thesis, Michigan State University, 1965
47. P. Bella, D. Král, B. Mohar, K. Quitterová, Labelling planar graphs with a condition at distance two. *Eur. J. Combin.* **28**, 2201–2239 (2007)
48. C. Berge, Perfect graphs, in *Six Papers on Graph Theory* (Indian Statistical Institute, Calcutta, 1963), pp. 1–21
49. X. Bian, L. Miao, H. Shang, C. Duan, G. Ma, Adjacent vertex-distinguishing edge partition of graphs. *J. East China Norm. Univ. Nat. Sci. Ed.* 2009(4), 16–20 (2009)
50. H.L. Bodlaender, T. Kloks, R.B. Tan, J.V. Leeuwen, λ -Coloring of graphs. *Lect. Notes Comput. Sci.* **1770**, 395–406 (2000)
51. H.L. Bodlaender, T. Kloks, R.B. Tan, J.V. Leeuwen, Approximations for λ -colorings of graphs. *Comput. J.* **47**, 193–204 (2004)
52. T. Bohman, D. Kravitz, On the irregularity strength of trees. *J. Graph Theory* **45**, 241–254 (2004)
53. T. Böhme, B. Mohar, M. Stiebitz, Dirac's map-color theorem for choosability. *J. Graph Theory* **32**, 327–339 (1999)
54. P. Boiron, E. Sopena, L. Vignal, Acyclic improper colorings of graphs. *J. Graph Theory* **32**, 97–107 (1999)
55. O.V. Borodin, On acyclic coloring of planar graphs. *Discrete Math.* **25**, 211–236 (1979)
56. O.V. Borodin, On the total coloring of planar graphs. *J. Reine Angew. Math.* **394**, 180–185 (1989)
57. O.V. Borodin, Structural properties of plane graphs without adjacent triangles and an application to 3-colorings. *J. Graph Theory* **21**, 183–186 (1996)
58. O.V. Borodin, Acyclic 3-choosability of planar graphs with no cycles with length from 4 to 12. *Diskretn. Anal. Issled. Oper.* **116**(5), 26–33 (2009) (in Russian)
59. O.V. Borodin, Acyclic 4-colorability of planar graphs without 4- and 5-cycles. *Diskretn. Anal. Issled. Oper.* **17**(2), 20–38 (2010) (in Russian)
60. O.V. Borodin, H.J. Broersma, A. Glebov, J.V.D. Heuvel, Stars and bunches in planar graphs. Part II: General planar graphs and colourings. CDAM researches report 2002-05, 2002
61. O.V. Borodin, M. Chen, A.O. Ivanova, A. Raspaud, Acyclic 3-choosability of sparse graphs with girth at least 7. *Discrete Math.* **310**, 2426–2434 (2010)

62. O.V. Borodin, D.G. Fon-Der Flaass, A.V. Kostochka, A. Raspaud, E. Sopena, Acyclic list 7-coloring of planar graphs. *J. Graph Theory* **40**, 83–90 (2002)
63. O.V. Borodin, A.N. Glebov, A.O. Ivanova, T.K. Neustroeva, V.A. Tashkinov, Sufficient conditions for the 2-distance $(\Delta+1)$ -colorability of plane graphs. *Sib. Elektron. Mat. Izv.* **1**, 129–141 (2004) (in Russian)
64. O.V. Borodin, A.N. Glebov, M. Montassier, A. Raspaud, Planar graphs without 5- and 7-cycles and without adjacent triangles are 3-colorable. *J. Combin. Theory Ser. B* **99**, 668–673 (2009)
65. O.V. Borodin, A.N. Glebov, A. Raspaud, M.R. Salavatipour, Planar graphs without cycles of length from 4 to 7 are 3-colorable. *J. Combin. Theory Ser. B* **93**, 303–311 (2005)
66. O.V. Borodin, A.O. Ivanova, Planar graphs without triangular 4-cycles are 4-choosable. *Sib. Elektron. Mat. Izv.* **5**, 75–79 (2008)
67. O.V. Borodin, A.O. Ivanova, Acyclic 5-choosability of planar graphs without adjacent short cycles. *J. Graph Theory* **68**, 169–176 (2011)
68. O. V. Borodin, A. O. Ivanova, Acyclic 5-choosability of planar graphs without 4-cycles. *Siberian Math. J.* **52**, 411–425 (2011)
69. O.V. Borodin, A.O. Ivanova, 2-Distance $(\Delta + 2)$ -coloring of planar graphs with girth six and $\Delta \geq 18$. *Discrete Math.* **309**, 6496–6502 (2009)
70. O.V. Borodin, A.O. Ivanova, List 2-distance $(\Delta + 2)$ -coloring of planar graphs with girth six. *Eur. J. Combin.* **30**, 1257–1262 (2009)
71. O.V. Borodin, A.O. Ivanova, Acyclic 4-choosability of planar graphs with no 4- and 5-cycles. *J. Graph Theory* **72**, 374–397 (2013)
72. O.V. Borodin, A.O. Ivanova, List injective colorings of planar graphs. *Discrete Math.* **311**, 154–165 (2011)
73. O.V. Borodin, A.O. Ivanova, T.K. Neustroeva, List 2-distance $(\Delta + 1)$ -coloring of planar graphs with given girth. *J. Appl. Ind. Math.* **2**, 317–328 (2008)
74. O.V. Borodin, A.O. Ivanova, A. Raspaud, Acyclic 4-choosability of planar graphs with neither 4-cycles nor triangular 6-cycles. *Discrete Math.* **310**, 2946–2950 (2010)
75. O.V. Borodin, A.V. Kostochka, On an upper bound of the graph's chromatic number depending on the graph's degree and density. *J. Combin. Theory Ser. B* **23**, 247–250 (1977)
76. O.V. Borodin, A.V. Kostochka, A. Raspaud, E. Sopena, Acyclic colouring of 1-planar graphs. *Discrete Appl. Math.* **114**, 29–41 (2001)
77. O.V. Borodin, A.V. Kostochka, D.R. Woodall, Total colorings of planar graphs with large maximum degree. *J. Graph Theory* **26**, 53–59 (1997)
78. O.V. Borodin, A.V. Kostochka, D.R. Woodall, Acyclic colourings of planar graphs with large girth. *J. Lond. Math. Soc.* **60**(2), 344–352 (1999)
79. O.V. Borodin, M. Montassier, A. Raspaud, Planar graphs without adjacent cycles of length at most seven are 3-colorable. *Discrete Math.* **310**, 167–173 (2010)
80. M. Borowiecki, A. Fiedorowics, Acyclic edge colouring of planar graphs without short cycles. *Discrete Math.* **310**, 1445–1455 (2010)
81. H. Broersma, F.V. Fomin, P.A. Golovach, G.J. Woeginger, Backbone coloring for networks, in *Proceeding of WG 2003*. Lecture Notes in Computer Science, vol. 2880, (Springer, Berlin, 2003), pp. 131–142
82. H. Broersma, F.V. Fomin, P.A. Golovach, G.J. Woeginger, Backbone coloring for graphs: tree and path backbones. *J. Graph Theory* **55**, 137–152 (2007)
83. H.J. Broersma, J. Fujisawa, L. Marchal, A.N.M. Salman, K. Yoshimoto, λ -Backbone colorings along pairwise disjoint stars and matchings. *Discrete Math.* **309**, 5596–5609 (2009)
84. Y. Bu, D. Chen, A. Raspaud, W. Wang, Injective coloring of planar graphs. *Discrete Appl. Math.* **157**, 663–672 (2009)
85. Y. Bu, N.W. Cranston, M. Montassier, A. Raspaud, W. Wang, Star-coloring of sparse graphs. *J. Graph Theory* **62**, 201–219 (2009)
86. Y. Bu, K.-W. Lih, W. Wang, Adjacent vertex distinguishing edge-colorings of planar graphs with girth at least six. *Discuss. Math. Graph Theory* **31**, 429–439 (2011)

87. Y. Bu, S. Zhang, Backbone coloring for C_4 -free planar graphs. *Sci. Sin. Math.* **41**, 197–206 (2011) (in Chinese)
88. A.C. Burris, Vertex-distinguishing edge-colorings. Ph.D. Dissertation, Memphis State University, 1993
89. A.C. Burris, R.H. Schelp, Vertex-distinguishing proper edge-coloring. *J. Graph Theory* **26**, 73–82 (1997)
90. M.I. Burštejn, Every 4-valent graph has an acyclic 5 coloring. *Soobshch. Akad. Nauk Gruzinskogo SSR* **93**, 21–24 (1979) (in Russian)
91. L. Cai, W. Wang, X. Zhu, Choosability of toroidal graphs without short cycles. *J. Graph Theory* **65**, 1–15 (2010)
92. T. Calamoneri, The $L(h, k)$ -labelling problem: a survey and annotated bibliography. *Comput. J.* **49**, 585–608 (2006)
93. T. Calamoneri, R. Petreschi, $L(h, 1)$ -labelling subclasses of planar graphs. *J. Parallel Distrib. Comput. Math.* **220**, 414–426 (2003)
94. Y. Card, Y. Roditty, Acyclic edge-colorings of sparse graphs. *Appl. Math. Lett.* **7**, 63–67 (1994)
95. P.A. Catlin, Hajós' graph-coloring conjecture: variations and counterexamples. *J. Combin. Theory Ser. B* **26**, 268–274 (1979)
96. J. Černý, M. Horňák, R. Soták, Observability of a graph. *Math. Slovaca* **46**, 21–31 (1996)
97. G.J. Chang, W.-T. Ke, D. Kuo, D.D.-F. Liu, R.K. Yeh, On $L(d, 1)$ -labellings of graphs. *Discrete Math.* **220**, 57–66 (2000)
98. G.J. Chang, D. Kuo, The $L(2, 1)$ -labelling problem on graphs. *SIAM J. Discrete Math.* **9**, 309–316 (1996)
99. G.J. Chang, C. Lu, J. Wu, Q. Yu, Vertex-coloring edge-weightings of graphs. *Taiwan. J. Math.* **15**, 1807–1813 (2011)
100. G. Chartrand, M.S. Jacobson, J. Lehel, O.R. Oellermann, S. Ruiz, F. Saba, Irregular networks. *Congr. Numer.* **64**, 197–210 (1988)
101. G. Chartrand, H.V. Kronk, The point-arboricity of planar graphs. *J. Lond. Math. Soc.* **44**, 612–616 (1969)
102. X. Chen, On the adjacent vertex distinguishing total coloring numbers of graphs with $\Delta = 3$. *Discrete Math.* **308**, 4003–4007 (2008)
103. M. Chen, X. Guo, Adjacent vertex-distinguishing edge and total chromatic numbers of hypercubes. *Inform. Process. Lett.* **109**, 599–602 (2009)
104. M. Chen, G. Hahn, A. Raspaud, W. Wang, Some results on the injective chromatic number of graphs. *J. Comb. Optim.* (2011). doi:10.1007/s10878-011-9386-2
105. M. Chen, H. Lu, Y. Wang, A note on 3-choosability of planar graphs. *Inform. Process. Lett.* **105**, 206–211 (2008)
106. M. Chen, A. Raspaud, A sufficient condition for planar graphs to be acyclically 5-choosable. *J. Graph Theory* **70**, 135–151 (2012)
107. M. Chen, A. Raspaud, N. Roussel, X. Zhu, Acyclic 4-choosability of planar graphs. *Discrete Math.* **311**, 92–101 (2011)
108. M. Chen, A. Raspaud, W. Wang, Three-coloring planar graphs without short cycles. *Inform. Process. Lett.* **101**, 134–138 (2007)
109. D. Chen, W. Wang, The $(2, 1)$ -total labelling of the product of two kinds of graphs. *J. Zhejiang Norm. Univ. Nat. Sci.* **29**, 26–31 (2006)
110. D. Chen, W. Wang, $(2, 1)$ -Total labelling of outerplanar graphs. *Discrete Appl. Math.* **155**, 2585–2593 (2007)
111. M. Chen, A. Raspaud, Planar graphs without 4- and 5-cycles are acyclically 4-choosable, *Discrete Appl. Math.* in press, (2013)
112. M. Chen, A. Raspaud, W. Wang, 6-Star-coloring of subcubic graphs. *J. Graph Theory* **72**, 128–145 (2013)
113. M. Chen, W. Wang, Acyclic 5-choosability of planar graphs without 4-cycles. *Discrete Math.* **308**, 6216–6225 (2008)

114. M. Chudnovsky, N. Robertson, P. Seymour, R. Thomas, The strong perfect graph theorem. *Ann. Math.* **164**, 51–229 (2006)
115. N. Cohen, F. Havet, T. Müller, Acyclic edge-colouring of planar graphs. Extended abstract. *Elettron. Note Discrete Math.* **34**, 417–421 (2009)
116. T. Coker, K. Johannson, The adjacent vertex distinguishing total chromatic number. arXiv:1009.1785v2
117. T.F. Coleman, J. Cai, The cyclic coloring problem and estimation of sparse Hessian matrices. *SIAM J. Algebraic Discrete Method* **7**, 221–235 (1986)
118. K.L. Collins, J.P. Hutchinson, Four-coloring six-regular graphs on the torus. in: P. Hansen, O. Marcotte (Eds.), *Graph Colouring and Applications, CRM Proceedings and Lecture Notes*, **23**, 21–34 (1999)
119. D.W. Cranston, S.-J. Kim, List-coloring the square of a subcubic graph. *J. Graph Theory* **57**, 65–78 (2008)
120. D.W. Cranston, S.-J. Kim, G. Yu, Injective colorings of sparse graphs. *Discrete Math.* **310**, 2965–2973 (2010)
121. D.W. Cranston, S.-J. Kim, G. Yu, Injective colorings of graphs with low average degree. *Algorithmica* **60**, 553–568 (2011)
122. B. Cuckler, F. Lazebnik, Irregularity strength of dense graphs. *J. Graph Theory* **58**, 299–313 (2008)
123. Y. Dai, Y. Bu, An upper bound on the adjacent vertex-distinguishing chromatic number of graph. *Math. Econ.* **26**, 107–110 (2009)
124. E. Dedò, D. Torri, N. Zagaglia Salvi, The observability of the Fibonacci and the Lucas cubes. *Discrete Math.* **255**, 55–63 (2002)
125. B. Descartes, A three colour problem. *Eureka* **21**, 459–470 (1947)
126. R. Diestel, *Graph Theory*, 3rd ed. (Springer, Berlin, 2005)
127. J.H. Dinitz, D.K. Garnick, A. Gyárfás, On the irregularity strength of the $m \times n$ grid. *J. Graph Theory* **16**, 355–374 (1992)
128. G.A. Dirac, A property of 4-chromatic graphs and some remarks on critical graphs. *J. Lond. Math. Soc.* **27**, 85–92 (1952)
129. G.A. Dirac, Map colour theorems related to the Heawood colour formula. *J. Lond. Math. Soc.* **31**, 460–471 (1956)
130. G.A. Dirac, S. Schuster, A theorem of Kuratowski. *Indag. Math.* **16**, 343–348 (1954)
131. W. Dong, B. Xu, Some results on acyclic edge coloring of plane graphs. *Inform. Process. Lett.* **110**, 887–892 (2010)
132. W. Dong, B. Xu, X. Zhang, Improved bounds on linear coloring of plane graphs. *Sci. China Ser. A Math.* **53**, 1895–1902 (2010)
133. A. Doyon, G. Hahn, A. Raspaud, Some bounds on the injective chromatic number of graphs. *Discrete Math.* **310**, 585–590 (2010)
134. Z. Dvořák, D. Král, P. Nejedlý, R. Škrekovski, Coloring squares of planar graphs with girth six. *Eur. J. Combin.* **29**, 838–849 (2008)
135. Z. Dvořák, B. Lidický, R. Škrekovski, 3-Choosability of triangle-free planar graphs with constraints on 4-cycles. *SIAM J. Discrete Math.* **24**, 934–945 (2010)
136. Z. Dvořák, B. Lidický, R. Škrekovski, Planar graphs without 3-, 7- and 8-cycles are 3-choosable. *Discrete Math.* **309**, 5899–5904 (2009)
137. N. Eaton, T. Hull, Defective list colorings of planar graphs. *Bull. Inst. Combin. Appl.* **25**, 79–87 (1999)
138. J.R. Edmonds, Minimum partition of a matroid into independent subsets. *J. Res. Nat. Bur. Stand.* **69B**, 65–72 (1965)
139. K. Edwards, M. Horňák, M. Woźniak, On the neighbour distinguishing index of a graph. *Graphs Combin.* **22**, 341–350 (2006)
140. P. Erdős, A.L. Rubin, H. Taylor, Choosability in graphs. *Congr. Numer.* **26**, 125–157 (1979)
141. L. Esperet, M. Montassier, A. Raspaud, Linear choosability of graphs. *Discrete Math.* **306**, 3938–3950 (2008)

142. L. Esperet, A. Pinlou, Acyclic improper choosability of outerplanar graphs. Research report RR-1405-06, LaBRI, Université Bordeaux 1, 2006
143. L. Esperet, A. Pinlou, Acyclic improper choosability of graphs. *Electron. Note Discrete Math.* **28**, 251–258 (2007)
144. B. Farzad, Planar graphs without 7-cycles are 4-choosable. *SIAM J. Discrete Math.* **23**, 1179–1199 (2009)
145. R.J. Faudree, J. Lehel, Bound on the irregularity strength of regular graphs. *Colloq. Math. Soc. János Bolyai* (North-Holland, Amsterdam) 52, 247–256 (1988)
146. M. Ferrara, R. Gould, M. Karoński, F. Pfender, An interactive approach to graph irregularity strength. *Discrete Appl. Math.* **158**, 1189–1194 (2010)
147. G. Fertin, E. Godard, A. Raspaud, Acyclic and k -distance coloring of the grid. *Inform. Process. Lett.* **87**, 51–58 (2003)
148. G. Fertin, A. Raspaud, Acyclic colorings of graphs of maximum degree $\Delta(G)$, in *European Conference on Combinatorics, Graph Theory and Applications*, June 2005, pp. 389–396
149. G. Fertin, A. Raspaud, B. Reed, On star-coloring of graphs. *Lect. Notes Comput. Sci.* **2204**, 140–153 (2001)
150. J. Fiala, J. Kratochvíl, On the computational complexity of the $L(2, 1)$ -labelling problem for regular graphs. *Lect. Notes Comput. Sci.* **3701**, 228–236 (2005)
151. J. Fiamčík, The acyclic chromatic class of a graph. *Math. Slovaca* **28**, 139–145 (1978)
152. J. Fiamčík, Acyclic chromatic index of a graph with maximum valency three. *Arch. Math. (Brno)* **16**, 81–87 (1980) (in Russian)
153. A. Fiedorowicz, Acyclic edge colourings of graphs with the number of edges linearly bounded by the number of vertices. *Inform. Process. Lett.* **111**, 287–290 (2011)
154. A. Fiedorowicz, M. Haluszczak, N. Narayanan, About acyclic edge colourings of planar graphs. *Inform. Process. Lett.* **108**, 412–417 (2008)
155. G. Fijavž, M. Juvan, B. Mohar, R. Škrekovski, Planar graphs without cycles of specific lengths. *Eur. J. Combin.* **23**, 377–388 (2002)
156. P. Franklin, A six-color problem. *J. Math. Phys.* **13**, 363–369 (1934)
157. A. Frieze, R.J. Gould, M. Karoński, F. Pfender, On graph irregularity strength. *J. Graph Theory* **41**, 120–137 (2002)
158. Y. Gao, D. Yu, Acyclic edge coloring of planar graphs without cycles of specific lengths. *J. Appl. Math. Comput.* **37**, 533–540 (2011)
159. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979)
160. M.R. Garey, D.S. Johnson, L. Stockmeyer, Some simplified NP-complete graph problems. *Theoret. Comput. Sci.* **1**, 237–267 (1976)
161. A.H. Gebremedhin, F. Manne, A. Pothen, What color is your Jacobian? Graph coloring for computing derivatives. *SIAM Rev.* **47**, 629–705 (2005)
162. A.H. Gebremedhin, A. Pothen, A. Tarafdar, A. Walther, Efficient computation of sparse Hessians using coloring and automatic differentiation. *INFORMS J. Comput.* **21**, 209–223 (2009)
163. S. Gerke, C. Greenhill, N. Wormald, The generalized acyclic edge chromatic number of random regular graphs. *J. Graph Theory* **53**, 101–125 (2006)
164. S. Gerke, M. Raemy, Generalised acyclic edge colourings of graphs with large girth. *Discrete Math.* **307**, 1668–1671 (2007)
165. M. Ghandehari, H. Hatami, Two upper bounds for the strong edge chromatic number. Preprint. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.8996>
166. D. Gonçalves, On the $L(p, 1)$ -labelling of graphs. *Discrete Math.* **308**, 1405–1414 (2008)
167. C. Greenhill, O. Pokrovskiy, Bounds on generalized acyclic chromatic numbers of bounded degree graphs. *Graphs Combin.* **21**, 407–419 (2005)
168. C. Greenhill, A. Ruciński, Neighbour-distinguishing edge colourings of random regular graphs. *Electron. J. Combin.* **13**, #R77 (2006)
169. J.R. Griggs, R.K. Yeh, Labelling graphs with a condition with at distance 2. *SIAM J. Discrete Math.* **5**, 586–595 (1992)

170. H. Grötzsch, Ein Drefarbensatz für dreikreisfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Mat-Natur. Reche.* **8**, 109–120 (1959)
171. B. Grünbaum, Acyclic colorings of planar graphs. *Israel J. Math.* **14**, 390–408 (1973)
172. F. Guldan, Acyclic chromatic index and linear arboricity of graphs. *Math. Slovaca* **41**, 21–27 (1991)
173. S. Gutner, The complexity of planar graph choosability. *Discrete Math.* **159**, 119–130 (1996)
174. E. Győri, M. Horňák, C. Palmer, M. Woźniak, General neighbour-distinguishing index of a graph. *Discrete Math.* **308**, 827–831 (2008)
175. E. Győri, C. Palmer, A new type of edge-derived vertex coloring. *Discrete Math.* **309**, 6344–6352 (2009)
176. G. Hajós, Über eine Konstruktion nicht n-färbbarer Graphen. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg. Math.-Nat. Reihe.* **10**, 116–117 (1961)
177. F. Harary, M. Plantholt, The point-distinguishing chromatic index, in *Graphs and Application*, ed. by F. Harary, J.S. Maybee (Wiley-Interscience, New York, 1985), pp. 147–162
178. T. Hasunumaa, T. Ishiib, H. Onoc, Y. Unod, A tight upper bound on the $(2, 1)$ -total labeling number of outerplanar graphs. *J. Discrete Algorithms*, **14**, 189–206 (2012)
179. H. Hatami, $\Delta + 300$ is a bound on the adjacent vertex distinguishing edge chromatic number. *J. Combin. Theory Ser. B* **95**, 246–256 (2005)
180. I. Havel, O zbarvitelnosti rovinných grafů třemi barvami. *Math. Geometrie Theorie Grafů*, 89–91 (1970)
181. F. Havet, J.V.D. Heuvel, C. McDiarmid, B. Reed, List colouring squares of planar graphs. *Electron. Note Discrete Math.* **29**, 515–519 (2007)
182. F. Havet, B. Reed, J.S. Sereni, $L(2, 1)$ -labelling of graphs, in *Proceedings of the ACM-SIAM Symposium on Discrete Algorithm (SODA 08)*, San Francisco, 20–22, 2008, pp. 621–630
183. F. Havet, S. Thomassé, Complexity of $(p, 1)$ -total labelling. *Discrete Appl. Math.* **157**, 2859–2870 (2009)
184. F. Havet, M.-L. Yu, $(d, 1)$ -Total labelling of graphs. Technical report 4650, INRIA, 2002
185. F. Havet, M.-L. Yu, $(p, 1)$ -Total labelling of graphs. *Discrete Math.* **308**, 496–513 (2008)
186. F. Havet, M.-L. Yu, Corrigendum to “ $(p, 1)$ -total labelling of Graphs”. *Discrete Math.* **310**, 2219–2220 (2010)
187. P.J. Heawood, Map colour theorem. *Q. J. Pure Appl. Math.* **24**, 332–338 (1890)
188. P. Hell, A. Raspaud, J. Stacho, On injective colouring of chordal graphs. In *LATIN2008: Theoretical informatics, Lecture Notes in Computer Science*, vol. 4957 (Springer, Berlin, 2008), pp. 520–530
189. J.V.D. Heuvel, S. McGuinness, Coloring of the square of planar graph. *J. Graph Theory* **42**, 110–124 (2003)
190. H. Hind, M. Molloy, B. Reed, Colouring a graph frugally. *Combinatorica* **17**, 469–482 (1997)
191. H. Hind, M. Molloy, B. Reed, Total coloring with $\Delta + \text{poly}(\log \Delta)$ colors. *SIAM J. Comput.* **28**, 816–821 (1999)
192. H. Hocquard, M. Montassier, Every planar graph without cycles of length 4 to 12 is acyclically 3-choosable. *Inform. Process. Lett.* **109**, 1193–1196 (2009)
193. H. Hocquard, M. Montassier, Acyclic coloring of graphs with maximum degree five, 2010. <http://hal.archives-ouvertes.fr/hal-00375166/fr/>
194. H. Hocquard, M. Montassier, A. Raspaud, A note on the acyclic 3-choosability of some planar graphs. *Discrete Appl. Math.* **158**, 1104–1110 (2010)
195. M. Horňák, R. Soták, Observability of complete multipartite graphs with equipotent parts. *Ars Combin.* **41**, 289–301 (1995)
196. M. Horňák, R. Soták, The fifth jump of the point-distinguishing chromatic index of $K_{n,n}$. *Ars Combin.* **42**, 233–242 (1996)
197. M. Horňák, R. Soták, Asymptotic behaviour of the observability of Q_n . *Discrete Math.* **176**, 139–148 (1997)
198. M. Horňák, R. Soták, Localization of jumps of the point-distinguishing chromatic index of $K_{n,n}$. *Discuss. Math. Graph Theory* **17**, 243–251 (1997)

199. M. Horňák, R. Soták, General neighbour-distinguishing index via chromatic number. *Discrete Math.* **310**, 1733–1736 (2010)
200. M. Horňák, N. Zagaglia Salvi, On the point-distinguishing chromatic index of complete bipartite graphs. *Ars Combin.* **80**, 75–85 (2006)
201. J. Hou, Some topics on restricted coloring problems of graphs. Ph.D. Dissertation, Shandong University, Jinan, 2009
202. J. Hou, J. Wu, G. Liu, B. Liu, Acyclic edge colorings of planar graphs and series-parallel graphs. *Sci. China Ser. A Math.* **52**, 605–616 (2009)
203. J. Hou, J. Wu, G. Liu, B. Liu, Acyclic edge chromatic number of outerplanar graphs, *J. Graph Theory* **64**, 22–36 (2010)
204. J. Hou, Y. Zhu, G. Liu, J. Wu, M. Lan, Total colorings of planar graphs without small cycles. *Graphs Combin.* **24**, 91–100 (2008)
205. J. Huang, H. Sun, W. Wang, D. Chen, (2,1)-Total labelling of trees with spares vertices of maximum degree. *Inform. Proc. Lett.* **109**, 119–203 (2009)
206. D. Hudák, F. Kardö, B. Lužar, R. Soták, R. Škrekovski, Acyclic edge coloring planar graphs with Δ colors. *Discrete Appl. Math.* **160**, 1356–1368 (2012)
207. J. Hulgan, Concise proofs for adjacent vertex-distinguishing total colorings. *Discrete Math.* **309**, 2548–2550 (2009)
208. J.P. Hutchinson, B. Richter, P. Seymour, Coloring eulerian triangulations. *J. Combin. Theory Ser. B* **84**, 225–239 (2002)
209. R.E. Jamison, G.L. Matthews, Acyclic colorings of products of cycles. *Bull. Inst. Combin. Appl.* **54**, 59–76 (2008)
210. R.E. Jamison, G.L. Matthews, On the acyclic chromatic number of hamming graphs. *Graphs Combin.* **24**, 349–360 (2008)
211. R.E. Jamison, G.L. Matthews, J. Villalpando, Acyclic colorings of products of trees. *Inform. Process. Lett.* **99**, 7–12 (2006)
212. T.R. Jensen, B. Toft, *Graph Coloring Problems* (Wiley, New York, 1995)
213. T.R. Jensen, B. Toft, Choosability versus chromaticity. *Geombinatorics* **5**, 45–64 (1995)
214. P.K. Jha, A. Narayanan, P. Sood, K. Sundaram, V. Sunder, On $L(2, 1)$ -labelling of the Cartesian product of a cycle and a path. *Ars Combin.* **55**, 81–89 (2000)
215. B. Joshi, K. Kothapalli, On acyclic vertex coloring of grid like graphs, in *International Conference on Recent Trends in Graph Theory and Combinatorics*, Cochin, 2010, submitted for publication. <http://estar.iiit.ac.in/~kkishore/grid.pdf>
216. M. Kalkowski, A note on the 1,2-conjecture. *Electronic J. Combin.* (In Press)
217. M. Kalkowski, M. Karoński, F. Pfender, Vertex-coloring edge-weightings: towards the 1-2-3-conjecture. *J. Combin. Theory Ser. B* **100**, 347–349 (2010)
218. M. Kalkowski, M. Karoński, F. Pfender, A new upper bound for the irregularity strength of graphs. *SIAM J. Discrete Math.* **25**, 1319–1321 (2011)
219. J.-H. Kang, $L(2, 1)$ -labelling of Hamiltonian graphs with maximum degree 3. *SIAM J. Discrete Math.* **22**, 213–230 (2008)
220. R.J. Kang, T. Müller, Frugal, Acyclic and star colourings of graphs. *Discrete Appl. Math.* **159**, 1806–1814 (2011)
221. M. Karoński, T. Łuczak, A. Thomason, Edge weights and vertex colours. *J. Combin. Theory Ser. B* **91**, 151–157 (2004)
222. K. Kawarabayashi, B. Mohar, Star coloring and acyclic coloring of locally planar graphs. *SIAM J. Discrete Math.* **24**, 56–71 (2010)
223. J. Kelly, L. Kelly, Path and circuits in critical graphs. *Am. J. Math.* **76**, 786–792 (1954)
224. A.B. Kempe, On the geographical problem of four colours. *Am. J. Math.* **2**, 193–200 (1879)
225. S. Khanna, N. Linial, S. Safra, On the hardness of approximating the chromatic number. *Combinatorica* **20**, 393–415 (2000)
226. H.A. Kierstead, A. Kündgen, C. Timmons, Star coloring bipartite planar graphs. *J. Graph Theory* **60**, 1–10 (2009)

227. S.-J. Kim, S. Oum, Injective chromatic number and chromatic number of the square of graphs. Preprint (2009)
228. D. König, Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Math. Ann.* **77**, 453–465 (1916)
229. A.V. Kostochka, Acyclic 6-coloring of planar graphs. *Metody Diskret Anal.* **28**, 40–56 (1976) (in Russian)
230. A.V. Kostochka, The total coloring of a multigraph with maximum degree 4. *Discrete Math.* **17**, 161–163 (1977)
231. A.V. Kostochka, Upper bounds of chromatic functions on graphs. Ph.D. Dissertation, Novosibirsk, 1978 (in Russian)
232. A.V. Kostochka, The total chromatic number of any multigraph with maximum degree five is at most seven. *Discrete Math.* **162**, 199–214 (1996)
233. A.V. Kostochka, L.S. Mel'nikov, Note to the paper of Grünbaum on acyclic colorings. *Discrete Math.* **14**, 403–406 (1976)
234. A.V. Kostochka, D.R. Woodall, Choosability conjectures and multicircuits. *Discrete Math.* **240**, 123–143 (2001)
235. D. Král, Coloring powers of chordal graphs. *SIAM J. Discrete Math.* **18**, 451–461 (2004)
236. D. Král, P. Nejedly, Distance constrained labelings of K_4 -minor free graphs. *Discrete Math.* **309**, 5745–5756 (2009)
237. D. Král, R. Škrekovski, A theorem about channel assignment problem. *SIAM J. Discrete Math.* **16**, 426–437 (2003)
238. D. Král, R. Škrekovski, The last excluded case of Dirac's Map-Color Theorem for choosability. *J. Graph Theory* **51**, 319–354 (2006)
239. W. Kristiana, Slamin, Total vertex irregular labelings of wheels, fans, suns and friendship graphs. *J. Combin. Math. Combin. Comput.* **65**, 103–112 (2008)
240. H.V. Kronk, A.T. White, A 4-color theorem for toroidal graphs. *Proc. Am. Math. Soc.* **34**, 83–85 (1972)
241. A. Kündgen, C. Timmons, Star coloring planar graphs from small lists. *J. Graph Theory* **63**, 324–337 (2010)
242. K. Kuratowski, Sur le problème des courbes gauches en topologie. *Fund. Math.* **15**, 271–283 (1930)
243. P.C.B. Lam, W.C. Shiu, Z.M. Song, The 3-choosability of plane graphs of girth 4. *Discrete Math.* **294**, 297–301 (2005)
244. P.C.B. Lam, B. Xu, J. Liu, The 4-choosability of plane graphs without 4-cycles. *J. Combin. Theory Ser. B* **76**, 117–126 (1999)
245. J. Lehel, Facts and quests on degree irregular assignments, in *Graph Theory, Combinatorics and Applications* (Wiley, New York, 1991), pp. 765–782
246. X. Li, R. Luo, Edge coloring of embedded graphs with large girth. *Graphs Combin.* **19**, 393–401 (2003)
247. C. Li, W. Wang, A. Raspaud, Upper bounds on the linear chromatic number of a graph. *Discrete Math.* **311**, 232–238 (2011)
248. K.-W. Lih, D.-F. Liu, W. Wang, On $(d, 1)$ -total numbers of graphs. *Discrete Math.* **309**, 3767–3773 (2009)
249. K.-W. Lih, Z. Song, W. Wang, K. Zhang, A note on list improper coloring planar graphs. *Appl. Math. Lett.* **14**, 269–273 (2001)
250. K.-W. Lih, W. Wang, Coloring the square of an outerplanar graph. *Taiwan. J. Math.* **10**, 1015–1023 (2006)
251. K.-W. Lih, W. Wang, X. Zhu, Coloring the square of a K_4 -minor free graph. *Discrete Math.* **269**, 303–309 (2003)
252. X. Liu, M. An, Y. Gao, An upper bound for the adjacent vertex-distinguishing total chromatic number of a graph. *J. Math. Res. Expos.* **29**, 343–348 (2009)
253. D.-F. Liu, X. Zhu, Circular distance two labeling and the λ -number for outerplanar graphs. *SIAM J. Discrete Math.* **19**, 281–293 (2005)

254. B. Liu, G. Liu, Vertex-distinguishing edge colorings of graphs with degree sum conditions. *Graphs Combin.* **26**, 781–791 (2010)
255. B. Liu, G. Liu, On the adjacent vertex distinguishing edge colourings of graphs. *Int. J. Comput. Math.* **87**, 726–732 (2010)
256. L. Liu, Z. Zhang, J. Wang, On the adjacent strong edge coloring of outer plane graphs. *J. Math. Res. Expos.* **25**, 255–266 (2005)
257. L. Lovász, A characterization of perfect graphs. *J. Combin. Theory Ser. B* **13**, 95–98 (1972)
258. L. Lovász, Three short proofs in graph theory. *J. Combin. Theory Ser. B* **19**, 269–271 (1975)
259. H. Lu, Y. Wang, W. Wang, Y. Bu, M. Montassier, A. Raspaud, On the 3-colorability of planar graphs without 4-, 7- and 9-cycles. *Discrete Math.* **309**, 4596–4607 (2009)
260. H. Lu, Q. Yu, C. Zhang, Vertex-coloring 2-edge-weighting of graphs. *Eur. J. Combin.* **32**, 21–27 (2011)
261. X. Luo, The 4-choosability of toroidal graphs without intersecting triangles. *Inform. Process. Lett.* **102**, 85–91 (2007)
262. X. Luo, Chromatic number of square of maximal outerplanar graphs. *Appl. Math. J. Chin. Univ. Ser. B* **22**, 163–168 (2007)
263. X. Luo, M. Chen, W. Wang, On 3-colorable planar graphs without cycles of four lengths. *Inform. Process. Lett.* **103**, 150–156 (2007)
264. B. Luzar, R. Škrekovski, M. Tancer, Injective colorings of planar graphs with few colors. *Discrete Math.* **309**, 5636–5649 (2009)
265. B. Luzar, Planar graphs with largest injective chromatic numbers, in: IMFM Preprint Series, **48**, 9–11 (2010)
266. Y. Makarychev, A short proof of Kuratowski's graph planarity criterion. *J. Graph Theory* **25**, 129–131 (1997)
267. C. McDiarmid, B. Reed, On total colourings of graphs. *J. Combin. Theory Ser. B* **57**, 122–130 (1993)
268. M. Mirzakhani, A small non-4-choosable planar graph. *Bull. Inst. Combin. Appl.* **17**, 15–18 (1996)
269. J. Mitchem, Every planar graph has an acyclic 8-coloring. *Duke Math. J.* **41**, 177–181 (1974)
270. J. Miškuf, R. Škrekovski, M. Tancer, Backbone colorings and generalized Mycielski graph. *SIAM J. Discrete Math.* **23**, 1063–1070 (2009)
271. J. Miškuf, R. Škrekovski, M. Tancer, Backbone coloring of graphs with bounded degree. *Discrete Appl. Math.* **158**, 534–542 (2010)
272. B. Mohar, Acyclic colorings of locally planar graphs. *Eur. J. Combin.* **26**, 491–503 (2005)
273. M. Molloy, B. Reed, Further algorithmic aspects of the local lemma, in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (ACM New York, New York, 1998), pp. 524–529
274. M. Molloy, B. Reed, A bound on the total chromatic number. *Combinatorica* **18**, 241–280 (1998)
275. M. Molloy, B. Reed, *Graph Coloring and the Probabilistic Method* (Springer, Berlin, 2002)
276. M. Molloy, M.R. Salavatipour, A bound on the chromatic number of the square of a planar graph. *J. Combin. Theory Ser. B* **94**, 189–213 (2005)
277. M. Montassier, A note on the not 3-choosability of some families of planar graphs. *Inform. Process. Lett.* **99**, 68–71 (2006)
278. M. Montassier, Acyclic 4-choosability of planar graphs with girth at least 5, in *Trends in Mathematics: Graph Theory in Paris* (Birkhäuser, Basel, 2007), pp. 299–310
279. M. Montassier, P. Ochem, A. Raspaud, On the acyclic choosability of graphs. *J. Graph Theory* **51**, 281–300 (2006)
280. M. Montassier, A. Raspaud, $(d, 1)$ -Total labelling of graphs with a given maximum average degree. *J. Graph Theory* **51**, 93–109 (2006)
281. M. Montassier, A. Raspaud, A note on 2-facial coloring of plane graphs. *Inform. Process. Lett.* **98**, 235–241 (2006)

282. M. Montassier, A. Raspaud, W. Wang, Acyclic 4-choosability of planar graphs without cycles of specific lengths, in *Topics in Discrete Mathematics: Algorithms and Combinatorics*, vol. 26 (Springer, Berlin, 2006), pp. 473–491
283. M. Montassier, A. Raspaud, W. Wang, Bordeaux 3-color conjecture and 3-choosability. *Discrete Math.* **306**, 573–579 (2006)
284. M. Montassier, A. Raspaud, W. Wang, Acyclic 5-choosability of planar graphs without small cycles. *J. Graph Theory* **54**, 245–260 (2007)
285. R. Muthu, N. Narayanan, C.R. Subramanian, Improved bounds on acyclic edge coloring. *Electron. Note Discrete Math.* **19**, 171–177 (2005)
286. R. Muthu, N. Narayanan, C.R. Subramanian, Improved bounds on acyclic edge coloring. *Discrete Math.* **307**, 3063–3069 (2007)
287. R. Muthu, N. Narayanan, C.R. Subramanian, Acyclic edge coloring of outerplanar graphs, in *Algorithmic Aspects in Information and Management, 3rd AAIM*, Portland. Lecture Notes in Computer Science, vol. 4508, 2007, pp. 144–152
288. R. Muthu, N. Narayanan, C.R. Subramanian, Acyclic edge colouring of partial 2-trees. <http://www.imsc.res.in/~narayan/p2t.pdf>
289. J. Mycielski, On the coloring of graphs. *Colloq. Math.* **3**, 161–162 (1955)
290. T. Nierhoff, A tight bound on the irregularity strength of graphs. *SIAM J. Discrete Math.* **13**, 313–323 (2000)
291. J. Nešetřil, P. Ossana de Mendez, Colorings and homomorphisms of minor closed classes, in *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, Algorithms and Combinatorics (Spring, Berlin, 2003), pp. 651–664
292. J. Nešetřil, N.C. Wormald, The acyclic edge chromatic number of a random d -regular graph is $d + 1$. *J. Graph Theory* **49**, 69–74 (2005)
293. C.T. Palmer, Edge weightings and the chromatic number. Ph.D. Dissertation, Central European University, 2008
294. J. Przybyło, A note on a neighbour-distinguishing regular graphs total-weighting. *Electron. J. Combin.* **15**, #R35 (2008)
295. J. Przybyło, Irregularity strength of regular graphs. *Electron. J. Combin.* **15**, #R82 (2008)
296. J. Przybyło, Linear bound on the irregularity strength and the total vertex irregularity strength of graphs. *SIAM J. Discrete Math.* **23**, 511–516 (2009)
297. J. Przybyło, M. Woźniak, On a 1,2 conjecture. *Discrete Math. Theor. Comput. Sci.* **12**, 101–108 (2010)
298. J. Przybyło, M. Woźniak, Total weight choosability of graphs. *J. Graph Theory* **66**, 198–212 (2011)
299. A. Raspaud, W. Wang, Linear coloring of planar graphs with large girth. *Discrete Math.* **309**, 5678–5686 (2009)
300. B. Reed, ω , Δ , and χ . *J. Graph Theory* **27**, 177–212 (1998)
301. B. Reed, A strengthening of Brooks' theorem. *J. Combin. Theory Ser. B* **76**, 136–149 (1999)
302. G. Ringel, *Map Color Theorem* (Springer, New York, 1974)
303. G. Ringel, J.W.T. Youngs, Solution of the Heawood map-color problem. *Proc. Natl. Acad. Sci. USA* **60**, 438–445 (1968)
304. F.S. Roberts, T -colorings of graphs: recent results and open problems. *Discrete Math.* **93**, 229–245 (1991)
305. N. Robertson, P. Seymour, R. Thomas, Hadwiger's conjecture for K_5 -free graphs. *Combinatorica* **14**, 279–361 (1993)
306. M. Rosenfeld, On the total coloring of certain graphs. *Israel J. Math.* **9**, 396–402 (1971)
307. J. Rudašová, R. Soták, Vertex-distinguishing proper edge colourings of some regular graphs. *Discrete Math.* **308**, 795–802 (2008)
308. D. Sakai, Labelling chordal graphs with a condition at distance two. *SIAM J. Discrete Math.* **7**, 133–140 (1994)
309. A.N.M. Salman, λ -Backbone colorings numbers of split graphs with tree backbones, in *Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications*, Universiti Sains Malaysia, Penang, 13–15 June 2006

310. D.P. Sanders, Y. Zhao, A note on the three coloring problem. *Graphs Combin.* **11**, 92–94 (1995)
311. D.P. Sanders, Y. Zhao, On total 9-coloring planar graphs of maximum degree seven. *J. Graph Theory* **31**, 67–73 (1999)
312. D.P. Sanders, Y. Zhao, Planar graphs of maximum degree seven are class I. *J. Combin. Theory Ser. B* **83**, 201–212 (2001)
313. C. Schwarz, D.S. Troxell, $L(2, 1)$ -labelling of products of two cycles. *Discrete Appl. Math.* **154**, 1522–1540 (2006)
314. L. Shen, Y. Wang, A sufficient condition for a planar graph to be 3-choosable. *Inform. Process. Lett.* **104**, 146–151 (2007)
315. L. Shen, Y. Wang, On the 7 total colorability of planar graphs with maximum degree 6 and without 4-cycles. *Graphs Combin.* **25**, 401–407 (2009)
316. L. Shen, Y. Wang, Total colorings of planar graphs with maximum degree at least 8. *Sci. China Ser. A* **52**, 1733–1742 (2009)
317. Q. Shu, W. Wang, Acyclic chromatic indices of planar graphs with girth at least five. *J. Comb. Optim.* **23**, 140–157 (2012)
318. Q. Shu, W. Wang, Y. Wang, Acyclic chromatic indices of planar graphs with girth at least four. *J. Graph Theory.* (2012) doi: 10.1002/jgt.21683
319. R. Škrekovski, A Grötzsch-type theorem for list colorings with impropriety one. *Combin. Probab. Comput.* **8**, 493–507 (1999)
320. R. Škrekovski, List improper colorings of planar graphs. *Combin. Prob. Comput.* **8**, 293–299 (1999)
321. S. Skulrattanakulchai, Acyclic colorings of subcubic graphs. *Inform. Process. Lett.* **92**, 161–167 (2004)
322. R. Steinberg, The state of the three color problem, in *Quo Vadis, Graph Theory?* ed. by J. Gimbel, J.W. Kennedy, L.V. Quintas. Annals of Discrete Mathematics, vol 55 (1993) pp. 211–248
323. X. Sun, J. Wu, Acyclic edge colorings of planar graphs without short cycles, in *The 7-th ISORA'08*, Lijiang, 2008, pp. 325–329
324. G. Szekeres, H.S. Wilf, An inequality for the chromatic number of a graph. *J. Combin. Theory* **4**, 1–3 (1968)
325. K. Taczuk, M. Woźniak, A note on the vertex-distinguishing index for some cubic graphs. *Opuscula Math.* **24**, 223–229 (2004)
326. C. Thomassen, Planarity and duality of finite and infinite graphs. *J. Combin. Theory Ser. B* **29**, 244–271 (1980)
327. C. Thomassen, Kuratowski's theorem. *J. Graph Theory* **5**, 225–241 (1981)
328. C. Thomassen, Every planar graph is 5-choosable. *J. Combin. Theory Ser. B* **62**, 180–181 (1994)
329. C. Thomassen, Grötzsch 3-color theorem and its counterparts for the torus and the projective plane. *J. Combin. Theory Ser. B* **62**, 268–279 (1994)
330. C. Thomassen, Five-coloring graphs on the torus. *J. Combin. Theory Ser. B* **62**, 11–33 (1994)
331. C. Thomassen, 3-List-coloring planar graphs of girth 5. *J. Combin. Theory Ser. B* **64**, 101–107 (1995)
332. C. Thomassen, Applications of Tutte cycles. Technical report, Technical University of Denmark, 2001
333. C. Thomassen, Some remarks on Hajós' conjecture. *J. Combin. Theory Ser. B* **93**, 95–105 (2005)
334. C. Timmons, Star-coloring planar graphs. Master's Thesis, California State University San Marcos, May 2007
335. C. Timmons, Star coloring high girth planar graphs. *Electron. J. Combin.* **15**, #R124 (2008)
336. O. Togni, Irregularity strength of the toroidal grid. *Discrete Math.* **165/166**, 609–620 (1997)
337. S. Varagani, V. Ch. Venkaiah, K. Yadav, K. Kothapalli, Acyclic vertex WQ3Qcoloring of graphs of maximum degree six. *Electron. Note Discrete Math.* **35**, 177–182 (2009)

338. N. Vijayaditya, On total chromatic number of a graph. *J. Lond. Math. Soc.* **3**(2), 405–408 (1971)
339. V.G. Vizing, On the estimate of the chromatic class of a p -graphs. *Metody Diskret. Analiz.* **3**, 25–30 (1964) (in Russian)
340. V.G. Vizing, Some unsolved problems in graph theory. *Uspekhi Mat. Nauk* **23**, 117–134 (1968) (in Russian)
341. V.G. Vizing, Vertex coloring with given colors. *Metody Diskret. Analiz.* **29**, 3–10 (1976) (in Russian)
342. M. Voigt, List colourings of planar graphs. *Discrete Math.* **120**, 215–219 (1993)
343. M. Voigt, A not 3-choosable planar graph without 3-cycles. *Discrete Math.* **146**, 325–328 (1995)
344. M. Voigt, A non-3-choosable planar graph without cycles of length 4 and 5. *Discrete Math.* **307**, 1013–1015 (2007)
345. K. Wagner, Über eine Eigenschaft der ebene Komplexe. *Math. Ann.* **114**, 570–590 (1937)
346. P.J. Wan, Near-optimal conflict-free channel set assignments for an optical cluster-based hypercube network. *J. Comb. Optim.* **1**, 179–186 (1997)
347. H. Wang, On the adjacent vertex distinguishing total chromatic number of the graphs with $\Delta(G) = 3$. *J. Comb. Optim.* **14**, 87–109 (2007)
348. W. Wang, The $L(2, 1)$ -labelling of trees. *Discrete Appl. Math.* **154**, 598–603 (2006)
349. W. Wang, Total chromatic number of planar graphs with maximum degree ten. *J. Graph Theory* **54**, 91–102 (2007)
350. W. Wang, Y. Bu, M. Montassier, A. Raspaud, On backbone coloring of graphs. *J. Comb. Optim.* **23**, 79–93 (2012)
351. W. Wang, L. Cai, Labelling planar graphs without 4-cycles with a condition on distance two. *Discrete Appl. Math.* **156**, 2241–2249 (2008)
352. W. Wang, D. Chen, (2, 1)-Total number of trees with maximum degree three. *Inform. Process. Lett.* **109**, 805–810 (2009)
353. W. Wang, M. Chen, Planar graphs without 4, 6 and 8-cycles are 3-colorable. *Sci. China Ser. A* **50**, 1552–1562 (2007)
354. W. Wang, M. Chen, On 3-colorable planar graphs without prescribed cycles. *Discrete Math.* **307**, 2820–2825 (2007)
355. W. Wang, M. Chen, Planar graphs without 4-cycles are acyclically 6-choosable. *J. Graph Theory* **61**, 307–323 (2009)
356. W. Wang, J. Huang, H. Sun, D. Huang, (2,1)-Total number of joins of paths and cycles. *Taiwan. J. Math.* **16**, 605–619 (2012)
357. W. Wang, C. Li, Linear coloring of graphs embeddable in a surface of nonnegative characteristic. *Sci. China Ser. A* **52**, 991–1003 (2009)
358. W. Wang, K.-W. Lih, The 4-choosability of planar graphs without 6-cycles. *Australas. J. Combin.* **24**, 157–164 (2001)
359. W. Wang, K.-W. Lih, Choosability and edge choosability of planar graphs without five cycles. *Appl. Math. Lett.* **15**, 561–565 (2002)
360. W. Wang, K.-W. Lih, Choosability and edge choosability of planar graphs without intersection triangles. *SIAM J. Discrete Math.* **15**, 538–545 (2003)
361. W. Wang, K.-W. Lih, Note on coloring the square of an outerplanar graph. *Ars Combin.* **86**, 89–95 (2008)
362. Y. Wang, H. Lu, M. Chen, Planar graphs without cycles of length 4, 5, 8, or 9 are 3-choosable. *Discrete Math.* **310**, 147–158 (2010)
363. W. Wang, X. Luo, Some results on distance two labelling of outerplanar graphs. *Acta Math. Appl. Sin. English Ser.* **25**, 21–32 (2009)
364. Y. Wang, X. Mao, H. Lu, W. Wang, On 3-colorability of planar graphs without adjacent short cycles. *Sci. China Ser. A* **53**, 1129–1132 (2010)
365. W. Wang, Q. Shu, K. Wang, P. Wang, Acyclic chromatic indices of planar graphs with large girth. *Discrete Appl. Math.* **159**, 1239–1253 (2011)

366. W. Wang, Q. Shu, Acyclic chromatic indices of K_4 -minor free graphs (in Chinese). *Sci. Sin. Math.* **41**, 733–744 (2011)
367. W. Wang, P. Wang, Acyclic chromatic indices of planar graphs with large girth Preprint (2008)
368. W. Wang, P. Wang, Adjacent vertex distinguishing total colorings of K_4 -minor free graphs. *Sci. China Ser. A* **39**, 1462–1467 (2009) (in Chinese)
369. W. Wang, Y. Wang, $L(p, q)$ -labelling of K_4 -minor free graphs. *Inform. Process. Lett.* **98**, 169–173 (2006)
370. W. Wang, Y. Wang, Adjacent vertex distinguishing total colorings of graphs with lower average degree. *Taiwan. J. Math.* **12**, 979–990 (2008)
371. W. Wang, Y. Wang, Adjacent vertex distinguishing edge-colorings of graphs with smaller maximum average degree. *J. Comb. Optim.* **19**, 471–485 (2010)
372. W. Wang, Y. Wang, Adjacent vertex distinguishing edge colorings of K_4 -minor free graphs. *Appl. Math. Lett.* **24**, 2034–2037 (2011)
373. Y. Wang, W. Wang, Adjacent vertex distinguishing total colorings of outerplanar graphs. *J. Comb. Optim.* **19**, 123–133 (2010)
374. P. Wang, J. Wu, A note on total colorings of planar graphs without 4-cycles. *Discuss. Math. Graph Theory* **24**, 125–135 (2004)
375. Y. Wang, Q. Wu, L. Shen, Planar graphs without cycles of length 4, 7, 8, or 9 are 3-choosable. *Discrete Appl. Math.* **159**, 232–239 (2011)
376. T. Wang, Q. Yu, On vertex-coloring 13-edge-weighting. *Front. Math. China* **3**, 1–7 (2008)
377. G. Wegner, Note on a paper by B. Grunbaum on acyclic colorings. *Israel J. Math.* **14**, 409–412 (1973)
378. G. Wegner, Graphs with given diameter and a coloring problem. Technical report, University of Dortmund, Dortmund, 1977
379. D. B. West, *Introduction to Graph Theory* (Pearson Education, USA, 2002)
380. C. Whitehead, N. Zagaglia Salvi, Observability of the extended Fibonacci cubes. *Discrete Math.* **266**, 431–440 (2003)
381. M.A. Whittlesey, J.P. Georges, D.W. Mauro, On the λ -number of Q_n and related graphs. *SIAM J. Discrete Math.* **8**, 99–506 (1995)
382. T.-L. Wong, X. Zhu, D. Yang, List total weighting of graphs. *Bolyai Soc. Math. Stud.* **20**, 337–353 (2010)
383. T.-L. Wong, X. Zhu, Total weight choosability of graphs. *J. Graph Theory* **66**, 198–212 (2011)
384. B. Xu, H. Zhang, Every toroidal graph without adjacent triangles is $(4, 1)^*$ -choosable. *Discrete Appl. Math.* **155**, 74–78 (2007)
385. K. Yadava, S. Varagani, K. Kothapalli, V. Ch. Venkaiah, Acyclic vertex coloring of graphs of maximum degree 4. <http://www.jaist.ac.jp>
386. K. Yadava, S. Varagani, K. Kothapalli, V. Ch. Venkaiah, Acyclic vertex coloring of graphs of maximum degree 5. *Discrete Math.* **311**, 342–348 (2011)
387. H.P. Yap, Total colourings of graphs, in *Lecture Notes in Mathematics*, vol. 1623 (Springer, London, 1996)
388. R.G. Yeh, A survey on labelling graphs with a condition at distance two. *Discrete Math.* **306**, 1217–1231 (2006)
389. H.G. Yeh, X. Zhu, 4-Colorable 6-regular toroidal graphs. *Discrete Math.* **273**, 261–274 (2003)
390. D. Yu, J. Hou, G. Liu, B. Liu, L. Xu, Acyclic edge coloring of planar graphs with large girth. *Theor. Comput. Sci.* **410**, 5196–5200 (2009)
391. R. Yuster, Linear coloring of graphs. *Discrete Math.* **185**, 293–297 (1998)
392. N. Zagaglia Salvi, On the point-distinguishing chromatic index of $K_{n,n}$. *Ars Combin.* **25**, 93–104 (1988)
393. N. Zagaglia Salvi, On the value of the point-distinguishing chromatic index of $K_{n,n}$. *Ars Combin.* **29**, 235–244 (1990)
394. L. Zhang, Every planar graph with maximum degree 7 is of class 1. *Graphs Combin.* **16**, 467–495 (2000)

395. Z. Zhang, X. Chen, J. Li, B. Yao, X. Lu, J. Wang, On adjacent-vertex-distinguishing total coloring of graphs. *Sci. China Ser. A* **48**, 289–299 (2005)
396. Z. Zhang, L. Liu, J. Wang, Adjacent strong edge coloring of graphs. *Appl. Math. Lett.* **15**, 623–626 (2002)
397. Z. Zhang, D.R. Woodall, B. Yao, J. Li, X. Chen, L. Bian, Adjacent strong edge colorings and total colorings of regular graphs. *Sci. China Ser. A* **52**, 973–980 (2009)
398. L. Zhang, B. Wu, Three-coloring planar graphs without certain small cycles. *Graph Theory Notes NY* **46**, 27–30 (2004)
399. L. Zhang, B. Wu, A note on 3-choosability of planar graphs without certain cycles. *Discrete Math.* **297**, 206–209 (2005)
400. H. Zhang, B. Xu, Acyclic 5-choosability of planar graphs with neither 4-cycles nor chordal 6-cycles. *Discrete Math.* **309**, 6087–6091 (2009)
401. H. Zhang, B. Xu, Z. Sun, Every plane graph with girth at least 4 without 8- and 9-circuits is 3-choosable. *Ars Combin.* **80**, 247–257 (2006)
402. A. Zykov, On some properties of linear complexes. *Mat. Sbornik* **24**, 163–188 (1949) (in Russian)

Online and Semi-online Scheduling*

Zhiyi Tan and An Zhang

Contents

1	Introduction	2192
1.1	Scheduling	2192
1.2	Online Problems and Competitive Analysis	2194
1.3	Structure and Notation	2195
2	Online Over List	2196
2.1	Non-preemptive Scheduling	2196
2.2	Preemptive Scheduling	2201
2.3	Randomized Algorithm	2202
2.4	Other Objectives	2204
3	Semi-online Scheduling	2205
3.1	Motivation and Taxonomy	2205
3.2	Results on Basic Semi-online Models of Type I	2209
3.3	Results on Basic Semi-online Models of Type II	2222
3.4	Results on Combined Semi-online Models	2226
3.5	Results on Disturbed Semi-online Models	2231
4	Online Over Time	2231
4.1	Single Machine Scheduling	2231
4.2	Results on Parallel Machine Scheduling	2234

*Supported by the National Natural Science Foundation of China (10971191, 11271324), Zhejiang Provincial Natural Science Foundation of China (LR12A01001), and Fundamental Research Funds for the Central Universities.

Z. Tan (✉)

Department of Mathematics, Zhejiang University, Hangzhou, People's Republic of China
e-mail: tanz@zju.edu.cn

A. Zhang

Institute of Operational Research and Cybernetics, Hangzhou Dianzi University, Hangzhou,
People's Republic of China
e-mail: anzhanghz@yahoo.com.cn

5 Other Variants of Online Scheduling.....	2236
5.1 Online Shop Scheduling.....	2236
5.2 Batch Scheduling.....	2237
5.3 Online Scheduling with Machine Eligibility.....	2239
6 Conclusion.....	2242
Cross-References.....	2245
Recommended Reading.....	2245

Abstract

During the last two decades, online and semi-online scheduling problems have received considerable research interest. This chapter provides definitions of several online paradigms, including online over list, online over time, and semi-online. This chapter presents details of the basic online algorithms and referencing other significant results.

1 Introduction

1.1 Scheduling

Scheduling is the earliest studied branch in combinatorial optimization and also one of the problems with the most fruitful results achieved. Early scheduling problems are motivated from manufacturing systems. Later, more and more scheduling problems are formalized in the areas of information science, supply chain management, public management, and so on. In the following, we summarize the basic concepts and notations which will be used later. For more detailed introduction on scheduling theory, please refer to [21, 36, 117, 143].

Scheduling problems can be generally described as follows. There is a sequence \mathcal{J} of n jobs J_1, J_2, \dots, J_n , which need to be processed on a set \mathcal{M} of m machines M_1, M_2, \dots, M_m . A *schedule* is an assignment for each job to one or more time intervals of one or more machines. Schedules that satisfy various requirements of the problem are called *feasible*. Scheduling problems are specified by the machine environment, the job characteristics, and an optimality criterion.

There are two basic types of machine systems: single-stage system and multi-stage system. In a single-stage system, each job requires one operation, while in multiple-stage system the jobs may require several operations at different stages. There are also two classical single-stage systems: single machine and parallel machines. In a single machine system, job J_j must be processed on the unique machine M_1 with a *processing time* p_j , $j = 1, 2, \dots, n$. In a parallel machines system, each job can be processed by *one* of the m machines. The time needed for job J_j to be processed on M_i is p_{ji} , $j = 1, \dots, n$, $i = 1, \dots, m$. If $p_{ji} = p_j$ for any $i = 1, \dots, m$ and $j = 1, \dots, n$, then the machine system is called *identical*. If there exist s_i , $i = 1, \dots, m$ such that $\frac{p_{j_1}}{p_{j_2}} = \frac{s_{i_2}}{s_{i_1}}$ for any $j = 1, \dots, n$, then the machine system is called *uniform*, and s_i is called the *speed* of M_i , $i = 1, \dots, m$. W.l.o.g., we assume $s_1 \geq s_2 \geq \dots \geq s_m = 1$ and let $p_j = p_{jm}$, which is the

normalized *processing time* of job J_j . In the case of two uniform machines, it is of more convenience for discussion to replace the speed parameters s_1 and $s_2 (= 1)$ by the *speed ratio* $s = \frac{s_1}{s_2}$, and both are equivalent. Parallel machines system that does not fall into above categories is called *unrelated*. In a multi-stage system, each operation of a job has a separate processing time, and different operations of the same job cannot be scheduled at the same time. The multi-stage system is usually distinguished into *open shop*, when different operations of the same job may be scheduled in any order; *flow shop*, if the order of the operations is fixed and same for all the jobs; and *job shop*, if the order of the operations is fixed and possibly different for each job.

Apart from the processing time of the jobs, the processing mode can also be classified into the job characteristics. Some scheduling models allow *preemption* of jobs, that is, the processing of a job may be interrupted and resumed later on possibly a different machine. If each job can be arbitrarily split between the machines and parts of the same job can run on different machines in parallel, this model is called *fractional assignment*. There is another possibility allowing jobs to *restart*, which means that a processing job can be stopped and restarted later to finish the full processing time.

Given a schedule σ , let $C_j(\sigma)$ (or C_j if there is no confusing) be the completion time of J_j , $j = 1, 2, \dots, n$. Some commonly used objective functions include:

The makespan: $\max_{1 \leq j \leq n} C_j(\sigma)$.

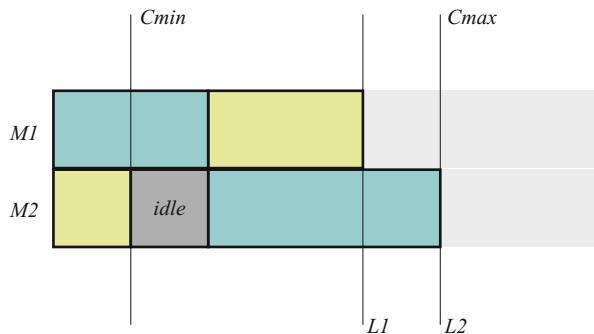
The total completion time: $\sum_{j=1}^n C_j(\sigma)$,

The total weighted completion time: $\sum_{j=1}^n w_j C_j(\sigma)$, where w_j is the weight of J_j . All these objectives are for minimization problems. There is another criteria which maximizes the continuous period of time when all machines are busy. Such criteria have applications in the sequencing of maintenance actions for modular gas turbine aircraft engines [49] and fairly allocation of indivisible goods [15]. Problems with this objective are often called *machine covering*.

The period when one machine is not assigned to any job during the scheduling process is called *idle* time. For most non-preemptive problems, there is no benefit to introduce idle times. But idle time is of great help if preemption is allowed, especially for uniform machines scheduling. Let L_i be the completion time of machine M_i in schedule σ . The makespan of σ equals to $\max_{i=1, \dots, m} L_i$. However, $\min_{i=1, \dots, m} L_i$ is not always identical with the objective value of the machine covering problem if idle times exist (Fig. 1).

In general, scheduling problems are specified in terms of a three-field notation [88]. Here, we adopt the notation from it. In this chapter, we will use $1, P, Q, R$ to denote single, identical, uniform, unrelated machine system and F, O, J to denote flow shop, open shop, and job shop system. Problems allowing preemption, fractional assignment, and restart will be denoted by *pmtn*, *frac*, and *res*, respectively. The objectives of minimizing makespan, minimizing the total completion time, and minimizing the total weighted completion time will be shortly denoted by C_{\max} , $\sum C_j$, and $\sum w_j C_j$, respectively. The objective of machine covering is denoted C_{\min} .

Fig. 1 Machine completion times and objective value of a schedule



1.2 Online Problems and Competitive Analysis

Online problems began to draw attention in the middle period of the eighties of the twentieth century. The basic character of online models is “lack of information,” while the previously studied problems having the access to full information are called *offline*. For more detailed introduction of online theories and variants, please refer to [20, 75].

For scheduling problems, there are a variety of online models, among which the following two are the most common.

Online over list: Jobs are ordered in a list and arrived one by one (at time 0). The information of a job is known to the scheduler only when all the jobs before it have been assigned. And the assignment of jobs is irrevocable.

Online over time: Each job has a *release time* after which the information of the job is known and can be processed. The release time of J_j is denoted by r_j , $j = 1, 2, \dots, n$.

Since almost all problems considered in this chapter are online, we will ignore *online over list* in the second field of the three-field notation. On the other hand, r_j will be included in the second field of the three-field notation for the online over time model.

The *competitive analysis* is the most common method in analysis of online algorithms, which is formally introduced by Sleator and Tarjan [161]. The performance of an online algorithm is measured by its *competitive ratio*, which has some similarity as *worst-case ratio* for offline problems. For a job sequence \mathcal{J} and an online algorithm A , if A produces a schedule with objective value $C^A(\mathcal{J})$ for the minimization (or maximization) problem and the optimal offline objective value is $C^*(\mathcal{J})$, then the competitive ratio of A is defined as

$$r_A = \inf_{\mathcal{J}} \{r \mid C^A(\mathcal{J}) \leq rC^*(\mathcal{J})\} \quad (\text{or } r_A = \inf_{\mathcal{J}} \{r \mid C^*(\mathcal{J}) \leq rC^A(\mathcal{J})\}).$$

If there is an online algorithm A for the problem with a competitive ratio r , then A is called an r -*competitive* algorithm. On the contrary, if no online algorithm has

a competitive ratio smaller than ρ , then the problem must have a *lower bound* ρ . Therefore, an online algorithm is called *optimal* or *best possible* for the problem if its competitive ratio just matches the lower bound. Here, optimal does not mean that it always produces an optimal schedule. It only indicates the fact that no online algorithm can be better. An online algorithm is stated to possess competitive ratio 1 if it always produces an optimal schedule, but such algorithm seldom exists. In this chapter, we use *optimal bound* to denote the competitive ratio of the optimal algorithm of the problem for simplicity. Symmetrically, a problem has *upper bound* r if there exists an online algorithm for the problem with competitive ratio r .

A *randomized algorithm* is one which somehow bases its decision making on the outcome of random coin flips. For a randomized algorithm, the objective value $C^A(\mathcal{J})$ is a random variable. Thus, the competitive ratio of a randomized algorithm (for a minimization problem) is defined as

$$r_A = \inf_{\mathcal{J}} \{r \mid E(C^A)(\mathcal{J}) \leq r C^*(\mathcal{J})\},$$

where the expectation is taken over the random choices of the algorithm. In addition, concepts of randomized upper and lower bound can also be defined accordingly. Algorithms which do not use randomization are called *deterministic algorithm*. Since deterministic algorithm can be viewed as a special randomized algorithm, the randomized lower bound cannot be larger than deterministic lower bound. In this chapter, without special mention, all algorithms (competitive ratio, lower bound) are deterministic. For a general introduction on randomized algorithms, please refer to [134].

1.3 Structure and Notation

Online scheduling boasts of extensive results; thus, it is impossible to cover all the problems in one chapter, and only part of paradigms together with the according main results is included. There have been two excellent surveys [144, 156] concerning online scheduling, and some paradigms discussed elaborately there will be omitted. We will mainly introduce paradigms approaching classical scheduling problems and withal active in recent research. There are several interesting and important problems not covered due to limited space and number of references, and we will sketch some of them in the following. Firstly, scheduling problems with deadline constraints, including interval scheduling, are not included in this chapter. Secondly, it does not cover problems concerning other objective functions, for example, the (weighted) total flow time and the L_p norm of machine completion times, together with objectives deriving from bicriteria situations, for example, scheduling with rejection and scheduling with machine cost. Thirdly, performance measures other than competitive analysis, including resource argumentation, will not be contained in this chapter. Finally, we skip problems which have specific

applications, such as scheduling problems in power management, broadcasting, and supply chain management.

The chapter is organized as follows. In Sect. 2, we survey main results for the online over list model. Section 3 is devoted to semi-online variants of online over list model. In Sect. 4, we introduce online and semi-online problems of the online over time model. Other variants of online scheduling are contained in Sect. 5.

The following notations as well as those defined above will be used frequently in the remainder of this chapter. For any $i = 1, \dots, m$, let $S_i = \sum_{l=1}^i s_l$ be the total speeds of the first i machines in uniform machine system, and we simplify S_m as S . For any $j = 1, \dots, n$, let \mathcal{J}_j be the subset of \mathcal{J} containing the first j jobs and P_j be the total processing times of the first j jobs. We simplify P_n as P . Let $p_{(j)}$ be the processing time of the job which has the j th largest processing time, that is, $p_{(1)} \geq p_{(2)} \geq \dots \geq p_{(n)}$. For any $i = 1, \dots, m$ and $j = 1, \dots, n$, let L_i^j be the completion time of machine M_i after the job J_j is assigned. L_i^j is also called the *load* of M_i if no idle time exists. Clearly, L_i^n equals to L_i .

2 Online Over List

This section summarizes main results for online over list model on parallel machines. In this paradigm, the makespan, which is the most common objective function with no doubt, will be considered throughout this section except the last subsection.

2.1 Non-preemptive Scheduling

2.1.1 Identical Machines

$Pm||C_{max}$ is the most studied and historical problem for online scheduling. In his epoch making paper, Graham [86] designed the first approximation algorithm in combinatorial optimization called *List Scheduling* (*LS* for short). *LS* uses greedy idea and has a very simple structure. It always assigns the current job to the machine where it can be started to process the earliest, that is, the machine with the smallest current load, ties are broken arbitrary. Such principle of assigning jobs is sometimes called *LS rule* in the literature. Graham proved the following result.

Theorem 1 ([86]) *The competitive ratio of LS for $Pm||C_{max}$ is $2 - \frac{1}{m}$.*

Proof (Sketch) Without loss of generalization, the last job J_n determines the makespan. Let s_n be the start time of J_n . By *LS* rule, no machine will idle during $[0, s_n]$. Hence, $s_n \leq \frac{P-p_n}{m}$. Clearly,

$$C^* \geq \frac{P}{m} \tag{1}$$

and

$$C^* \geq \max_{1 \leq j \leq n} p_j. \quad (2)$$

Hence,

$$C^{LS} = s_n + p_n \leq \frac{P - p_n}{m} + p_n = \frac{P}{m} + \frac{m-1}{m} p_n \leq C^* + \frac{m-1}{m} C^* = \left(2 - \frac{1}{m}\right) C^*. \quad \square$$

LS is a typical online algorithm. When assigning a new job, it does not use any information of jobs that have not arrived yet. Not only the rule of *LS* but also the ideas behind the proof are frequently used in design and analysis of other online algorithms. Twenty years have passed since the appearance of *LS*, and online problems are drawing more and more attentions. Feige et al. [74] proved that *LS* is the optimal algorithm for $Pm||C_{max}$ when $m = 2, 3$. It should be emphasized that no other algorithm has been proved to be optimal for any $m \geq 4$ up till now.

Though *LS* is irreplaceable in online scheduling, it is gradually recognized that *LS* cannot be the optimal algorithm for $Pm||C_{max}$ when $m \geq 4$. Hence, algorithms and lower bounds have been refined bit by bit, and brief results and history are summarized in Table 1. In these new algorithms, jobs are not always assigned to the least loaded machine. Sometimes, the current job may be assigned to the second smallest or the machine whose load lies approximately in middle of all machines. Almost all these algorithms contain some parameters which have been carefully selected, and the analysis of the algorithm is much more involved. The instances used to prove the lower bound are likewise very sophisticated. The current best lower bound is obtained even by exhaustive search using computers.

It seems that improvement of the algorithm could be carried on since there is still a gap 0.066 between the current best upper and lower bounds. However, there is a turning point when Albers [2] proposed her important results. When proving the competitive ratio of an online algorithm, it is in fact that some *lower bounds* instead of the exact value of the optimum itself are used. For example, lower bounds

Table 1 Improvement of lower and upper bounds of $Pm||C_{max}$

Reference	Competitive ratio	Reference	Lower bound
Graham [86]	$2 - \frac{1}{m}$	Falgle et al. [74]	$\frac{1+\sqrt{2}}{2}(m \geq 4)$
Galambos and Woeginger [82]	$2 - \frac{1}{m} - \epsilon_m^a$	Bartal et al. [17]	$1.837(m \geq 3454)$
Bartal et al. [18]	$2 - \frac{1}{70} \approx 1.986^b$	Albers [1]	$1.852(m \geq 80)$
Karger et al. [110]	1.945	Gormley et al. [85]	1.85358^d
Albers [1]	1.923		
Fleischer and Wahl [77]	1.9201 ^c		

^a $\epsilon_m \rightarrow 0(m \rightarrow \infty)$, the first algorithm which beats *LS* for $m \geq 4$

^bThe first algorithm with competitive ratio strictly less than 2 for any m

^cThe best current known algorithm

^dThe best current known lower bound

(1) and (2) are used in the proof of [Theorem 1](#). Another useful lower bound on the optimum is

$$C^* \geq 2p_{(m+1)}. \quad (3)$$

Theorem 2 ([2]) *If only using (1), (2) and (3) as lower bounds on the optimum, it is impossible to prove an online algorithm A has a competitive ratio less than 1.919.*

Note that the competitive ratio of the algorithm could be smaller than 1.919, but it cannot be proved using existing techniques. The bound 1.919 in [Theorem 2](#) is very close to the current best upper bound 1.9201. Hence, it seems more urgent to find new lower bounds on the optimum than to designing more delicate algorithms. Though [Theorem 2](#) does not give new lower bounds, it does point out the direction of future research. Such ideas can be adopted to other hard online scheduling problems as well.

When the number of machines is small, better lower bounds and algorithms can be obtained. For example, Chen et al. [31] presented an algorithm with competitive ratio at most

$$\max \left\{ \frac{4m^2 - 3m}{2m^2 - 2}, \frac{2(m-1)^2 + \sqrt{1 + 2m(m-1)} - 1}{(m-1)^2 + \sqrt{1 + 2m(m-1)} - 1} \right\}$$

for $Pm||C_{max}$, $m \geq 4$. Rudin III and Chandrasekaran [147] proved the lower bound of $P4||C_{max}$ is at least $\sqrt{3}$, which is in line with the conjecture that the optimal bound of $P4||C_{max}$ is $\sqrt{3}$. Numerical bounds are summarized in [Table 2](#).

2.1.2 Uniform and Unrelated Machines

The competitive analysis for $Qm||C_{max}$ is even more difficult than that for $Pm||C_{max}$. Both the competitive ratios of online algorithms and lower bounds of the problems are functions of machine speeds s_i , $i = 1, \dots, m$. This might be the reason why optimal or tight bounds can only be obtained for small values of m . For larger m or arbitrary number of machines, it is of more significance and practical that either to consider some special combinations of s_i or to obtain *overall* upper and lower bounds, that is, the maximal value among all possible choice of s_i , $i = 1, \dots, m$ for both.

Since the machines have different speeds, the machine where a job can be started to process the earliest may not be the machine where it can be completed the earliest. Therefore, it is necessary to distinguish two variants of LS , namely, LSc and LSs . LSc and LSs assign the arriving job to the machine which the job can be completed or started the earliest, respectively. It is evident to see that for $Qm||C_{max}$, LSc is better than LSs .

The competitive ratio of LSc for $Qm||C_{max}$ is at most

$$\min_{1 \leq k \leq m} \left\{ \frac{\sum_{i=1}^m s_i + (k-1)s_1}{\sum_{i=1}^k s_i} \right\}$$

Table 2 Current best upper and lower bounds for online over list model on identical machines

m	Non-preemptive			Preemptive		
	Deterministic	Upper bounds	Lower bounds	Randomized	Upper bounds	Optimal bounds
2	$\frac{3}{2} = 1.5000$ [74]	$\frac{3}{2} = 1.5000$ [86]	$\frac{4}{3} \approx 1.3333$ [18]	$\frac{4}{3} \approx 1.3333$ [18]	$\frac{4}{3} \approx 1.3333$ [18]	$\frac{4}{3} \approx 1.3333$ [33]
3	$\frac{5}{3} \approx 1.6667$ [74]	$\frac{5}{3} = 1.5000$ [86]	$> \frac{27}{19}$ [174]	1.53727 [153]	$\frac{27}{19} \approx 1.42105$ [33]	
4	$\sqrt{3} \approx 1.732$ [147]	$\frac{26}{15} \approx 1.7333$ [31]	$\frac{256}{175} \approx 1.46286$ [32, 155]	1.65669 [153]	$\frac{256}{175} \approx 1.46286$ [33]	
5	1.74625 [31]	$\frac{85}{48} \approx 1.77083$ [31]	$\frac{3.125}{2.101} \approx 1.48739$ [32, 155]	1.73376 [151]	$\frac{3.125}{2.101} \approx 1.48739$ [33]	
6	1.77301 [31]	$\frac{9}{5} = 1.8000$ [31]	$\frac{46.656}{31.031} \approx 1.50353$ [32, 155]	1.78295 [151]	$\frac{46.656}{31.031} \approx 1.50353$ [33]	
7	1.79103 [31]	$\frac{175}{96} \approx 1.82292$ [31]	$\frac{823.543}{543.607} \approx 1.51496$ [32, 155]	1.81681 [151]	$\frac{823.543}{543.607} \approx 1.51496$ [33]	
∞	1.85358 [85]	1.9201 [77]	$\frac{e-1}{e} \approx 1.582$ [32, 155]	1.916 [2]	$\frac{e-1}{e} \approx 1.582$ [33]	

[90]. The overall competitive ratio is at most

$$\begin{cases} \frac{1+\sqrt{5}}{2}, & m = 2, \\ \frac{2+\sqrt{2m-2}}{2}, & m \geq 3, \end{cases}$$

and the bound is tight in the overall sense when $m \leq 6$ [44]. For large m , the bound $O(\log m)$ obtained by using an alternative method different from those in Theorem 1 is better [10]. Specifically, for $m = 2$, the bound is

$$\begin{cases} \frac{2s+1}{s+1}, & 1 \leq s \leq \frac{1+\sqrt{5}}{2}, \\ \frac{s+1}{s}, & s > \frac{1+\sqrt{5}}{2}, \end{cases}$$

and LSc is the optimal algorithm [72]. For $m = 3$, the bound is

$$\begin{cases} \frac{s_1 + s_2 + s_3}{s_1}, & s_1^2 \geq s_2^2 + s_1s_2 + s_2s_3, \\ \frac{2s_1 + s_2 + s_3}{s_1}, & s_1^2 < s_2^2 + s_1s_2 + s_2s_3 \text{ and } s_1^2 + s_1s_2 \geq s_3^2 + s_1s_3 + 2s_2s_3, \\ \frac{3s_1 + s_2 + s_3}{s_1 + s_2 + s_3}, & s_1^2 + s_1s_2 < s_3^2 + s_1s_3 + 2s_2s_3, \end{cases}$$

and it is tight for any combinations of machine speeds [90]. However, it has been shown that LSc is optimal only for the case when $s_1 = s_2 = 1 \geq (\sqrt{2} - 1)s_3$ and $s_1^2 \geq s_2^2 + s_1s_2 + s_2$ [90, 136], and there does exist another online algorithm which beats LSc when $\frac{1+\sqrt{97}}{8} < \frac{s_1}{s_2} = \frac{s_1}{s_3} < 2$ [90]. It implies that the optimality of LS for $P3||C_{max}$ cannot be generalized to the problem with different machine speeds.

Since the competitive ratio of LSc tends to infinite when m becomes very large [44], it is natural to raise the question that whether an algorithm with finite competitive ratio exists. The first such algorithm with competitive ratio 8 was obtained by using a so-called *doubling strategy* [10]. Their results were later improved by Berman [19], where a new algorithm with overall competitive ratio $3 + \sqrt{8} \approx 5.828$ and a overall lower bound 2.4380 are given. The lower bound was recently improved to θ by Ebenlendr and Sgall [60], where $\theta \approx 2.564$ is the solution of the equation $\int_0^1 \frac{\ln \theta}{\ln(1-\theta-x)} dx = -1$.

For the most studied special case of $s_1 > s_2 = s_3 = \dots = s_m = 1$, the overall competitive ratio of LSc for $Qm||C_{max}$ is at most $\frac{3m-1}{m+1}$, where the maximum value achieves at $s_1 = 2$ [44]. The bound matches the overall lower bound 2 when $m = 3$ [118]. For $m \geq 4$, Li and Shi [118] designed a new algorithm which has a smaller competitive ratio around $s_1 = 2$. Thus, the overall competitive ratio can be decrease to 2.8795. Cheng et al. [42] further raised an algorithm which has a competitive ratio of 2.45 if the speed of the fastest machine is restricted to $1 \leq s_1 \leq 2$.

For $Rm||C_{max}$, Aspnes et al. [10] proved that the competitive ratio of LSc is at most m , and thus, LSc is optimal when $m = 2$. They also designed a new algorithm

with competitive ratio $O(\log m)$. Since the lower bound of the problem is at least $\lceil \log_2(m+1) \rceil$ [13], their algorithm is optimal up to a constant factor.

2.2 Preemptive Scheduling

Results of preemptive online scheduling are more complete than those of non-preemptive problems. The main reason may be that the optimal offline makespan can be obtained in polynomial time. The following lemma plays an important role in designing online algorithms for $Pm|pmtn|C_{max}$ and $Qm|pmtn|C_{max}$.

Lemma 1 ([84, 101]) *For $Qm|pmtn|C_{max}$,*

$$C^* = \min \left\{ \frac{\sum_{j=1}^n p_j}{\sum_{i=1}^m s_i}, \max_{1 \leq l \leq m-1} \frac{\sum_{i=1}^l p(i)}{\sum_{i=1}^l s_i} \right\}.$$

Chen et al. [33] designed an optimal algorithm *preemptive* for $Pm|pmtn|C_{max}$. Let $\Gamma_m = \frac{m^m}{m^m - (m-1)^m}$. Note that $\lim_{m \rightarrow \infty} \Gamma_m = \frac{e-1}{e} \approx 1.582$. Recall that for any j , $C^*(\mathcal{J}_j)$ can be calculated by Lemma 1.

Algorithm Preemptive

Let the current job be J_j . Reindex the machines such that $L_1^{j-1} \leq L_2^{j-1} \leq \dots \leq L_m^{j-1}$. If $L_m^{j-1} + p_j \leq \Gamma_m C^*(\mathcal{J}_j)$, then assign J_j to M_m . Otherwise, let $l = \min\{i | L_i^{j-1} + p_j \geq \Gamma_m C^*(\mathcal{J}_j)\}$. Assign part of J_j of processing time $\Gamma_m C^*(\mathcal{J}_j) - L_l^{j-1}$ to M_l and the remaining part of J_j of processing time $L_l^{j-1} + p_j - \Gamma_m C^*(\mathcal{J}_j)$ to M_{l-1} .

Theorem 3 ([33]) *The competitive ratio of Preemptive for $Pm|Pmtn|C_{max}$ is Γ_m .*

Proof (Sketch) Prove by induction that for any j , the following two inequalities hold

$$\begin{aligned} L_m^j &\leq \Gamma_m C^*(\mathcal{J}_j), \\ \sum_{i=1}^k L_i^j &\leq \frac{(\frac{m}{m-1})^k - 1}{(\frac{m}{m-1})^m - 1} P_j, k = 1, \dots, m. \end{aligned}$$

Then, it can be confirmed that the schedule produced by *Preemptive* is feasible, and the competitive ratio holds. \square

Several attempts have been made to modify and generalize *Preemptive* to handle uniform machines, which resulted in an optimal algorithm for $Q2|pmtn|C_{max}$ with competitive ratio $\frac{s^2+2s+1}{s^2+s+1}$ [72, 180] and an optimal algorithm for $Qm|pmtn|C_{max}$ with nondecreasing speed ratios, that is, $\frac{s_{i-1}}{s_i} \leq \frac{s_i}{s_{i+1}}$, $2 \leq i \leq m-1$ [64]. But it seems difficult to make further generalizations using similar ideas.

Recently, Ebenlendr et al. [62] introduced a novel technique which leads to an optimal algorithm of $Qm|pmtn|C_{max}$ for any combinations of machine speeds. The competitive ratio $\Gamma(s_1, \dots, s_m)$ of the algorithm is the optimal value of the following linear programming with decision variables $q_1, \dots, q_m, O_1, \dots, O_m$ and parameters s_1, \dots, s_m .

$$\begin{aligned} & \max q_1 + q_2 + \dots + q_m \\ & \text{s.t. } q_1 + \dots + q_k \leq (s_1 + s_2 + \dots + s_m)O_k, \quad k = 1, \dots, m, \\ & \quad q_j + q_{j+1} + \dots + q_k \leq (s_1 + s_2 + \dots + s_{k-j+1})O_k, \quad 2 \leq j \leq k \leq m, \\ & \quad 1 = s_1O_m + s_2O_{m-1} + \dots + s_mO_1, \\ & \quad q_j \leq q_{j+1}, \quad j = 2, \dots, m-1, \\ & \quad q_1 \geq 0, q_2 \geq 0. \end{aligned} \quad (4)$$

However, it can only be proved that the algorithm is optimal for any m and any combinations of machine speeds, but the concrete expression of the optimal bound $\Gamma(s_1, \dots, s_m)$ is not known yet, even the overall bounds are hard to get. The currently known best lower and upper bound for arbitrary m is 2.054 (achieves at $m = 100$ and is obtained by numerical calculation) and $e \approx 2.718$, respectively [62]. When the number of machines is small or some assumptions are made on the values of s_i , (4) can be solved theoretically [62]. For example, the optimal bounds for $Q3|pmtn|C_{max}$ are

$$\Gamma(s_1, s_2, s_3) = \begin{cases} \left(\frac{s_1}{S} + (1 - \frac{s_1}{S})\frac{s_2}{S} + (1 - \frac{s_1}{S})^2\frac{s_3}{S}\right)^{-1}, & \text{if } \frac{s_1}{s_2} \leq \frac{s_2}{s_3} + 1; \\ \frac{s_2^2}{s_1^2 + s_2^2 + s_3^2 + s_1s_2 + s_1s_3 + s_2s_3}, & \text{if } \frac{s_1}{s_2} \geq \frac{s_2}{s_3} + 1. \end{cases}$$

Another question requiring answer is that whether the optimal bounds will increase if idle time is prohibited. Note that algorithms in [33, 64, 72, 180] all do not introduce idle time, but no algorithm which allows idle time can have a smaller competitive ratio under certain machine environment mentioned there. However, the algorithm given in [62] does use idle time. It is not known yet that whether an algorithm which does not use idle time will necessarily have a strict larger competitive ratio than $\Gamma(s_1, \dots, s_m)$ for general $Qm|pmtn|C_{max}$.

2.3 Randomized Algorithm

As far as ever known results are concerned, for case on parallel machine scheduling problem with objective to minimize makespan, any lower bound for deterministic preemptive online algorithm (allowing idle time) is also a lower bound for randomized preemptive or non-preemptive online algorithms. For example, the randomized lower bound for $Pm||C_{max}$ and $Qm||C_{max}$ is $\frac{(m-1)^m}{m^m - (m-1)^m}$ [32, 155] and $\Gamma(s_1, \dots, s_m)$ [62], respectively. But the converse is not true. Tichý [174] showed that there does not exist a randomized algorithm for $P3||C_{max}$ with competitive

ratio $\frac{27}{19}$. However, his proof does not yield a new randomized lower bound with a strict larger value.

The first randomized algorithm for parallel machine scheduling is due to [18], where an optimal randomized algorithm for $P2||C_{max}$ is given. The algorithm was restated in a simpler version in [155] as follows.

Algorithm Random

1. If possible, assign the current job J_j randomly so that afterwards the expected makespan equals $\frac{2}{3}P_j$.
2. Otherwise, assign job J_j always on the less loaded machine.

Theorem 4 ([18, 155]) *The competitive ratio of Random for $P2||C_{max}$ is $\frac{4}{3}$.*

Proof (Sketch) Prove by induction that for any j , the following two invariants hold

$$\begin{aligned} E(C^{\text{Random}}(\mathcal{J}_j)) &\geq \frac{2}{3}P_j, \\ \text{If } E(C^{\text{Random}}(\mathcal{J}_j)) > \frac{2}{3}P_j, \text{ then } \max_{1 \leq l \leq j} p_l &\geq \frac{3}{4}E(C^{\text{Random}}(\mathcal{J}_j)). \end{aligned} \quad (5)$$

Clearly, (5) is also valid for $j = n$. Then, by (1) and (2), the theorem is thus proved. \square

Design effective randomized algorithm for $Pm||C_{max}$ is rather a challenging work. When the number of machines is small, the competitive ratios of algorithms proposed by Seiden [151, 153] can be less than the best known deterministic algorithms (See Table 2). For arbitrary m , Albers [2] designed a randomized algorithm with competitive ratio 1.916, which is a combination of two deterministic algorithms with equal probability, and it is better than any known deterministic algorithm.

There is even less study on randomized algorithm for uniform machines. For $Q2||C_{max}$, Epstein et al. [72] showed that randomization does not help when $s \geq 2$. They also presented numerical randomized lower bounds for $1 \leq s \leq 2$ by solving linear programmings and a smaller analytical randomized lower bound $\frac{(s+1)^2(s+2)}{3s^2+5s+1}$ for $1 \leq s \leq \sqrt{2}$. Three randomized algorithms are proposed, and thus, an overall upper bound 1.52778 can be achieved, but the gap between the lower and upper bounds for any $s \geq 1$ is still relatively large. For $Q3||C_{max}$ with restricted machine speeds $s_1 = s_2 \geq s_3 = 1$, Musitelli and Nicoletti [136] presented a randomized algorithm with competitive ratio

$$\begin{cases} \frac{4s_2 + 2}{3s_2}, & 1 \leq s_2 < \frac{\sqrt{10}+2}{3}, \\ \frac{6s_2 + 2}{3s_2 + 2}, & s_2 \geq \frac{\sqrt{10}+2}{3}. \end{cases}$$

It is only a little smaller than the competitive ratio of LSc , and no randomized lower bound has been reported.

2.4 Other Objectives

2.4.1 Total Completion Time

The property of the objective of minimizing the total completion time is different from that of minimizing makespan. The objective value of the schedule will be affected by not only the set of jobs assigned to each machine but also the process sequence of it. As a result, there exists no algorithm with constant competitive ratio, even for the single machine case.

Theorem 5 ([76]) *For every function $f : N \rightarrow R^+$ that fulfills conditions:*

1. $f(n)$ is nondecreasing,
2. $\sum_{n=1}^{\infty} \frac{1}{nf(n)}$ converges,

there exists an online algorithm for $1||\sum C_j$ with competitive ratio at most $O(f(n))$, where n is the number of jobs. On the other hand, let $g : N \rightarrow R^+$ be a function that fulfills condition:

1. $g(n)$ is nondecreasing,
2. $\sum_{n=1}^{\infty} \frac{1}{ng(n)}$ diverges,
3. $g(n) = O(\log^2 n)$,
4. $g\left(\frac{n}{\log^2 n}\right) = \Omega(g(n))$,

then there does not exist an online algorithm for $1||\sum C_j$ with competitive ratio $o(f(n))$.

By selecting specific functions which satisfy the respective conditions, it can be found that the gap between the lower and upper bounds is rather small.

Corollary 1 ([76]) *For every $\epsilon > 0$, there exists an online algorithm with competitive ratio at most $(\log n)^{1+\epsilon}$, but no online algorithm with competitive ratio $\log n$ can exist.*

2.4.2 Machine Covering

The online version of non-preemptive machine covering problem is relatively easy. It should be mentioned that for machine covering problems, LSS is better than LSc in most cases. Woeginger [181] and Epstein [66] proved that LSS is the optimal algorithm for $Pm||C_{min}$ and $Q2||C_{min}$, respectively. In fact, their analysis can be generalized to $Qm||C_{min}$. LSS remains the optimal algorithm with competitive ratio $\sum_{i=1}^m s_i$. Azar and Epstein [12] designed a randomized algorithm for $Pm||C_{min}$ with an overall competitive ratio $O(\sqrt{m} \log m)$ and proved an overall randomized lower bound $\frac{\sqrt{m}}{4}$.

Surprisingly, machine covering problem becomes much more difficult if preemption is allowed, even though the optimal offline objective value

$$\min_{0 \leq j \leq m-1} \frac{\sum_{l=j+1}^n p(l)}{\sum_{l=j+1}^m s_l}$$

for $Qm|pmtn|C_{min}$ can be calculated similarly as $Qm|pmtn|C_{max}$ [108]. Jiang et al. [108] proved that $\sum_{i=1}^m \frac{1}{i}$ is a lower bound of $Pm|pmtn|C_{min}$ and m is an overall lower bound of $Qm|pmtn|C_{min}$, but no algorithm with matched competitive ratio has been given. For $Q2|pmtn|C_{min}$, they designed an optimal algorithm with competitive ratio $\frac{2s+1}{s+1}$. However, their algorithm may introduce idle time, even when it assigns the first job. If idle time is not allowed, then no deterministic algorithm can have a smaller competitive ratio than

$$\begin{cases} \frac{2s+1}{s+1}, & 1 \leq s < \frac{1+\sqrt{5}}{2}, \\ s, & s \geq \frac{1+\sqrt{5}}{2}. \end{cases}$$

A corresponding algorithm prohibiting idle time with matched competitive ratio was also given in [108]. Therefore, whether idle time is allowed or not does affect the optimal bounds for machine covering problems. It should be noted that different from algorithms allowing idle time, for non-idle time situation, there is no evidence yet that the lower bound also holds for randomized algorithm. Such phenomenon also appears in similar situations in the following.

3 Semi-online Scheduling

3.1 Motivation and Taxonomy

There are two basic characteristics for the online over list model. The first is that jobs arrive one by one and it is required to assign each job without any knowledge of the jobs that arrive later. The second is that the assignment of the job should be determined immediately upon its arrival and cannot be changed later. The reason of the first characteristic is “lack of information,” while the essence of the second is “restriction of scheduling.” There are often voices of criticism on the model for the sake of both theoretical study and practical application. In practice, most problems are not pure offline or online but somehow in between. Theoretically, the competitive ratio of an online algorithm tends to be over large in contract with the actual performance, since offline algorithms are more powerful than online algorithms. Moreover, the lower bound of the problem can also be ineffective, namely, too large, as the adversary can arbitrarily determine the instances without any restriction.

The reasons aforementioned bring semi-online scheduling about an active research area. Semi-online can be looked upon as relaxation of online or an intermediate state of offline and online. The primary significance of study on semi-online settings is as follows. First and foremost, without reasonable understanding of semi-online problems, there will be no more than online algorithm to implement faced with semi-online cases, which will give rise to inevitable loss definitely, just as *LS* algorithm performs not satisfactorily on offline problems. Furthermore, deep

study on semi-online problems will lead to initiative search for crucial factors in algorithm design and improving, which will certainly save the cost and energy on many pointless factors. In theory, research on semi-online scheduling is of help to clarify influence of various restrictions of both information accessed to and algorithm design, and to reveal some deep and dominant properties. For example, it is interesting to consider the question that whether it is possible or under what kind of circumstance that an *online approximation scheme*, that is, a class of semi-online algorithms with competitive ratio arbitrarily close to 1 can exist. In contrast to *PTAS* of offline problems, whose time complexity increases as the worst-case ratio of the algorithm decreases, more partial information or more freedom of scheduling is required in online cases when pursuing a better competitive ratio. In fact, one such approximation scheme has already been obtained [160]. Last but not least, it will be meaningful for algorithm design and analysis on online problems when researching on semi-online models, as will see later.

Though semi-online model can be viewed as an intermediate state between online and offline, its properties and research approach are almost the same as the online model. Competitive analysis is still the primary technique adopted, and lower bound of the problem, together with competitive ratio of the algorithm, can be defined accordingly.

The value of a semi-online model, denoted by Π_s , is evaluated by comparing its upper (or lower) bound with that of a corresponding online model reacting to the same machine environment and objective function, which is represented by Π_o . The semi-online model is called *valuable* if there exists an online algorithm for Π_s whose competitive ratio is smaller than the optimal bound (or lower bound) of the Π_o . On the other hand, if the lower bound of Π_s is no smaller than the competitive ratio of an algorithm for Π_o , it is called *valueless*. Of course, a semionline model is valuable or may not be determined by certain machine environment or objective functions. In this section, mainly identical and uniform machines, as well as minimizing makespan and maximizing minimum machine load, are considered. It is also possible to compare variants of semi-online models qualitatively, according to the extent to which the optimal bound of pure online problem can be improved. However, the outcomes may be quite sensitive to the machine environment or objective function.

Various semi-online paradigms have been studied in the literature, which can be divided into several categories. The taxonomy and their notations, which will be included in the middle field of the three-field notation, are summarized in the [Table 3](#). Their specific implications will be introduced in the rest of the subsection.

3.1.1 Basic Semi-online Models of Type I

Basic semi-online models of Type I modify the first assumption of the online over list model, that is, some partial information about the future jobs are known in advance. The following four kinds of partial information are the most heated.

decr [154]: The jobs are arriving in nonincreasing order of their processing times, that is, $p_1 \geq p_2 \geq \dots \geq p_n$.

Table 3 Semi-online models

Basic	Type I	<i>Valuable:</i> sum, opt, max, decr <i>Valueless:</i> num, LB, UB, min, incr, lookahead
	Type II	buffer, parallel, reassignment
Combined	Type I and Type I	UB & LB, UB & sum, LB & max, max & sum, max & opt, decr & sum, decr & opt, end of sequence
	Type I and Type II	sum & buffer, sum & parallel, sum & reassignment
Disturbed	inexact information	inexact sum, inexact opt, inexact max

sum [111]: The total processing time of all jobs $P = \sum_{j=1}^n p_j$ is known before the first job is arrived.

opt [14]: The optimal offline makespan C^* is known before the first job is arrived.

max [111]: The maximal processing time of all jobs $p_{\max} = \max_{j=1,\dots,n} p_j$ is known before the first job is arrived.

The motivation of *decr*, *sum*, and *max* is obvious, whereas it may seem to be strange at first glance that the optimal offline makespan can be known in advance. In fact, it can be interpreted from the realistic scenario of remote file transfer [14]. Besides, online problems with C^* is known in advance have already been studied before semi-online began to draw attention, since the following lemma is useful in competitive analysis of difficult online problems [10].

Lemma 2 *For some scheduling problem of online over list model with objective to minimize makespan, if there exists an algorithm for the opt variant with competitive ratio r, then there exists an algorithm for the pure online model with competitive ratio at most 4r.*

Some variants reacting to other partial knowledge of input are also reasonable, which is quite common in practice. For example,

num: The number of all jobs n is known before the first job is arrived.

incr: The jobs are arriving in nondecreasing order of their processing times, that is, $p_1 \leq p_2 \leq \dots \leq p_n$.

UB(LB): The upper (lower) bound on the processing time of all jobs $p_{UB} \geq \max_{j=1,\dots,n} p_j$ ($p_{LB} \leq \min_{j=1,\dots,n} p_j$) is known before the first job is arrived.

min: The minimal processing time of all jobs $p_{\min} = \min_{j=1,\dots,n} p_j$ is known before the first job is arrived.

However, such paradigms, which seem worthless under certain machine environment and objective functions discussed so far, have not drawn much attention.

Another kind of partial information which is common in practice is called *lookahead* [111]. When assigning the current job, the information of the next k jobs is known, where k is a constant number. Lookahead tends to be of benefit for scheduling according to practical experience, whereas it turns out to be still worthless as far as competitive ratio is concerned.

3.1.2 Basic Semi-online Models of Type II

There are other variants which lie between online and offline. However, they have nothing to do with the partial information of the input but more freedom acquired in scheduling. These variants belong to *Type II* semi-online models. It is a much more potential research area compared with Type I semi-online models, which is fruitful of results.

In the *buffer* [111] paradigm, there is a buffer of size K which can store at most K jobs. The incoming job can either be scheduled immediately or be stored in the buffer, which enables it to be scheduled later. The root cause of the improved performance attributed to buffer lies in the fact that it can reorder the job sequence. Pure online and offline models are equivalent to problems with buffer size 0 and ∞ , respectively. In another related class of models *reassignment* [160, 167], it is allowed to reassign some of the jobs during or after the process. Recall that in the pure online model, no job can be reassigned.

A technique frequently used in designing algorithm for the offline problems is to run several procedures in parallel and then select the best result as output. However, it does not fit pure online setting. Semi-online variant which allows to use such technique is called *parallel*. A real-world system which resembles such situation was claimed to exist in [111]. Kellerer et al. [111] proved that the optimal bound of $P2|parallel(2)|C_{max}$ is $\frac{4}{3}$; here, *parallel(2)* means that two procedures are run in parallel.

3.1.3 Combined Semi-online Models

Since some semi-online variants are indeed of value, it is natural to raise the question whether a more efficient algorithm can be designed for a problem possessing characteristics of several semionline variants in the meantime. A combination is *useful*, if there exists an online algorithm for the combined semi-online problem whose competitive ratio is smaller than the optimal bound (or lower bound) of any one of the basic semi-online model. Otherwise, the combination is *useless*. The combination could be of two basic semi-online variants of Type I, as well as that of Type I and II, respectively. In spite of the fact that combination consisting of more than three variants can certainly be concerned with, such cases are always quite complicated and seldom studied.

3.1.4 Disturbed Semi-online Models

The *disturbed* semi-online models were first studied in [165]. In some cases, it is allowed to use some additional information or slightly more resources than the offline algorithm, whereas the information or resources might be unreliable. For example, it is known in advance that the total processing time lies in the interval $[P_0, \alpha P_0]$, but the exact value of P is still unknown. It can be believed that the value of inexact partial information lies largely on the value of α , which is called

disturbance parameter. For problem with inexact partial information, special stress is laid on determining the value of α which insures the information to be useful and further designing algorithm making use of the beneficial information acquired.

3.2 Results on Basic Semi-online Models of Type I

3.2.1 Identical Machines

It has been widely known for a long time that in combinatorial optimization, the performance of an algorithm can be improved after appropriate preprocessing for the input. Graham [87] proved that the worst-case ratio of algorithm *Longest Processing Time first* (*LPT* for short), where *LS* is implemented after a preprocessing step, namely, reordering the jobs in a nonincreasing sequence of their processing time, is $\frac{4}{3} - \frac{1}{3m}$. However, it is not allowed to reorder the sequence in the pure online setting. But if the jobs are arrived in nonincreasing order of their processing times, the performance of *LS* will be just the same as that of *LPT*. Above situation was reformulated as a semi-online variant by Seiden et al. [154]. They proved that *LS* is the optimal algorithm for $P2|decr|C_{max}$. However, *LS* is no longer optimal when $m \geq 3$, since there exists an algorithm with competitive ratio $\frac{5}{4}$ for $m \geq 4$ and an optimal algorithm with competitive ratio $\frac{1+\sqrt{37}}{6}$ for $m = 3$ [43].

If the total processing time of all jobs P is known in advance, a lower bound $\frac{P}{m}$ on the optimal makespan is readily available, which will be of great help in designing algorithm with a better competitive ratio. Take the optimal algorithm reacting to two machines [111] for example. Jobs are successively assigned to M_1 until there exists a crucial job, such that the load of M_1 would exceed a threshold $\frac{4}{3} \cdot \frac{P}{2}$ if the crucial job is still assigned to M_1 ; here, $\frac{4}{3}$ is the expected competitive ratio of the algorithm. Then, the crucial job is assigned to M_2 , and the remaining jobs will no longer have any negative effect on the competitive ratio. The core idea of the algorithm is to keep one machine lightly loaded so as to serve the long jobs which might present later. The significance of *sum* lies in the fact that it is impossible to determine the threshold without access to P .

For $Pm|sum|C_{max}$, Angelelli et al. [6] proposed an algorithm with competitive ratio $\frac{\sqrt{6}+1}{2} \approx 1.725$ and proved that the lower bound of the problem is 1.565 for sufficiently large m . The lower bound was recently improved to 1.585 by Albers and Hellwig [3]. By more careful classification according to the processing time of the jobs as well as the current load of the machines, Cheng et al. [41] designed an improved algorithm with competitive ratio $\frac{8}{5}$. They also showed that $\frac{3}{2}$ is a lower bound of the problem for any $m \geq 6$. For the case of $m = 3$, the current best lower and upper bounds are $\frac{\sqrt{129}-3}{6} \approx 1.393$ and $\frac{27}{19} \approx 1.421$, respectively [9].

The semi-online variant *opt* is closely associated with the variant *sum*. In fact, many instances used to show the tightness of the competitive ratio of some algorithms satisfy $C^* = \frac{P}{m}$. However, the instance with $C^* > \frac{P}{m}$ does exist. Therefore, it is not easy to answer whether corresponding two problems regarding

sum and *opt* variants have the same optimal bound. Nevertheless, Dósa et al. [54] proved the following result.

Lemma 3 ([54])

- (i) If ρ is a lower bound of $Qm|opt|C_{max}$, then the lower bound of $Qm|sum|C_{max}$ is at least ρ .
- (ii) If there is an algorithm for $Qm|sum|C_{max}$ with competitive ratio r , then there also exists an algorithm for $Qm|sum|C_{opt}$ with competitive ratio at most r .

By Lemma 3, the algorithm for $Pm|sum|C_{max}$ given in [41] can be modified to solve $Pm|opt|C_{max}$ with at most the same competitive ratio $\frac{8}{5}$, which improves the former algorithm for $Pm|opt|C_{max}$ with a competitive ratio of $\frac{13}{8}$ [14]. When the number of machines is small, another algorithm also given in [14] has a smaller competitive ratio of $\frac{5m-1}{3m+1}$. On the other hand, the lower bound of $Pm|sum|C_{max}$ is no longer valid for $Pm|opt|C_{max}$. Though $\frac{4}{3}$ is clearly a lower bound of $Pm|opt|C_{max}$ for any $m \geq 2$ [14], no larger lower bound has been reported for $m \geq 3$.

There are fewer results on the *max* variant. He and Zhang [97] presented an optimal algorithm for $P2|max|C_{max}$ with competitive ratio $\frac{4}{3}$. Note that for the *max* variant, not only the processing times of all jobs are no more than p_{max} , but also at least one job with processing time p_{max} will arrive sooner or later. Hence, the first job with processing time p_{max} , which is denoted as J_{max} , can be shifted to the beginning of the job sequence by assumption. Thus, the optimal algorithm given in [97] can be simplified by using above idea.

Algorithm PLS

1. Let J_{max} be scheduled on M_1 from time 0 to p_{max} .
2. Assign the remaining jobs by *LS* rule.

Theorem 6 *The competitive ratio of PLS is $\frac{4}{3}$ for $P2|max|C_{max}$.*

Proof If $\max\{L_1, L_2\} \leq \frac{2}{3}P$, then $\frac{C^{PLS}}{C^*} \leq \frac{\max\{L_1, L_2\}}{\frac{P}{2}} \leq \frac{4}{3}$ by (1). Otherwise,

$$\begin{aligned} \max\{L_1, L_2\} + (L_1 + L_2) + |L_1 - L_2| &= 3 \max\{L_1, L_2\} > 2P = (L_1 + L_2) \\ &\quad + \max\{L_1, L_2\} + \min\{L_1, L_2\}. \end{aligned} \quad (6)$$

Since the remaining jobs are assigned by *LS* rule, $|L_1 - L_2| \leq p_{max}$. Combining it with (6), $\min\{L_1, L_2\} < |L_1 - L_2| \leq p_{max}$. Recall that J_{max} is assigned to M_1 , $L_1 \geq p_{max}$. Hence, $L_2 < p_{max}$ and thus no job other than J_{max} will be assigned to M_1 , which implies that *PLS* produces an optimal schedule. \square

However, for general m , no algorithm performing better than the best known algorithm for pure online model has been obtained. It is also unknown that whether

Table 4 Results of semi-online variants on arbitrary m identical machines

Variant	Non-preemptive				Preemptive
	Makespan minimization		Machine covering		Makespan minimization
	Lower bound	Upper bound	Lower bound	Upper bound	Optimal bound
<i>decr</i>	$\frac{1+\sqrt{37}}{6}$ [154]	$\frac{5}{4}$ [43]	Open	$\frac{4m-2}{3m-1}$ [47]	$\max_{0 \leq k \leq m} \frac{2m^2+2mk}{2m^2+k^2+k}$ [154]
<i>sum</i>	1.585 [3]	$\frac{8}{5} = 1.6000$ [41]		$m-1$ [22, 166]	1 [98]
<i>opt</i>	$\frac{4}{3} \approx 1.333$ [14]	$\frac{8}{5} = 1.6000$ [41]	$\frac{43}{24} \approx 1.791$ [61]	$\frac{11}{6} \approx 1.834$ [61]	1 [58]
<i>max</i>	Open	Open		$m-1$ [22, 166]	$\max_{0 \leq k \leq m} \frac{2m^2+2mk}{2m^2+k^2+k}$ [154]

the partial information is still valuable for sufficiently large m , since the effect of the information about the single job J_{\max} tend to reduce as m increases.

As far as the problem of machine covering is concerned, the four semi-online variants differ hugely both in property and difficulty. Optimal algorithms for $Pm|sum|C_{\min}$ and $Pm|max|C_{\min}$ can be obtained, both having a competitive ratio $m-1$ for $m \geq 3$ [22, 166]. However, the research on $Pm|opt|C_{\min}$ seems to be much more difficult than that of $Pm|sum|C_{\min}$. Azar and Epstein [12] proposed an algorithm with competitive ratio $2 - \frac{1}{m}$, which is optimal for $m = 2, 3, 4$. A more involved algorithm with competitive ratio $\frac{11}{6} \approx 1.834$ was given in [61]. The current best lower bound is $\frac{7}{4}$ for $m > 4$ [12] and $\frac{43}{24} \approx 1.791$ for sufficiently large m [61]. These bounds are far smaller than those for $Pm|sum|C_{\min}$. For the problem $Pm|decr|C_{\min}$, LS has a competitive ratio of $\frac{4m-2}{3m-1}$ [47] and is optimal for $m = 2, 3$ [96].

Tables 4 and 5 summarize the current best results of different semi-online variants on identical machines for arbitrary m and small number of m , respectively. As is shown in the table, different semi-online variants are diverse in value. In addition, optimal algorithm for two machines has almost all been found, whereas for three or more machines, nearly all problems are waiting to be answered, which is quite thought-provoking.

3.2.2 Two Uniform Machines

Similarly as pure online problems, the upper and lower bounds for most semi-online problems on two uniform machines are piece-wise rational or irrational functions of s . But the number of the pieces could be considerably large, often exceeding 10. There may be large difference among optimal algorithms for the same problem with different value of s , which make the analysis of these problems extremely difficult. Most results are obtained mainly by case by case analysis. Up till now, no universal or revolutionary method has been developed to handle these problems efficiently.

Table 5 Results of semi-online variants on a small number of identical machines

Model ^a	$m = 2^b$						$m = 3$						$m = 2$ Makespan Non-pmtm OB	
	Makespan			Makespan			MC			Non-pmtm				
	Non-pmtm	pmtm	OB	Non-pmtm	LB	UB	OB	OB	OB	OB	OB	OB		
<i>decr</i>	$\frac{7}{6} [86, 154]$	$\frac{9}{5} [154]$		$\frac{1+\sqrt{57}}{6} [154]$		$\frac{1+\sqrt{57}}{6} [43]$	$\frac{6}{5} [154]$	$\frac{5}{3} [12]$	<i>parallel(2)</i>	$\frac{4}{3} [111]$				
<i>sum</i>	$\frac{4}{3} [111]$	$1 [98]$		$\frac{\sqrt{129}-3}{6} [9]$		$\frac{27}{19} [9]$	$1 [98]$	$2 [166]$	<i>reassignFE</i>	$\sqrt{2} [167]$				
<i>opt</i>	$\frac{4}{3} [14]$	$1 [65]$		$\frac{4}{3} [14]$		$\frac{7}{5} [14]$	$1 [58]$	$\frac{5}{4} [47, 96]$	<i>reassignSQ</i>	$\frac{4}{3} [55]$				
<i>max</i>	$\frac{4}{3} [97]$	$\frac{6}{5} [154]$	Open		Open		$\frac{6}{5} [154]$	$2 [166]$	<i>sum & buffer</i>	$\frac{6}{5} [53]$				
<i>buffer</i>	$\frac{4}{3} [111]$	$\frac{4}{3} [52]$	c				$\frac{9}{7} [52]$	d	<i>sum & reassignFE</i>	$\frac{5}{4} [132]$				
<i>reassignFA</i>	$\frac{4}{3} [167]$	Open		$\frac{15}{11} [4]$		$\frac{15}{11} [4]$	Open	Open	<i>sum & parallel(2)</i>	$\frac{5}{4} [53]$				
<i>sum & max</i>	$\frac{6}{5} [164]$	$1 [98]$		$\frac{4}{3} [182]$		$\frac{4}{3} [182]$	$1 [98]$	$\frac{3}{2} [166]$	<i>decr & sum</i>	$\frac{10}{9} [164]$				
<i>opt & max</i>	$\frac{6}{5} [164]$	$1 [65]$		$\frac{4}{3} [182]$		$\frac{4}{3} [182]$	$1 [98]$	$\frac{3}{2} [166]$	<i>decr & opt</i>	$\frac{10}{9} [65]$				
<i>eos</i>	$\sqrt{2} [190]$	Open		$\frac{3}{2} [190]$		$\frac{3}{2} [190]$	Open	Open						

^aMC machine covering, OB optimal bound, LB lower bound, UB upper bound^bFor most semi-online variants, the makespan minimization problem and the machine covering problem on two identical machines are equivalent. The competitive ratios r_1 and r_2 of the same algorithm for corresponding two problems satisfy $r_1 + \frac{1}{r_2} = 2$. Therefore, results about machine covering problems on two identical machines are omitted. However, there does exist semi-online model that above two problems are not equivalent, such as *UB & LB*^cThe lower bound is $\frac{15}{11}$ for any buffer size K [63], and there exists an algorithm with matched competitive ratio which uses a buffer of size 6 [112]^dThe lower bound is $\frac{11}{6}$ for any buffer size K , and the upper bound is $\frac{5}{2}$ for $K \geq 2$ [73]

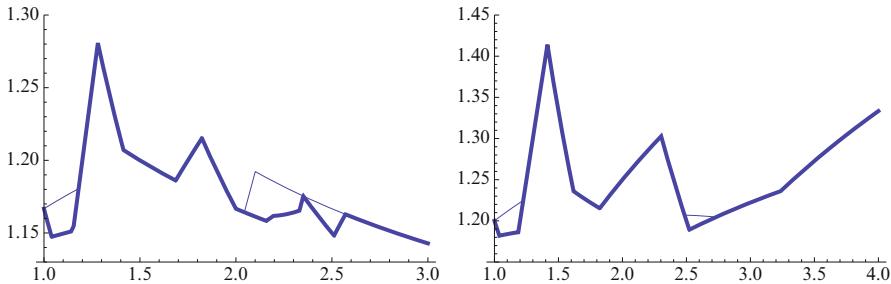


Fig. 2 The optimal bounds (thick) and competitive ratios of LS (dash) for $Q2|decr|C_{max}$ (left) and $Q2|decr|C_{min}$ (right)

Among the four basic semi-online variants of Type I, *decr* is the only one which optimal bounds for all values of $s \geq 1$ are obtained for both minimization and maximization problems. Based on the worst-case ratio of *LPT* for the offline version [133], Epstein and Favrholdt proved that LSc is optimal for the majority value of s . For the remaining two intervals which is close to 1 and nearby 2.5 where LSc is not optimal, they designed three new algorithms and proved their optimality. Symmetrically, for $Q2|decr|C_{min}$, LSs is also optimal except for two similar intervals of s , and new optimal algorithms have been proposed for these intervals [39]. The figures depicting two optimal bounds (as functions of s) bear some similarities (Fig. 2).

The other two problems which optimal algorithms for any $s \geq 1$ have been obtained are $Q2|sum|C_{min}$ and $Q2|opt|C_{min}$. The optimal bound of $Q2|sum|C_{min}$ [163] is

$$\begin{cases} \frac{s+2}{s+1}, & s \in [1, \sqrt{2}], \\ s, & s \in [\sqrt{2}, \frac{1+\sqrt{5}}{2}], \\ \frac{s^2+s+1+\sqrt{5s^4+6s^3+3s^2+2s+1}}{2s(s+1)}, & s \in (\frac{1+\sqrt{5}}{2}, \infty), \end{cases}$$

while the optimal bound of $Q2|opt|C_{min}$ [66] is

$$\begin{cases} \frac{s+2}{s+1}, & s \in [1, \sqrt{2}], \\ s, & s \in [\sqrt{2}, \frac{1+\sqrt{5}}{2}], \\ \frac{2s+1}{s+1}, & s \in (\frac{1+\sqrt{5}}{2}, \infty). \end{cases}$$

Note that the optimal bound of $Q2|sum|C_{min}$ is strict smaller than that of $Q2|opt|C_{min}$ for $s > \frac{1+\sqrt{5}}{2}$. The overall optimal bound of $Q2|sum|C_{min}$ is also strict less than that of $Q2|opt|C_{min}$ (See Fig. 3). These conclusions turn out to be opposite to both the results of Lemma 3 concerning problems with objective to minimize makespan and that of machine covering problems on m identical machines.

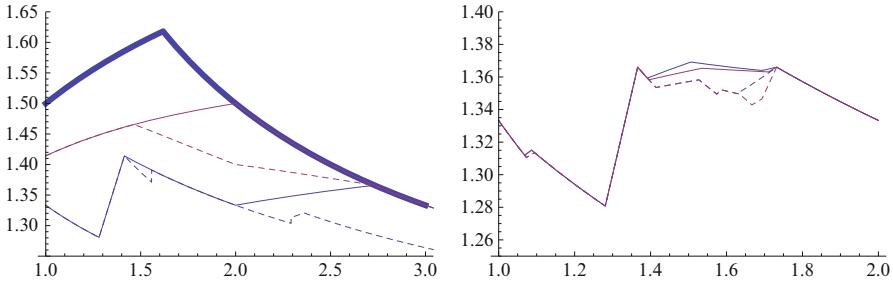


Fig. 3 Left: the optimal bound of $Q2||C_{max}$ (thick) and the upper bounds (solid) and lower bounds (dash) of $Q2|max|C_{max}$ (bottom) and $Q2|eos|C_{max}$ (top). Right: the upper bounds (solid) and lower bounds (dash) of $Q2|sum|C_{max}$ (top) and $Q2|opt|C_{max}$ (bottom)

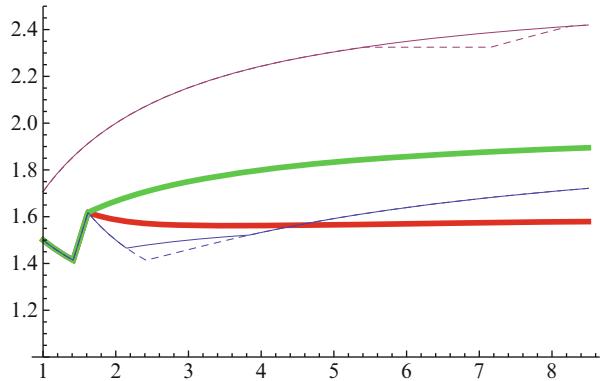


Fig. 4 The optimal bounds (thick) of $Q2|sum|C_{min}$ (bottom) and $Q2|opt|C_{min}$ (top) and the upper bounds (solid) and lower bounds (dash) of $Q2|max|C_{min}$ (bottom) and $Q2|eos|C_{min}$ (top)

For the remaining four problems, there are still gaps between upper and lower bounds for some values of s , though ceaseless efforts have been put into narrowing the gap. The gaps of some problems seem to be small, but it is usually the case that the gap might appear exactly be the place where the analysis becomes toughest. For the sake of conciseness, only best known bounds will be listed in the following. General information of these bounds is summarized in Table 6 (also see Figs. 4 and 3). It can be seen that $Q2|sum|C_{max}$ and $Q2|opt|C_{max}$ are much more complicated than the corresponding machine covering problems. For $Q2|max|C_{max}$, the effect of the information about the single job J_{max} declines when s is sufficiently large. There is a noteworthy fact that for majority makespan minimization problems, the overall bound is achieved at the point where the argument s values exactly the overall bound itself and lies in the interval $[1.2, 1.5]$. For machine covering problems, the overall bounds all achieve at $s = \infty$ except $Q2|sum|C_{max}$.

Table 6 General information and current status of upper and lower bounds on two uniform machines

Problem	Objective	Variant	Overall lower bound		Overall upper bound		Largest gap		Intervals where gap exists	
			Lower bound	s	Upper bound	s	Gap	s	Intervals	Length
C_{max}	<i>decr</i>	$\frac{1+\sqrt{17}}{4}$	$\frac{1+\sqrt{17}}{4}$	$\frac{1+\sqrt{17}}{4}$	$\frac{1+\sqrt{17}}{4}$	$\frac{1+\sqrt{17}}{4}$	0	\emptyset	0	0
		$\frac{1+\sqrt{3}}{2}$	$\frac{1+\sqrt{3}}{2}$	1.3692	1.5062	0.0176	1.5744	(1.071, 1.0946) \cup (1.3915, 1.732)	0.3641	
	<i>opt</i>	$\frac{1+\sqrt{3}}{2}$	$\frac{1+\sqrt{3}}{2}$	$\frac{1+\sqrt{3}}{2}$	$\frac{1+\sqrt{3}}{2}$	0.0219	$\frac{5}{3}$	(1.071, 1.0946) \cup (1.3956, 1.732)	0.3600	
C_{min}	<i>max</i>	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$	0.081	$1 + \sqrt{3}$	(1.414, 1.558) \cup (2, 3.56)	1.704	
		1.4656	1.4656	$\frac{3}{2}$	2	0.1000	2	(1.4657, 1.732)	0.2663	
	<i>eos</i>	2	∞	2	∞	0	\emptyset	\emptyset	0	0
C_{min}	<i>decr</i>	$\frac{1+\sqrt{5}}{2}$	$\frac{1+\sqrt{5}}{2}$	$\frac{1+\sqrt{5}}{2}$	$\frac{1+\sqrt{5}}{2}$	0	\emptyset	\emptyset	0	0
		$\frac{5}{2}$	∞	$\frac{5}{2}$	∞	0	\emptyset	\emptyset	0	0
	<i>opt</i>	2	∞	2	∞	0	\emptyset	\emptyset	0	0
C_{max}	<i>max</i>	2	∞	2	∞	0.064	$1 + \sqrt{2}$	(2.148, 3.836)	1.688	
		$\frac{5}{2}$	∞	$\frac{5}{2}$	∞	0.0624	7.1519	(5.4043, 8.2426)	2.8383	

The lower and upper bounds of $Q2|sum|C_{max}$ [54, 65, 137] are

$$\left\{ \begin{array}{ll} \frac{3s+1}{3s}, & s \in \left[1, \frac{5+\sqrt{205}}{18} \approx 1.0732 \right], \\ \frac{8s+5}{5s+5}, & s \in \left[\frac{5+\sqrt{205}}{18}, \frac{1+\sqrt{31}}{6} \approx 1.0946 \right], \\ \frac{2s+2}{2s+1}, & s \in \left[\frac{1+\sqrt{31}}{6}, \frac{1+\sqrt{17}}{4} \approx 1.280 \right] \\ s, & s \in \left[\frac{1+\sqrt{17}}{4}, \frac{1+\sqrt{3}}{2} \approx 1.36603 \right] \\ \frac{2s+1}{2s}, & s \in \left[\frac{1+\sqrt{3}}{2}, \sqrt{2} \approx 1.41421 \right] \\ \frac{3s+5}{2s+4}, & s \in \left[\sqrt{2}, \frac{\sqrt{21}}{3} \approx 1.52753 \right] \\ \frac{3s+3}{3s+1}, & s \in \left[\frac{\sqrt{21}}{3}, \frac{5+\sqrt{193}}{12} \approx 1.57437 \right] \\ \frac{4s+2}{2s+3}, & s \in \left[\frac{5+\sqrt{193}}{12}, \frac{7+\sqrt{145}}{12} \approx 1.5868 \right] \\ \frac{5s+2}{4s+1}, & s \in \left[\frac{7+\sqrt{145}}{12}, \frac{9+\sqrt{193}}{14} \approx 1.63509 \right] \\ c(s), & s \in \left[\frac{9+\sqrt{193}}{14}, \sqrt{3} \right] \\ \frac{s+2}{s+1}, & s \in [\sqrt{3}, \infty) \end{array} \right.$$

and

$$\left\{ \begin{array}{ll} \frac{3s+1}{3s}, & s \in [1, x_1 \approx 1.071], \\ \frac{7s+6}{4s+6}, & s \in \left[x_1, \frac{1+\sqrt{145}}{12} \approx 1.0868 \right], \\ \frac{2s+2}{2s+1}, & s \in \left[\frac{1+\sqrt{145}}{12}, \frac{1+\sqrt{17}}{4} \approx 1.280 \right] \\ s, & s \in \left[\frac{1+\sqrt{17}}{4}, \frac{1+\sqrt{3}}{2} \approx 1.36603 \right] \\ \frac{2s+1}{2s}, & s \in \left[\frac{1+\sqrt{3}}{2}, x_2 \approx 1.3915 \right] \\ \frac{s+\sqrt{29s^2+59s+30}}{4s+5}, & s \in [x_2, x_3 \approx 1.5062] \\ \frac{\sqrt{36s^4+9s^3-32s^2-2s+9}+6s^2+9s+3}{9s^2+7s}, & s \in [x_3, x_4 \approx 1.6932] \\ \frac{\sqrt{4s^4+8s^3+s^2+4+2s^2+3s+2}}{2s^2+6s}, & s \in [x_4, \sqrt{3}] \\ \frac{s+2}{s+1}, & s \in [\sqrt{3}, \infty), \end{array} \right.$$

respectively, where $c(s) = \frac{\sqrt{s^2x^2 + 4s(s+1-x)} + sx}{2s}$, x is a root of the equation $\frac{\sqrt{s^2x^2 + 4s(s+1-x)} + sx}{2s} = \frac{s(2s+2-x)}{(s+1)(x+2)}$, x_1, x_2, x_3, x_4 are the positive roots of

$$3s^2(9s^2 - s - 5) = (3s+1)(5s+5 - 6s^2),$$

$$\begin{aligned} \frac{2s+1}{2s} &= \frac{s + \sqrt{29s^2 + 59s + 30}}{4s+5}, \\ \frac{s + \sqrt{29s^2 + 59s + 30}}{4s+5} &= \frac{\sqrt{36s^4 + 9s^3 - 32s^2 - 2s + 9} + 6s^2 + 9s + 3}{9s^2 + 7s}, \\ &\quad \frac{\sqrt{36s^4 + 9s^3 - 32s^2 - 2s + 9} + 6s^2 + 9s + 3}{9s^2 + 7s} \\ &= \frac{\sqrt{4s^4 + 8s^3 + s^2 + 4} + 2s^2 + 3s + 2}{2s^2 + 6s}, \end{aligned}$$

respectively.

The lower and upper bounds of $Q2|opt|C_{max}$ [54, 65, 137] are

$$\left\{ \begin{array}{l} \frac{3s+1}{3s}, s \in \left[1, \frac{5+\sqrt{205}}{18} \approx 1.0732 \right], \\ \frac{8s+5}{5s+5}, s \in \left[\frac{5+\sqrt{205}}{18}, \frac{1+\sqrt{31}}{6} \approx 1.0946 \right], \\ \frac{2s+2}{2s+1}, s \in \left[\frac{1+\sqrt{31}}{6}, \frac{1+\sqrt{17}}{4} \approx 1.280 \right] \\ s, \quad s \in \left[\frac{1+\sqrt{17}}{4}, \frac{1+\sqrt{3}}{2} \approx 1.36603 \right] \\ \frac{2s+1}{2s}, s \in \left[\frac{1+\sqrt{3}}{2}, \sqrt{2} \approx 1.41421 \right] \\ \frac{3s+5}{2s+4}, s \in \left[\sqrt{2}, \frac{\sqrt{21}}{3} \approx 1.52753 \right] \\ \frac{3s+3}{3s+1}, s \in \left[\frac{\sqrt{21}}{3}, \frac{5+\sqrt{193}}{12} \approx 1.57437 \right] \\ \frac{4s+2}{2s+3}, s \in \left[\frac{5+\sqrt{193}}{12}, \frac{7+\sqrt{145}}{12} \approx 1.58668 \right] \\ \frac{5s+2}{4s+1}, s \in \left[\frac{7+\sqrt{145}}{12}, \frac{9+\sqrt{193}}{14} \approx 1.63509 \right] \\ \frac{7s+4}{7s}, s \in \left[\frac{9+\sqrt{193}}{14}, \frac{5}{3} \right] \\ \frac{7s+4}{4s+5}, s \in \left[\frac{5}{3}, \frac{5+\sqrt{73}}{8} \approx 1.89300 \right] \\ \frac{s+1}{2}, s \in \left[\frac{5+\sqrt{73}}{8}, \sqrt{3} \right] \\ \frac{s+2}{s+1}, s \in [\sqrt{3}, \infty) \end{array} \right.$$

and

$$\begin{cases} \frac{3s+1}{3s}, & s \in [1, s_8 \approx 1.071], \\ \frac{7s+6}{4s+6}, & s \in \left[s_8, \frac{1+\sqrt{145}}{12} \approx 1.0868 \right], \\ \frac{2s+2}{2s+1}, & s \in \left[\frac{1+\sqrt{145}}{12}, \frac{1+\sqrt{17}}{4} \approx 1.280 \right] \\ s, & s \in \left[\frac{1+\sqrt{17}}{4}, \frac{1+\sqrt{3}}{2} \approx 1.36603 \right] \\ \frac{2s+1}{2s}, & s \in \left[\frac{1+\sqrt{3}}{2}, \frac{1+\sqrt{21}}{4} \approx 1.39562 \right] \\ \frac{6s+6}{4s+5}, & s \in \left[\frac{1+\sqrt{21}}{4}, \frac{1+\sqrt{13}}{3} \approx 1.535187 \right] \\ \frac{12s+10}{9s+7}, & s \in \left[\frac{1+\sqrt{13}}{3}, \frac{5+\sqrt{241}}{12} \approx 1.71035 \right] \\ \frac{2s+3}{s+3}, & s \in \left[\frac{5+\sqrt{241}}{12}, \sqrt{3} \right] \\ \frac{s+2}{s+1}, & s \in [\sqrt{3}, \infty), \end{cases}$$

respectively.

The lower and upper bounds of $Q2|max|C_{max}$ [23, 24, 128] are

$$\begin{cases} \frac{2s+2}{2s+1}, & s \in \left[1, \frac{1+\sqrt{17}}{4} \approx 1.280 \right], \\ s, & s \in \left[\frac{1+\sqrt{17}}{4}, \sqrt{2} \right], \\ \frac{1+\sqrt{4s^2+1}}{2s}, & s \in [\sqrt{2}, x_5 \approx 1.558] \\ \frac{s+2}{s+1}, & s \in [x_5, x_6 \approx 2.292] \\ \frac{(s+1)(9s+5) - \sqrt{(s+1)(-3s^3+7s^2+7s+1)}}{2(s+1)(3s+2)}, & s \in [x_6, x_7 \approx 2.360] \\ \frac{s+\sqrt{s^2+4s}}{2s}, & s \in [x_7, \frac{3+\sqrt{17}}{2} \approx 3.56] \\ \frac{s+1}{s}, & s \in \left[\frac{3+\sqrt{17}}{2}, \infty \right) \end{cases}$$

and

$$\begin{cases} \frac{2s+2}{2s+1}, & s \in \left[1, \frac{1+\sqrt{17}}{4} \approx 1.280\right], \\ s, & s \in \left[\frac{1+\sqrt{17}}{4}, \sqrt{2}\right], \\ \frac{s+2}{s+1}, & s \in [\sqrt{2}, 2] \\ \frac{3s+2}{2s+2}, & s \in [2, 1 + \sqrt{3}] \\ \frac{s+1}{s}, & s \in [1 + \sqrt{3}, \infty), \end{cases}$$

respectively, where x_5, x_6, x_7 are the roots of

$$2s^4 + 2s^3 - 6s^2 - 5s + 3 = 0,$$

$$\begin{aligned} 12s^6 - s^5 - 32s^4 - 12s^3 + 7s^2 + s - 1 &= (6s^4 - 3s^3 - 6s^2 - 3s - 1) \\ &\times \sqrt{(s+1)(-3s^3 + 7s^2 + 7s + 1)}, \\ 8s^3 + 15s^2 + 13s + 4 &= (6s^2 + 9s + 3)\sqrt{s^2 + 4s}, \end{aligned}$$

respectively.

The lower and upper bounds of $Q2|max|C_{min}$ [27, 129] are

$$\begin{cases} \frac{s+2}{s+1}, & s \in [1, \sqrt{2}], \\ s, & s \in \left[\sqrt{2}, \frac{1+\sqrt{5}}{2}\right], \\ \frac{s+1}{s}, & s \in \left[\frac{1+\sqrt{5}}{2}, 1 + \sqrt{2}\right] \\ \frac{s^2 + s + 1 + \sqrt{s^4 - s^2 + 2s + 1}}{s^2 + 2s}, & s \in [1 + \sqrt{2}, \infty) \end{cases}$$

and

$$\begin{cases} \frac{s+2}{s+1}, & s \in [1, \sqrt{2}], \\ s, & s \in \left[\sqrt{2}, \frac{1+\sqrt{5}}{2}\right], \\ \frac{s+1}{s}, & s \in \left[\frac{1+\sqrt{5}}{2}, x_8 \approx 2.148\right] \\ \frac{s+1 + \sqrt{5s^2 + 6s + 1}}{2(s+1)}, & s \in [x_8, x_9 \approx 3.836] \\ \frac{s^2 + s + 1 + \sqrt{s^4 - s^2 + 2s + 1}}{s^2 + 2s}, & s \in [x_9, \infty). \end{cases}$$

respectively, where x_8, x_9 are the positive roots of

$$\frac{s+1}{s} = \frac{s+1+\sqrt{5s^2+6s+1}}{2(s+1)}$$

and

$$\frac{s+1+\sqrt{5s^2+6s+1}}{2(s+1)} = \frac{s^2+s+1+\sqrt{s^4-s^2+2s+1}}{s^2+2s},$$

respectively.

3.2.3 Preemption

In regard to preemptive semi-online scheduling problems, there were some results on optimal algorithms [56, 67, 93, 94]. But breakthrough was not made until Ebenlendr and Sgall introduced the general framework for preemptive online scheduling, through which for the majority of semi-online variants, preemptive scheduling problems with the objective function of makespan can be solved. The optimal algorithm for these semi-online problems is identical with the optimal algorithm for the pure online problem; only the optimal bound needs to be recalculated by using the reformed linear programming (Eq. 4). However, as in the pure online case, there are still two points at issue. The first is that the analytically optimal bounds can hardly be obtained when m is large, even for the identical machines case. The second is that it is necessary for the algorithm to use idle times, and whether it is inevitable still remains unknown.

For $Qm|pmtn, opt|C_{max}$, there exists an algorithm which always produces an optimal schedule [58]. Similar result for two uniform machines was obtained by Epstein [65] before. $Pm|pmtn, sum|C_{max}$ also has an algorithm with competitive ratio 1 [98], but it does not hold for $Qm|pmtn, sum|C_{max}$ [59].

For $Pm|pmtn, decr|C_{max}$, Seiden et al. designed an optimal algorithm with competitive ratio $\max_{0 \leq k \leq m} \frac{2m^2+2mk}{2m^2+k^2+k}$ [154]. The bound tends to $\frac{1+\sqrt{3}}{2}$ when $m \rightarrow \infty$, and the algorithm does not introduce idle times. Interestingly, it is also the optimal algorithm for $Pm|pmtn, max|C_{max}$. However, above two variants do not share optimal algorithm for uniform machines. The optimal bounds for $Q2|pmtn, decr|C_{max}$ [67] and $Q2|pmtn, max|C_{max}$ [94] are

$$\begin{cases} \frac{3s+3}{3s+2}, & 1 \leq s \leq 3, \\ \frac{2s^2+2s}{2s^2+s+1}, & s > 3 \end{cases}$$

and $\frac{2s^2+3s+1}{2s^2+2s+1}$, respectively (Fig. 5). Moreover, no deterministic algorithm that never uses idle time can have the same competitive ratio as those use idle time when $s > 2$ for the former problem and $s > \frac{1+\sqrt{5}}{2}$ for the latter. But the corresponding lower and upper bounds are still unknown. More optimal bounds for different semi-online variants on $m = 2, 3, 4$ uniform machines can be found in [57, 59].

Preemptive machine covering problems are much more complicated than corresponding makespan minimization problems. Due to the particularity of the definition of the objective, whether idle time is allowed should be taken into special consideration. The lower bound of $Pm|pmtn,sum|C_{min}$ is $\frac{2m-3}{m-1}$, and there exists optimal algorithm with matched competitive ratio when $m = 2, 3$ [98]. For $Q2|pmtn,max|C_{min}$, the optimal bound is $\frac{s^2+3s+1}{s^2+2s+1}$ [99]. However, the algorithm may introduce idle time before J_{max} arrives when $s > s' \approx 1.247$. Here, s' is the positive root of $s^3+s^2-2s-1 = 0$. For $Q2|pmtn,decr|C_{min}$, the optimal bound is

$$\begin{cases} \frac{2s+3}{2s+2}, & 1 \leq s \leq 3, \\ \frac{s^2+3s}{s^2+2s+1}, & s > 3. \end{cases}$$

The optimal algorithm also needs to introduce idle time when assigning the first job if $s > \frac{\sqrt{6}}{2}$. Whether there exist algorithms for $Q2|pmtn,sum|C_{min}$ and $Q2|pmtn,opt|C_{min}$ that can always obtain the optimal schedule remains open (Fig. 5).

3.2.4 Other Results

Due to the difficulty in competitive analysis with multiple parameters, there is little study on non-preemptive semi-online problems on more than two uniform machines. Azar and Epstein [11] proposed algorithms with competitive ratio both m for $Qm|opt|C_{min}$ and $Qm|decr|C_{min}$, respectively. These algorithms are optimal in the overall sense.

Even less is known about randomized algorithms for semi-online problems. Seiden et al. [154] proposed a barely random algorithm for $P2|decr|C_{max}$ with competitive ratio $\frac{8}{7}$, and no randomized algorithm can achieve a better competitive ratio.

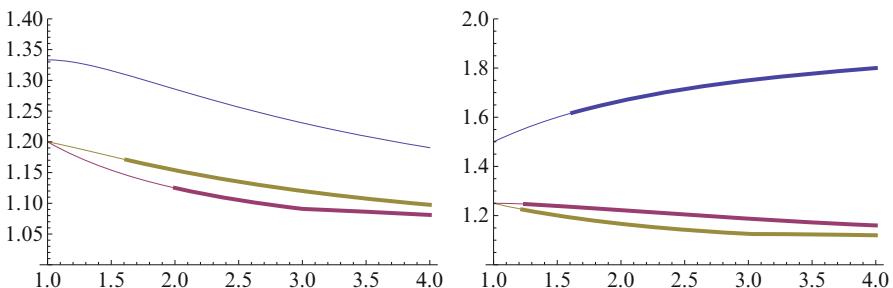


Fig. 5 Left: the optimal bounds of $Q2|pmtn|C_{max}$ (top), $Q2|max, pmtn|C_{max}$ (middle), and $Q2|decr, pmtn|C_{max}$ (bottom). Right: the optimal bounds of $Q2|pmtn|C_{min}$ (top), $Q2|max, pmtn|C_{min}$ (middle), and $Q2|decr, pmtn|C_{min}$ (bottom). Thick segments represent the intervals of s where idle time is necessary

3.3 Results on Basic Semi-online Models of Type II

3.3.1 Buffer

Evidently, it is more likely to design algorithm with a better performance if larger buffer is allowed. However, a buffer of size more than 1 is dispensable for two identical machines. That is to say, the lower bound of the problem with a buffer of size arbitrary large equals to the competitive ratio of an algorithm using a buffer of size only 1, which is $\frac{4}{3}$ [111, 187]. However, such phenomenon does not occur in the case where there are more than two machines or machines have different speeds. Therefore, the algorithm, as well as its corresponding competitive ratios, may be relevant to the size of the buffer. However, it seems impossible and also redundant to obtain optimal algorithms for any values of K . An alternate way is to find an algorithm satisfying *super optimality*. An algorithm with competitive ratio r which uses a buffer of size K is *super optimal* if no algorithm which uses a buffer of size arbitrary large can have a competitive ratio smaller than r , where K is a constant number. Clearly, if two algorithms are both super optimal, then the algorithm which uses a smaller buffer is better. However, research on the problem with a buffer of limited size is also of significance, since in practical application, the buffer size is limited by the cost, space, or other additional restrictions.

Main results on the *buffer* variant on arbitrary number of machines are summarized in Table 7. For $Pm|buffer|C_{max}$ [63], $Pm|pmtn, buffer|C_{max}$ [51], and $Qm|buffer|C_{min}$ [73], super optimal algorithms have been obtained (in the overall sense for the last problem on uniform machines). Note that the super optimal bound for $Pm|pmtn, buffer|C_{max}$ is not an increasing function of m , and the problem cannot be solved by using the general framework included in [59].

For $Q2|buffer|C_{max}$, it is possible to obtain lower and upper bounds as functions of s [52]. A buffer of size 2 is enough to achieve the super optimal bound

$$\begin{cases} \frac{s^2 + 2s + 1}{s^2 + s + 1}, & 1 \leq s \leq \frac{1+\sqrt{5}}{2}, \\ \frac{s^2}{s^2 - s + 1}, & \frac{1+\sqrt{5}}{2} \leq s \leq 2, \\ \frac{s+2}{s+1}, & s \geq 2. \end{cases}$$

Moreover, the optimal algorithm only needs a buffer of size 1 when $s \geq 2$. However, when the buffer size reduces to 1, the optimal bound increases to $\frac{s+2}{s+1}$ for $\sqrt{2} < s < 2$, and there exists an algorithm with competitive ratio $\frac{2(s+1)}{s+2}$ while the lower bound is $\max \left\{ s, \frac{s^2 + 2s + 1}{s^2 + s + 1} \right\}$ $1 \leq s \leq \sqrt{2}$.

Table 7 Main results for the *buffer* variant on arbitrary number of machines

Problem	Buffer size/lower bound		Buffer size	Lower bound	Buffer size/competitive ratio
	Buffer size	Lower bound			
$P_m buffer C_{max}$ [63] (except marked)	Constant	γ_m^a		$\left\lceil \left(1 + \frac{2}{\gamma_m}\right)m \right\rceil + 1^a$	γ_m^a
	$\lfloor \frac{m}{2} \rfloor - 1$	$\frac{3}{2}$		$\frac{3m}{2}$	$\frac{3}{2}$ [112]
	$\lfloor \frac{m}{8} \rfloor - 1 (m \geq 8)$	$1 + \frac{1}{\sqrt{2}}$		m	$1 + \frac{\gamma_m a}{2}$
			$K \in [1, \frac{m+1}{2}]$		$2 - \frac{1}{m-K}$
$Q^m buffer C_{max}$				$m - 1$	$2 + \epsilon (\epsilon > 0)$ [63]
				m	$2 - \frac{1}{m} + \epsilon (\epsilon > 0)$ [112]
$Pm pmtn,$ <i>buffer</i> [51]	m is even	Constant	$\frac{4}{3}$	$\frac{m}{2} - 1$	$\frac{4}{3}$
	m is odd	Constant	$\frac{4m^2}{3m^2+1}$	$\frac{m-1}{2}$	$\frac{4m^2}{3m^2+1}$
			b		
	General m	$\leq \lceil \frac{m-2}{2} \rceil$			
		$1 (m \geq 5)$	$\frac{4m^3-12m^2+4m}{3m^3-11m^2+18m-24}$		
$Pm buffer C_{min}$ [73]	Constant		$\sum_{i=1}^m \frac{1}{i}$	$m - 1$	$1 + \sum_{i=1}^{m-1} \frac{1}{i}$
		$K = f(m) < m$	$\frac{m}{f(m)+1}$		
$Qm buffer C_{min}$ [73]	constant		m	$m + 1$	m
		$m - 2$	∞	$m - 1$	$\frac{2m-1+\sqrt{4m^2-8m+5}}{2}$

^a γ_m is the smallest positive solution of the equation

$$\left\lceil m - \frac{m}{\gamma_m} \right\rceil \frac{\gamma_m}{m} + (\gamma_m - 1) \sum_{i=m-\frac{m}{\gamma_m}}^{m-1} \frac{1}{i} = 1.$$

For any given m , γ_m can be calculated numerically, for example, $\gamma_3 \approx 1.3666$, $\gamma_4 \approx 1.375$. Moreover, $\lim_{m \rightarrow \infty} \gamma_m = \gamma_\infty \approx 1.4659$, where γ_∞ is the smallest solution of equation $\gamma_\infty e^{\gamma_\infty} = -\frac{1}{e^2}$

$$b_{\max K+1 \leq i \leq m+1} \frac{m'}{(i+IK-im-Km)m^{K-i}(m-1)^{-K-i}+(K+m)m^{i-1}}$$

3.3.2 Reassignment

Recall that the online over list model is typically characterized by the disallowance of reassignment. Since semi-online can be viewed as relaxation of online, it is natural to consider the situation where some jobs can be actually reassigned, which is also of realistic interest from an application perspective.

Variants in this category can be classified into two main types: sequential reassignment and final one-off reassignment. For the first type, reassignment can be done when every new job arrives, while for the second one, reassignment can only be done after all jobs have been assigned. For both cases, however, there should be some constraints on the reassignment; otherwise, the problem will reduce to an offline fashion.

In [160], Sanders et al. considered the *proportional sequential reassignment* model. When a job with processing time p_j arrives, some jobs with total processing time at most δp_j can be reassigned, where δ is the *migration factor*. Obviously, the competitive ratio r of an algorithm will decrease when δ increases. For identical machines with objective to minimize makespan, they designed algorithms for different value of δ . Some combinations of r and δ are $(r, \delta) = (\frac{3}{2}, \frac{4}{3})$, $(r, \delta) = (\frac{3}{2} - \frac{1}{2m}, 2)$, and $(r, \delta) = (\frac{4}{3}, \frac{5}{2})$, respectively. They even obtained a family of online algorithms with competitive ratio $1 + \varepsilon$, where $\varepsilon > 0$ can be arbitrarily close to 0, while δ only depends exponentially on $\frac{1}{\varepsilon}$. However, only for the case of $m = 2$ and $\delta = 1$, an algorithm with competitive ratio $\frac{7}{6}$ has been proved to be optimal. For the machine covering problem, they obtained an algorithm with competitive ratio 2 for $\delta = 1$.

Epstein and Levin [71] studied the above variants in a preemptive setting. Since a job can be split, the scheduler is allowed to reassign only part of the job. As a result, the migration factor does not reckon in the entire processing time but rather only the reassigned fraction of that. They presented a $(r, \delta) = (1, 1 - \frac{1}{m})$ algorithms for m identical machines, and the migration factor cannot be improved. For m uniform machines, a migration factor at least $m - 1$ is needed to obtain an algorithm with competitive ratio 1.

The restriction of reassigned job can be made through numbers of jobs instead of total processing times. That is at most G already assigned jobs can be reassigned when a new job arrives. Such variant can be called as *sequential reassignment with quantitative restriction* and is denoted *reassignSQ*. It is stronger than *buffer* since the former can simulate the latter by the following steps: First, temporarily set the jobs on any one of the machines as if they were stored in a buffer. And then reassign them on the machine as if jobs in the buffer were assigned. Contrariwise, a job can no longer be reassigned unless it is in the buffer. In brief, *reassignSQ* can do what *buffer* can, but not vice versa.

For $Q2|reassignSQ|C_{max}$, Dósa et al. [55] proved that the optimal bound when $G \geq 2$ coincides with the optimal bound of $Q2|buffer|C_{max}$ with buffer size $K \geq 2$. However, when $G = 1$, the lower bound is

$$\begin{cases} \frac{s^2 + 2s + 1}{s^2 + s + 1}, & 1 \leq s \leq \frac{1 + \sqrt{5}}{2}, \\ \frac{s + 1}{2}, & \frac{1 + \sqrt{5}}{2} \leq s \leq \sqrt{3}, \\ \frac{s + 2}{s + 1}, & s \geq \sqrt{3}, \end{cases}$$

while the upper bound is

$$\begin{cases} \frac{s^2 + 2s + 1}{s^2 + s + 1}, & 1 \leq s \leq s'', \\ \frac{s + \sqrt{5s^2 + 8s + 4}}{2(s + 1)}, & s'' \leq s \leq \sqrt{3}, \\ \frac{s + 2}{s + 1}, & s \geq \sqrt{3}, \end{cases}$$

where s'' is the root of equation $s^3 - s - 1 = 0$. The algorithm is not optimal when $s \in [s'', \sqrt{3}]$, and it is not identical with the bound of $Q2|buffer|C_{max}$ with buffer size $K = 1$ (Fig. 6).

Several final one-off reassignment models were first proposed by Tan and Yu [167]. One is that when all jobs have been assigned, at most H arbitrary jobs can be reassigned. The model is denoted as *reassignFA*. Tan and Yu proved that optimal bound for two identical machines with objective to minimize makespan is $\frac{4}{3}$, and the optimal algorithm only needs to reassign at most one job. Albers and Hellwig [4] generalized the model to m identical machines. Interestingly, the competitive ratio γ_m of the given algorithm is the same as that of the optimal algorithm for $Pm|buffer|C_{max}$, and the algorithm only reassigns at most $\left(\lceil \frac{2-\gamma_m}{(\gamma_m-1)^2} \rceil + 4\right)m$ jobs. Moreover, no algorithm can achieve a better competitive ratio if $H = o(n)$. They also obtained algorithms with different value of r and H , such as $(r, H) = (\frac{5}{3}, 4m)$ and $(r, H) = (\frac{7}{4}, \frac{5}{2}m)$. For two uniform machines, the optimal bound of $Q2|reassignFA|C_{max}$ when $H \geq 2$ is again the same as that of $Q2|buffer|C_{max}$ when $K \geq 2$. Also, the optimal algorithm reassigns at most one job when $s > 2$. However, if only one job can be reassigned, the competitive ratio of the best known algorithm is also the same as that of $Q2|buffer|C_{max}$ with buffer size $K = 1$, but it is larger than that of $Q2|reassignSQ|C_{max}$ with $G = 1$ (Fig. 6).

Another final one-off reassignment model is denoted *reassignFE*. After all jobs have been arrived, the *last* job assigned to *each* machine can be reassigned. For two identical machines with objective to minimize makespan, the optimal bound is $\sqrt{2}$ [167]. In fact, in such situation at most one job is needed to be reassigned [132]. Cao and Liu [25] generalized the result to two uniform machines and obtain the optimal bound

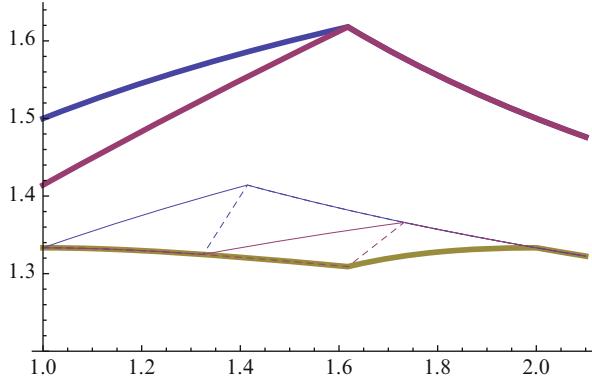


Fig. 6 The optimal bounds (thick) of $Q2||C_{max}$ (top), $Q2|reassignFE|C_{max}$ (middle), and $Q2|buffer|C_{max}$ with $K \geq 2$ (also $Q2|reassignSQ|C_{max}$ with $G \geq 2$ and $Q2|reassignFA|C_{max}$ with $H \geq 2$) (bottom), the upper bounds (solid) and lower bounds (dash) of $Q2|buffer|C_{max}$ with $K = 1$ (also $Q2|reassignFA|C_{max}$ with $H = 1$) (top) and $Q2|reassignSQ|C_{max}$ with $G = 1$ (bottom)

$$\begin{cases} \sqrt{s+1}, & 1 \leq s \leq \frac{1+\sqrt{5}}{2}, \\ \frac{s+1}{s}, & s > \frac{1+\sqrt{5}}{2}. \end{cases}$$

Comparing it with the optimal bound of the pure online problem [72], reassignment is useless when $s > \frac{1+\sqrt{5}}{2}$ (Fig. 6).

3.4 Results on Combined Semi-online Models

3.4.1 UB and LB

Though *UB* and *LB* are both valueless, their combination can be valuable. Such situation is very common in practice, since schedulers tend to estimate the processing time of jobs. However, it would be of little significance if the estimation is too rough to neglect the error. Hence, it is necessary to take influence of a parameter β , defined as $\beta = \frac{UB}{LB} \geq 1$, into consideration when dealing with competitive analysis. Main results of this variant are summarized in Table 8.

Note that *LS* remains optimal for any value of β for $P2|UB\&LB|C_{max}$ and $Pm|UB\&LB|C_{min}$. It is not true for $P3|UB\&LB|C_{max}$, though *LS* is also the optimal algorithm for $P3||C_{max}$. He and Dósa [92] proved the competitive ratio of *LS* is

$$\begin{cases} \frac{2\beta+1}{3}, & \beta \in [1, \frac{3}{2}], \\ \frac{2\beta+3}{\beta+3}, & \beta \in [\frac{3}{2}, \sqrt{3}], \\ \frac{\beta+1}{2}, & \beta \in [\sqrt{3}, 2], \\ 2 - \frac{1}{\beta}, & \beta \in [2, 3], \\ \frac{5}{3}, & \beta \in [3, +\infty), \end{cases}$$

and LS is optimal only when $\beta \in [1, \frac{3}{2}] \cup [\sqrt{3}, 2] \cup [6, +\infty)$. When $\beta \in [2, 6]$, there exists an improved algorithm with competitive ratio

$$\begin{cases} \frac{3}{2}, & \beta \in [2, \frac{5}{2}], \\ \frac{4\beta+2}{2\beta+3}, & \beta \in [\frac{5}{2}, 3], \\ \frac{5}{3} - \frac{1}{18} \min \left\{ \frac{6-\beta}{18}, \frac{3}{103} \right\}, & \beta \in [3, 6]. \end{cases}$$

Nevertheless, the new upper bound matches the lower bound of the problem only when $\beta \in [2, \frac{5}{2}]$. By the definition of the paradigm, the lower bound of the problem is obviously a nondecreasing function of β . For the remaining intervals of β where optimal algorithm has not been obtained, nonconstant lower bound of the problem, equaling to $\frac{7\beta+4+\sqrt{\beta^2+8\beta+4}}{2\beta+2+2\sqrt{\beta^2+8\beta+4}}$, can be found only in the situation where $\beta \in [\frac{5}{2}, 3]$. It is implied again from the results stated above that scheduling for two machines is of enormous difference in essence from that of three machines, while the latter is much more complicated.

A similar model of LB and max is proposed by Cao et al. [26]. It is assumed that $\frac{p_{max}}{\beta} \leq p_j \leq p_{max}$ for any j , and there always exists a job of processing time p_{max} . They proved that the optimal bound of $P2|LB\&max|C_{max}$ is

$$\begin{cases} \frac{\beta+1}{2}, & \beta \in [1, \frac{4}{3}], \\ \frac{4\beta+4}{3\beta+4}, & \beta \in [\frac{4}{3}, \sqrt{2}], \\ \frac{2\beta}{\beta+1}, & \beta \in [\sqrt{2}, 2], \\ \frac{4}{3}, & \beta \in [2, +\infty). \end{cases}$$

It is smaller than that of $P2|LB\&UB|C_{max}$ for some value of β (Fig. 7).

3.4.2 sum/opt and max/UB

Angelelli et al. [7, 8] studied the problem $P2|sum\&UB|C_{max}$. Lower bounds as functions of $\gamma = \frac{2UB}{P}$ and several algorithms are proposed. Optimal bounds are achieved for majority value of γ (see Figs. 4–6 of [8]).

Table 8 Main results for the *UB&LB* variant

Problem	Interval of β where the partial information is valuable and the optimal bound	Interval of β where the partial information is valueless ^a
$P2 UB\&LB C_{max}$ [97]	$\beta \in [1, 2], \frac{1+\beta}{2}$	$\beta \in [2, +\infty]$
$Pm UB\&LB C_{max}, m \geq 3$ [92] ^b	$\beta \in [1, \frac{m}{m-1}], 1 + \frac{(m-1)(r-1)}{m}$	
$Pm UB\&LB C_{min}$ [91]	$\beta \in [1, m], \beta$	$\beta \in [m, +\infty)$
$Pm pmtn, UB\&LB C_{max}$ [94]	^c	$\beta \in [(\frac{m}{m-1})^{m-1}, +\infty)$
$Q2 pmtn, UB\&LB C_{max}$ [56, 93]	$\beta \in [1, 2s],^d$	$\beta \in [2s, +\infty)$
$Q2 pmtn, UB\&LB C_{min}$ [105]	$\beta \in [1, 2s], \frac{2s+2+\beta}{2s+2}$	$\beta \in [2s, +\infty)$

^aValueless implies that the optimal bound is the same as the optimal bound of the corresponding pure online problem

^bOnly partial results about this problem have been obtained

^cOptimal algorithm can be obtained by using the general framework included in [59]. The analytical lower bound for $\beta \in \left[1, \left(\frac{m}{m-1}\right)^{m-1}\right]$ is at least

$$\frac{m \left(\frac{m}{m-1}\right)^k + (m-k-1)\beta}{(2m-k-1) \left(\left(\frac{m}{m-1}\right)^k - 1\right) + (m-k) + \frac{(m-k)(m-k-1)\beta}{2m}},$$

$$\left(\frac{m}{m-1}\right)^k \leq \beta \leq \left(\frac{m}{m-1}\right)^{k+1}, k = 0, 1, \dots, m-2.$$

Whether it is the optimal bounds remains open except for $m = 2, 3$.

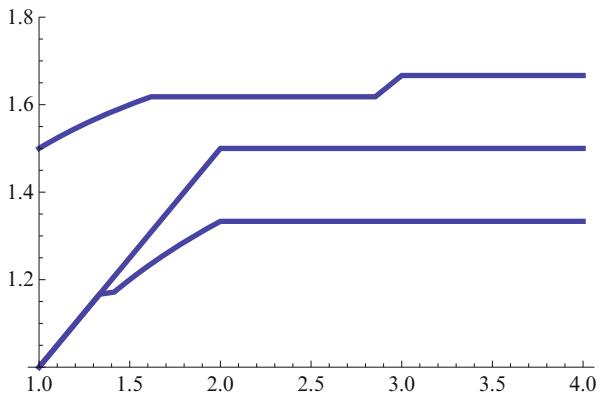
^dThe optimal bound is

$$\begin{cases} \frac{1+s+\frac{\beta}{2}+\frac{\beta s}{2}}{1+s+\frac{\beta s}{2}}, & 1 \leq s \leq 2 \text{ and } \beta \in [1, 2s], s > 2 \text{ and } \beta \in \left[1, \frac{2}{s-1}\right] \cup [2s-2, 2s] \\ \frac{(1+\beta)(1+s)s}{1+s+s^2+s\beta^2}, & s \geq 2 \text{ and } \beta \in \left[\frac{2}{s-1}, s\right] \\ \frac{s^2+s}{s^2+1}, & s \geq 2 \text{ and } \beta \in [s, 2s-2]. \end{cases}$$

If idle time is not allowed, then the (deterministic) lower bound will be larger, but no online algorithm with matched competitive ratio has been obtained [93]

There are other papers considering the combination of *sum(opt)* and *max*. However, only overall bounds (the maximum bound among all combinations of P (or C^*) and p_{max}) are obtained. For example, the optimal bound for $Pm|sum\&max|C_{min}$ [166] is

Fig. 7 The optimal bounds of $P2|\mathcal{M}_j(GoS), LB\&UB|C_{max}$ (top), $P2|LB\&UB|C_{max}$ (middle), and $P2|LB\&max|C_{max}$ (bottom)



$$\begin{cases} \frac{5}{4}, & m = 2, \\ \frac{3}{2}, & m = 3, \\ m - 2, & m \geq 4. \end{cases}$$

For the makespan minimization problem, optimal bounds are obtained only on a small number of machines (See [Table 5](#)). If preemption is allowed, optimal bound of $Q3|pmtn, sum\&max|C_{max}$ for any combinations of P and p_{max} and any values of $s_i, i = 1, 2, 3$, can be obtained by using the general framework included in [\[59\]](#).

3.4.3 End of Sequence

Zhang and Ye [\[190\]](#) proposed an interesting semi-online variant, which was called “end of sequence” (*eos* for short) later [\[70\]](#). It is known that the last job has the largest processing time, and the scheduler will be informed whether the current arrived job is the last one. Such variant can be viewed as a weaker version of the combined semi-online variant *num&incr*. Recall that both *num* and *incr* are valueless if they are considered solely. Thus, it is likely that *eos* is adjacent to the boundary between valuable semi-online variants and valueless ones.

Zhang and Ye [\[190\]](#) proved the optimal bounds of $P2|eos|C_{max}$ and $P3|eos|C_{max}$ are $\sqrt{2}$ and $\frac{3}{2}$, respectively. Note that the lower bound of $P2|eos|C_{max}$ does not hold for $P2|num\&incr|C_{max}$, the optimal bound for the latter problem remains open. Epstein and Ye considered the problems $Q2|eos|C_{max}$ and $Q2|eos|C_{min}$. Optimal algorithms for almost all values of s are given [\[70\]](#) ([Table 6](#), [Figs. 3 and 4](#)).

Table 9 Results for the non-preemptive problems with inexact partial information

Problem	The interval of α where the inexact information is valuable	The interval of α where the obtained algorithm is optimal	The gap between the upper and lower bound	The total length of non-optimal intervals	The interval of α where the inexact information is valueless
$P2 inexact sum C_{max}$	$[1, \frac{3}{2})$	$[\frac{3+\sqrt{21}}{6}, \frac{3}{2})$	≤ 0.0303	≤ 0.2635	$[\frac{3}{2}, \infty)$
$P2 inexact opt C_{max}$	$[1, \frac{3}{2})$	$[\frac{3+\sqrt{21}}{6}, \frac{1+\sqrt{10}}{3}] \cup [\frac{6-\sqrt{10}}{2}, \frac{3}{2})$	≤ 0.0303	≤ 0.2957	$[\frac{3}{2}, \infty)$
$P2 inexact max C_{max}$	$[1, 2)$	$[1, \sqrt{5} - 1]$	≤ 0.0445	≤ 0.7640	$[2, \infty)$

Table 10 Results for the preemptive problems with inexact partial information

Problem	The interval of α where the inexact information is valuable and the optimal bound	The interval of α where the inexact information is valueless
$Q2 pmtn, inexact opt C_{max}$	$\alpha \in \left[\left(\frac{m}{m-1} \right)^k, \left(\frac{m}{m-1} \right)^{k+1} \right],$ $\frac{1}{1 - \left(\frac{m}{m-1} \right)^k + (m-k) + \frac{m-k}{m\alpha}},$ $k = 0, 1, \dots, m-2$	$\left[\left(\frac{m}{m-1} \right)^{m-1}, \infty \right)$
$Pm pmtn, inexact opt C_{max}$	$[1, s+1), \frac{\alpha(s+1)}{\alpha s + 1}$	$[s+1, \infty)$
$Q2 pmtn, inexact max C_{max}$	$[1, s+1), \frac{(s+1)(1+s+\alpha s)}{1+2s+s^2+rs^2}$	$[s+1, \infty)$

3.4.4 Miscellanies

The optimal bounds of $P2|decr\&sum|C_{max}$ and $P2|decr\&opt|C_{max}$ are both $\frac{10}{9}$ [65, 164]. Azar and Epstein [11] designed an algorithm for $Qm|decr\&opt|C_{min}$ with an overall competitive ratio 2, which is optimal in the overall sense.

By using the general framework included in [59], it can be proved that the overall optimal bound of $Qm|pmtn, decr\&sum|C_{max}$ is $\frac{12}{11}$ when $m = 3$ and $\frac{10}{9}$ when $m = 4$. For the problem $Pm|pmtn, decr\&buffer|C_{max}$, Dósa and Epstein [51] proved that a buffer of size $m-1$ is enough for an online algorithm to produce the optimal schedule, and the optimal bounds for any buffer size $K < m-1$ is $\max_{0 \leq \mu \leq m-K-1} \frac{2m(m+\mu)}{2m^2+2K\mu+\mu^2+\mu}$.

Dósa and He [53] studied two problems $P2|sum\&buffer|C_{max}$ and $P2|sum\¶llel(2)|C_{max}$. They designed two optimal algorithms with competitive ratio $\frac{5}{4}$ and $\frac{6}{5}$, respectively. Min et al. [132] studied the problem $P2|sum\&reassignFE|C_{max}$ and proposed an optimal algorithm with competitive ratio $\frac{5}{4}$.

3.5 Results on Disturbed Semi-online Models

In regard to disturbed semi-online model, there are fewer variants brought forward and results achieved. Study on problems with inexact partial information is much more difficult than the problem with exact partial information. One reason is that it is expected to find the upper and lower bound as functions of disturbance parameter α . If preemption is not allowed, only problems on two identical machines have been studied [165]. Related results are summarized in [Table 9](#), and the detailed bounds are omitted here. If preemption is allowed, some research approaches and techniques of previous variants can be generalized to m identical machines or two uniform machines [106]. These results are summarized in [Table 10](#). By using the general framework included in [59], even the optimal bound of $Q3|pmtn, \text{inexact sum}|C_{max}$ for any α and any values of s_i , $i = 1, 2, 3$ can be obtained. However, the known algorithms for the non-preemptive problem are not optimal for all values of α , and also there is no study on the problems of $Pm|pmtn, \text{inexact max}|C_{max}$, $Pm|pmtn, \text{inexact sum}|C_{max}$, $Q2|pmtn, \text{inexact sum}|C_{max}$.

4 Online Over Time

Scheduling jobs that arrive over time is usually called online over time model. A number of researches are done with respect to the minimization problems for the three following classical objective functions: the makespan C_{max} , the total completion time $\sum_{j=1}^n C_j$, and the total weighted completion time $\sum_{j=1}^n w_j C_j$. For these objective functions, and some special cases, deterministic and randomized online algorithms are analyzed deriving different approximation guarantees. Results on online over time model are summarized in [Table 11](#).

4.1 Single Machine Scheduling

4.1.1 Minimize the Total Completion Time

Problems of minimizing the total completion time on a single machine are extensively studied. Phillips et al. [142] presented a 2-competitive algorithm based on the optimal preemptive schedule, which can be found by the *SRPT* (Shortest Remaining Processing Time first) rule in polynomial time [148]. There is another 2-competitive algorithm, namely, delayed SPT ($D - SPT$), provided by Hoogeveen and Vestjens [100].

Algorithm D-SPT

At any time t a machine is idle and a job is available, choose a job with shortest processing time, say J_i . If there are more than one job with the same processing

Table 11 The best known results on over time model

Problem	Jobs	Lower bound	Upper bound	Randomized lower bound	Randomized upper bound
$1 r_j \sum C_j$	None	2	[100]	2	[100, 127, 142]
pmtn	1	[148]	1	[148]	1
res	1.2108	[69]	1.5	[175]	1.1068
$UB \& LB$	Φ	[170]	$\frac{\sqrt{1+\beta(\beta-1)}+\beta-1}{\beta}$	[170]	$\frac{e}{e-1}$
$1 r_j \sum w_j C_j$	None	2	[100]	2	[5]
pmtn	1.0730	[69]	1.57	[158]	1.0389
res	1.2232	[69]	2	[5]	1.1161
$UB \& LB$	$1 + \frac{\sqrt{4\beta^2+1}-1}{2\beta}$	[171]	$1 + \frac{\sqrt{4\beta^2+1}-1}{2\beta}$	[171]	[69]
release ^a	Ψ	[89]	Ψ	[89]	[89]
$P r_j \sum C_j$	None	1.309	[176]	1.791 + $o(m)$	[159]
pmtn	$\frac{21}{19}$	[184]	$\frac{5}{4}$	[45, 159]	1.047
$P r_j \sum w_j C_j$	None	1.309	[176]	1.791 + $o(m)$	[159]
pmtn	1.114	[184]	1.791 + $o(m)$	[159]	1.157
$P r_j C_{max}$	None	1.3473	[30]	1.5	[30]
$P2 r_j C_{max}$	None	$\frac{5-\sqrt{5}}{2}$	[30]	$\frac{5-\sqrt{5}}{2}$	[138]
$Q r_j \sum C_j$	None	1.309	[176]	$\frac{\sqrt{4m-3}+3}{2}$	[122, 124]
$Q2 r_j \sum w_j C_j$	pmtn	1	2	2	[122]

^aThe semi-online problems that release times of all jobs are known in advance

time, choose the one with the smallest release time. If $p_i \leq t$, then process J_i ; otherwise, wait until time p_i or until a new job arrives, whichever happens first.

A very important idea in $D - SPT$ algorithm is to shift the release time of jobs before scheduling, which was adopted by Stougie as well (cited in [176]). Lu et al. [127] generalized this idea and proposed a class of online algorithms that have the same competitive ratio of 2. Note that for this problem, it is shown by Hoogeveen and Vestjens [100] that any online algorithm must have a competitive ratio no smaller than 2, and hence, all above algorithms are best possible.

In online over time problems, it is commonly interesting to know whether or how much the competitive ratio would decrease if restarts are allowed. Vestjens [176] first showed a lower bound for the competitive ratio of any online algorithm. Later, Epstein and van Stee [69] improved the bound from 1.112 to 1.2108. The algorithm provided by van Stee and Poutré [175] indicates that the upper bound of the problem allowing restarts is at most $\frac{3}{2}$.

In contrast with online over list, only a few researches are found involving in semi-online variants, even though there are actually more semi-online models than those in online over list model. Tao et al. [170] investigated the problem with the knowledge of the upper and lower bounds of the processing times, $UB \& LB$. A semi-online algorithm with competitive ratio $\frac{\sqrt{1+\beta(\beta-1)}+\beta-1}{\beta}$, as well as a lower bound

$$\Phi = \min_{s \geq 0} \max \left\{ 1 + s, \max_{k \in \mathbb{Z}^+} \left\{ 1 + \frac{2k(\beta - 1)}{2\beta(k + 1)s + 2\beta + k^2 + 3k} \right\} \right\},$$

is proposed, where $\beta = \frac{UB}{LB} \geq 1$.

4.1.2 Minimize the Total Weighted Completion Time

Indeed, the weighted problem is more general and hence more difficult. Anderson and Potts [5] considered the delayed weighted shortest processing time rule ($D - WSPT$), which is a direct generalization of the $D - SPT$ algorithm. By introducing a new proof technique, they showed that $D - WSPT$ has a competitive ratio of 2 and therefore matches the known lower bound [100]. If restarts are allowed, the lower and upper bounds of this problem are 1.2232 and 2, respectively [5, 69].

If preemption is allowed, unlike the unweighted version, Vestjens [176] showed that any preemptive algorithm must have a competitive ratio no smaller than 1.0333. The lower bound is later improved to 1.0730 by Epstein and van Stee [69]. Several algorithms are proved to be 2-competitive, including $D - WSPT$, $P - WSPT$, and $WSRPT$ [130, 146]. Here, $P - WSPT$ (preemptive $WSPT$) schedules at any time the job with the largest ratio of weight over processing time, while $WSRPT$ (weighted shortest remaining processing time first) schedules at any time the job with the largest ratio of weight over remaining processing time. The instance given by Xiong and Chung [184] recently showed that the competitive ratio of

WSRPT cannot be smaller than 1.215. Though there is a big gap between the lower and upper bounds, developing an online algorithm with a competitive ratio better than 2 is a hard work. Breakthrough on this problem is finally made by Sitters [158]. By using a parameter $c \geq 1$ and applying the $P - WSPT$ rule with a restriction that a job cannot be preempted at time t if it can be completed before time ct , a class of online algorithms, namely, $ONLINE(c)$, are obtained and shown to be c -competitive for any $c \geq \xi$, where $\xi \approx 1.57$ is the real root of $2\xi^3 - 4\xi^2 + 2\xi - 1 = 0$. Hence, there exists a ξ -competitive algorithm for the preemptive problem.

If the release times of all jobs are known in advance, Hall et al. [89] showed that the problem has a lower bound of

$$\Psi = \max_{1 \leq j \leq n} \min_{0 \leq l \leq j-1} \frac{r_{l+1} + r_l + \sqrt{(r_{l+1} - r_l)^2 + 4r_j r_{l+1}}}{2r_{l+1}}$$

where $r_1 \leq r_2 \dots \leq r_n$ are the release times of all jobs. It can be proved that Ψ is in between $\frac{\sqrt{5}+1}{2}$ and 2. An online algorithm with matched competitive ratio is provided as well. Clearly, this variant will never be classified into online over list model. The semi-online problem *UB&LB* is considered by Tao et al. [171], where they presented an optimal algorithm with competitive ratio $1 + \frac{\sqrt{4\beta^2+1}-1}{2\beta}$.

4.1.3 Randomized Algorithm and Lower Bound

Chekuri et al. [29] designed a randomized algorithm to minimize the total completion time with competitive ratio $\frac{e}{e-1}$. Their algorithm is intriguing as it beats the lower bound 2 for deterministic online algorithms [100]. Moreover, it is also optimal since its competitive ratio matches the randomized lower bound given by Stougie and Vestjens [162]. If restarts are permitted, Epstein and van Stee [69] proved that any randomized algorithm must have a competitive ratio at least 1.1068. In the same paper, they also considered problems aiming at minimizing the total weighted completion time, where a lower bound 1.0389 and a lower bound 1.1161 for the preemptive problem and the problem allowing restarts are proposed, respectively. Regarding the upper bounds, Schulz and Skutella [149] provided a $\frac{4}{3}$ -competitive randomized algorithm for the preemptive problem. Goemans et al. [83] presented a randomized algorithm with competitive ratio 1.6853 for the problem allowing restarts.

4.2 Results on Parallel Machine Scheduling

4.2.1 Minimize the Makespan

For the makespan minimization problems, it should be emphasized that Shmoys et al. [157] have described a general method to use offline algorithms to obtain online algorithms for problems of online over time model. However, the method

always produces a competitive ratio at least 2, even if the corresponding offline problem can be solved to optimality. In the same paper [157], Shmoys et al. proved a lower bound of $\frac{10}{9}$ for the problem on m identical machines. Chen and Vestjens [30] improved the lower bound to $1 + \alpha$ for $m \geq 3$ and to $\frac{5-\sqrt{5}}{2}$ for $m = 2$, where $\alpha \approx 0.3473$ is the solution of $\alpha^3 - 3\alpha + 1 = 0$. Moreover, they proposed a simple algorithm, namely, online Longest Processing Time first (online *LPT*).

Algorithm Online *LPT*

At any time a machine becomes idle for processing, schedule an available job with the longest processing time.

The competitive ratio of online *LPT* is shown to be $\frac{3}{2}$. As far as we know, this is the best result on this problem except that in [138], where Noga and Seiden presented an improved algorithm *SLEEP* for $m = 2$. The difference between *SLEEP* and online *LPT* is that it might wait for some time even when there is an idle machine and an available job. The idea makes the algorithm optimal for $m = 2$ and, hence, beat online *LPT*. For further study, one may ask how to extend the algorithm and whether it can beat online *LPT* for a general number of machines. Nevertheless, there are still no such study by now. The randomized lower bounds on makespan minimization problem are studied in [162] and [138], where a lower bound of $4 - 2\sqrt{2}$ for a general number of machines and a lower bound of 1.21207 for $m = 2$ are provided, respectively.

4.2.2 Minimize the Total (Weighted) Completion Time

For the problem of minimizing the total completion time, any deterministic algorithm is at least $\frac{21}{19} \approx 1.105$ -competitive for the preemptive version and is at least 1.309-competitive for the non-preemptive version [176, 184]. Moreover, any randomized algorithm is at least 1.047-competitive for the preemptive version and is at least 1.157-competitive for the non-preemptive version [176].

The simple and well-known *SRPT* algorithm plays a significant rule in preemptive scheduling of minimizing the total completion time. As we have seen before, it produces an optimal schedule for the single machine case [148]. The upper bound of *SRPT* for m identical machines was known to be 2 [142] for a long time until recently, Chung et al. [45] claimed that it is at most 1.86. Shortly after that, Sitters [159] improved it to $\frac{5}{4}$.

For the preemptive problem of minimizing the total weighted completion time, 2-competitive online algorithm is found in [131]. The lower bound of the problem is proved to be $\frac{16-\sqrt{14}}{11} \approx 1.114$ [184]. For the non-preemptive version, a randomized algorithm with competitive ratio 2 as well as a deterministic algorithm with competitive ratio 2.62 are obtained by Schulz and Skutellathe [150] and Correa and Wagner [46], respectively. The best known result for this objective is due to Sitters [159], who gave a deterministic algorithm with competitive ratio of $1.791 + o(m)$ for both versions. Even applying the algorithm to unweighted problem, the competitive ratio remains the same, and the algorithm beats the previously best one with competitive ratio 2 [121]. If the number of

machines is small, randomized algorithms with smaller competitive ratios are found in [46].

With respect to machines with different speeds, Liu and Lu [122] studied the two machine case. They presented a $\frac{\sqrt{5}+3}{2}$ -competitive algorithm for the non-preemptive problem of minimizing the total completion time. If preemption is allowed, they gave an algorithm with competitive ratio of 2 to minimize the total weighted completion time. The non-preemptive algorithm in [122] for two uniform machines was extended to m uniform machines by Liu et al. [124], who established a competitive ratio of $\frac{\sqrt{4m-3}+3}{2}$.

5 Other Variants of Online Scheduling

5.1 Online Shop Scheduling

Researches on online shop scheduling are mainly focused on two or three machines. Chen and Woeginger [34] considered the two-machine open shop scheduling of online over list model. A 1.875-competitive algorithm, as well as a lower bound $\frac{1+\sqrt{5}}{2} \approx 1.618$, was given. By restricting to *permutation algorithms* that always produce schedules with the same job sequence on both machines, the upper and lower bound can be further improved to 1.848 and $\frac{23-2\sqrt{13}}{9} \approx 1.754$ [37], respectively. If preemption is allowed, Chen and Woeginger [34] presented a $\frac{4}{3}$ -competitive algorithm for two machines, and a $\frac{27}{19}$ -competitive algorithm for three machines was provided by Chen et al. [37]. Since the lower bound Γ_m of $Pm|pmtn|C_{max}$ is also a lower bound of $Om|pmtn|C_{max}$ [155], both algorithms are optimal. For two-machine flow shop and job shop problems, optimal algorithms were proven to have a competitive ratio of 2 by Chen and Woeginger [34].

If jobs arrive over time, Chen et al. [35] considered the two-machine open shop scheduling and proved that greedy-like algorithm has a competitive ratio $\frac{3}{2}$ and is optimal for non-preemptive version. Another algorithm, namely, SLICE, was shown to have a competitive ratio of $\frac{5}{4}$ and best possible for preemptive version. Liu et al. [125] further extended SLICE to the semi-online problem *UB & LB*. They proved the corresponding algorithm is $\frac{5\beta-1}{4\beta}$ -competitive and matches the lower bound of the considered problem with $\beta = \frac{UB}{LB}$. Stougie and Vestjens [162] proved that 1.25 is a randomized lower bound of $O2|r_j|C_{max}$.

Recently, Zhang and Velde [188, 189] investigated problems with time lags, where the time lag of a job is defined as the maximum allowable delay between the completion time of the first and the start time of the second operation of the job. For these problems, it might benefit from allowing a machine to be idle while an operation is available for assignment. Such algorithms are called delay algorithms. The greedy-like algorithm has a competitive ratio 2 for two-machine open shop, job shop, and flow shop problems [188, 189]. However, the lower bound of delay algorithms is only proved to be $\sqrt{2}$ for open shop [189] and $\frac{1+\sqrt{5}}{2}$ for job shop or

flow shop [188]. It is hopeful that a delay algorithm might beat the greedy algorithm in this model. Nevertheless, it is still an open question so far.

5.2 Batch Scheduling

In this problem, jobs arrive over time and can be processed together by a batch processing machine which can handle up to B jobs simultaneously. All jobs in a batch start and complete at the same time. The processing time of a batch is given by the longest processing time of any job in the batch. There are two models considered in the literature: the unbounded model where $B = \infty$ and the bounded model where B is bounded. For both models, problems of minimizing makespan on single or parallel identical machines are widely studied.

5.2.1 Unbounded and Bounded Model on a Single Machine

Suppose that there is a single batch processing machine, Deng et al. [48] and Zhang et al. [191] independently provided an optimal online algorithm (denoted as H^∞) with competitive ratio $\frac{\sqrt{5}+1}{2}$.

Algorithm H^∞

0. Set $t = 0$.
1. Let $U(t)$ be the set of all unscheduled jobs available at time t and J_k be the job with the longest processing time in $U(t)$, compute $\alpha_k = \frac{\sqrt{5}+1}{2}r_k + \frac{\sqrt{5}-1}{2}p_k$ and $s = \max\{t, \alpha_k\}$.
2. In the time interval $[t, s]$, whenever a new job J_h arrives (at time t') with $p_h > p_k$, then reset $k = h$ and reset α_k and s accordingly. Let $U(t') = U(t) \cup \{J_h\}$. Reset $t = t'$.
3. At time s , schedule all jobs in $U(s)$ as a single batch. If some new jobs arrives by $s + p_k$, let $t = s + p_k$; otherwise, wait until a new job arrives and let t be the arrival time of such a job. Go to Step 1.

In fact, the lower bound $\frac{\sqrt{5}+1}{2}$ even holds when $B = 2$, and all jobs have only two distinct arrival times [191]. It seems much more challenging to derive optimal algorithms for the bounded model. When all jobs have exactly two distinct arrival time, Zhang et al. [191] gave an optimal one with competitive ratio $\frac{\sqrt{5}+1}{2}$. When jobs have arbitrary arrival times, Poon and Yu [145] obtained a $\frac{7}{4}$ -competitive algorithm for $B = 2$. For the general B , we note there exist several 2-competitive algorithms. The first one is the greedy-like algorithm GRLPT, which was proposed by Lee and Uzsoy [113], and was shown to be 2-competitive by Liu and Yu [123]. Another two are due to Zhang et al. [191], which can be seen as a generalization of H^∞ . Finally, Poon and Yu [145] claimed all the FBLPT-based (full-batch longest processing time) algorithms, including the above three, can achieve the same competitive ratio of 2.

5.2.2 Unbounded and Bounded Model on Identical Machines

For batch scheduling on m parallel identical machines, researches are mostly focused on the unbounded model. Zhang et al. [191] gave a lower bound $\sqrt[m+2]{2}$ and

an online algorithm $PH(\beta_m)$ with a competitive ratio $1 + \beta_m$, where $0 < \beta_m < 1$ is a solution to the equation $\beta_m = (1 - \beta_m)^{m-1}$. In the same paper, the dense algorithms, which always immediately assign the currently available job to one idle machine as long as there are two or more idle machines, are proposed. Zhang et al. proved that the competitive ratio of any dense algorithm is no smaller than $\sqrt{2}$ and there exists one dense algorithm with competitive ratio $\frac{\sqrt{5}+1}{2}$. In a later paper [192], Zhang et al. gave an optimal algorithm with competitive ratio $1 + \xi_m$ for the special case that all jobs have unit processing time, where ξ_m is the positive solution of the equation $(1 + \xi_m)^{m+1} = \xi_m + 2$.

When jobs have arbitrary processing times, the optimal algorithm for two-machine case has a competitive ratio of $\sqrt{2}$ [139, 173]. The optimal algorithm for the general case is due to Liu et al. [126] and Tian et al. [172] independently, who both proved the competitive ratio to be $1 + \frac{\sqrt{m^2+4-m}}{2}$. In addition, Tian et al. [172] provided a dense algorithm with competitive ratio $\frac{3}{2}$ and showed it is best possible.

For the bounded model, Zhang et al. [192] observed that the optimal algorithm is $\frac{\sqrt{5}+1}{2}$ -competitive if all jobs have equal processing times. Apart from this, neither algorithm results nor lower bounds are found in the literature.

5.2.3 Other Variants

There are some other discussions with respect to the online batch scheduling problems. One interesting variant is allowing to restart a batch, which means a running batch may be interrupted, losing all the work done on it, and the jobs in the interrupted batch are released and become independently unscheduled. Allowing restarts reduces the impact of a wrong decision. For the unbounded model on single machine, Fu et al. [78] showed the competitive ratio of any online algorithm is no less than $\frac{5-\sqrt{5}}{2}$ and gave a $\frac{3}{2}$ -competitive algorithm. Shortly after that, Yuan et al. [185] gave a $\frac{5-\sqrt{5}}{2}$ -competitive algorithm for the problem, and thus it is optimal. Fu et al. [79] studied the same problem with an additional assumption that any job cannot be restarted twice. Optimal algorithm with competitive ratio $\frac{3}{2}$ is obtained. In another paper [81], Fu et al. extended the result to identical machines, where a lower bound of 1.298, as well as a $\frac{1+\sqrt{3}}{2}$ -competitive online algorithm, was given.

Nong et al. [140] introduced *job families* to the batch scheduling problem, which means jobs from different families cannot be processed in the same batch. Online algorithms with competitive ratio 2 are presented for both bounded and unbounded models on single machine. Besides, they showed that the algorithm is best possible for the unbounded model in the sense that jobs consist of an infinite number of families. For the bounded model, there is no online algorithm with competitive ratio smaller than $\max\{\frac{2B}{B+1}, \frac{1+\sqrt{5}}{2}\}$. Fu et al. [80] revisited the unbounded model and gave a best possible algorithm with competitive ratio $\frac{\sqrt{17}+3}{4}$ for the special case of two families of jobs.

Yuan et al. [186] and Li et al. [119] studied a lookahead semi-online model. Yuan et al. [186] considered the unbounded problem on a single batch machine.

With the information of the next longest job at any time t , they presented a best possible online algorithm with competitive ratio $\frac{5-\sqrt{5}}{2}$. If jobs are of unit length and an online algorithm can foresee all the jobs that will arrive in the time segment $(t, t + \beta]$ at any time t , Li et al. [119] presented a best possible online algorithm for the unbounded problem on m parallel-batch machines.

Chen et al. [38] considered a batch scheduling problem on single machine to minimize the total weighted completion times of jobs. They developed a linear time online algorithm with competitive ratio of $\frac{10}{3}$ for the unbounded model and an efficient algorithm with competitive ratio $4 + \epsilon$ for any $\epsilon > 0$ for the bounded model. Also, they observed that there exists a 2.89-competitive and a $(2.89 + \epsilon)$ -competitive randomized algorithm for unbounded and bounded models, respectively.

5.3 Online Scheduling with Machine Eligibility

Online scheduling with machine eligibility constraints has received much attention in recent years. Unlike the classical parallel machine scheduling, job J_j cannot be processed on any one among the machine set \mathcal{M} in this model. Instead, it has a specific set $\mathcal{M}_j \subseteq \mathcal{M}$ that can be assigned to. Set \mathcal{M}_j is so-called the *eligible processing set* of job J_j . Depending on the structure of \mathcal{M}_j , $j = 1, \dots, n$, there are five classes of eligibility constraints that have been studied extensively [116]:

1. Arbitrary eligible processing sets
2. Tree-hierarchical processing sets
3. Grade of Service (GoS) processing sets
4. Interval processing sets
5. Nested processing sets.

With tree-hierarchical processing sets, each machine is represented by a node, and the nodes are connected in the form of a rooted tree. Any job assignable to a node is also assignable to the node's ancestors in the tree. The GoS processing set structure can be seen as a special case of the tree-hierarchical processing set structure where the rooted tree actually forms a chain. Problems with a GoS processing set are also called *online hierarchical scheduling* in the literature. With regard to interval processing sets, job J_j is associated with two integers a_j and $b_j \geq a_j$ such that $\mathcal{M}_j = \{M_{a_j}, M_{a_j+1}, \dots, M_{b_j}\}$. A special case of interval processing set structure is a nested processing set structure, where, for any pair of jobs J_j and J_k , we either have $\mathcal{M}_j \subseteq \mathcal{M}_k$ or $\mathcal{M}_k \subseteq \mathcal{M}_j$ or $\mathcal{M}_j \cap \mathcal{M}_k = \emptyset$. Clearly, the GoS processing set structure is also a special case of nested processing set structure.

Problems with one of the above classes of eligibility constraints will be denoted as “ \mathcal{M}_j ,” “ $\mathcal{M}_j(\text{tree})$,” “ $\mathcal{M}_j(\text{GoS})$,” “ $\mathcal{M}_j(\text{interval})$ ” and “ $\mathcal{M}_j(\text{nested})$ ” in the middle field of the three-field notation, respectively. It is clear that $P||C_{max}$ ($Q||C_{max}$) is a special case of $P|\mathcal{M}_j|C_{max}$ ($Q|\mathcal{M}_j|C_{max}$) with $\mathcal{M}_j = \mathcal{M}$ for all J_j , while $R|\mathcal{M}_j|C_{max}$ is a special case of $R||C_{max}$. However, in the rest of this subsection, we will mostly consider the identical machine environment, unless stated otherwise.

5.3.1 Online Over List

For arbitrary eligibility, Azar et al. [13] contributed the first result. They consider a greedy algorithm, namely, AW , as follows: When a job arrives the algorithm assigns it to an eligible machine that has the smallest current load among all eligible machines. The competitive ratio of AW is shown to be at most $\lceil \log_2 m \rceil + 1$, and the lower bound is proved to be at least $\lceil \log_2(m + 1) \rceil$. Hwang et al. [102] improved the analysis of AW and obtained a slightly smaller competitive ratio of $\log_2 m + 1$. The best known upper and lower bounds for this problem are found by Lim et al. [120] recently. They showed that AW has a competitive ratio no greater than $EU_m = \lfloor \log_2 m \rfloor + \frac{m}{2^{\lfloor \log_2 m \rfloor}}$, and the lower bound of the problem is EL_m , where $EL_1 = 1$ and

$$EL_m = EL_{\theta_m} + \frac{1}{\lfloor \frac{\theta_m}{m-\theta_m} \rfloor}, \theta_m = \operatorname{argmax}_{\lfloor \frac{m}{2} \rfloor \leq i \leq m-1} \left\{ EL_i + \frac{1}{\lfloor \frac{i}{m-i} \rfloor} \right\}.$$

The gap between the two bounds does not exceed 0.1967 and AW is optimal when the number of machines can be written as a sum of two powers of 2. Lee et al. [114] proved the optimal bound of $Q2|\mathcal{M}_j|C_{max}$ is $1 + \min\{s, \frac{1}{s}\}$. Note that for $Q2|\mathcal{M}_j|C_{max}$, the index of M_1 and M_2 cannot be reordered, and thus the speed ratio s can be an arbitrary positive number.

Bar-Noy et al. [16] analyzed online scheduling problem subject to tree-hierarchical eligibility. A 5-competitive online algorithm is proposed. Furthermore, if jobs are of unit size, then the competitive ratio can be improved to 4. Randomized algorithms are also proposed, where the competitive ratios are shown to be $e + 1$ and e , respectively. A lower bound of $1 + \frac{1}{2} \lceil \log_2 m \rceil$ for $Pm|\mathcal{M}_j(\text{nested})|C_{max}$ was given in [120].

There is a large number of researches discussing problems with a GoS eligibility, since it is very natural and has application in the service industry [103]. In [16], Bar-Noy et al. gave an algorithm with competitive ratio $e + 1$ for the general problem $P|\mathcal{M}_j(\text{GoS})|C_{max}$ and showed that it is e -competitive if either jobs have unit size or can be fractionally processed. Moreover, they proved a lower bound of e for the fractional assignment case. However, the lower bound is valid only in the case when the number of machines tends to infinity. If the number of machines is fixed, Tan and Zhang [169] proposed an improved algorithm with competitive ratio GU_m and lower bound GL_m for the fractional assignment case; both are based on the solutions of mathematical programming. The algorithm can be modified to solve the general problem $Pm|\mathcal{M}_j(\text{GoS})|C_{max}$ by using the same technique included in [16]. When the number of machines is small, algorithms for the general problem can be further improved. For $m = 5$ and 4, Tan and Zhang [169] proposed two algorithms with competitive ratios of 2.610 and $\frac{7}{3}$, respectively. Lim et al. [120] improved the two results to 2.501 and 2.294. For $m = 3$, Zhang et al. [193] showed that there exists an optimal algorithm with competitive ratio 2. For $m = 2$, Park et al. [141] and Jiang et al. [109] independently proposed an optimal algorithm with competitive ratio $\frac{5}{3}$. If machines have different speeds, the optimal

bound

$$h_1(s) = \begin{cases} \min\{1 + s, \frac{2+2s+s^2}{1+s+s^2}\}, & 0 < s \leq 1, \\ \min\{\frac{1+s}{s}, \frac{1+3s+s^2}{1+s+s^2}\}, & 1 \leq s < \infty, \end{cases}$$

of $Q2|\mathcal{M}_j(GoS)|C_{max}$ was given by Tan and Zhang [169] (Fig. 8).

If preemption is allowed, Jiang et al. [109] designed a $\frac{3}{2}$ -competitive algorithm for the problem $P2|\mathcal{M}_j(GoS), pmtn|C_{max}$ and showed that it is optimal if idle time is not allowed. For general m , Dósa and Epstein [50] proved that $\frac{2m}{m+1}$ is a lower bound even if idle time is allowed. An algorithm for three machines with matched competitive ratio was also given. In the same paper, they also studied the problem $Q2|\mathcal{M}_j(GoS), pmtn|C_{max}$. An online algorithm with competitive ratio

$$h_2(s) = \max \left\{ \frac{(1+s)^2}{1+s+s^2}, \frac{s(1+s)^2}{1+s^2+s^3} \right\}$$

is proposed and is shown to be optimal. Both algorithms need to introduce idle time. For $Q2|\mathcal{M}_j(GoS), \text{frac}|C_{max}$, Chassid and Epstein [28] designed an optimal algorithm with competitive ratio $\frac{(1+s)^2}{1+s+s^2}$ (Fig. 8).

For semi-online scheduling, Park et al. [141] studied the problem on two machines with the knowledge of the total processing time of jobs, where an optimal algorithm with competitive ratio $\frac{3}{2}$ is presented. Wu et al. [183] considered two semi-online problems on two machines. The first one is known the optimal makespan, for which they showed an optimal algorithm with competitive ratio $\frac{3}{2}$. The second is known the maximum processing time, for which an algorithm with competitive ratio $\frac{\sqrt{5}+1}{2}$, as well as a matched lower bound, is given. Liu et al. [125] considered the semi-online problem on two machines where upper and lower bounds on the processing time of jobs are known in advance. An online algorithm, as well as a lower bound of the problem, was presented. Optimal bound of the problem

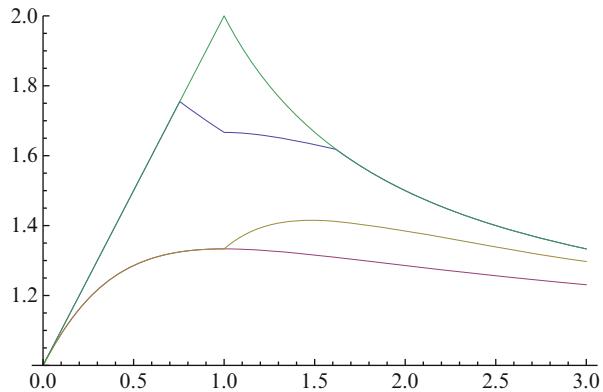
$$h_3(\beta) = \begin{cases} \frac{2\beta+1}{\beta+1}, & 1 \leq \beta < \frac{\sqrt{5}-1}{2}, \\ \frac{\sqrt{5}+1}{2}, & \frac{\sqrt{5}+1}{2} \leq \beta < \frac{3\sqrt{5}-1}{2}, \\ \frac{\beta+2}{\beta^3}, & \frac{3\sqrt{5}-1}{2} \leq \beta < 3, \\ \frac{5}{3}, & r \geq 3, \end{cases}$$

is given by Jiang and Zhang [107], where $\beta = \frac{UB}{LB}$ (Fig. 7).

There are several researches considering problems with two GoS levels (denoted as $2GoS$). In this problem, jobs that can be processed on first k machines are called high level and the other jobs which can run on all m machines are called low level. Jiang [104] first showed that the AW algorithm is at least $(4 - \frac{1}{m})$ -competitive and then provided an online algorithm with a competitive ratio of $\frac{12+4\sqrt{2}}{7} \approx 2.522$. Later, Zhang et al. [194] improved the result to $1 + \frac{m^2-m}{m^2-km+k^2} < \frac{7}{3}$.

For machine covering problems, Chassid and Epstein [28] proved that no algorithm can have a constant competitive ratio even for the most special

Fig. 8 The optimal bounds of $Q2|\mathcal{M}_j|C_{max}$, $Q2|\mathcal{M}_j(GoS)|C_{max}$, $Q2|\mathcal{M}_j(GoS), pmtn|C_{max}$, and $Q2|\mathcal{M}_j(GoS)$, $frac|C_{max}$ (from top to bottom)



problem $Q2|\mathcal{M}_j(GoS)|C_{min}$. For problems $Q2|\mathcal{M}_j(GoS)$, $frac|C_{min}$ and $Q2|\mathcal{M}_j(GoS)$, $sum|C_{min}$, they gave optimal bounds of $\frac{2s+1}{s+1}$ and $\max\{1, s\} + \frac{1}{s}$, respectively. Main results on online over list scheduling with machine eligibility are summarized in Tables 12–14.

5.3.2 Online Over Time

In contrast with problems of online over list, there are very few results concerning problems of online over time. Lee et al. [115, 116] considered the problems allowing restart or preemption of jobs. Various lower bounds, as well as a full study on two machines, are given. They also observed that the general method introduced by Shmoys et al. [157] that uses offline algorithms to obtain online algorithms for problems with job release times can work for scheduling with machine eligibility as well [116], although it produces algorithms with competitive ratio at least 2. All related results are summarized in Table 15.

Wang et al. [178] introduced another interesting problem with respect to scheduling with GoS eligibility, which is so-called *online service scheduling*. The problem arises from the service industry where customers (jobs) are classified as either “ordinary” or “special.” Ordinary customers can be served on any service facility (machines), while special customers can be served only on a flexible service facility. The difference between their model and the problem with two GoS levels is that customers arrive over time and the order of the assignment should be consistent with the order of arrival time of jobs in the same class. For several service policies used in practice, Wang et al. [178] and Wang and Xing [177] analyzed and compared their performance in the sense of competitive ratios.

6 Conclusion

This chapter surveyed different paradigms of online scheduling, including online over list and online over time, and gave a relatively complete picture to the semi-online scheduling problem. Most of detailed algorithms and proofs are not given in

Table 12 Results of online over list scheduling with machine eligibility on two identical and uniform machines

Problems	Optimal bounds	Problems	Optimal bounds
$P2 \mathcal{M}_j C_{max}$	2 [13]	$Q2 \mathcal{M}_j C_{max}$	$1 + \min\{s, \frac{1}{s}\}$ [114]
$P2 \mathcal{M}_j(GoS) C_{max}$	$\frac{5}{3}$ [109, 141]	$Q2 \mathcal{M}_j(GoS) C_{max}$	$h_1(s)$ [168]
$P2 \mathcal{M}_j(GoS), frac C_{max}$	$\frac{4}{3}$ [169]	$Q2 \mathcal{M}_j(GoS), frac C_{max}$	$\frac{(1+s)^2}{1+s+s^2}$ [28]
$P2 \mathcal{M}_j(GoS), pmtn C_{max}^a$	$\frac{4}{3}$ [50]	$Q2 \mathcal{M}_j(GoS), pmtn C_{max}$	$h_2(s)$ [50]
$P2 \mathcal{M}_j(GoS), sum C_{max}$	$\frac{3}{2}$ [141]	$Q2 \mathcal{M}_j(GoS) C_{min}$	∞ [28]
$P2 \mathcal{M}_j(GoS), opt C_{max}$	$\frac{3}{2}$ [183]	$Q2 \mathcal{M}_j(GoS), frac C_{min}$	$\frac{2s+1}{s+1}$ [28]
$P2 \mathcal{M}_j(GoS), max C_{max}$	$\frac{\sqrt{5}+1}{2}$ [183]	$Q2 \mathcal{M}_j(GoS), sum C_{min}$	$\max\{1, s\} + \frac{1}{s}$ [28]
$P2 \mathcal{M}_j(GoS), UB \& LB C_{max}$	$h_3(\beta)$ [107]		

^aIf idle time is not allowed, the optimal bound increases to $\frac{3}{2}$ [109]

Table 13 Results of online over list scheduling with machine eligibility on a small number of identical machines

Problem	$m = 3$		$m = 4$		$m = 5$		$m = 6$		$m = 7$	
	LB	UB	LB	UB	LB	UB	LB	UB	LB	UB
$Pm \mathcal{M}_j C_{max}$ [13, 120]	2.5	2.5	3	3	3.25	3.25	3.5	3.5	3.667	3.75
$Pm \mathcal{M}_j$ (interval) $ C_{max}$ [120]	$1 + \sqrt{2}$		$\frac{3+\sqrt{5}}{2}$		3					
$Pm \mathcal{M}_j$ (nested) $ C_{max}$ [120]	$\frac{7}{3}$		$1 + \sqrt{2}$		$\frac{5}{2}$		3			
$Pm \mathcal{M}_j(GoS) C_{max}$ [120, 169, 193]	2	2	2		2.294	2	2.501	2	2.778	2
$Pm \mathcal{M}_j(GoS),$ $frac C_{max}$ [169]	$\frac{3}{2}$		$\frac{3}{2}$	$\frac{44}{27}$	$\frac{44}{27}$	$\frac{245}{143}$	$\frac{245}{143}$	$\frac{16}{9}$	$\frac{16}{9}$	$\frac{1,071}{586}$
$Pm \mathcal{M}_j(GoS),$ $pmtn C_{max}$ [50]	$\frac{3}{2}$		$\frac{3}{2}$	$\frac{8}{5}$		$\frac{5}{3}$		$\frac{12}{7}$		$\frac{7}{4}$

Table 14 Main results on online over list scheduling with machine eligibility on general number of machines

Problems	Jobs	Lower bound	Upper bound	Gap
$Pm \mathcal{M}_j C_{max}$	None	EL_m	[120]	EU_m [120] ≤ 0.1967
$P \mathcal{M}_j(tree) C_{max}$	None	e	[16]	5 [16] $5 - e$
	$p_j \equiv 1$	e	[16]	4 [16] $4 - e$
$P \mathcal{M}_j(GoS) C_{max}$	None	e	[16]	$e + 1$ [16] 1
	$p_j \equiv 1$	e	[16]	e [16] 0
	frac	e	[16]	e [16] 0
$Pm \mathcal{M}_j(GoS) C_{max}$	None	GL_m	[169]	$GU_m + 1$ [169] 1
	frac	GL_m	[169]	GU_m [169] 0
	2 GoS	2	[104, 194]	$1 + \frac{m^2 - m}{m^2 - km + k^2}$ [194] $\leq 1/3$
	2 GoS, $p_j \equiv 1$	3/2	[193]	3/2 [193] 0

Table 15 Main results on online over time scheduling with machine eligibility

Problem	Jobs	Lower bound	Upper bound
$P r_j, \mathcal{M}_j C_{max}$	None	2	$4 - 2/m$
	res	1.5687	$3 - 1/m$
	pmtn	$1 + \frac{(m-1)(m-2)}{2m(2m-3)}$	2
$P2 r_j, \mathcal{M}_j C_{max}$	None	2	2
	res	1.5687	2
	pmtn	1.125	2
	$p_j = p$	$\frac{\sqrt{5}+1}{2}$	$\frac{\sqrt{5}+1}{2}$
$P r_j, \mathcal{M}_j(GoS) C_{max}$	None	1.5550	$2 + \epsilon$
	res	4/3	$2 + \epsilon$
	pmtn	1.0917	2
$P2 r_j, \mathcal{M}_j(GoS) C_{max}$	None	1.5550	2
	res	4/3	2
	pmtn	1	1
	$p_j = p$	$\sqrt{2}$	$\sqrt{2}$
$P r_j, \mathcal{M}_j(nested) C_{max}$	None	1.5550	$2 + \epsilon$
	res	4/3	$2 + \epsilon$
	pmtn	1.148	2
$P r_j, \mathcal{M}_j(tree) C_{max}$	None	1.5550	$8/3$
	res	4/3	$7/3$

the chapter; please refer to the reference for more details. Online and semi-online scheduling is relatively young when compared to offline scheduling. Yet they have generated tremendous interest and promise to have more results in the future.

Cross-References

- [Advances in Scheduling Problems](#)
- [Greedy Approximation Algorithms](#)
- [Job Shop Scheduling with Petri Nets](#)

Recommended Reading

1. S. Albers, Better bounds for online scheduling. *SIAM J. Comput.* **29**, 459–473 (1999)
2. S. Albers, On randomized online scheduling, in *Proceedings of the 34th ACM Symposium on Theory of Computing* (ACM, New York, 2002), pp. 134–143
3. S. Albers, M. Hellwig, Semi-online scheduling revisited. *Theor. Comput. Sci.* **443**, 1–9 (2012)
4. S. Albers, M. Hellwig, On the value of job migration in online makespan minimization. *Proceedings of the 20th Annual European Symposium on Algorithms*. Lecture Notes in Computer Science (Springer, Berlin/New York, 2012)
5. E.J. Anderson, C.N. Potts, Online scheduling of a single machine to minimize total weighted completion time. *Math. Oper. Res.* **29**, 686–697 (2004)
6. E. Angelelli, A.B. Nagy, M.G. Speranza, Z. Tuza, The on-line multiprocessor scheduling problem with known sum of the tasks. *J. Sched.* **7**, 421–428 (2004)
7. E. Angelelli, M.G. Speranza, Z. Tuza, Semi on-line scheduling on two parallel processors with upper bound on the items. *Algorithmica* **37**, 243–262 (2003)
8. E. Angelelli, M.G. Speranza, Z. Tuza, New bounds and algorithms for on-line scheduling: two identical processors, known sum and upper bound on the tasks. *Discret. Math. Theor. Comput. Sci.* **8**, 1–16 (2006)
9. E. Angelelli, M.G. Speranza, Z. Tuza, Semi on-line scheduling on three processors with known sum of the tasks. *J. Sched.* **10**, 263–269 (2007)
10. J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, O. Waarts, On-line load balancing with applications to machine scheduling and virtual circuit routing. *J. ACM* **44**, 486–504 (1997)
11. Y. Azar, L. Epstein, On-line machine covering, in *Proceeding of the 5th Annual European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 1284 (Springer, Berlin/New York, 1997), pp. 23–36
12. Y. Azar, L. Epstein, On-line machine covering. *J. Sched.* **1**, 67–77 (1998)
13. Y. Azar, J. Naor, R. Rom, The competitiveness of on-line assignments. *Algorithmica* **18**, 221–237 (1995)
14. Y. Azar, O. Regev, On-line bin-stretching. *Theor. Comput. Sci.* **168**, 17–41 (2001)
15. N. Bansal, M. Sviridenko, The Santa Claus problem, in *Proceeding of the 38th ACM Symposium on Theory of Computing* (ACM, New York, 2006), pp. 31–40
16. A. Bar-Noy, A. Freund, J. Naor, Online load balancing in a hierarchical server topology. *SIAM J. Comput.* **31**, 527–549 (2001)
17. Y. Bartal, H. Karloff, Y. Rabani, A better lower bound for on-line scheduling. *Inf. Process. Lett.* **50**, 113–116 (1994)
18. Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an Ancient scheduling problem. *J. Comput. Syst. Sci.* **51**, 359–366 (1995)
19. P. Berman, M. Charikar, M. Karpinski, On-line load balancing for related machines. *J. Algorithms* **35**, 108–121 (2000)
20. A. Borodin, R.E. Yaniv, *Online Computation and Competitive Analysis* (Cambridge University Press, Cambridge/New York, 2005)
21. P. Brucker, *Scheduling Algorithms* (Springer, Berlin/New York, 2007)
22. S.Y. Cai, Semi-online machine covering. *Asia-Pac. J. Oper. Res.* **24**, 373–382 (2007)
23. S.Y. Cai, Q.F. Yang, Semi-online scheduling on two uniform machines with the known largest size. *J. Comb. Optim.* **21**, 393–408 (2011)

24. Q. Cao, Z.H. Liu, Semi-online scheduling with known maximum job size on two uniform machines. *J. Comb. Optim.* **20**, 369–384 (2010)
25. Q. Cao, Z.H. Liu, Online scheduling with reassignment on two uniform machines. *Theor. Comput. Sci.* **411**, 2890–2898 (2010)
26. Q. Cao, Z. Liu, T.C.E. Cheng, Semi-online scheduling with known partial information about job sizes on two identical machines. *Theor. Comput. Sci.* **412**, 3731–3737 (2011)
27. S. Cao, Z.Y. Tan, Online uniform machine covering with the known largest size. *Prog. Nat. Sci.* **17**, 1271–1278 (2007)
28. O. Chassid, L. Epstein, The hierarchical model for load balancing on two machines. *J. Comb. Optim.* **15**, 305–314 (2008)
29. C. Chekuri, R. Motwani, B. Natarajan, C. Stein, Approximation techniques for average completion time scheduling. *SIAM J. Comput.* **31**, 146–166 (2001)
30. B. Chen, A.P.A. Vestjens, Scheduling on identical machines how good is LPT in an on-line setting. *Oper. Res. Lett.* **21**, 165–169 (1997)
31. B. Chen, A. van Vliet, G.J. Woeginger, New lower and upper bounds for on-line scheduling. *Oper. Res. Lett.* **16**, 221–230 (1994)
32. B. Chen, A. van Vliet, G.J. Woeginger, A lower bound for randomized on-line scheduling algorithms. *Inf. Process. Lett.* **51**, 219–222 (1994)
33. B. Chen, A. van Vliet, G.J. Woeginger, An optimal algorithm for preemptive on-line scheduling. *Oper. Res. Lett.* **18**, 127–131 (1995)
34. B. Chen, G.J. Woeginger, A study of on-line scheduling two-stage shops, in *Minimax and Applications*, ed. by D.-Z. Du, P.M. Pardalos (Kluwer Academic, Dordrecht/Boston, 1995), pp. 97–107
35. B. Chen, P.A.V. Arjen, G.J. Woeginger, On-line scheduling of two-machine open shops where jobs arrive over time. *J. Comb. Optim.* **1**, 355–365 (1998)
36. B. Chen, C.N. Potts, G.J. Woeginger, A review of machine scheduling: complexity, algorithms and approximability, in *Handbook of Combinatorial Optimization*, ed. by D.-Z. Du, P.M. Pardalos (Springer, 1998), pp. 21–169
37. B. Chen, D. Du, J. Han, J. Wen, On-line scheduling of small open shops. *Discret. Appl. Math.* **110**, 133–150 (2001)
38. B. Chen, X. Deng, W. Zang, On-line scheduling a batch processing system to minimize total weighted job completion time. *J. Comb. Optim.* **8**, 85–95 (2004)
39. X.Y. Chen, L. Epstein, Z.Y. Tan, Semi-online machine covering for two uniform machines. *Theor. Comput. Sci.* **410**, 5047–5062 (2009)
40. X. Chen, Y. Lan, A. Benko, G. Dosa, X. Han, Optimal algorithms for online scheduling with bounded rearrangement at the end. *Theor. Comput. Sci.* **412**, 6269–6278 (2011)
41. T.C.E. Cheng, H. Kellerer, V. Kotov, Semi-on-line multiprocessor scheduling with given total processing time. *Theor. Comput. Sci.* **337**, 134–146 (2005)
42. T.C.E. Cheng, C.T. Ng, V. Kotov, A new algorithm for online uniform-machine scheduling to minimize the makespan. *Inf. Process. Lett.* **99**, 102–105 (2006)
43. T.C.E. Cheng, H. Kellerer, V. Kotov, Algorithms better than LPT for semi-online scheduling with decreasing processing times. *Oper. Res. Lett.* **40**, 349–352 (2012)
44. Y. Cho, S. Sahni, Bounds for list schedules on uniform processors. *SIAM J. Comput.* **9**, 91–103 (1980)
45. C. Chung, T. Nonner, A. Souza, SRPT is 1.86-competitive for completion time scheduling, in *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms* (ACM, New York, 2010), pp. 1373–1388
46. J.R. Correa, M.R. Wagner, LP-based online scheduling: from single to parallel machines. *Math. Program.* **119**, 109–136 (2009)
47. J. Csirik, H. Kellerer, G. Woeginger, The exact LPT-bound for maximizing the minimum machine completion time. *Oper. Res. Lett.* **11**, 281–287 (1992)
48. X. Deng, C.K. Poon, Y. Zhang, Approximation algorithms in batch processing. *J. Comb. Optim.* **7**, 247–257 (2003)

49. B. Deuermeyer, D. Friesen, M. Langston, Scheduling to maximize the minimum processor finish time in a multiprocessor system. *SIAM J. Discret. Methods* **3**, 190–196 (1982)
50. G. Dósa, L. Epstein, Preemptive scheduling on a small number of hierarchical machines. *Inf. Comput.* **206**, 602–619 (2008)
51. G. Dósa, L. Epstein, Preemptive online scheduling with reordering, in *Proceeding of the 17th Annual European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 5757 (Springer, Berlin/New York, 2009), pp. 456–467
52. G. Dósa, L. Epstein, Online scheduling with a buffer on related machines. *J. Comb. Optim.* **20**, 161–179 (2010)
53. G. Dósa, Y. He, Semi-online algorithms for parallel machine scheduling problems. *Computing* **72**, 355–363 (2004)
54. G. Dósa, M.G. Speranza, Z. Tuza, Two uniform machines with nearly equal speeds: unified approach to known sum and known optimum in semi on-line scheduling. *J. Comb. Optim.* **21**, 458–480 (2011)
55. G. Dósa, Y.X. Wang, X. Han, H. Guo, Online scheduling with rearrangement on two related machines. *Theor. Comput. Sci.* **412**, 642–653 (2011)
56. D. Du, Optimal preemptive semi-online scheduling on two uniform processors. *Inf. Process. Lett.* **92**, 219–223 (2004)
57. T. Ebenlendr, Semi-online preemptive scheduling: study of special cases. *Proceeding of the 8th International Conference on Parallel Processing and Applied Mathematics, Part II*. Lecture Notes in Computer Science, vol. 6068 (Springer, Berlin, 2010), pp. 11–20
58. T. Ebenlendr, J. Sgall, Optimal and online preemptive scheduling on uniformly related machines. *J. Sched.* **12**, 517–527 (2009)
59. T. Ebenlendr, J. Sgall, Semi-online preemptive scheduling: one algorithm for all variants. *Theory Comput. Syst.* **48**, 577–613 (2011)
60. T. Ebenlendr, J. Sgall, A lower bound on deterministic online algorithms for scheduling on related machines without preemption, in *Proceeding of the 9th Workshop on Approximation and Online Algorithms*. Lecture Notes in Computer Science (Springer, Berlin/New York, 2012), pp. 102–108
61. T. Ebenlendr, J. Noga, J. Sgall, G.J. Woeginger, A note on semi-online machine covering. *Proceeding of the 3rd Workshop on Approximation and Online Algorithms*. Lecture Notes in Computer Science, vol. 3879 (Springer, Berlin/New York, 2006), pp. 110–118
62. T. Ebenlendr, W. Jawor, J. Sgall, Preemptive online scheduling: optimal algorithms for all speeds. *Algorithmica* **53**, 504–522 (2009)
63. M. Englert, D. Özmen, M. Westermann, The power of reordering for online minimum makespan scheduling, in *Proceeding of the 48th IEEE Symposium Foundations of Computer Science*, Providence, RI (2008), pp. 603–612
64. L. Epstein, Optimal preemptive scheduling on uniform processors with non-decreasing speed ratios. *Oper. Res. Lett.* **29**, 93–98 (2001)
65. L. Epstein, Bin stretching revisited. *Acta Inform.* **39**, 97–117 (2003)
66. L. Epstein, Tight bounds for on-line bandwidth allocation on two links. *Discret. Appl. Math.* **148**, 181–188 (2005)
67. L. Epstein, L.M. Favrholt, Optimal preemptive semi-online scheduling to minimize makespan on two related machines. *Oper. Res. Lett.* **30**, 269–275 (2002)
68. L. Epstein, L.M. Favrholt, Optimal non-preemptive semi-online scheduling on two related machines. *J. Algorithms* **57**, 49–73 (2005)
69. L. Epstein, R. van Stee, Lower bounds for on-line single-machine scheduling. *Theor. Comput. Sci.* **299**, 439–450 (2003)
70. L. Epstein, D. Ye, Semi-online scheduling with “end of sequence” information. *J. Comb. Optim.* **14**, 45–61 (2007)
71. L. Epstein, A. Levin, Robust algorithms for preemptive scheduling. *Proceeding of the 19th Annual European Symposium on Algorithms*. Lecture Notes in Computer Science, vol. 6942 (Springer, Berlin/New York, 2011), pp. 567–578

72. L. Epstein, J. Noga, S. Seiden, J. Sgall, G.J. Woeginger, Randomized on-line scheduling on two uniform machines. *J. Sched.* **4**, 71–92 (2001)
73. L. Epstein, A. Levin, R. van Stee, Max-min online allocations with a reordering buffer. *SIAM J. Discret. Math.* **25**, 1230–1250 (2011)
74. U. Faigle, W. Kern, G. Turan, On the performance Of on-line algorithm for particular problem. *Acta Cybern.* **9**, 107–119 (1989)
75. A. Fiat, G.J. Woeginger, *Online Algorithms: The State of the Art*. Lecture Notes in Computer Science, vol. 1442 (Springer, Berlin/New York, 1998)
76. A. Fiat, G.J. Woeginger, On-line scheduling on a single machine: minimizing the total completion time. *Acta Inform.* **36**, 287–293 (1999)
77. R. Fleischer, M. Wahl, Online scheduling revisited. *J. Sched.* **3**, 343–353 (2000)
78. R. Fu, J. Tian, J.J. Yuan, Y. Lin, Online scheduling in a parallel batch processing system to minimize makespan using restarts. *Theor. Comput. Sci.* **374**, 196–202 (2007)
79. R. Fu, J. Tian, J.J. Yuan, C. He, On-line scheduling on a batch machine to minimize makespan with limited restarts. *Oper. Res. Lett.* **36**, 255–258 (2008)
80. R. Fu, J. Tian, J.J. Yuan, On-line scheduling on an unbounded parallel batch machine to minimize makespan of two families of jobs. *J. Sched.* **12**, 91–97 (2009)
81. R. Fu, T.C.E. Cheng, C.T. Ng, J.J. Yuan, Online scheduling on two parallel-batching machines with limited restarts to minimize the makespan. *Inf. Process. Lett.* **110**, 444–450 (2010)
82. G. Galambos, G.J. Woeginger, An on-line scheduling heuristic with better worst case ratio than Graham's list scheduling. *SIAM J. Comput.* **22**, 349–355 (1993)
83. M.X. Goemans, M. Queyranne, A.S. Schulz, M. Skutella, Y. Wang, Single machine scheduling with release dates. *SIAM J. Discret. Math.* **15**, 165–192 (2002)
84. T.F. Gonzales, S. Sahni, Preemptive scheduling of uniform processor systems. *J. ACM* **25**, 92–101 (1978)
85. T. Gormley, N. Reingold, E. Torng, J. Westbrook, Generating adversaries for request-answer games, in *Proceeding of the 11th ACM-SIAM Symposium on Discrete Algorithms* (ACM, New York/Society for Industrial and Applied Mathematics, Philadelphia, 2000), pp. 564–565
86. R.L. Graham, Bounds for certain multiprocessing anomalies. *Bell Syst. Tech. J.* **45**, 1563–1581 (1966)
87. R.L. Graham, Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* **17**, 416–429 (1969)
88. R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discret. Math.* **5**, 287–326 (1979)
89. N.G. Hall, M.E. Posner, C.N. Potts, Online scheduling with known arrival times. *Math. Oper. Res.* **34**, 92–102 (2009)
90. F.Q. Han, Z.Y. Tan, Y. Yang, On the optimality of list scheduling for online uniform machines scheduling. *Optim. Lett.* **6**, 1551–1571 (2012)
91. Y. He, The optimal on-line parallel machine scheduling. *Comput. Math. Appl.* **39**, 117–121 (2000)
92. Y. He, G. Dósa, Semi-online scheduling jobs with tightly-grouped processing times on three identical machines. *Discret. Appl. Math.* **150**, 140–159 (2005)
93. Y. He, Y.W. Jiang, Optimal algorithms for semi-online preemptive scheduling problems on two uniform machines. *Acta Inform.* **40**, 367–383 (2004)
94. Y. He, Y.W. Jiang, Preemptive semi-online scheduling with tightly-grouped processing times. *J. Comput. Sci. Technol.* **19**, 733–739 (2004)
95. Y. He, Y.W. Jiang, Optimal semi-online preemptive algorithms for machine covering on two uniform machines. *Theor. Comput. Sci.* **339**, 293–314 (2005)
96. Y. He, Z.Y. Tan, Randomized on-line and semi-on-line scheduling on identical machine. *Asia-Pac. J. Oper. Res.* **20**, 31–40 (2003)
97. Y. He, G.C. Zhang, Semi on-line scheduling on two identical machines. *Computing* **62**, 179–187 (1999)
98. Y. He, H. Zhou, Y.W. Jiang, Preemptive semi-online algorithms with known total size. *Acta Math. Sin. (English Series)* **22**, 587–594 (2006)

99. Y. He, Y.W. Jiang, H. Zhou, Optimal preemptive online algorithm for scheduling with known largest size on two uniform machines. *Acta Math. Sin. (English Series)* **23**, 165–174 (2007)
100. J.A. Hoogeveen, A.P.A. Vestjens, Optimal online algorithms for single-machine scheduling, in *Proceedings of the 5th International Conference on Integer Programming and Combinatorial Optimization*. Lecture Notes in Computer Science, vol. 1084 (Springer, Berlin/New York, 1996), pp. 404–414
101. E. Horwath, E.C. Lam, R. Sethi, A level algorithm for preemptive scheduling. *J. ACM* **24**, 32–43 (1977)
102. H.C. Hwang, S.Y. Chang, Y. Hong, A posterior competitiveness for list scheduling algorithm on machines with eligibility constraints. *Asia-Pac. J. Oper. Res.* **21**, 117–125 (2004)
103. H.C. Hwang, S.Y. Chang, K. Lee, Parallel machine scheduling under a grade of service provision. *Comput. Oper. Res.* **31**, 2055–2061 (2004)
104. Y.W. Jiang, Online scheduling on parallel machines with two GoS levels. *J. Comb. Optim.* **16**, 28–38 (2008)
105. Y.W. Jiang, Y. He, Optimal preemptive online algorithm for scheduling tightly-grouped jobs on two uniform machines. *Asia-Pac. J. Oper. Res.* **23**, 77–88 (2006)
106. Y.W. Jiang, Y. He, Optimal semi-online algorithms for preemptive scheduling problems with inexact partial information. *Acta Inform.* **44**, 571–590 (2007)
107. Y.W. Jiang, A. Zhang, Optimal online algorithms on two hierarchical machines with tightly-grouped processing times, Technical Report, 2010
108. Y. Jiang, Z.Y. Tan, Y. He, Preemptive machine covering on parallel machines. *J. Comb. Optim.* **10**, 345–363 (2005)
109. Y. Jiang, Y. He, C. Tang, Optimal online algorithms for scheduling on two identical machines under a grade of service. *J. Zhejiang Univ. Sci. (A)* **7**, 309–314 (2006)
110. D.R. Karger, S.J. Phillips, E. Torn, A better algorithm for an ancient scheduling problem. *J. Algorithms* **20**, 400–430 (1996)
111. H. Kellerer, V. Kotov, M.G. Speranza, Z. Tuza, Semi on-line algorithms for the partition problem. *Oper. Res. Lett.* **21**, 235–242 (1997)
112. Y. Lan, X. Chen, N. Ding, G. Dosa, X. Han, Online minimum makespan scheduling with a buffer, in *Proceeding of the Joint International Conference on Frontiers in Algorithmics and Algorithmic Aspects in Information and Management*. Lecture Notes in Computer Science, vol. 7285 (2012), pp. 161–171
113. C.Y. Lee, R. Uzsoy, Minimizing makespan on a single batch processing machine with dynamic job arrivals. *Int. J. Prod. Res.* **37**, 219–236 (1999)
114. K. Lee, J.Y.T. Leung, M. Pinedo, Online scheduling on two uniform machines subject to eligibility constraints. *Theor. Comput. Sci.* **410**, 3975–3981 (2009)
115. K. Lee, J.Y.T. Leung, M. Pinedo, Scheduling jobs with equal processing times subject to machine eligibility constraints. *J. Sched.* **14**, 27–38 (2011)
116. K. Lee, J.Y.-T. Leung, M.L. Pinedo, Makespan minimization in online scheduling with machine eligibility. *4OR-A Q. J. Oper. Res.* **8**, 331–364 (2010)
117. J.Y.T. Leung (ed.), *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (CRC, Boca Raton, 2004)
118. R.H. Li, L.J. Shi, An online algorithm for some uniform professor scheduling. *SIAM J. Comput.* **27**, 414–422 (1998)
119. W. Li, Z. Zhang, S. Yang, Online algorithms for scheduling unit length jobs on parallel-batch machines with lookahead. *Inf. Process. Lett.* **112**, 292–297 (2012)
120. K. Lim, K. Lee, S.Y. Chang, Improved bounds for online scheduling with eligibility constraints. *Theor. Comput. Sci.* **412**, 5211–5224 (2011)
121. P. Liu, X. Lu, On-line scheduling of parallel machines to minimize total completion times. *Comput. Oper. Res.* **36**, 2647–2652 (2009)
122. P. Liu, X. Lu, Online scheduling of two uniform machines to minimize total completion times. *J. Ind. Manag. Optim.* **5**, 95–102 (2009)
123. Z.H. Liu, W. Yu, Scheduling one batch processor subject to job release dates. *Discret. Appl. Math.* **105**, 129–136 (2000)

124. M. Liu, C. Chu, Y. Xu, F. Zheng, Online scheduling on m uniform machines to minimize total (weighted) completion time. *Theor. Comput. Sci.* **410**, 3875–3881 (2009)
125. M. Liu, C. Chu, Y. Xu, F. Zheng, An optimal online algorithm for two-machine open shop preemptive scheduling with bounded processing times. *Optim. Lett.* **4**, 227–237 (2010)
126. P. Liu, X. Lu, Y. Fang, A best possible deterministic on-line algorithm for minimizing makespan on parallel batch machines. *J. Sched.* **15**, 77–81 (2012)
127. X. Lu, R.A. Sitters, L. Stougie, A class of on-line scheduling algorithms to minimize total completion time. *Oper. Res. Lett.* **31**, 232–236 (2003)
128. R. Luo, S. Sun, Semi on-line scheduling problem with the largest processing time of jobs on two uniform machines known. *J. Syst. Sci. Math. Sci.* **26**, 729–736 (2006). (in Chinese)
129. R. Luo, S. Sun, W. Huang, Semi on-line scheduling problem for maximizing the minimum machine completion time on two uniform machines. *J. Syst. Sci. Complex.* **19**, 101–107 (2006)
130. N. Megow, *Coping with Incomplete Information in Scheduling Stochastic and Online Models*, Ph.D. thesis, Technische Universität Berlin, October 2006, Published by Cuvillier Verlag Göttingen, Germany, 2007
131. N. Megow, A.S. Schulz, On-line scheduling to minimize average completion time revisited. *Oper. Res. Lett.* **32**, 485–490 (2004)
132. X. Min, J. Liu, Y.Q. Wang, Optimal semi-online algorithms for scheduling problems with reassignment on two identical machines. *Inf. Process. Lett.* **111**, 423–428 (2011)
133. P. Mireault, J.B. Orlin, R.V. Vohra, A parametric worst case analysis of the LPT heuristic for two uniform machines. *Oper. Res.* **45**, 116–125 (1997)
134. R. Motwani, P. Raghavan, *Randomized Algorithms* (Cambridge University Press, Cambridge/New York, 1995)
135. G. Muratore, U.M. Schwarz, G.J. Woeginger, Parallel machine scheduling with nested job assignment restrictions. *Oper. Res. Lett.* **38**, 47–50 (2010)
136. A. Musitelli, J.M. Nicoletti, Competitive ratio of list scheduling on uniform machines and randomized heuristics. *J. Sched.* **14**, 89–101 (2011)
137. C.T. Ng, Z.Y. Tan, Y. He, T.C.E. Cheng, Two semi-online scheduling problems on two uniform machines. *Theor. Comput. Sci.* **410**, 776–792 (2009)
138. J. Noga, S.S. Seiden, An optimal online algorithm for scheduling two machines with release times. *Theor. Comput. Sci.* **268**, 133–143 (2001)
139. Q. Nong, T.C.E. Cheng, C.T. Ng, An improved on-line algorithm for scheduling on two unrestrictive parallel batch processing machines. *Oper. Res. Lett.* **36**, 584–588 (2008)
140. Q. Nong, J.J. Yuan, R. Fu, L. Lin, J. Tian, The single-machine parallel-batching on-line scheduling problem with family jobs to minimize makespan. *Int. J. Prod. Econ.* **111**, 435–440 (2008)
141. J. Park, S.Y. Chang, K. Lee, Online and semi-online scheduling of two machines under a grade of service provision. *Oper. Res. Lett.* **34**, 692–696 (2006)
142. C. Phillips, C. Stein, J. Wein, Minimizing average completion time in the presence of release dates. *Math. Program.* **82**, 199–223 (1995)
143. M.L. Pinedo, *Scheduling, Theory, Algorithms, and Systems* (Springer, New York/London, 2008)
144. K. Pruhs, J. Sgall, E. Torng, Online scheduling, in *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, ed. by J.Y.T. Leung, vol. 15 (Chapman & Hall/CRC, Boca Raton, 2004)
145. C.K. Poon, W. Yu, On-line scheduling algorithms for a batch machine with finite capacity. *J. Comb. Optim.* **9**, 167–186 (2005)
146. M. Queyranne, On the Anderson–Potts single machine on-line scheduling algorithm, manuscript (2001)
147. J.F. Rudin III, R. Chandrasekaran, Improved bound for the online scheduling problem. *SIAM J. Comput.* **32**, 717–735 (2003)
148. L. Schrage, A proof of the optimality of the shortest remaining processing time discipline. *Oper. Res.* **16**, 199–223 (1968)

149. A.S. Schulz, M. Skutella, The power of α -points in preemptive single machine scheduling. *J. Sched.* **5**, 121–133 (2002)
150. A.S. Schulz, M. Skutella, Scheduling unrelated machines by randomized rounding. *SIAM J. Discret. Math.* **15**, 450–469 (2002)
151. S.S. Seiden, Online randomized multiprocessor scheduling. *Algorithmica* **28**, 173–216 (2000)
152. S.S. Seiden, A guessing game and randomized online algorithms, in *Proceedings of the 32nd ACM Symposium on Theory of Computing* (ACM, New York, 2000), pp. 592–601
153. S.S. Seiden, Barely random algorithms for multiprocess scheduling. *J. Sched.* **6**, 309–334 (2003)
154. S.S. Seiden, J. Sgall, G.J. Woeginger, Semi online scheduling with decreasing job sizes. *Oper. Res. Lett.* **27**, 215–221 (2000)
155. J. Sgall, A lower bound for randomized on-line multiprocessor scheduling. *Inf. Process. Lett.* **63**, 51–55 (1997)
156. J. Sgall, On-line scheduling, in *Online Algorithms: The State of the Art*, ed. by A. Fiat, G.J. Woeginger (Springer, Berlin/New York, 1998), pp. 196–231
157. D.B. Shmoys, J. Wein, D.P. Williamson, Scheduling parallel machines online. *SIAM J. Comput.* **24**, 1313–1331 (1995)
158. R. Sitters, Competitive analysis of preemptive single-machine scheduling. *Oper. Res. Lett.* **38**, 585–588 (2010)
159. R. Sitters, Efficient algorithms for average completion time scheduling, in *Proceedings of the 14th International Conference on Integer Programming and Combinatorial Optimization*. Lecture Notes in Computer Science, vol. 6080 (Springer, Berlin, 2010), pp. 411–423
160. N. Sivadasan, P. Sanders, M. Skutella, Online scheduling with bounded migration. *Math. Oper. Res.* **34**, 481–498 (2009)
161. D.D. Sleator, R.E. Tarjan, Amortized efficiency of list update and paging rules. *Commun. ACM* **28**, 202–208 (1985)
162. L. Stougie, A.P.A. Vestjens, Randomized algorithms for on-line scheduling problems how low can't you go. *Oper. Res. Lett.* **30**, 89–96 (2002)
163. Z.Y. Tan, S.J. Cao, Semi-online machine covering on two uniform machines with known total size. *Computing* **78**, 369–378 (2006)
164. Z.Y. Tan, Y. He, Semi-on-line problems on two identical machines with combined partial information. *Oper. Res. Lett.* **30**, 408–414 (2002)
165. Z.Y. Tan, Y. He, Semi-online scheduling problems on two identical machines with inexact partial information. *Theor. Comput. Sci.* **377**, 110–125 (2007)
166. Z.Y. Tan, Y. Wu, Optimal semi-online algorithms for machine covering. *Theor. Comput. Sci.* **372**, 69–80 (2007)
167. Z.Y. Tan, S. Yu, Online scheduling with reassignment. *Oper. Res. Lett.* **36**, 250–254 (2008)
168. Z.Y. Tan, A. Zhang, A note on hierarchical scheduling on two uniform machines. *J. Comb. Optim.* **20**, 85–95 (2010)
169. Z.Y. Tan, A. Zhang, Online hierarchical scheduling: an approach using mathematical programming. *Theor. Comput. Sci.* **412**, 246–256 (2011)
170. J. Tao, Z. Chao, Y. Xi, A semi-online algorithm and its competitive analysis for a single machine scheduling problem with bounded processing times. *J. Ind. Manag. Optim.* **6**, 269–282 (2010)
171. J. Tao, Z. Chao, Y. Xi, Y. Tao, An optimal semi-online algorithm for a single machine scheduling problem with bounded processing time. *Inf. Process. Lett.* **110**, 325–330 (2010)
172. J. Tian, T.C.E. Cheng, C.T. Ng, J.J. Yuan, Online scheduling on unbounded parallel-batch machines to minimize the makespan. *Inf. Process. Lett.* **109**, 1211–1215 (2009)
173. J. Tian, R. Fu, J.J. Yuan, A best online algorithm for scheduling on two parallel batch machines. *Theor. Comput. Sci.* **410**, 2291–2294 (2009)
174. T. Tichý, Randomized on-line scheduling on three processors. *Oper. Res. Lett.* **32**, 152–158 (2004)

175. R. van Stee, J.A. La Poutré, Minimizing the total completion time on-line on a single machine, using restarts. *J. Algorithms* **57**, 95–129 (2005)
176. A.P.A. Vestjens, *On-Line Machine Scheduling*, Ph.D. thesis, Department of Mathematics and Computing Science, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, 1997
177. Z. Wang, W. Xing, Worst-case analysis for on-line service polices. *J. Comb. Optim.* **19**, 107–122 (2010)
178. Z. Wang, W. Xing, B. Chen, On-line service scheduling. *J. Sched.* **12**, 31–43 (2009)
179. Y. Wang, A. Benko, X. Chen, G. Dosa, H. Guo, X. Han, C.S. Lanyi, Online scheduling with one rearrangement at the end: revisited. *Inf. Process. Lett.* **112**, 641–645 (2012)
180. J. Wen, D. Du, Preemptive on-line scheduling for two uniform processors. *Oper. Res. Lett.* **23**, 113–116 (1998)
181. G. Woeginger, A polynomial time approximation scheme for maximizing the minimum machine completion time. *Oper. Res. Lett.* **20**, 149–154 (1997)
182. Y. Wu, Z.Y. Tan, Q.F. Yang, Optimal semi-online scheduling algorithms on a small number of machines, in *Proceedings of the 1st International Symposium on Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, Hangzhou. Lecture Notes in Computer Science, vol. 4614 (2007), pp. 504–515
183. Y. Wu, M. Ji, Q.F. Yang, Optimal semi-online scheduling algorithms on two parallel identical machines under a grade of service provision. *Int. J. Prod. Econ.* **135**, 367–371 (2012)
184. B. Xiong, C. Chung, Completion time scheduling and the WSRPT algorithm, in *Proceedings of the 2nd International Symposium on Combinatorial Optimization*, Athens. Lecture Notes in Computer Science, vol. 7422 (2012), pp. 416–426
185. J.J. Yuan, R. Fu, C.T. Ng, T.C.E. Cheng, A best online algorithm for unbounded parallel-batch scheduling with restarts to minimize makespan. *J. Sched.* **14**, 361–369 (2011)
186. J.J. Yuan, C.T. Ng, T.C.E. Cheng, Best semi-online algorithms for unbounded parallel batch scheduling. *Discret. Appl. Math.* **159**, 838–847 (2011)
187. G.C. Zhang, A simple semi on-line algorithm for $P2||C_{max}$ with a buffer. *Inf. Process. Lett.* **61**, 145–148 (1997)
188. X. Zhang, S. van de Velde, On-line two-machine job shop scheduling with time lags. *Inf. Process. Lett.* **110**, 510–513 (2010)
189. X. Zhang, S. van de Velde, On-line two-machine open shop scheduling with time lags. *Eur. J. Oper. Res.* **204**, 14–19 (2010)
190. G.C. Zhang, D. Ye, A note on on-line scheduling with partial information. *Comput. Math. Appl.* **44**, 539–543 (1999)
191. G.C. Zhang, X. Cai, C.K. Wong, On-line algorithms for minimizing makespan on batch processing machines. *Nav. Res. Logist.* **48**, 241–258 (2001)
192. G.C. Zhang, X. Cai, C.K. Wong, Optimal on-line algorithms for scheduling on parallel batch processing machines. *IIE Trans.* **35**, 175–181 (2003)
193. A. Zhang, Y.W. Jiang, Z.Y. Tan, Optimal algorithms for online hierarchical scheduling on parallel machines, Technical Report, 2009
194. A. Zhang, Y.W. Jiang, Z.Y. Tan, Online parallel machines scheduling with two hierarchies. *Theor. Comput. Sci.* **410**, 3597–3605 (2009)

Online Frequency Allocation and Mechanism Design for Cognitive Radio Wireless Networks*

Xiang-Yang Li and Ping Xu

Contents

1	Introduction	2254
2	System Model, Problem Formulation.....	2257
2.1	Network and System Models.....	2257
2.2	Problem Formulation.....	2258
2.3	Requests Arrival Following a Distribution.....	2260
3	Spectrum Allocation and Mechanism Design with Known Δ , β and γ Only.....	2262
3.1	Upper Bounds on All Online Methods.....	2262
3.2	Efficient Online Allocation Methods.....	2264
3.3	More General Networks.....	2271
3.4	Dealing With Selfish Users.....	2273
3.5	Simulation Studies.....	2276
4	Spectrum Allocation and Mechanism Design with Known Distribution on Requests.....	2282
4.1	Semi-Arbitrary-Arrival Case.....	2282
4.2	Random-Arrival Case.....	2288
4.3	General Conflict Graph Model.....	2292
4.4	Simulation Results.....	2293
5	Literature Reviews.....	2296
6	Conclusion.....	2297
	Recommended Reading.....	2298

*The research of authors is partially supported by NSF CNS-0832120, NSF CNS-1035894, National Natural Science Foundation of China under Grant No. 60828003, program for Zhejiang Provincial Key Innovative Research Team, program for Zhejiang Provincial Overseas High-Level Talents (One-hundred Talents Program). Any opinions, findings, conclusions, or recommendations expressed in this chapter are those of author(s) and do not necessarily reflect the views of the funding agencies (NSF, and NSFC).

X.-Y. Li • P. Xu

Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA

e-mail: xiangyang.li@gmail.com; pxu3@iit.edu; xli@cs.iit.edu

Abstract

In wireless networks, we need to allocate spectrum efficiently. One challenge is that the spectrum usage requests often come in an online fashion. The second challenge is that the secondary users in a cognitive radio network are often selfish and prefer to maximize their own benefits. In this chapter, we review results in the literature that address these two challenges, namely, (1) *TOFU*, semi-truthful online frequency allocation method for wireless networks when primary users can sublease the spectrums to secondary users, and (2) *SALSA*, an online truthful mechanism when the distributions of user arrivals are known.

In *TOFU* protocol [36], secondary users are required to submit the spectrum bid α time slots before its usage. Upon receiving an online spectrum request, the protocol will decide whether to grant its exclusive usage or not, within at least γ time slots of requests' arrival. It is assumed that existing spectrum usage can be preempted with some compensation. For various possible known information, authors in [36] analytically proved that the competitive ratios of their methods are within small constant factors of the optimum online method. Furthermore, in their mechanisms, no selfish users will gain benefits by bidding lower than its willing payment. The extensive simulation results show that they perform almost optimum: their methods get a total profit that is more than 95 % of the offline optimum when γ is about the duration of spectrum usage Δ .

In [39], Xu, Wang, and Li proposed *SALSA*, strategyproof online spectrum admission for wireless networks. They assume that the requests arrival follows the Poisson distribution. Preempting existing spectrum usage is *not* allowed. They proposed two protocols that have guaranteed performances for two different scenarios: (1) *random-arrival case*: the bid values and requested time durations follow some distributions that can be learned, or (2) *semi-arbitrary-arrival case*: the bid values could be arbitrary, but the request arrival sequence is random. They analytically proved that their protocols are strategyproof and are both approximately social efficient and revenue efficient. Their extensive simulation results show that they perform almost optimum. Their method for semi-arbitrary-arrival model achieves social efficiency and revenue efficiency almost 20–30 % of the optimum, while it has been proven that no mechanism can achieve social efficiency ratio better than $1/e \simeq 37\%$. Their protocol for the random-arrival case even achieves social efficiency and revenue efficiency 2–6 times the expected performances by the celebrated VCG mechanism.

1 Introduction

The current fixed spectrum allocation scheme leads to significant spectrum *white spaces*. Several new spectrum management models [28] have been proposed to improve the spectrum usage. One promising technology is the opportunistic spectrum usage. Subleasing is widely regarded as another potential way to share

spectrum, if we ensure that primary users' spectrum usage is not interfered with. Assume that there is a set of n secondary nodes V that will share a spectrum channel. Each secondary user v_i may wish to pay (by leasing) for the usage of the available spectrum.

Previous studies on spectrum assignment (e.g., [25, 38, 42, 43]) often assume that the information of all spectrum requests is known before allocation is made. However, the spectrum usage requests often arrive online and the central authority (typically a primary user) needs to *quickly* decide whether the requests are granted or not. Here, assume that upon receiving an online spectrum request, the protocol needs to decide whether to grant its exclusive usage or not, within at least γ time slots. The rejection typically could not be revoked. Two different approaches of granting the request could be used: *preemptive* and *non-preemptive*. If the requests are non-preemptive, then the central authority could not stop the current running request(s) to satisfy this new request. In the first part of the chapter (Sect. 3), we review the results with preemptive requests where the central authority could potentially terminate the current running request(s) to satisfy this new request to potentially make more profits. Assume that requests terminated cannot be restarted or resumed later (so it is also called “abortion”). Terminating a running request e_i , the central authority has to pay a penalty that is $\beta > 0$ times of the amount paid for the remaining unserved time slots. In the second part of the chapter (Sect. 4), we assume that the central authority could not preempt existing requests to satisfy new requests.

Without any known constraint on requests, it was shown in [35–37] that the competitive ratio of *any* online spectrum allocation method can be arbitrarily bad when preemption is not allowed. This is because any online method has to accept the first coming request (it is possible that no other requests come), therefore miss the following conflicting requests with an arbitrarily large value. In the first part of this chapter, we review the results in [36] on the case when the maximum time requirement is Δ time slots. Always assume that every request arrives at the beginning of a time slot and the number of time slots requested is an integer $t \in [1, \Delta]$. To the best of our knowledge, [35–37] are the first to study online spectrum allocation with cancelation and preemption penalty.

Assume that a sequence of spectrum requests arrives online: the i th request e_i arrives at time a_i , requesting the usage of the spectrum channel for a duration t_i (starting at time s_i), with a bid b_i . The central authority needs to design an online method which, upon receiving the i th bidding request, decides before time $s_i - \alpha + \gamma$ whether the request will be granted or not, and if it is granted, how much will be charged. Here, γ is called *delay factor*. If allowed, terminating a running request e_i , the central authority has to pay some penalty, denoted as $\mu(b_i, \ell_i, t_i) = \beta \frac{\ell_i}{t_i} b_i$, to the user who initiated the request e_i , where $\beta \geq 0$ is a constant. In other words, the penalty is proportional to the amount $\frac{\ell_i}{t_i} b_i$ paid by the requestor for the service of remaining ℓ_i unserved time slots.

The main contributions of the protocol *TOFU* by Xu and Li [36] are as follows. They design several efficient online spectrum allocation methods, called *TOFU*, that

perform almost optimum in all cases. When $0 \leq \beta < 1$, their protocol has a constant competitive ratio depending on β . When $\beta > 1$, they show that the competitive ratio of any online method is at most $O(\frac{\gamma+1}{\Delta})$. They then show that their protocol TOFU achieves the bound $\Omega(\frac{\gamma+1}{\Delta})$. When $\beta = 1$, they show that the competitive ratio of any online algorithm is at most $O((\frac{\gamma+1}{\Delta})^{1/2})$. They then show that their online method TOFU has competitive ratios that match the bounds asymptotically, i.e., $\Theta((\frac{\gamma+1}{\Delta})^{1/2})$. Their extensive simulations show that their methods have competitive ratios almost 95 % in most cases, even compared with the optimum offline spectrum allocation method that knows all future inputs. They also extend their protocol to a more general case in which the secondary users are distributed arbitrarily and the required service regions are heterogeneous disks. They show that the same asymptotic lower bounds and upper bounds apply to this general model.

They also design efficient auction mechanism for spectrum allocation when secondary users are selfish. In such a mechanism, each user will be charged an amount, say \mathbf{p}_i , if its request \mathbf{e}_i is granted. They show that in their online mechanism, to maximize its profit, no secondary user will bid lower than its actual valuation. However, a user may have a chance to improve its profit by bidding higher than its actual valuation. They call this property as *semi-truthful*. Recall that a protocol is called *truthful* in the literature if no user can lie about its bid to improve its profit. Their methods still achieve similar results when the requested region of each user is a sectoral cell.

The main contributions of the *SALSA* protocol [39] are as follows. For both requests arrival models, they designed an admission and charging protocol, called *SALSA* (for strategyproof online spectrum admission), such that the expected social efficiency and revenue are within small constant factors of that of the optimum *offline* method in which all information is known in advance. They also proved that their protocol is strategyproof, which implies that any secondary user cannot improve its profit by lying on his request (bid value and time requirement). They also extended their results to a more general model in which secondary users are distributed in a domain arbitrarily and the conflict graph formed by secondary users is growth-bounded. They showed that their results still hold for this general model.

Their extensive simulations show that their methods perform almost optimum, even compared with the optimum *offline* methods that know all bids in advance. For example, for semi-arbitrary-arrival model, their protocol gets a total profit that is almost 20–30 % of the optimum, while it has been proven in [7] that no mechanism can achieve social efficiency ratio better than $1/e$ in the worst case. The spectrum utilization is also around 20–40 % for this model. For the random-arrival model, their protocol achieves social efficiency and revenue efficiency that is 2–6 times of that by the celebrated VCG mechanism. The spectrum efficiency is around 90 % even for slightly loaded systems.

The rest of the chapter is organized as follows. In Sect. 2, we define in detail the problems to be studied.

Section 3 reviews the *TOFU* protocol when the central authority only knows the time ratio Δ . Specifically, in Sect. 3.1, we review upper bounds on the competitive

ratios of any online allocation methods. Then, we review their solutions in Sect. 3.2 and their analytical proof on the performance bounds of their methods. Their methods for networks with general conflict graphs are discussed in Sect. 3.3. We review their mechanisms to deal with selfish users in Sect. 3.4. We summarize their simulation studies in Sect. 3.5.

Section 4 reviews the protocol SALSA when the central authority knows some distribution information about the future requests. Specifically, in Sect. 4.1, the SALSA protocol with semi-arbitrary-arrival is presented, and Sect. 4.2 presents the SALSA protocol for the case with random-arrival. Section 4.3 reviews the SALSA protocols to networks with growth-bounded conflict graphs. The simulation results are reported in Sect. 4.4.

The related work is reviewed in Sect. 5. We conclude the chapter in Sect. 6.

2 System Model, Problem Formulation

2.1 Network and System Models

Consider a wireless network consisting of some primary users $\mathcal{U} = \{u_1, u_2, \dots, u_p\}$ who hold the right of some spectrum channels for subleasing. There is a central authority who decides the spectrum assignment on behalf of these primary users. In other words, here assume that each primary user will trust the central authority and is satisfied with what he/she will get from the auction based on the mechanism designed. If each primary user has an asking price for its spectrums, we need to design mechanisms (such as double auction in which buyers enter competitive bidders and sellers enter competitive offers simultaneously) that also take into account the selfish behavior of primary users. Note that Wang et al. [31] proposed some truthful double-auction mechanisms with known distributions on bids. The wireless network also consists of some secondary users $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$. Those secondary users may lease the spectrum usage in some region for a time period at any time. In this chapter, assume a simple scenario where there is only one channel available. It is interesting to design allocation and auction schemes when multiple channels are available and a secondary user may request for a number of channels at same time.

Secondary users may reside at different geometry locations. Here, assume that each request is associated with a disk region. Whether a secondary user's request of channels conflicts with the request of another secondary user depends on their locations, in addition to the required time period. This location-dependent conflict will be modeled by a conflict graph $H = (\mathcal{V}, E)$, where two nodes v_i and v_j form an edge (v_i, v_j) in H if and only if they cannot use same channel simultaneously. We will first study the networks with a complete conflict graph and then extend the methods to cases when requested disks are of arbitrary sizes. We will show that the methods have asymptotically optimum performance guarantees in both cases.

2.2 Problem Formulation

Assume that secondary users $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ will ask for spectrum usage on the fly. A user could ask for spectrum usage at different time slots. Let $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_i, \dots$ be the sequence of all requests. Each request $\mathbf{e}_i = (v, b_i, a_i, s_i, t_i)$ is claimed by a secondary user v at time a_i , who bids b_i for the usage of the channel from time s_i to time $s_i + t_i$. Here, we omit the region requirement since this chapter will first address networks with complete conflict graph. For most discussions, we will omit the user v when it is clear from the context. In other words, $\mathbf{e}_i = (b_i, a_i, s_i, t_i)$ denotes the i th request. Obviously, $a_i \leq s_i$ for all requests, which means only spectrum usage in future can be requested. Assume time is slotted and time requirement t_i must be an integer. Assume that every user will ask for the spectrum usage for at most Δ time slots ($t_i \leq \Delta$ for all i). They call Δ the *time bound*.

Since the objective for spectrum auction/lease is to improve spectrum utilization and improve the revenue from the spectrum auction/lease, the central authority (i.e., spectrum auctioneer) can post a requirement that all requests must be submitted in advance of a certain time slots. In this chapter, assume that for every request \mathbf{e}_i , $s_i - a_i \geq \alpha$ for a value α given by the auctioneer. Hereafter, call α the *advance factor*. Each request is claimed at least α time slots before its required usage. Intuitively, a larger advance factor α will give more advantage to the central authority. Later, we will show that the asymptotic performances of the methods do not depend on α as long as $\alpha \geq \gamma$, the delay factor.

The advance factor puts some constraints on the secondary users on when they should submit their bids. To be fair, the system will also put some condition on the central authority about when the central authority should make a decision on each request. In this work, it is always required that the central authority should make a decision on a request within γ time slots of its arrival time. Call γ *delay factor* in the spectrum allocation problem. Obviously, they need $\gamma \leq \alpha$ to make the system meaningful. When $\gamma = 0$, central authority has to make decision *immediately* on request \mathbf{e}_i at time a_i . When $\gamma = \alpha$, central authority could make decision as late as possible. If a request has been rejected, it will never be reconsidered and accepted later. [Figure 1](#) illustrates the model of advance and delay factor. For each request, its arrival time a_i must be before time t_1 in the figure, and central authority must make a decision before time t_2 in the figure.

Since central authority is not able to estimate accurate price for spectrum usage, the model lets the central authority accept a reservation in advance and lets the secondary user get a reasonable guarantee. Crucially, to improve the spectrum usage and revenue, let the central authority terminate a reserved/running request at a later time for new request with higher priority at any time. Cancelation and preemption is necessary to take advantage of a spike in demand and rising prices for spectrum channels and not be forced to sell the spectrum below the market because of an a priori contract. When current spectrum usage is terminated, penalty should be paid to compensate the preemption. Here, assume that the penalty $\mu(b, \ell, t)$ is a linear function of the unfinished time $\ell \leq t$ of that request $\mathbf{e}(v, b, a, s, t)$, i.e., $\mu(b, \ell, t) = \beta \frac{\ell}{t} b$ for a constant $\beta \geq 0$. Notice that the cancelation is modeled as $\ell = t$ since the spectrum usage was not started at all. [Figure 2](#) illustrates the unfinished time ℓ

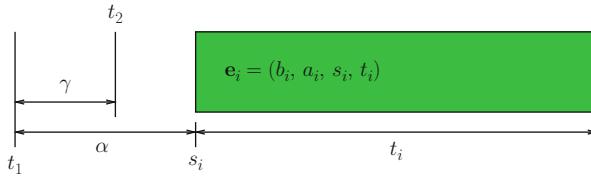


Fig. 1 Advance factor α and delay factor γ

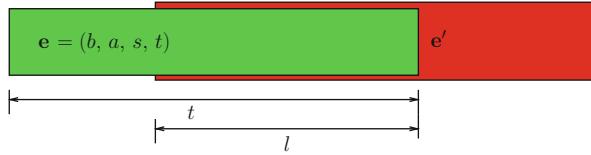


Fig. 2 Unfinished time ℓ when \mathbf{e} is preempted by \mathbf{e}'

Table 1 Symbols used in this chapter

Symb.	Meaning	Symb.	Meaning
\mathbf{e}_i	Request	b_i	Bid value
a_i	Arrival time	s_i	Start time
t_i	Required service time	ℓ_i	Unfinished service time
Δ	Time bound	α	Advance factor
γ	Delay factor	β	Penalty factor

when a request \mathbf{e} is preempted by another request \mathbf{e}' . Here, the constant $\beta \geq 0$ is the *penalty factor*. **Table 1** summarizes the symbols used in this chapter.

As stated before, all requests arrive on the fly, thus any information about the future, e.g., the distribution of future bids, the arrival times, the start times, and the required time durations, are all unknown. The objective of the spectrum allocation is to find an allocation which maximizes the total net profit, i.e., the total payment collected minus the total penalties caused by preemption and cancelation.

To evaluate the performance of an online algorithm, it is usually compared with an optimal *offline* algorithm. The following lemma indicates the hardness of the offline version of the problem.

Lemma 1 *Even knowing all requests, it is NP-hard to find an optimal spectrum allocation that maximizes the total bids of admitted requests.*

Proof Observe that the maximum weighted independent set (MWIS) problem in geometry graph is a special case of the spectrum allocation problem: assume that all spectrum requests are for the same time duration, the graph for MWIS problem is the conflict graph H , and the weight of a node is the bid value of the spectrum request by this node. Recall that MWIS in geometry graph itself is an NP-complete problem [24]. Thus, the lemma follows. \square

For all online spectrum allocation problems studied here, online algorithms with constant competitive ratios will be designed if advance factor is chosen carefully.

The performance of an online algorithm \mathcal{A} for spectrum allocation is measured by its *competitive ratio* $\varrho(\mathcal{A}) = \min_{\mathcal{I}} \frac{\mathcal{A}(\mathcal{I})}{\text{OPT}(\mathcal{I})}$, where \mathcal{I} is *any* possible sequence of requests arrival, $\mathcal{A}(\mathcal{I})$ is the profit produced by *online algorithm* \mathcal{A} on input \mathcal{I} , and $\text{OPT}(\mathcal{I})$ is the profit produced by optimum *offline algorithm* OPT on input \mathcal{I} when the sequence \mathcal{I} is known in advance by OPT. In this work, when we study the competitive ratio, we assume all users are truthful. Given a sequence of spectrum requests, at any time instant t , an online spectrum allocation method only knows all spectrum requests that arrived from time $t - \gamma$ to time slot t . On the other hand, for the same sequence of spectrum requests, although the optimum offline method also has to make decision for a request within γ time slots of its arrival, it does know *all future* requests it will get, thus makes a smarter decision.

As we will see later that the performances of methods and the lower bounds on the performance of any online algorithm vary when the penalty factor β , the advance factor α , and the delay factor γ are different, for certain parameters β , α , and γ , call it (β, α, γ) problem in this chapter.

2.3 Requests Arrival Following a Distribution

In this chapter, two different scenarios will be reviewed. The first scenario assumes that the mechanism designer does not know any distribution information about requests, except the delay factor γ , the penalty factor β , the advance factor α , and the time ratio Δ . [Section 3](#) will focus on this scenario. The second scenario assumes that the mechanism designer does know some distribution information about the arrived requests, such as the expected bid values by all requests. [Section 4](#) will focus on the second scenario.

In the second scenario, it is assumed that the true bid values b_i and time requirement t_i are always independent variables. This assumption is reasonable in practice since b_i is the amount a user is willing to pay for a unit of time. Also assume that arrival time a_i is *memoryless*, i.e., the distribution of arrival time is Poissonian. The arrival rate λ is not necessarily known in advance.

In this chapter, two different cases, random-arrival case, and semi-arbitrary-arrival case, will be studied. The simplest case is *random-arrival* case, in which the true bid values b_i and required time slots t_i are sampled *i.i.d.* from some *stationary* distributions that are not necessarily known to the central authority in advance. For example, true bid values b_i of all requests could be randomly and independently drawn from a normal distribution with mean μ and variance σ , and the values of μ and σ could be known or unknown to the central authority in advance. Or the central authority may also know that the arrival of requests follows the Poisson process with a rate λ , although the value of λ may be unknown to the central authority in advance.

Although motivated by this *i.i.d.* model, it may be more robust to work in *semi-arbitrary-arrival model*. This model also implies some fundamental limits to this online spectrum allocation problem. Assume that bid values b_i , and required time durations t_i are generated by an adversary who wants to beat these protocols. Let n

be the number of requests that will be posted. The adversary generates a (potentially random) set $\mathcal{A} = \{a_1, a_2, \dots, a_n\}$ of arrival times that are produced from Poisson distribution with unknown rate and a set $\mathcal{T} = \{t_1, t_2, \dots, t_n\}$ of required time durations that are produced from an unknown distribution. And the adversary could generate an arbitrary set $\mathcal{B} = \{b_1, b_2, \dots, b_n\}$ which is not necessarily sampled from any stationary distributions. After these three sets are generated, the bid values, arrival times, and required time durations are matched together using a *random permutation*, which is *not* controlled by the adversary. Call this the *random ordering hypothesis*. The hypothesis is reasonable since we concern the *expected* performance of these mechanisms. When this matching is also controlled by the adversary, we can present examples of requests such that, for *any* online spectrum allocation and auction method, the expected social efficiency or revenue could be arbitrarily bad comparing with that of *offline* methods.

The protocols need to compute how much each request needs to pay. When a request is accepted, the secondary user who issued the request will be granted the usage of channel, and will need to pay some monetary value for this. We will design mechanisms to compute the payment by secondary users. Let \mathbf{v}_i be the true valuation of the spectrum usage requested by \mathbf{e}_i and \mathbf{p}_i be the payment that request \mathbf{e}_i should pay. Then, the final profit (or the utility) of request \mathbf{e}_i is $\mathcal{U}(i) = x_i \cdot \mathbf{v}_i - \mathbf{p}_i$. When \mathbf{e}_i is admitted, $x_i = 1$; otherwise, $x_i = 0$. Notice that $b_i t_i = \mathbf{v}_i$ when secondary user issued request \mathbf{e}_i truthfully.

In the online auction, the secondary users seek to place requests to maximize their individual utility in equilibrium. As a standard approach, in this chapter, we will only focus on direct revelation mechanisms in which secondary users directly submit their bid values and time requirement (although not necessarily truthfully). We assume that each secondary user cannot announce an earlier arrival time than its true arrival. At every time t , given the set of submitted requests θ that are not processed yet, the central authority decides an allocation $x_i(\theta, t) \in \{0, 1\}$ and a payment $\mathbf{p}_i(\theta, t)$ for request \mathbf{e}_i . We place the following natural conditions on the allocation and payments: allocations cannot be revoked, so $x_i(\theta, t)$ is a nondecreasing function of t . The allocation and payments must be online computable, in that they can only depend on information available at time t . We are interested in mechanisms with dominant-strategy equilibrium, such that every secondary user has a single optimal strategy regardless of the strategies of others. This is a particularly robust solution concept. Notice that a secondary user may lie lower or lie higher its bid value b_i (compared with its true bid value b'_i) and can only lie its time requirement t_i to a larger value.

Definition 1 A mechanism is value-strategyproof and time-strategyproof if a secondary user's dominant strategy is to report its true bid value (called *value-SP*) and true time requirement (called *time-SP*).

Competitive ratio is always used to evaluate the performance of an online algorithm. Most of the previous work focused on the worst-case competitive ratio of the online algorithm; however, the probability that the worst case will happen is very small in practice. Thus, we study the expected competitive ratio instead, which

could better reflect the practical performance of an online algorithm. The *average competitive ratio* of an online algorithm is defined as the ratio between its average performance and the average performance of the optimal *offline* algorithm for *all* possible inputs. We focus on the *average competitive ratio* since it is easy to prove that without any known constraint, the worst-case competitive ratio of any online method can be arbitrarily bad.

To evaluate the performance of an online mechanism, we usually compare it with *offline Vickrey* mechanism which will maximize the total social efficiency (but not necessarily strategyproof in the online model). Notice that *offline* mechanism knows all future inputs before it makes decisions, which surely improves the performance significantly. The performance of an online mechanism \mathcal{M} is measured by its *social efficiency* $\text{Eff}_s(\mathcal{M})$ and *revenue efficiency* $\text{Eff}_r(\mathcal{M})$. The *social efficiency* of mechanism \mathcal{M} is defined as the total true valuations of all winners, i.e., $\text{Eff}_s(\mathcal{M}) = \sum_i x_i v_i$. For a mechanism \mathcal{M} , let \mathbf{p}_i be the payment collected from secondary request \mathbf{e}_i . And the *revenue efficiency* of mechanism \mathcal{M} is simply $\text{Eff}_r(\mathcal{M}) = \sum_i \mathbf{p}_i$. The benchmark mechanisms that we adopt for the purpose of competitive analysis are the offline Vickrey mechanisms, denoted as *Vick*. Notice that for offline Vickrey mechanism, it will always select a set of winners that maximize the social efficiency.

Definition 2 The *social efficiency competitiveness* of an admission mechanism \mathcal{M} is defined as the ratio of the total valuations of winners under mechanism \mathcal{M} over the total valuations of winners under *offline Vickrey* mechanism. In other words, $\text{Eff}_s(\mathcal{M}, \text{Vick}) = \text{Eff}_s(\mathcal{M})/\text{Eff}_s(\text{Vick})$. Similarly, the *revenue efficiency competitiveness* of mechanism \mathcal{M} is defined as the ratio of the total payments of winners under mechanism \mathcal{M} over the total payments of winners under *offline Vickrey* mechanism. In other words, $\text{Eff}_r(\mathcal{M}, \text{Vick}) = \text{Eff}_r(\mathcal{M})/\text{Eff}_r(\text{Vick})$. We say that mechanism \mathcal{M} is $\frac{1}{\text{Eff}_s(\mathcal{M}, \text{Vick})}$ -competitive for social efficiency and $\frac{1}{\text{Eff}_r(\mathcal{M}, \text{Vick})}$ -competitive for revenue efficiency.

The objective is to design *online strategyproof spectrum admission mechanisms* in different cases which maximize the *expected* social efficiency and revenue efficiency. The expectations are computed over *all* possible permutations of bid values b_i and *all* possible permutations of time requirements.

3 Spectrum Allocation and Mechanism Design with Known Δ , β and γ Only

3.1 Upper Bounds on All Online Methods

In this section, we present upper bounds on the competitive ratios of online allocation methods in cases depending on whether β is less than 1, equal to 1, or larger than 1. These bounds will be proved to be almost tight later.

3.1.1 Upper Bound for $(\beta < 1, \gamma, \alpha)$ Problem

We first show upper bounds on the competitive ratios when $\beta < 1$. Note that in this case, we assume that every preempted job must be executed for at least one time slot.

The main approach of the proofs in this section is *adversary argument*: an adversary carefully constructs sequences of requests to make the performance of online algorithms as bad as possible. The trick is that the adversary can choose the future requests based on the current decision of online algorithms. Then, no matter what decision is made, certain performance cannot be achieved.

Theorem 1 *Let $\theta = \frac{1}{c\beta}$. No online algorithm has a competitive ratio $> \theta$ for $(\beta < 1, \gamma, \alpha)$ problem when $\gamma \leq \frac{\beta}{(c+1)(2+\beta)}\Delta$ for $c > 1/\beta$.*

3.1.2 Upper Bound for $(\beta = 1, \gamma, \alpha)$ Problem

When the penalty factor $\beta = 1$, there are two different cases: $\gamma = o(\Delta)$ or $\gamma = \Omega(\Delta)$.

- (1) When $\gamma = o(\Delta)$, i.e., $\lim \frac{\gamma}{\Delta} = 0$, we first show that no online algorithm can achieve competitive ratio more than $\sqrt[3]{2(\gamma+1)}\Delta^{-\frac{1}{3}}$ for $(1, \alpha, \gamma)$ problem. Then, we improve this bound to $\sqrt{2(\gamma+1)}\Delta^{-\frac{1}{2}}$.
- (2) When $\gamma = \Omega(\Delta)$, we show that $(1, \alpha, \gamma)$ problem cannot have a competitive ratio $1 - \epsilon$ for an arbitrary $\epsilon > 0$.

Theorem 2 ([37]) *No online algorithm has a competitive ratio $> \sqrt[3]{2(\gamma+1)}\Delta^{-\frac{1}{3}}$ for $(\beta = 1, \gamma, \alpha)$ problem when $\gamma = o(\Delta)$.*

Theorem 3 ([37]) *No online algorithm has competitive ratio $> \frac{1}{c}$ for $(\beta = 1, \gamma, \alpha)$ problem, where $\sqrt{2(\gamma+1)}\Delta^{-\frac{1}{2}} < \frac{1}{c} \leq \sqrt[3]{2(\gamma+1)}\Delta^{-\frac{1}{3}}$ for $(\beta = 1, \gamma, \alpha)$ problem when $\gamma = o(\Delta)$.*

Theorem 4 ([37]) *No online algorithm has a competitive ratio $\geq 1 - \epsilon$ for an arbitrary small $\epsilon > 0$, for $(\beta = 1, \gamma, \alpha)$ problem when $\gamma = \Omega(\Delta)$.*

3.1.3 Upper Bound for $(\beta > 1, \gamma, \alpha)$ Problem

For $(\beta > 1, \gamma, \alpha)$ problem, we show that upper bound of competitive ratios is $O(\frac{\gamma+1}{\Delta})$ when $\gamma = o(\Delta)$.

Theorem 5 ([37]) *No online algorithm has competitive ratio more than $\frac{2\beta}{(\beta-1)^2}\frac{\gamma+1}{\Delta}$ for $(\beta > 1, \gamma, \alpha)$ problem, when $\gamma = o(\Delta)$.*

A surprising result from the analysis in this section is that the performance upper bounds do not depend on the *advance factor* α . In other words, no matter how many time slots the secondary users claim their requests in advance, the theoretical upper bounds will not be improved asymptotically if the *delay factor* γ does not change.

3.2 Efficient Online Allocation Methods

In this section, we present several methods for efficient online spectrum allocation. We then analytically study the performances of the methods and prove that they have asymptotically best competitive ratios. All results in this section assume that the conflict graph of the network is a complete graph.

3.2.1 Method \mathcal{K} for $(\beta < 1, \gamma, \alpha)$ Problem

When $\beta < 1$, whenever we admit a request e_i at time t , even we preempt it at next time slot $t + 1$, we still receive a profit $(1 - \beta)b_i$. Thus, we will adopt the following strategy \mathcal{K} : at any time t , let e_i be the request from the set of all currently available requests, denoted as $\mathcal{R}_a(t)$, with the largest bid b_i among all requests whose starting time is $t - \gamma + \alpha$. We admit request e_i at time t . It is easy to prove the following Lemma.

Lemma 2 *Strategy \mathcal{K} is at least $(1 - \beta)$ -competitive.*

3.2.2 Method \mathcal{G} for $(\beta = 1, \gamma, \alpha)$ Problem

We then consider the case $\beta = 1$, i.e., the preemption penalty is exactly the remaining portion of the bid. Since we do not know anything about the future, to maximize the total profits, we shall accept request with large bid or large bid per time slot (called *density*) intuitively. The main difficulty is the trade-off between requests with large bid and requests with large *density*. The results on upper bounds of competitive ratio in Sect. 3.1 illustrate some intuitions. In this and following subsection, we design online algorithms whose competitive ratios match corresponding upper bounds.

Consider current time t , and let $\mathcal{R}_a(t)$ be all spectrum requests submitted before time t , which are not processed. Based on the delay requirement, we know that all requests in $\mathcal{R}_a(t)$ must be submitted during the time slots $[t - \gamma, t]$, and their starting times must be in the time interval $[t - \gamma + \alpha, t + \alpha]$. Among all spectrum requests in $\mathcal{R}_a(t)$, let $\mathcal{R}(t) \subseteq \mathcal{R}_a(t)$ be all spectrum requests whose starting times are in the time interval $[t, t + \gamma]$. Recall that we always have $\gamma \leq \alpha$. The online algorithms will only make decisions at time t using the information about requests $\mathcal{R}(t)$, although it knows a superset of requests $\mathcal{R}_a(t)$. See Fig. 3 for illustration, where $\mathcal{R}_a(t) = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_5\}$ and $\mathcal{R}(t) = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. We will show that this method can achieve a competitive ratio that is already asymptotically optimum. In other words, knowing the information α will not enable us to get a method with a better competitive ratio asymptotically.

Candidate Requests Set

For requests $\mathcal{R}(t)$, we will find some subsets, called *candidate requests set*, using dynamic programming to optimize some objective functions. This method will then make decisions on whether to admit these subsets of requests under some conditions involving the currently running spectrum usages.

Fig. 3 All requests $\mathcal{R}_a(t)$ and a subset of requests $\mathcal{R}(t)$ with starting time in $[t, t + \gamma]$ (denoted by green color)

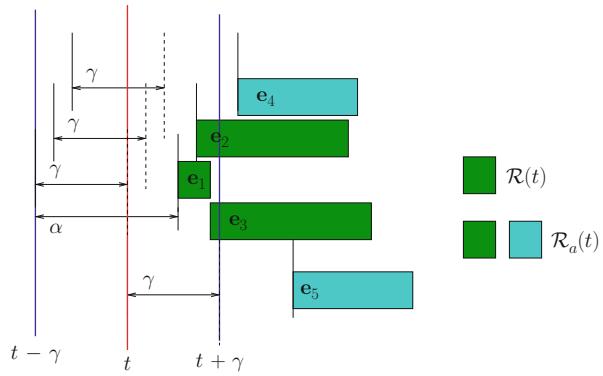
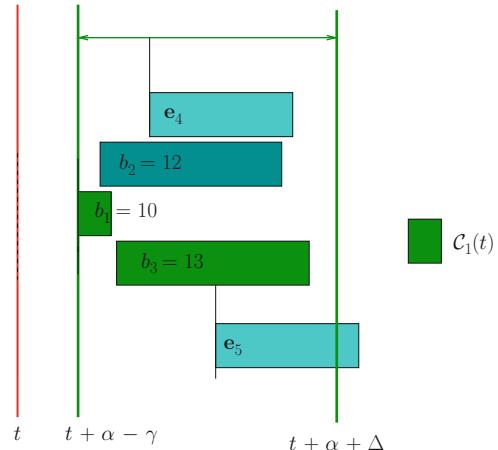


Fig. 4 Strong candidate set $\mathcal{C}_1(t)$ at time t (color green)



Definition 3 Candidate Requests Set:

- (1) A *strong candidate requests set* at time t , denoted as $\mathcal{C}_1(t)$, is a subset of requests from $\mathcal{R}(t)$ that has the largest total bids if $\mathcal{C}_1(t)$ were allowed to run from time $t - \gamma + \alpha$ to time at most $t + \alpha + \Delta$. See [Fig. 4](#) for illustration where $\mathcal{C}_1(t) = \{e_1, e_3\} \subseteq \mathcal{R}(t)$ since the total profit during time interval $[t - \gamma + \alpha, t + \alpha + \Delta]$ is maximized. We abuse the notation little bit here by also letting $\mathcal{C}_1(t)$ denote the profit made by $\mathcal{C}_1(t)$.

Also let $\mathcal{P}(\mathcal{C}_1(t), t')$ denote the profit made from $\mathcal{C}_1(t)$ if $\mathcal{C}_1(t)$ are admitted and then possibly being preempted at a time slot $t' \in [t - \gamma + \alpha, t + \alpha + \Delta]$.

- (2) A *weak candidate requests set* at time t , denoted as $\mathcal{C}_2(t)$, is a subset of requests from $\mathcal{R}(t)$ that has the largest total bids if $\mathcal{C}_2(t)$ were allowed to run during time interval $[t - \gamma + \alpha, t + \alpha]$ (assume they are preempted at time $t + \alpha + 1$). We abuse the notation little bit here by also letting $\mathcal{C}_2(t)$ denote the profit made by $\mathcal{C}_2(t)$.

Algorithm 1 Find $\mathcal{C}_1(t)$ by dynamic programming

Input: $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{n(t)}$ such that $s_1 \leq s_2 \leq \dots \leq s_{n(t)}$.

Output: Strong candidate set $\mathcal{C}_1(t)$.

```

1: for  $i = 1$  to  $n(t)$  do
2:   for  $j = 1$  to  $i$  do
3:      $\mathcal{Q}(j, s_i) = \max_{k \in [1, n]} \{\mathcal{Q}(k, s_j) + b_j - \mu(\mathbf{e}_k, s_j)\}$ 
4:    $\mathcal{C}_1(t) = \max_{k \in [1, n]} \{\mathcal{Q}(k, s_{n(t)})\}$ 

```

Briefly speaking, $\mathcal{C}_1(t)$ maximizes the total bids at time t , and $\mathcal{C}_2(t)$ maximizes the total bid per time slot in the predictable future (since the requests are claimed in advance). Observe that the starting times of candidate sets are in the interval $[t, t + \alpha]$. Since $\gamma \leq \alpha$, at time t , we should know all requests whose starting times are from t to $t + \gamma$. So we could find the *candidate requests set* $\mathcal{C}_1(t)$ and $\mathcal{C}_2(t)$ by dynamic programming as following.

Consider requests $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n$ in the increasing order of their starting times, i.e., $t - \gamma + \alpha \leq s_1 \leq s_2 \leq \dots \leq s_n \leq t + \alpha$. When we compute the strong candidate set $\mathcal{C}_1(t)$, we will *not* consider any requests that are currently running or have been rejected before. Additionally, for the scheduling of requests in $\mathcal{C}_1(t)$, we may use one request to preempt another request in $\mathcal{C}_1(t)$. Let $\mathcal{Q}(j, s_i)$ denote the maximum possible profit made during time interval $[t, t + \gamma + \Delta]$ from $\mathcal{R}(t)$ when \mathbf{e}_j is the last accepted request before or at time s_i . If the starting time s_j of \mathbf{e}_j satisfies $s_j > s_i$, $\mathcal{Q}(i, s_j) = 0$ since it is not a feasible solution. Then,

$$\mathcal{C}_1(t) = \max_{k \in [1, n]} \mathcal{Q}(k, s_n).$$

For each i and j such that $i \geq j$, we have

$$\mathcal{Q}(j, s_i) = \max_{k \in [1, n]} \{\mathcal{Q}(k, s_j) + b_j - \mu(\mathbf{e}_k, s_j)\}.$$

Here, $\mu(\mathbf{e}_k, s_j)$ is the paid penalty when request \mathbf{e}_k is preempted at time s_j .

Algorithm 1 summarizes the method finding $\mathcal{C}_1(t)$. For each i and j , the algorithm takes $O(n(t))$ time to compute $\mathcal{Q}(j, s_i)$. Therefore, it takes $O(n(t)^3)$ time to find $\mathcal{C}_1(t)$ where $n(t)$ is the total number of spectrum requests in $\mathcal{R}(t)$. The method can be easily extended to find $\mathcal{C}_2(t)$ with same running time.

The Greedy Algorithm \mathcal{G}

We then propose an online algorithm \mathcal{G} . At each time t , the input of our algorithm \mathcal{G} is $\mathcal{R}_a(t)$. Algorithm \mathcal{G} should decide whether a request that arrived at time slot $t - \gamma$ will be accepted or rejected. The basic idea of our method is as follows.

Assume that time 0 is the reference point for time. First, before time γ , we do not need to make decisions on any arrived requests. At time γ , we need to decide whether to accept/reject requests that arrive at time 0. If channel is not occupied, we will simply accept the request from $\mathcal{C}_1(\gamma)$ with the earliest starting time being α , if there is any. We call $\mathcal{C}_1(\gamma)$ as *virtual meta job* requests currently being accepted.

For any time $t > \gamma$, our method will choose either a request from the strong candidate request $\mathcal{C}_1(t)$, or a request from weak candidate request $\mathcal{C}_2(t)$, or continue to run the request (which could be empty) chosen previously. The method deals with three complementary cases about the channel status at time $t - \gamma + \alpha$, separately.

Case 1: Non-occupied Channel If the channel will be *empty* at time $t - \gamma + \alpha$, we find $\mathcal{C}_1(t)$ with the maximum overall profit in $\mathcal{R}(t)$. We accept the request in $\mathcal{C}_1(t)$ with starting time $t - \gamma + \alpha$, and we say that the channel is being used by candidate requests set $\mathcal{C}_1(t)$. In other words, here we treat $\mathcal{C}_1(t)$ as a large *virtual meta* spectrum request. Note that at current time slot t , we only admit the first spectrum request from $\mathcal{C}_1(t)$ while leave the admission decisions on other requests in $\mathcal{C}_1(t)$ *pending*. Whether these pending “admitted” requests will be actually admitted depending on future coming requests, if future coming requests are better, we will preempt this virtual meta request $\mathcal{C}_1(t)$.¹ Thus, those preempted pending admissions from $\mathcal{C}_1(t)$ will not be processed at all.

Case 2: Occupied by Weak Candidate Requests If the channel will be used by a *weak candidate requests set* $\mathcal{C}_2(t)$ at time $t - \gamma + \alpha$ and this candidate requests set $\mathcal{C}_2(t)$ *weakly preempted* (exact definitions will be given later) some other candidate requests set before, all requests from $\mathcal{R}_a(t)$ submitted at time $t - \gamma$ will *not be admitted*.

Case 3: Occupied by Strong Candidate Requests The last case is that the channel will be used by a *strong candidate requests set* $\mathcal{C}_1(t)$ at time $t - \gamma + \alpha$. Assume the request to be run at time $t - \gamma + \alpha$ is \mathbf{e}_j from some virtual candidate requests set $\mathcal{C}_1(t_1)$. There are three possible steps in this case.

- (1) We first find the strong candidate requests set $\mathcal{C}_1(t)$. The first request $\mathbf{e}_i \in \mathcal{C}_1(t)$ such that $s_i = t - \gamma + \alpha$ will be accepted only if

$$\mathcal{C}_1(t) \geq c_1 \cdot \mathcal{C}_1(t_1), \quad (1)$$

where $c_1 > 1$ is a constant parameter. In other words, we use a virtual meta request $\mathcal{C}_1(t)$ to replace another virtual request $\mathcal{C}_1(t_1)$ if the potential profit from new virtual meta request is sufficiently larger. We call it a *strong-preemption*.

- (2) If strong-preemption cannot be applied, we find $\mathcal{C}_2(t)$. The request $\mathbf{e}_i \in \mathcal{C}_2(t)$ such that $s_i = t - \gamma + \alpha$ will be accepted only if

$$\mathcal{C}_2(t) + \mathcal{P}(\mathcal{C}_1(t_1), t - \gamma + \alpha) \geq c_2 \cdot \mathcal{C}(t_1), \quad (2)$$

where $c_2 > 0$ is an adjustable control parameter. In other words, we use a virtual candidate requests set $\mathcal{C}_2(t)$ to replace another virtual candidate requests set $\mathcal{C}_1(t_1)$ if the profit from this new meta request and the profit from the meta request $\mathcal{C}(t_1)$ being preempted at future time $t - \gamma + \alpha$ are at least a constant

¹Preempting $\mathcal{C}_1(t)$ at a time, say t' , will be implemented by preempting the requests from $\mathcal{C}_1(t)$ that are supposed to be running at and after time t' .

Algorithm 2 Online spectrum allocation \mathcal{G}

Input: A constant parameter $c_1 > 1$, an adjustable control parameter $c_2 > 0$, γ , α , Δ , $\mathcal{R}_a(t)$, $\mathcal{R}(t)$, $\mathcal{C}_1(t)$, and $\mathcal{C}_2(t)$.

Current candidate requests set \mathcal{C} from time $t' < t$. Here, $\mathcal{C} = \mathcal{C}_1(t')$ if $\mathcal{C}_1(t')$ strongly preempted others, or $\mathcal{C} = \mathcal{C}_2(t')$ if $\mathcal{C}_2(t')$ strongly preempted others.

Output: Whether requests submitted at time $t - \gamma$ will be admitted and new *current candidate requests set \mathcal{C}* .

```

1: if  $\mathcal{C} = \mathcal{C}_2(t')$  then
2:   if  $t - t' \geq \gamma$  then
3:      $\mathcal{C} = \emptyset$ ;
4:   else
5:     Accept request  $\mathbf{e}_i \in \mathcal{C}_2(t)$  such that  $s_i = t - \gamma + \alpha$ .
6: if  $\mathcal{C} = \mathcal{C}_1(t')$  or  $\emptyset$  then
7:   if  $\mathcal{C}_1(t) \geq c_1 \cdot \mathcal{C}_1(t')$  then
8:      $\mathcal{C} = \mathcal{C}_1(t)$ ;
9:     Accept request  $\mathbf{e}_i \in \mathcal{C}_1(t)$  such that  $s_i = t - \gamma + \alpha$ .
10:  else if  $\mathcal{C}_2(t) + \mathcal{P}(\mathcal{C}_1(t'), t) \geq c_2 \cdot \mathcal{C}_1(t)$  then
11:     $\mathcal{C} = \mathcal{C}_2(t)$ ;
12:    Accept request  $\mathbf{e}_i \in \mathcal{C}_2(t)$  such that  $s_i = t - \gamma + \alpha$ .
13:  else
14:    Accept request  $\mathbf{e}_i \in \mathcal{C}_1(t')$  such that  $s_i = t - \gamma + \alpha$ .

```

c_2 fraction of the profit by keeping running the meta request $\mathcal{C}(t_1)$. We call it a *weak-preemption*. In this subcase, all requests in $\mathcal{C}_2(t)$ will be accepted and the last request will be terminated at time $t + \alpha + 1$ automatically.

- (3) If the weak-preemption cannot be applied also, we accept the request in the previously used candidate requests set $\mathcal{C}(t_1)$, whose starting time is $t - \gamma + \alpha$ (if there is any) or continue the request \mathbf{e}_j that will run through the time slot $t - \gamma + \alpha$.

[Algorithm 2](#) summarizes the online spectrum allocation method \mathcal{G} .

Theoretical Performance Study

We then show that the algorithm \mathcal{G} is asymptotically optimal if we choose constant c_1 and control parameter c_2 carefully. We use $\mathcal{G}(\mathcal{I})$ to denote the profit made on requests sequence \mathcal{I} by the algorithm \mathcal{G} . Also use $\text{OPT}(\mathcal{I})$ to denote the profit made by the optimal offline algorithm. To analyze the performance of the method, we first give a definition *candidate sequence*.

Definition 4 Candidate Sequence: A *candidate sequence* is a sequence of *candidate requests sets* $\mathcal{C}_1(t_1), \mathcal{C}_1(t_{i+1}), \dots, \mathcal{C}_1(t_{j-1}), \mathcal{C}_2(t_j)$ or $\mathcal{C}_1(t_1), \mathcal{C}_1(t_{i+1}), \dots, \mathcal{C}_1(t_{j-1}), \mathcal{C}_1(t_j)$ which satisfies all of following three conditions:

- (1) Strong candidate requests set $\mathcal{C}_1(t_i)$ does not preempt another candidate requests set.
- (2) Strong candidate requests set $\mathcal{C}_1(t_{k+1})$ strongly preempts strong candidate requests set $\mathcal{C}_1(t_k)$, for $k = i, \dots, j - 2$.

- (3) Weak candidate set $\mathcal{C}_2(t_j)$ weakly preempts strong candidate set $\mathcal{C}_1(t_{j-1})$, or a strong candidate set $\mathcal{C}_1(t_j)$ strongly preempts $\mathcal{C}_1(t_{j-1})$ and $\mathcal{C}_1(t_j)$ is not preempted by another requests set.

Here, we use the parameters of the first and last *candidate requests set* to denote a *candidate sequence*, e.g., $\mathcal{S}(t_i, t_j)$. According to the definition, we can decompose the solution of algorithm \mathcal{G} , i.e., $\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_m\}$, into multiple *candidate sequences*. Notice that each spectrum request \mathbf{e} will appear in exactly one *candidate sequence*. We use $\mathcal{G}(\mathcal{S}(t_i, t_j))$ to denote the profit made on *candidate sequence* $\mathcal{S}(t_i, t_j)$ by algorithm \mathcal{G} . And we use $\text{OPT}[t_i, t_j]$ to denote the profit made by optimal offline algorithm on the requests whose starting times are in interval $[t_i, t_j]$.

Lemma 3 *For each candidate sequence $\mathcal{S}(s_i, s_j)$ in the solution given by algorithm \mathcal{G} , we have*

$$\mathcal{G}(\mathcal{S}(s_i, s_j)) \geq \min(c_1, c_2) \cdot \mathcal{C}_1(s_{j-1}).$$

Proof By definition of candidate sequences, there are two different cases: (1) $\mathcal{C}_1(s_j)$ strongly preempted $\mathcal{C}_1(s_{j-1})$ or (2) $\mathcal{C}_2(s_j)$ weakly preempted $\mathcal{C}_1(s_{j-1})$.

If request $\mathcal{C}_1(s_j)$ strongly preempted $\mathcal{C}_1(s_{j-1})$, \mathcal{G} makes at least $\mathcal{C}_1(s_j) \geq c_1 \cdot \mathcal{C}_1(s_{j-1})$. If request $\mathcal{C}_2(s_j)$ weakly preempted $\mathcal{C}_1(s_{j-1})$, \mathcal{G} makes at least $\mathcal{P}(\mathcal{C}_1(s_{j-1}), s_j) + \mathcal{C}_2(s_j) \geq c_2 \cdot \mathcal{C}_1(s_{j-1})$. So the lemma holds for either case. \square

Lemma 4 ([37]) *For each candidate sequence $\mathcal{S}(s_i, s_j)$ in the solution given by algorithm \mathcal{G} , for each $i \leq k < j$, we have*

$$\text{OPT}[s_k, s_{k+1}] \leq \left(c_1 + c_2 + \frac{c_2}{\sqrt{(fl+1)/\Delta}} \right) \mathcal{C}_1(s_k).$$

Lemma 5 *For each candidate sequence $\mathcal{S}(s_i, s_j)$ in the solution given by algorithm \mathcal{G} , we have*

$$\text{OPT}[s_i, s_j] \leq \left(c_1 + 1 + \frac{1}{c_1 - 1} \right) \left(c_1 + c_2 + \frac{c_2}{\sqrt{(fl+1)/\Delta}} \right) \mathcal{C}_1(s_{j-1}).$$

Proof Obviously, $\text{OPT}[s_i, s_j] = \sum_{k=i}^{j-1} \text{OPT}[s_k, s_{k+1}]$. From Lemma 4, we have $\text{OPT}[s_k, s_{k+1}] \leq \left(c_1 + c_2 + \frac{c_2}{\sqrt{(fl+1)/\Delta}} \right) \mathcal{C}_1(s_k)$. Thus, $\text{OPT}[s_i, s_j] \leq \left(c_1 + c_2 + \frac{c_2}{\sqrt{(fl+1)/\Delta}} \right) \sum_{k=i}^{j-1} \mathcal{C}_1(s_k)$. Based on the condition of strong-preemption, we have $\mathcal{C}_1(s_k) \geq c_1 \cdot \mathcal{C}_1(s_{k-1})$ for all $i \leq k \leq j$. The lemma then follows. \square

Theorem 6 *Algorithm \mathcal{G} is $\Theta(\sqrt{\gamma+1}\Delta^{-\frac{1}{2}})$ -competitive.*

Proof Given a request sequence \mathcal{I} , we decompose the solution by algorithm \mathcal{G} into multiple *candidate sequence* $\mathcal{S}(t_i, t_j)$. Then, the total profit made by algorithm \mathcal{G} is

$\sum_{\forall S(t_i, t_j)} \mathcal{S}(t_i, t_j)$. On the other hand, the total profit made by the optimal offline algorithm is $\sum_{\forall S(t_i, t_j)} \text{OPT}[t_i, t_j]$. From [Lemmas 3](#) and [5](#), the competitive ratio of algorithm \mathcal{G} is at least

$$\frac{\sum_{\forall S(t_i, t_j)} \mathcal{S}(t_i, t_j)}{\sum_{\forall S(t_i, t_j)} \text{OPT}[t_i, t_j]} \geq \frac{\min(c_1, c_2)}{(c_1 + 1 + \frac{1}{c_1 - 1})(c_1 + c_2 + \frac{c_2}{\sqrt{(\gamma+1)/\Delta}})}.$$

It is easy to show that the right-hand side gets the maximum value $\frac{1}{4(1+\sqrt{\Delta/(\gamma+1)})}$ if $c_1 = 2$ and $c_2 = \frac{2}{1+\sqrt{\Delta/(\gamma+1)}}$. Thus, the competitive ratio is at least $\frac{1}{8} \sqrt{\frac{\gamma+1}{\Delta}}$ when $\gamma \leq \Delta - 1$. \square

Let $n(t)$ be the cardinality of $\mathcal{R}_a(t)$. At time t , [Algorithm 2](#) takes $O(n(t)^3)$ time to find $\mathcal{C}_1(t)$ and $\mathcal{C}_2(t)$, then takes constant time to make decision. The time complexity of [Algorithm 2](#) is $\sum_{\forall t} n(t)^3$. Notice that each request could appear in at most $\Delta + \gamma$ different $\mathcal{R}_a(t)$. Let n be the number of all online requests. We have $\sum_{\forall t} n(t) \leq (\Delta + \gamma)n$ and $n(t) \leq n$ for all t , which implies following theorem.

Theorem 7 *Time complexity of [Algorithm 2](#) is $O((\Delta + \gamma)n^3)$.*

3.2.3 Method \mathcal{H} for $(\beta > 1, \gamma, \alpha)$ Problem

In this subsection, we propose a greedy algorithm \mathcal{H} for $(\beta > 1, \gamma, \alpha)$ problem. At time t , the method \mathcal{H} should decide whether a request whose start time is $t + \alpha - \gamma$ will be accepted.

Then, we show that algorithm \mathcal{H} is asymptotically optimal. The notations are used in previous subsection.

Theorem 8 *Algorithm \mathcal{H} is $\frac{(c-\beta-1)}{c^2} \frac{\gamma+1}{\Delta+\gamma+1}$ -competitive.*

Proof We divide the requests accepted by algorithm \mathcal{H} into multiple *candidate requests sets*. Let us consider the profit made by \mathcal{H} on each *candidate requests set* $\mathcal{C}_1(t)$. For each admitted $\mathcal{C}_1(t)$, we redistribute $(1 + \beta)\mathcal{C}_1(t')$ of the money from $\mathcal{C}_1(t)$ to $\mathcal{C}_1(t')$ if $\mathcal{C}_1(t)$ preempts $\mathcal{C}_1(t')$.

There are three complementary cases here.

Case 1: $\mathcal{C}_1(t)$ does not preempt others and is not preempted. \mathcal{H} makes $\mathcal{C}_1(t)$ profit on $\mathcal{C}_1(t)$. Let $t_s = t + \alpha - \gamma$. For the optimal offline algorithm OPT, if time $t_s + i(\gamma + 1)$ (for some integer $i > 0$) is before the end time of $\mathcal{C}_1(t)$, we have $\text{OPT}[t_s + i(\gamma + 1), t_s + (i + 1)(\gamma + 1)] \leq c \cdot \mathcal{C}_1(t)$. Otherwise, the requests whose starting times are during time interval $[t_s + i(\gamma + 1), t_s + (i + 1)(\gamma + 1)]$ should preempt $\mathcal{C}_1(t)$. We know that the total running time of $\mathcal{C}_1(t)$ is no more than $\Delta + \gamma + 1$. Thus, no more than $c \lceil \frac{\Delta+\gamma+1}{\gamma+1} \rceil \mathcal{C}_1(t)$ profit will be made by OPT during the running time of $\mathcal{C}_1(t)$.

Case 2: $\mathcal{C}_1(t)$ does not preempt others and is preempted in \mathcal{H} . \mathcal{H} makes $\mathcal{C}_1(t)$ profit on $\mathcal{C}_1(t)$ since the *candidate requests* which preempted $\mathcal{C}_1(t)$ will distribute $(1 + \beta)\mathcal{C}_1(t)$ profit to $\mathcal{C}_1(t)$. And at most $\beta\mathcal{C}_1(t)$, profit is paid as penalty.

No more than $c \lceil \frac{\Delta+\gamma+1}{\gamma+1} \rceil \mathcal{C}_1(t)$) profit will be made by OPT during the running time of $\mathcal{C}_1(t)$.

Case 3: $\mathcal{C}_1(t)$ preempts another candidate requests set in \mathcal{H} . \mathcal{H} makes at least $\frac{c-\beta-1}{c} \mathcal{C}_1(t)$ profit from $\mathcal{C}_1(t)$ (after considering redistributing some to the candidate requests set being preempted by $\mathcal{C}_1(t)$) since we distribute $(1 + \beta)\mathcal{C}_1(t') \leq \frac{1+\beta}{c} \mathcal{C}_1(t)$ to the candidate set preempted by $\mathcal{C}_1(t)$. If $\mathcal{C}_1(t)$ is not preempted, we make at least $\frac{c-\beta-1}{c} \mathcal{C}_1(t)$. If $\mathcal{C}_1(t)$ is preempted, we will receive $(1 + \beta)\mathcal{C}_1(t)$ from the future candidate requests set that preempts it. Thus, we always make at least $\mathcal{C}_1(t)$. For OPT, it is easy to show that OPT will make no more than $c \lceil \frac{\Delta+\gamma+1}{\gamma+1} \rceil \mathcal{C}_1(t)$ profit.

Thus, the competitive ratio for each candidate requests set is at least $\frac{(c-\beta-1)}{c^2} \lceil \frac{\gamma+1}{\Delta+\gamma+1} \rceil$. The competitive ratio of \mathcal{H} is at least $\frac{(c-\beta-1)}{c^2} \frac{\gamma+1}{\Delta+\gamma+1}$. We finish the proof. \square

When $\gamma = a\Delta - 1$, it is easy to show that

Theorem 9 *Method \mathcal{H} is at least $\frac{a}{4(1+a)(1+\beta)}$ -competitive (by choosing $c = 2(1 + \beta)$), when $\gamma = a\Delta - 1$.*

3.3 More General Networks

All studies in previous sections assumed that the conflict graph of networks, which models the location-dependent conflict, is a complete graph. In this section, we extend the algorithms for the networks where each request \mathbf{e}_i asks for spectrum usage in a disk-shaped region D_i centered at (x_i, y_i) with a radius r_i . For the performance upper bounds, all Lemmas and Theorems in Sect. 3.1 still hold since complete conflict graph is a special case.

If we can find the strong (weak) candidate sets $\mathcal{C}_1(t)$ ($\mathcal{C}_2(t)$) in polynomial time as we did by using Algorithm 1 in Sect. 3.2.2, then the method \mathcal{G} (or \mathcal{H}) still works since all previous analysis still holds. However, the main difficulty is that, in this case, the central authority may accept multiple requests at same time, which makes it hard to get the strong/weak candidate sets in polynomial time by using dynamic programming. Therefore, we propose to approximate strong/weak candidate sets at each time t in polynomial time by using *shifting strategy* [24] as follows.

We divide the request disks into different levels according to their radii. At same level, the radii of all request disks are within a constant factor λ of each other. Let d be the diameter of smallest request disk; level i includes all request disks whose diameters are within $[d\lambda^{i-1}, d\lambda^i]$.

First, we show that a PTAS can be achieved in *each* level i . In other words, here we approximate the strong/weak candidate sets when only request disks in level i are considered. Let $k > 1$ be an integer. At level i , a (p, q) partition is defined as the region is partitioned by a set of horizontal lines, $y = \dots, -2\lambda^i k + p, -\lambda^i k$

$+p, p, \lambda^i k + p, 2\lambda^i k + p, \dots$, and a set of vertical lines, $x = \dots, -2\lambda^i k + q, -\lambda^i k + q, q, \lambda^i k + q, 2\lambda^i k + q, \dots$. In each partition, the region is partitioned into a number of $\lambda^i k \times \lambda^i k$ squares. We ignore the requests whose disks are hit by any lines of the partition and find the strong/weak candidate sets in each $\lambda^i k \times \lambda^i k$ square. Together with strong/weak candidate sets in all squares, we get strong/weak candidate sets for a partition.

Lemma 6 *Strong/weak candidate sets for each partition can be computed in polynomial time.*

Proof According to area argument, the number of independent requests that can be accepted at same time is no more than $C = 4\lambda^2 k^2 / \pi$ which is a constant. Then, we can find the strong/weak candidate sets in each $\lambda^i k \times \lambda^i k$ square in polynomial time by using dynamic programming as we did in [Algorithm 1](#). Assume $n_{a,b}(t)$ requests located in a $\lambda^i k \times \lambda^i k$ square whose top left corner is $(a \lambda^i k + p, b \lambda^i k + q)$ in a (p, q) partition at time t . The main difference is that we need to define $\mathcal{Q}(e_1, e_2, \dots, e_C, s)$ as the maximum possible profit from $\mathcal{R}(t)$ when e_1, e_2, \dots, e_C are the last independent requests accepted by central authority before or at time s . Then,

$$\mathcal{C}_1(t) = \max \mathcal{Q}(e_1, e_2, \dots, e_C, \max s),$$

where $\max s$ the latest start time among $\mathcal{R}(t)$. Since there are at most $\binom{n_{a,b}(t)}{C+1} = O(n_{a,b}(t)^{C+1})$ different \mathcal{Q} , and each \mathcal{Q} can be computed within $\binom{n_{a,b}(t)}{C} = O(n_{a,b}(t)^C)$ time, the time complexity to compute strong/weak candidate sets in a $k \times k$ square whose left top corner is $(ak + i, bk + j)$ at time t is $O(n_{a,b}(t)^{2C+1})$. Let $n(t)$ denote the total number of requests in $\mathcal{R}(t)$. The time complexity to compute strong/weak candidate sets in all $\lambda^i k \times \lambda^i k$ squares of a partition (p, q) at time t is no more than $O(n(t)^{2C+1})$. \square

In each partition, some requests may be ignored. To achieve a better approximation, we shift the horizontal and vertical lines and compute strong/weak candidate sets for k^2 partitions where $p \in [1, k]$ and $q \in [1, k]$. We set the best strong/weak candidate sets among all partitions as the strong/weak candidate sets of time t . It takes $O(k^2 n(t)^{2C+1})$ time to find strong/weak candidate sets at each time. Then, the total running time is at most $O(k^2 n^{2C+2})$ where n is the number of total requests since we have to compute this at most n times.

On the other hand, the strong/weak candidate sets we find are at least a $1 - \frac{2}{k}$ approximation as following lemma.

Lemma 7 *At least one partition achieves at least $1 - \frac{2}{k}$ approximation.*

Proof We prove the lemma by using strong candidate set. The weak candidate set case follows same proof. Let $\mathcal{C}_1^{i,j}(t)$ denote the total profit made by all requests such that (1) appear in $\mathcal{C}_1(t)$ and (2) not ignored in partition (i, j) . Then,

$$\sum_{\forall i,j} \mathcal{C}_1^{i,j}(t) \geq (k^2 - 2k)\mathcal{C}_1(t)$$

since each request is ignored in at most 2 partitions.

On the other hand, $\mathcal{C}_1^{i,j}(t)$ is smaller than the profit of the strong candidate set for partition (i, j) , which is no larger than the profit of the approximated strong candidate set, called $\mathcal{C}'_1(t)$. In other words,

$$k^2\mathcal{C}'_1(t) \geq \sum_{\forall i,j} \mathcal{C}_1^{i,j}(t) \geq (k^2 - 2k)\mathcal{C}_1(t)$$

which implies $\mathcal{C}'_1(t) \geq (1 - \frac{2}{k})\mathcal{C}_1(t)$ and finishes the proof. \square

According to [Lemmas 6](#) and [7](#), shifting strategy gives a PTAS for *each* level i . The result can be extended when request disks in *all* levels are considered. The main technology is dynamic programming which is discussed in [Sect. 3.2](#) of [24]. We omit the details here.

Since the strong/weak candidate sets can be approximated well, we have the following theorem.

Theorem 10 *The methods $(\mathcal{K}, \mathcal{G}, \mathcal{H})$ will achieve asymptotically optimum competitive ratios in polynomial time for general networks modeled by disk graphs. The competitive ratios will have an additional factor $1 - \epsilon$ for any constant $\epsilon > 0$ comparing with previous analysis, when we set $\epsilon = \frac{2}{k}$.*

3.4 Dealing With Selfish Users

In this section, we show how to design a mechanism based on the allocation algorithm described in [Sect. 3.2](#) when secondary users could be selfish. For the spectrum allocation and auction problem, when each secondary user declares his request, he may lie on the bid and time requirement. We need to design rules such that each secondary user has incentives to declare his request truthfully. Each secondary user i has its own private information \mathbf{t}_i , including b_i , a_i , and t_i . Let $\mathbf{a}_i = (b'_i, a'_i, t'_i)$ be the value user i will report. For each vector of actions $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n)$, a mechanism $M = (\mathcal{A}, P)$ (including allocation algorithm \mathcal{A} and pricing scheme P) computes a spectrum allocation $\mathcal{A}(\mathbf{a}) = (\mathcal{A}_1(\mathbf{a}), \mathcal{A}_2(\mathbf{a}), \dots, \mathcal{A}_n(\mathbf{a}))$ and a payment vector $\mathbf{p}(\mathbf{a}) = (\mathbf{p}_1(\mathbf{a}), \mathbf{p}_2(\mathbf{a}), \dots, \mathbf{p}_n(\mathbf{a}))$. Each user i will be allocated $\mathcal{A}_i(\mathbf{a})$ and be charged $\mathbf{p}_i(\mathbf{a})$.

A mechanism is said to be truthful if it satisfies both incentive compatible (IC) and individual rational (IR) properties. A mechanism is incentive compatible if every user i will maximize its utility by reporting its private type \mathbf{t}_i truthfully. A mechanism is individual rational if the utility of an agent winning the auction is not less than its utility of losing the auction. We assume that no user will delay his/her request and a user will not lie about t_i and s_i . Unfortunately, in this work, we cannot

Algorithm 3 Online spectrum allocation \mathcal{H}

Input: A constant parameter $c > 1 + \beta, \gamma, \alpha, \Delta, \mathcal{R}_a(t), \mathcal{R}(t), \mathcal{C}_1(t)$. Previous *current candidate requests set* $\mathcal{C} = \mathcal{C}_1(t')$ where $t' < t$. Here, $\mathcal{C}_1(t')$ may be empty.

Output: Whether requests submitted at time $t - \gamma$ will be admitted and new *current candidate requests set* \mathcal{C} .

- 1: **if** $\mathcal{C}_1(t) \geq c \cdot \mathcal{C}_1(t')$ **then**
 - 2: $\mathcal{C} = \mathcal{C}_1(t)$;
 - 3: Accept request $\mathbf{e}_i \in \mathcal{C}_1(t)$ such that $a_i = t - \gamma$.
 - 4: **else**
 - 5: Accept request $\mathbf{e}_i \in \mathcal{C}_1(t')$ such that $a_i = t - \gamma$.
-

design an online spectrum allocation mechanism that is always truthful. Instead, we will show that under the mechanism TOFU, regardless of the actions of other users, no user i can improve its profit by reporting a bid b'_i that is smaller than its true bid value b_i . We call this property as *semi-truthful*.

For the spectrum auction problem, the final profit (or the utility) of a user i is 0 if its request is rejected. If its request is admitted, the final profit is

$$\text{utility}(i) = b_i - \mathbf{p}_i + \mu(b'_i, \ell'_i, t_i),$$

where b_i is the actual valuation, \mathbf{p}_i is the real payment, and $\mu(b'_i, \ell'_i, t_i)$ is the potential preemption penalty, where b'_i is the actual bid of user i .

It is a folklore result that the allocation method in a mechanism that can prevent lying must have the monotone property. Here, a spectrum allocation method \mathcal{A} is monotone if a user i is granted the spectrum usage under \mathcal{A} with a bid $\mathbf{e}_i = (b_i, a_i, t_i)$, then the user i will still be granted the spectrum usage under \mathcal{A} if the user increases bid b_i and/or decreases the required time duration t_i . First, the methods ([Algorithms 2](#) and [3](#)) presented in previous section do have the monotone property. In the algorithm, we need to find strong candidate requests set, and weak candidate requests set. Both sets will be found by dynamic programming. It is easy to show that these dynamic programming methods are monotone. Thus, it is possible to design a mechanism using the algorithms (\mathcal{G} and \mathcal{H}) as spectrum allocation methods.

Consider a user i , and assume the bids of all other users remain the same. We first propose the following definition based on the monotone property of the methods.

Definition 5 Let \underline{b}_i be the minimum bid that i has to bid to get admitted when its spectrum request is to be processed at γ time slots later. Let \bar{b}_i be the minimum bid that i has to bid to get admitted and not get preempted later.

Clearly, $\underline{b}_i \leq \bar{b}_i$. For a secondary user whose request is not granted, the value $\underline{b}_i = 0$. The values \underline{b}_i and \bar{b}_i clearly can be computed in polynomial time since the bid values of all other users are known. Here, \underline{b}_i can be computed at the time $a_i + \gamma$. To compute the value \bar{b}_i , we need to know whether request i will be preempted later. Since the latest job that can preempt \mathbf{e}_i must have starting time no later than $s_i + \Delta$,

Algorithm 4 TOFU semi-truthful online spectrum auction

-
- 1: Based on the setting, we use either [Algorithms 2](#) or [3](#) to decide whether a request \mathbf{e}_i will be admitted or not.
 - 2: The mechanism will charge user i a payment

$$\mathbf{p}_i = \underline{b}_i$$

we must process the potential jobs that can preempt \mathbf{e}_i no later than $s_i + \Delta - \alpha + \gamma$. In other words, the value \bar{b}_i can be computed within time $\Delta + \gamma$ after job i arrived. Then, the protocol TOFU works as follows.

Theorem 11 *The mechanism TOFU is semi-truthful, i.e., to maximize its profit, every secondary user will not bid a price lower than its actual value.*

Proof Consider a user i arrived at time t . Let b_i be its private willing bid and b'_i be its actual bid. When i bids some value $b'_i \in (\underline{b}_i, \bar{b}_i)$, i will get preempted with some unserved time ℓ'_i and receive a compensation $\mu(b'_i, \ell'_i, t_i) = \beta \frac{\ell'_i}{t_i} b'_i$.

Case 1: $b'_i \geq \bar{b}_i$. There are three strategies again for i .

- (1) If $b'_i \geq \bar{b}_i$, then user i will be admitted and will not be preempted. It will be charged a payment $\mathbf{p}_i < \bar{b}_i$ and thus will get a profit $b_i - \mathbf{p}_i > 0$.
- (2) If $b'_i < \underline{b}_i$, it will not get admitted and thus get 0 utility.
- (3) If $b'_i \in (\underline{b}_i, \bar{b}_i)$, then i will be admitted and then be preempted with unserved time ℓ'_i . It will get a compensation $\mu(b'_i, \ell'_i, t_i) = \beta \frac{\ell'_i}{t_i} b'_i$. Its utility then is $f \cdot b_i - \mathbf{p}_i + \mu(b'_i, \ell'_i, t_i) = (1 - \beta \frac{\ell'_i}{t_i}) b_i - \mathbf{p}_i + \beta \frac{\ell'_i}{t_i} b'_i = b_i - \mathbf{p}_i - \beta \frac{\ell'_i}{t_i} (b_i - b'_i)$, which is less than the profit $b_i - \mathbf{p}_i$ that it will get when reported b_i since $b'_i < \bar{b}_i \leq b_i$.

Thus, in all subcases, reporting untruthfully will not gain any benefit for i .

Case 2: $b'_i \in (\underline{b}_i, \bar{b}_i)$. There are five strategies for i .

- (1) If $b'_i = b_i$, it will be admitted and then be preempted. Let ℓ_i be the unserved time slots when bidding b_i . Then, its utility is $(1 - \beta \frac{\ell_i}{t_i}) b_i - \mathbf{p}_i + \beta \frac{\ell_i}{t_i} b_i = b_i - \mathbf{p}_i > 0$.
- (2) If $b'_i \geq \bar{b}_i$, then user i will be admitted and will not be preempted. It will be charged a payment $\mathbf{p}_i < \bar{b}_i$ and thus will get a profit $b_i - \mathbf{p}_i > 0$.
- (3) If $b'_i < \underline{b}_i$, it will not get admitted and thus get 0 utility.
- (4) If $b_i < b'_i < \bar{b}_i$, then i will be admitted and then be preempted with a shorter unserved time $\ell'_i \leq \ell_i$, due to the monotone property of the method. It will get a compensation $\mu(b'_i, \ell'_i, t_i) = \beta \frac{\ell'_i}{t_i} b'_i$. Its utility then is $f \cdot b_i - \mathbf{p}_i + \mu(b'_i, \ell'_i, t_i) = (1 - \beta \frac{\ell'_i}{t_i}) b_i - \mathbf{p}_i + \beta \frac{\ell'_i}{t_i} b'_i = b_i - \mathbf{p}_i - \beta \frac{\ell'_i}{t_i} (b_i - b'_i)$, which is larger than the profit $b_i - \mathbf{p}_i$ that it will get when reported b_i since $b'_i > b_i$.
- (5) If $\underline{b}_i < b'_i < b_i$, then i will be admitted and then be preempted with a longer unserved time $\ell'_i \geq \ell_i$. Its utility then is $f \cdot b_i - \mathbf{p}_i + \mu(b'_i, \ell'_i, t_i) = (1 - \beta \frac{\ell'_i}{t_i}) b_i - \mathbf{p}_i + \beta \frac{\ell'_i}{t_i} b'_i = b_i - \mathbf{p}_i - \beta \frac{\ell'_i}{t_i} (b_i - b'_i)$, which is smaller than the profit $b_i - \mathbf{p}_i$ that it will get when reported b_i since $b'_i < b_i$.

Thus, in all subcases, reporting lower will not gain any benefit for i .

Case 3: $b_i \leq \underline{b}_i$. There are several strategies again for i .

- (1) If $b'_i \leq \underline{b}_i$, it will not get admitted and thus has 0 profit.
- (2) If $b'_i \geq \bar{b}_i$, it will be admitted and not be preempted. In this case, its profit is $b_i - \mathbf{p}_i < 0$.
- (3) If $\underline{b}_i < b'_i < \bar{b}_i$, it will be admitted and be preempted with an unserved time ℓ'_i .

In this case, its profit is $b_i - \mathbf{p}_i - \beta \frac{\ell'_i}{t_i} (b_i - b'_i) < 0$ since $b_i < \mathbf{p}_i$ and $b_i < b'_i$. In all subcases, bidding untruthfully is not beneficial to user i . \square

Observe that, in the scheme, the only scenario (case 2.4) that a secondary user can gain benefit is when $b_i \in (\underline{b}_i, \bar{b}_i)$ and it bids a value $b'_i \in (b_i, \bar{b}_i)$. The user i must bid larger than its true valuation b_i , but not larger than \bar{b}_i . Observe that value \bar{b}_i is computed at time $a_i + \gamma + \Delta$ using the requests submitted during time interval $[a_i, a_i + \Delta]$, where user i does not know when it arrives at time a_i . Thus, we have

Theorem 12 *If any user i does not know the requests within Δ time after its arrival, the mechanism TOFU is truthful, i.e., to maximize its profit, every secondary user will bid truthfully even it learned history.*

It is not difficult to construct examples in which a user can gain profit by bidding larger than its true valuation if it can know the bids of other users. It will be an interesting future work to study the total payment of a mechanism compared with the best possible payments collected by a truthful mechanism. Notice that, when we know the distributions of all bids, and there is only one spectrum, and no spatial and temporal reuse, Myerson [26] presented a mechanism that guarantees the payment is optimal. We note that it is very challenging to design a mechanism with a total payment close to optimum for the problem studied here.

3.5 Simulation Studies

We conducted extensive simulations to study the performance of the algorithm \mathcal{G} , specifically, the competitive ratio of the algorithm in practice. We set $c_1 = 2$ and $c_2 = \sqrt{\frac{\gamma+1}{\Delta}}$ for algorithm \mathcal{G} . Suppose the total service time is always 2,000 time slots, which is large enough comparing with time requirements. In the experiment, we first generate n secondary nodes that are randomly placed. We randomly deployed 100 nodes in a 5×5 square area. Assume that all secondary users locate at these positions. The geometry location of a request is exactly the location of secondary user who claimed that request. We assume that any two requests within distance 1 will conflict with each other if they requested the same channel for some time slots. We then generate random requests with random bid values, and time requirement. The bid of each request is uniformly distributed in $[0, 1]$; the time requirement of each request is uniformly distributed in $[1, \Delta]$.

Observe that the average load of a node in the simulation is $\frac{J \cdot D}{T \cdot n}$, where J is total number of requests, D is the average duration of a request (thus, $D = \Delta/2$ in the

setting), T is total duration of scheduling (thus, $T = 2,000$ time slots here), and n is number of secondary users ($n = 100$ here). Observe that the expected number of users in a unit area is $n/5^2 = 4$ in the setting. Thus, varying J and Δ , we tested both lightly loaded system and also highly loaded system.

3.5.1 Competitive Ratio of Algorithm \mathcal{G}

A number of parameters may affect the performances of algorithm \mathcal{G} , e.g., the number of total requests, the *time bound* Δ , and the *delay factor* γ . To study the effect of these parameters, we first fix the *time bound* Δ and study the competitive ratios of \mathcal{G} while the *delay factor* γ varies. Then, we fix the *delay factor* γ and study the competitive ratios of \mathcal{G} while the *time bound* Δ changes. We set three different total numbers of requests in these experiments to study how the degree of competition affects \mathcal{G} 's performance.

In Fig. 5, we plot the competitive ratios of \mathcal{G} in various cases while the *delay factor* γ is fixed (see Fig. 5a–c); we also plot the competitive ratios of algorithm \mathcal{G} in various cases while *time bound* Δ is fixed (see Fig. 5d–f). In most cases, method \mathcal{G} makes a total profit that is more than 95 % of the optimum offline method when $\gamma \simeq \Delta$. Observe that although it is NP-hard to find the optimum offline solution, we implemented a method (omitted here due to space limit) that will find an almost optimum solution (within a factor $1 - \epsilon$ for an arbitrarily small $\epsilon > 0$) when the conflict graph is a disk-intersection graph. The experimental results are much better than the theoretical bound we proved in previous sections.

The competitive ratios decrease when Δ increases. This is exactly what the theoretical analysis implies. When $\gamma < \Delta$, the competitive ratio of \mathcal{G} increases significantly when *dfactor* increases. This implies that increasing the *delay factor* is a good way to achieve better performance if the *delay factor* is relatively small. We also observe that \mathcal{G} almost achieves the optimum offline solution when *delay factor* γ is close to or larger than Δ , which verifies the theoretical bound. Furthermore, the competitive ratio will not improve much when $\gamma > \Delta$. Thus, we recommend to set $\gamma \in [\Delta/2, \Delta]$.

The total number of requests also affects the performance of \mathcal{G} . We observe that the competitive ratios decrease when the total number of requests increases. This is because \mathcal{G} is conservative: the weak preemption in \mathcal{G} just tries to satisfy the theoretical bound, which may lose some profit potentially. Thus, when there are more requests, the optimal offline algorithm will make more profit but the profit made by \mathcal{G} will not increase so much.

3.5.2 Efficiency Ratio of Algorithm \mathcal{G}

Recall that to ensure that secondary users will not bid a price lower than their actual values, we do not charge what they bid. For secondary user whose request is accepted, we charge the minimum bid such that the request will still be accepted. Thus, the actual profit made is not the winners' total bids but the winners' total payments. Clearly, the total payments are no more than the total bids. Here, we define the *efficiency ratio* as the ratio between the profit made by the mechanism and the winners' total bids computed by the method \mathcal{G} . We also study the effect of

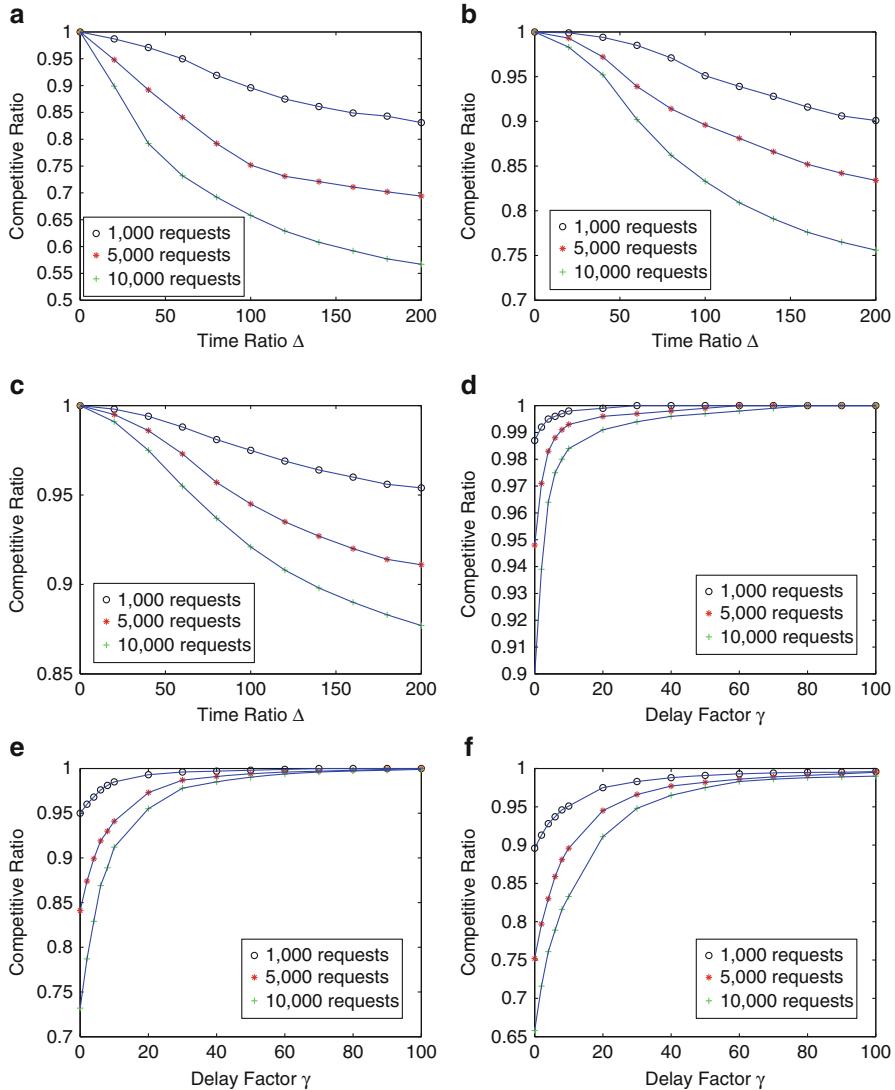


Fig. 5 The competitive ratios of method \mathcal{G} in various cases. **(a)** Delay factor $\gamma = 0$. **(b)** Delay factor $\gamma = 10$. **(c)** Delay factor $\gamma = 20$. **(d)** Time bound $\Delta = 20$. **(e)** Time bound $\Delta = 60$. **(f)** Time bound $\Delta = 100$

some parameters such as the number of total requests, the *time bound* Δ , and the *delay factor* γ .

In Fig. 6, we plot the efficiency ratios of the mechanism in various cases. In (a)–(c), we fix *time bound* Δ and plot the efficiency ratio while *delay factor* γ changes; in (d)–(f), we fix *delay factor* γ and plot the efficiency ratio while time bound Δ changes.

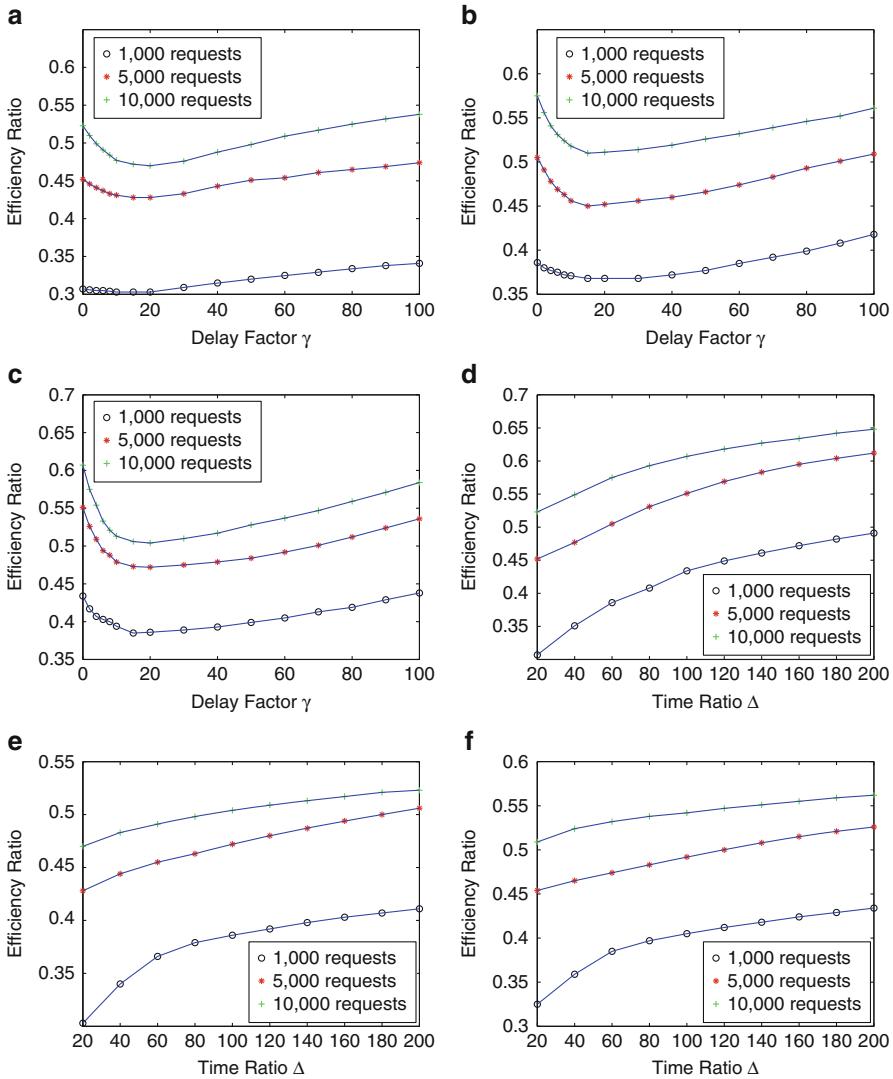


Fig. 6 The efficiency ratios of the mechanism in various cases. **(a)** Time bound $\Delta = 20$. **(b)** Time bound $\Delta = 60$. **(c)** Time bound $\Delta = 100$. **(d)** Delay factor $\gamma = 0$. **(e)** Delay factor $\gamma = 20$. **(f)** Delay factor $\gamma = 60$

We observe that the efficiency ratios increase when the Δ or the number of total requests increases. In this case, a request has more chance to conflict with other requests. Thus, it is easier to find a replacement which increases the minimum value a winner has to bid. We also find that the efficiency ratios first decrease then increase when the *delay factor* γ increases. Comparing with Fig. 5d-f, we know that the

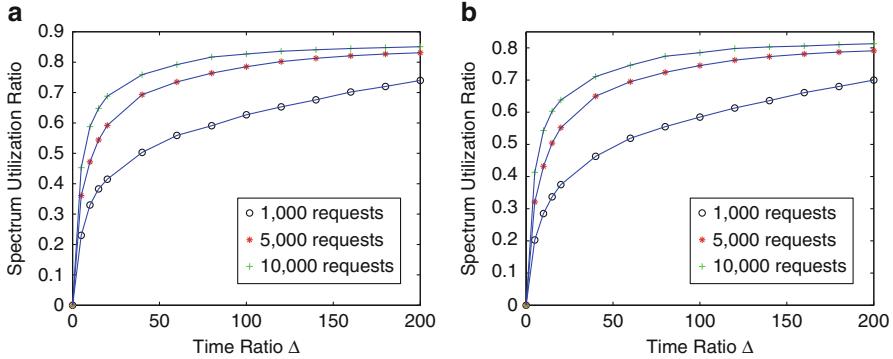


Fig. 7 The spectrum utilization ratios of method \mathcal{G} in various cases. (a) Delay factor $\gamma = 0$. (b) Delay factor $\gamma = 20$

efficiency ratios decrease because the competitive ratios increase dramatically when γ is relatively small. The total payments do not increase as fast as the total bids do. When the *delay factor* γ is larger than Δ , the efficiency ratios increase because the competitive ratios increase slowly.

3.5.3 Spectrum Utilization Ratio of Algorithm \mathcal{G}

We also studied the spectrum utilization of the method. The spectrum utilization in a time interval $[1, T]$ under a spectrum allocation method \mathcal{A} is defined as $\sum_{t=1}^T \ell_{\mathcal{A}}(t) / \sum_{t=1}^T n(t)$, where $\ell_{\mathcal{A}}(t)$ is the number of users who are using the channel (at different locations) at time t without conflict in spectrum allocation produced by \mathcal{A} and $n(t)$ is the maximum number of users who can use the channel concurrently without conflict. Figure 7 reports the results. We find that varying γ does not affect the spectrum utilization that much. Observe that the measurement of spectrum utilization compared to the method with the offline method that can arbitrarily preempt the spectrum usage. When the system is highly loaded, the spectrum utilization of the method is about 80 %, while the spectrum utilization is about 50 % for lightly loaded system.

3.5.4 Compare Algorithm \mathcal{G} with Simple Heuristics

We then compare algorithm \mathcal{G} with two simple greedy algorithms. One greedy algorithm \mathcal{B} always satisfies the virtual spectrum candidate request with the largest $C_1(t)$ value. When request with larger bid is coming, \mathcal{B} will terminate current assignments with smaller value if necessary. Another greedy algorithm \mathcal{D} always satisfies the virtual spectrum candidate request with the largest $C_2(t)$ value. When request with larger bid is coming, \mathcal{D} will terminate current assignments with smaller value if necessary. The competitive ratio of these two simple greedy algorithms could be arbitrarily bad theoretically. We conduct simulations to compare them with algorithm \mathcal{G} which is asymptotically optimal theoretically.

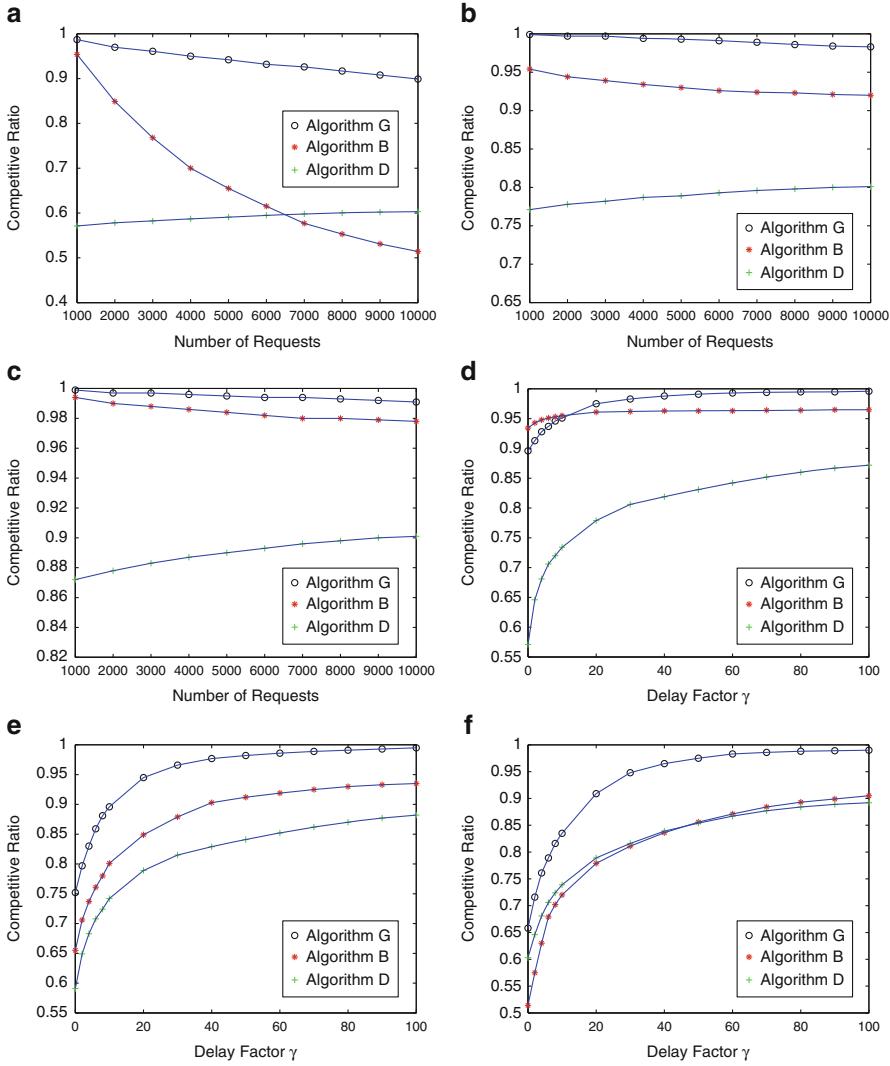


Fig. 8 Compare algorithm \mathcal{G} with two simple greedy algorithms. **(a)** Delay factor $\gamma = 0$. **(b)** Delay factor $\gamma = 10$. **(c)** Delay factor $\gamma = 20$. **(d)** 1,000 requests. **(e)** 5,000 requests. **(f)** 10,000 requests

In Fig. 8, we plot the competitive ratios of algorithm \mathcal{G} , simple greedy algorithm \mathcal{B} and \mathcal{D} to compare their performances. In Fig. 8a–c, we fix time bound $\Delta = 20$ and vary the number of total requests for different delay factors γ . In Fig. 8d–f, we fix time bound $\Delta = 100$ and vary delay factor γ for different number of total requests. We observe that the algorithm \mathcal{G} outperforms these two simple greedy algorithms significantly in most cases.

The effect of number of total requests is also investigated in Fig. 8a–c. We can see that the competitive ratio of algorithm \mathcal{G} is not affected much when the number of total requests increases. On the other hand, the performance of algorithm \mathcal{B} decreases significantly when the number of total requests increases and *delay factor* $\gamma = 0$. The competitive ratio of algorithm \mathcal{D} increases slightly when the number of total requests increases. However, the competitive ratio of algorithm \mathcal{D} is always worse than that of the algorithm \mathcal{G} .

The effect of *delay factor* γ is then investigated in Fig. 8a–c. We can see that the performances of all three algorithms increase significantly when the *delay factor* γ increases. When *delay factor* γ is increasing, the algorithm \mathcal{G} is always the best one whose performance is close to the optimum.

4 Spectrum Allocation and Mechanism Design with Known Distribution on Requests

In this section, we review the results on online spectrum allocation and mechanism design when some additional distribution information about online spectrum requests are known. Two different models will be studied here. The first model assumed that semi-arbitrary-arrival case, while the second model assume a random-arrival case.

4.1 Semi-Arbitrary-Arrival Case

In this section, we consider the semi-arbitrary-arrival case. We first show the lower bounds of competitive ratios under the *random ordering hypothesis*. The lower bound of social efficiency ratio comes from the well-known secretary problem [7] which is a special case in the model. The lower bound of revenue efficiency ratio comes from the result in [16]. Consider the model used in [16] (Sect. 5), in which a single item is being sold and an adversary specifies a set of bids that are randomly matched with arrival and departure intervals. That model is actually a special case of the model when $T = 1$. So its lower bound on the efficiency is also a lower bound in the model. We have following theorems.

Theorem 13 *For the online spectrum admission problem, no online mechanism is e -competitive for social efficiency.*

Theorem 14 *For the online spectrum admission problem, for any constant $\epsilon > 0$, there is no strategyproof online mechanism which is $(3/2 - \epsilon)$ -competitive for revenue efficiency.*

We are now ready to describe the online mechanism for spectrum allocation. The online mechanism \mathcal{M}_1 runs as following. It will divide the overall time times T into two phases. In the first phase (composed of the first δT time slots), it studies some

Algorithm 5 Learning algorithm $\mathcal{M}_1(\mathcal{L})$ for mechanism \mathcal{M}_1

Input: Requests arriving during $[0, \delta T]$.

Output: A parameter $\tau > 0$.

```

1: for  $\tau = 1$  to  $\lfloor \frac{T}{2} \rfloor$  do
2:   if  $\Pr(t_1 \leq t_i \leq 2t_1) \geq \frac{1}{2}$  then
3:     Return parameter  $\tau$ 

```

properties of the distribution of the requested time durations. Then, in the second phase, the mechanism will start to allocate the channel to the coming requests.

In the first phase, the mechanism \mathcal{M}_1 collects the information of coming requests and computes the information $\Pr(t_1 \leq t_i \leq 2t_1)$ for $t_1 \in [1, T/2]$. Here, we use $\Pr(t_1 \leq t_i \leq t_2)$ to denote the estimated probability that the required time duration of a coming request is at least t_1 and at most t_2 . We then learn a parameter τ which will be introduced later by using the *learning Algorithm 5* $\mathcal{M}_1(\mathcal{L})$. Here, $[\tau, 2\tau]$ will be a time interval such that at least a constant proportion of the requests whose requested duration of time slots will be in this interval. During the first δT time slots, we reject all coming requests. Later, we will show that it does not affect the performance significantly.

Notice that the adversary can pick up a common distribution (such as uniform distribution, normal distribution, exponential distribution) as the distribution of required time durations. The learning method ([Algorithm 5](#)) can find a feasible parameter τ without knowing the exact distribution. Notice that here the constant $1/2$ in [Algorithm 5](#) could be replaced by any positive constant $\in (0, 1)$. For example, if the required time durations are uniformly distributed in $[t_1, t_2]$, then $\tau = \frac{t_1+t_2}{2}$ is a feasible solution. If the required time durations are normally distributed with mean μ , then $\tau = \mu$ is a feasible solution. In the rest of chapter, we make the following assumption.

Assumption 1 *Given a distribution on the required time slots, we assume that a feasible value τ can always be found by [Algorithm 5](#).*

Since we only sampled the requests arrived in time $[0, \delta T]$, we first would like to bound the estimation error of these probabilities. Let Err be the maximum error

$$Err = \max_{t_1 \leq T/2} \|\Pr(t_1 \leq t_i \leq 2t_1) - \overline{\Pr}(t_1 \leq t_i \leq 2t_1)\|,$$

where $\overline{\Pr}(t_1 \leq t_i \leq t_2)$ is the actual probability that the required time slots of a coming request are in $[t_1, t_2]$.

Lemma 8 *The estimation error Err is neglectable.*

Proof According to the theory of VC-dimension [30], the bound on the test error of a classification model is

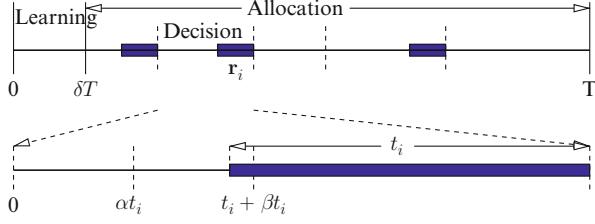


Fig. 9 In mechanism $\mathcal{M}_1(\mathcal{D})$, time is divided into several decision phases. A decision phase may not accept any request. For each decision phase with one accepted request e_i , the requested number of time slots should be $t_i \in [\tau, 2\tau]$

$$\Pr \left(Err \leq \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}} \right) \geq 1 - \eta,$$

where $h < N$ is the VC-dimension of the classification model, and N is the size of the training set. The formula is valid when $h < N$. In the case, the VC-dimension is at most $h = \log \frac{T}{2}$ since we only consider the inputs with requested time duration at most $T/2$. Assume the arrival rate is λ , then the size of training set is $N = \delta\lambda T$ with high probability. The bound on the test error is given by $\sqrt{\frac{\log \frac{T}{2} (\log(\delta\lambda T / \log \frac{T}{2}) + 1)}{\delta\lambda T}} = \sqrt{\frac{\log(2\delta\lambda - \log \log \frac{T}{2} + 1) \log \frac{T}{2}}{2\delta\lambda T}}$. Since δ and λ are constants, the error is neglectable when T is large enough. \square

After the first δT time slots, based on the results from learning algorithm, the decision algorithm $\mathcal{M}_1(\mathcal{D})$ of mechanism \mathcal{M}_1 decides which requests will be admitted and how much each admitted request should be charged. The following concept is needed for presenting mechanism \mathcal{M}_1 .

Definition 6 Let $p_{\leq \tau}^{(s)}$ denote the s -th highest bid value (for the usage of per unit time slot) among all requests received during time interval $[0, \tau]$.

The main idea of algorithm $\mathcal{M}_1(\mathcal{D})$ is to divide the time interval $[\delta T, T]$ into multiple non-overlap decision phases. Figure 9 illustrates the protocol. In each decision phase, the decision algorithm waits for some time slots, then tries to admit a request with the highest bid value in this decision phase. The reason why we do so is that the adversary could generate any kind of bid values set. In some cases, to guarantee the performance, we have to admit the highest bid value, e.g., all bid values are 0 except the highest one $\max_i b_i = 1$. By waiting a certain number of time slots, we can ensure that we will see the highest bid with a constant probability. Later, we will prove that the decision algorithm $\mathcal{M}_1(\mathcal{D})$ could admit the request with highest bid value in each decision phase with constant probability. Current decision

Algorithm 6 Decision algorithm $\mathcal{M}_1(\mathcal{D})$ for mechanism \mathcal{M}_1

Input: Parameter τ returned from [Algorithm 5](#), constant parameter $\alpha \leq 1$ and β , requests arrived during $[\delta T, T]$.

Output: Allocation and charging for each coming request.

- 1: Start a new decision phase at beginning.
- 2: A request $\mathbf{e}_i = (b_i, a_i, t_i)$ will be admitted during a decision phase if the following conditions are met:
 - 1) No conflict request is admitted and running,
 - 2) $\tau \leq t_i \leq 2\tau$,
 - 3) Request \mathbf{e}_i arrives during a time interval $[s_i, s_i + \beta t_i]$ for a s_i such that $\alpha t_i \leq s_i \leq t_i$ where $\alpha \leq 1$ is a constant, and
 - 4) The bid value b_i by the user satisfies that $b_i \geq p_{\leq s_i}^{(1)}$.
- 3: Decide if current decision phase will end or not. The clock of each new decision phase is reset as 0.
- 4: For each request \mathbf{e}_i admitted, it will be charged $p_{\leq s_i}^{(1)}$ per unit time. The total charge will be $\mathbf{p}_i = p_{\leq s_i}^{(1)} \cdot t_i$. For each request \mathbf{e}_i which is not admitted, it will be charged 0.

phase will end if (1) either a request is admitted and finished (2) or no request is admitted after $2(1 + \beta)\tau$ time slots for a given constant β . Another new decision phase will start when a previous decision phase ends. For simplicity of presentation, we assume that the clock of each new decision phase is reset as 0. This means that, in [Algorithm 6](#), when we say a request arrived in time t_i , it actually arrived t_i time slots after this new decision phase started. [Algorithm 6](#) summarizes the allocation method.

Theorem 15 Mechanism \mathcal{M}_1 is strategyproof.

Proof We should prove that no request could make more profit by bidding other than its true valuation or/and announcing larger time requirement. In other words, we need to prove *value-SP*, *time-SP*, and *value- and time-SP*. We will prove these properties separately.

Value-SP: According to the pricing scheme of \mathcal{M}_1 , we know that each request \mathbf{e}_i will be charged $p_{\leq s_i}^{(1)} t_i$ and make $\mathbf{v}_i - p_{\leq s_i}^{(1)} t_i$ profit if it is admitted. Request \mathbf{e}_i will be charged 0 and make 0 profit if it is not admitted. Since $p_{\leq s_i}^{(1)}$ does not depend on b_i , for a request \mathbf{e}_i admitted by mechanism \mathcal{M}_1 , it cannot improve its profit by lying on its bid value b_i .

On the other hand, for a request \mathbf{e}_i which is not admitted by mechanism \mathcal{M}_1 , there are two possible reasons causing its rejection. The first reason is that \mathbf{e}_i does not appear in a time interval $[s_i, s_i + \beta t_i]$ that satisfies the conditions in mechanism \mathcal{M}_1 . For this reason, no matter how request \mathbf{e}_i lies on its bid b_i , it will never be admitted. The second reason is that \mathbf{e}_i bids $b_i < p_{\leq s_i}^{(1)}$ while it bids truthfully, i.e., $\mathbf{v}_i = b_i t_i < p_{\leq s_i}^{(1)} t_i$. For this reason, if request \mathbf{e}_i bids lower than its true valuation, i.e., $b_i t_i < \mathbf{v}_i$, it still cannot be admitted and its profit is still 0; if request \mathbf{e}_i bided higher than its true valuation, i.e., $b_i t_i > \mathbf{v}_i$, there are two cases. Case 1: $b_i < p_{\leq s_i}^{(1)}$,

request \mathbf{e}_i is still not admitted and no more profit is made. Case 2: $b_i \geq p_{\leq s_i}^{(1)}$, request \mathbf{e}_i is admitted and make $\mathbf{v}_i - b_i t_i < 0$ profit, which implies request \mathbf{e}_i loses profit.

As stated above, no matter how request lies on its bid value, it cannot make more profit.

Time-SP: Similarly, we know each request \mathbf{e}_i will make $\mathbf{v}_i - p_{\leq s_i}^{(1)} t_i$ profit when it is admitted; otherwise, \mathbf{e}_i will make 0 profit when it is not admitted.

For request \mathbf{e}_i which is admitted by mechanism \mathcal{M}_1 , if it announces a larger required time duration $t'_i > t_i$, \mathbf{e}_i can be still admitted or not admitted. If \mathbf{e}_i is not admitted due to lying on t_i , it loses profit due to lying. If \mathbf{e}_i is still admitted, it needs to pay $p_{\leq s'_i}^{(1)}$ per unit time where $s'_i \geq s_i$ is the parameter described in mechanism \mathcal{M}_1 . According to the definition of $p_{\leq s'_i}^{(1)}$, we know that $p_{\leq s'_i}^{(1)} \geq p_{\leq s_i}^{(1)}$ when $s'_i \geq s_i$. Therefore, request \mathbf{e}_i makes no more profit by lying since $\mathbf{v}_i - p_{\leq s'_i}^{(1)} t'_i < \mathbf{v}_i - p_{\leq s_i}^{(1)} t_i$ when $p_{\leq s'_i}^{(1)} \geq p_{\leq s_i}^{(1)}$ and $t'_i > t_i$.

For request \mathbf{e}_i which is not admitted by mechanism \mathcal{M}_1 , there are two possible reasons causing its rejection. The first reason is that \mathbf{e}_i does not appear in a time interval $[s_i, s_i + \beta t_i]$ that satisfies the conditions in mechanism \mathcal{M}_1 . For this reason, no matter how request \mathbf{e}_i announces a larger t_i , it will never be admitted. The second reason is that \mathbf{e}_i bids $b_i < p_{\leq s_i}^{(1)}$ while it bids truthfully, i.e., $\mathbf{v}_i = b_i t_i < p_{\leq s_i}^{(1)} t_i$. For this reason, if request \mathbf{e}_i announces a larger $t'_i > t_i$, \mathbf{e}_i may be admitted when $b_i \geq p_{\leq s'_i}^{(1)}$ where s'_i is the new parameter when t'_i is announced. Since $t'_i > t_i$, we have $s'_i \geq s_i$ according to the mechanism. Therefore, even if \mathbf{e}_i is admitted due to lying higher on its required time slots, it makes $\mathbf{v}_i - b_i t'_i < \mathbf{v}_i - p_{\leq s'_i}^{(1)} t'_i \leq \mathbf{v}_i - p_{\leq s_i}^{(1)} t_i \leq 0$ since $p_{\leq s'_i}^{(1)} \geq p_{\leq s_i}^{(1)}$ when $s'_i \geq s_i$.

Value- and time-SP: When request lies on its bid value and time requirement, we can consider it as a lie on bid value and another lie on time requirement. As we proved above, neither of these could make more profit, we finish this part. \square

Lemma 9 At any decision phase, mechanism \mathcal{M}_1 could admit the request with highest bid value and charge that request at second highest bid value during that phase with a probability at least $\frac{\alpha\beta}{2(\beta+2)^2}$.

Proof To admit a request \mathbf{e}_i in a decision phase, the required time duration t_i should be within $[\tau, 2\tau]$. We know that $\Pr(\tau \leq t_i \leq 2\tau) \geq \frac{1}{2}$ according to the learning algorithm $\mathcal{M}_1(\mathcal{L})$. To admit a request \mathbf{e}_i with highest bid value with a charge equal to the second highest bid during a decision phase, the request with second highest bid value should arrive during time interval $[0, s_i]$ and the request with highest bid value should arrive during time interval $[s_i, s_i + \beta t_i]$.

Since the arrival of requests is memoryless, the number of arrival during a time interval only depends on the length of that interval. The probability that the request with second highest bid arrives during $[0, s_i]$ is at least $\frac{s_i}{s_i + \beta t_i + t_i} \geq \frac{\alpha}{2 + \beta}$. The probability that the request with highest bid value arrives during $[s_i, s_i + \beta t_i]$ is at least $\frac{\beta t_i}{s_i + \beta t_i + t_i} \geq \frac{\beta}{2 + \beta}$.

Thus, at each decision phase, the mechanism could admit the request with highest bid value and charge it at second highest bid value with constant probability $\frac{1}{2} \frac{\alpha}{2+\beta} \frac{\beta}{2+\beta} = \frac{\alpha\beta}{2(2+\beta)^2}$. \square

Lemma 10 At each decision phase, mechanism \mathcal{M}_1 is $\frac{2(2+\beta)^3}{\alpha\beta}$ -competitive during that phase with respect to offline Vickrey auction for both social and revenue efficiency.

Proof According to Lemma 9, we know that \mathcal{M}_1 has a constant probability $\frac{\alpha\beta}{2(2+\beta)^2}$ to admit the request with highest bid and charge it with second highest bid. We know that decision phase lasts at most $s_i + \beta t_i + t_i \leq (2 + \beta)t_i$ time when e_i is admitted. The maximum total valuation is at most $(2 + \beta)b_i t_i$. The total valuation of winner of \mathcal{M}_1 is at least $b_i t_i$ with probability $\frac{\alpha\beta}{2(2+\beta)^2}$. Therefore, the social efficiency ratio is at least $\frac{\alpha\beta}{2(2+\beta)^3}$ which implies \mathcal{M}_1 is at least $\frac{2(2+\beta)^3}{\alpha\beta}$ -competitive for social efficiency.

Similarly, it is easy to show that \mathcal{M}_1 is also at least $\frac{2(2+\beta)^3}{\alpha\beta}$ -competitive for revenue efficiency. We finish the proof. \square

This lemma shows that, in *single* decision phase, the protocol does manage to get a social efficiency and revenue within a constant factor of optimum. This does not directly imply that the protocol is overall a constant approximation for social and revenue efficiency because many decision phases exist in $[0, T]$.

Definition 7 We call a decision phase *good* decision phase if a request is admitted during this phase. Otherwise, we call it *bad* decision phase.

We proved that for *good* decision phase, \mathcal{M}_1 is $\frac{2(2+\beta)^3}{\alpha\beta}$ -competitive for both social and revenue efficiency. According to the mechanism, the length of a *bad* decision phase is always $2(1 + \beta)\tau$. In following lemma, we prove that the expected length of a *good* decision phase is at least $(1 + \alpha)\tau$.

Lemma 11 The expected length of a good decision phase is at least $(1 + \alpha)\tau$.

Proof When a request e_i is admitted in a decision phase, the length of that decision phase is at least $s_i + t_i \geq (1 + \alpha)t_i$. So the expected length of such decision phase is at least $(1 + \alpha)t_i \geq (1 + \alpha)\tau$ since the decision algorithm guarantees that $t_i \geq \tau$ for all admitted request e_i . \square

Theorem 16 Mechanism \mathcal{M}_1 is $\Theta(1)$ -competitive with respect to offline Vickrey auction for both social and revenue efficiency.

Proof According to Lemma 9, we know that during time interval $[\delta T, T]$, for each decision phase, with probability $P = \frac{\alpha\beta}{2(2+\beta)^2}$, we have a *good* decision phase with expected length at least $(1 + \alpha)\tau$, and with probability $1 - P$, we have a *bad* decision phase with length at most $2(1 + \beta)\tau$. Thus, after each decision phase, the expected total length of all *good* decision phases increases $P(1 + \alpha)\tau$ and the expected total

length of all *bad* decision phases increases $2(1-P)(1+\beta)\tau$. So the expected total length of all *good* decision phases is $\frac{P(1+\alpha)}{P(1+\alpha)+2(1-P)(1+\beta)}(1-\delta)T = \Theta(1)T$.

We know that \mathcal{M}_1 is $\frac{2(2+\beta)^3}{\alpha\beta}$ -competitive for both social and revenue efficiency during each of the *good* decision phases. Then, \mathcal{M}_1 is at least $\frac{2(2+\beta)^3}{\alpha\beta} \frac{P(1+\alpha)+2(1-P)(1+\beta)}{P(1+\alpha)} \frac{1}{1-\delta}$ -competitive for both social and revenue efficiency. \square

4.2 Random-Arrival Case

In this section, we consider the random-arrival case where the bid values and required time durations follow some known/unknown distributions. We will propose an online mechanism \mathcal{M}_2 with performance analysis. Note that in this section, the time is also assumed to be slotted.

If the distributions or arrival rate is unknown, similar to previous protocol \mathcal{M}_1 , in the first δT ($\delta < 1$) time, we learn the distributions and arrival rate based on the information of arriving requests [21]. We already proved that the sample size is large enough to achieve relatively small estimate error even that δ is very small. So in this section, we mainly focus on the case where the central authority knows the distributions and the arrival rate in advance.

Since we know all distributions, at each time t , we can compute the expected social efficiency of the offline Vickrey auction from the current time slot t to T of the spectrum channel by *dynamic programming (DP)*. Let $V(t)$ denote this expected social efficiency from time slot t to T of offline Vickrey auction, and let event X_k denote that k requests arrive at same time, event Y_t denote requested time duration of a request is t , and the event Z_b denote bid value of a request is b . It is not difficult to derive that

$$V(t) = \sum_{k=0}^{+\infty} \mathbf{Pr}(X_k) \left(\sum_{t_i=1}^{T-t} \mathbf{Pr}(Y_{t_i})(b_m(k) \cdot t_i + V(t+t_i)) \right).$$

Here, $b_m(k)$ is the expected highest bid value among k requests' bid values. It is easy to show that $\int_{-\infty}^{b_m(k)} \mathbf{Pr}(Z_x) dx = \frac{1}{k+1}$. And we have following lemma.

Lemma 12 *Function $V(t)$ is monotonic nonincreasing as t grows from 0 to T .*

Proof First, it is trivial to show that $\forall t \in [0, T]$, $V(t) \geq 0$. Let $V^k(t)$ denote the expected social efficiency where k requests arrive at time t . For any k , we have

$$\begin{cases} V^k(t+1) &= \sum_{t_i=1}^{T-t-1} \mathbf{Pr}(Y_{t_i})(b_m(k)t_i + V(t+1+t_i)) \\ V^k(t) &= \sum_{t_j=1}^{T-t} \mathbf{Pr}(Y_{t_j})(b_m(k)t_j + V(t+t_j)). \end{cases}$$

Algorithm 7 Decision algorithm $\mathcal{M}_2(\mathcal{D})$ for mechanism \mathcal{M}_2

Input: Probability distribution of bid values, discrete probability distribution of requested durations, arrival rate λ and requests arriving at time t .

Output: Allocation and charging scheme

- 1: Compute $V(t)$, $\forall t \in [0, T]$.
- 2: **if** the channel is being used **then**
- 3: Reject all coming requests.
- 4: **else**
- 5: Admit request \mathbf{e}_i s.t. $i = \arg \max V'(t)$ and reject others. Here, $V'(t) = b_i t_i + V(t + t_i)$ if \mathbf{e}_i is admitted.
- 6: If a request \mathbf{e}_i is admitted, charge it $b_j t_j + V(t + t_j) - V(t + t_i)$ where \mathbf{e}_j is the request that makes the second highest $V'(t)$; if a request \mathbf{e}_i is not admitted, charge it 0.

Then, $V^k(t+1) - V^k(t) = -\Pr(Y_{(T-t)})((b_m(k)(T-t) + V(T)) \leq 0$. From above, we prove that for any k , $V^k(t)$ is monotonic nonincreasing as t grows from 0 to T , which implies that $V(t)$ is monotonic nonincreasing. \square

Notice that if all secondary users announce requests truthfully, $V(0)$ is actually the expected social efficiency and revenue for Vickrey auction. Next, we are going to give a scheme that guarantees the truthfulness of both bid values and requested time durations.

The admission method of the protocol is as following. We assume a virtual request \mathbf{e}_0 with bid value $b_0 = 0$ and required time duration $t_0 = 1$ arrive at each time. If the protocol admits \mathbf{e}_0 at some time, then it means no request will be admitted at that time. For example, if k requests arrive, together with the virtual request, we have totally $k+1$ requests. Let $V'(t)$ denote the expected social efficiency from time t to T after a decision is made at time t . We always admit a request \mathbf{e}_i which maximizes the expected total social efficiency

$$V'(t) = b_i t_i + V(t + t_i)$$

and reject others.

Assume that \mathbf{e}_j is the request which made the *second highest* expected social efficiency. If request \mathbf{e}_i ($i \neq 0$) is admitted, then the secondary user who proposed \mathbf{e}_i will be charged:

$$\mathbf{p}_i = b_j t_j + V(t + t_j) - V(t + t_i),$$

i.e., it will be charged as the second highest expected social efficiency excluding the expected social efficiency after he finishes. **Algorithm 7** summarizes the method.

Theorem 17 *Mechanism \mathcal{M}_2 is strategyproof.*

Proof We will prove *value-SP*, *time-SP*, and *value- and time-SP*.

Value-SP: If a request \mathbf{e}_i is admitted when the bid value is truthful, since the charging scheme does not rely on the bid value of the request itself, so lying on bid value will not bring more profit.

If a request \mathbf{e}_i is not admitted when the bid value is truthful, there are two cases when it lies on the bid value. If \mathbf{e}_i is still not admitted, it is trivial that the profit will not increase (always be 0). If \mathbf{e}_i is admitted by bidding b'_i , then it will be charged $b_j t_j + V(t + t_j) - V(t + t_i)$. Since \mathbf{e}_i is not admitted if bidding is truthful, we know that $b_i t_i + V(t + t_i) < b_j t_j + V(t + t_j)$ which implies the profit $b_i t_i - \mathbf{p}_i$ by reporting untruthfully is no more than 0. Thus, no matter how a secondary user lies on its bid value, it cannot make more profit.

Time-SP: Recall that we assume each request could only claim a longer required time duration than its actual requirement. If request \mathbf{e}_i is admitted when it claims its true time requirement, it will still be admitted when claiming a longer required time $t'_i > t_i$. Then, it will have to pay: $V(t + t'_i) - V(t + t_i)$ more than before. Since $V(t)$ is monotonic nonincreasing by Lemma 12, it does not make more profit by lying.

If a request \mathbf{e}_i is not admitted when claiming its time requirement truthfully, there are two cases when it lies on the time requirement. If \mathbf{e}_i is still not admitted, it is trivial that the profit will not increase (always be 0). If \mathbf{e}_i is admitted when it lies its time requirement as $t'_i > t_i$, it will be charged $b_j t_j + V(t + t_j) - V(t + t'_i)$. Since \mathbf{e}_i is not admitted if bidding is truthful, we know that $b_i t_i + V(t + t_i) < b_j t_j + V(t + t_j)$. Together with $V(t + t'_i) \geq V(t + t_i)$, we have

$$b_i t_i \leq b_j t_j + V(t + t_j) - V(t + t'_i)$$

which implies the profit is no more than 0.

In brief, no matter how request lies on its time requirement, it cannot make more profit.

Value- and time-SP: When a request lies on its bid value and time requirement, we consider it as a lie on bid value and another lie on time requirement. As we proved above, neither of these could make more profit, we finish this part. \square

Lemma 13 *At each time, mechanism \mathcal{M}_2 will admit a request with constant probability if the channel is not being used.*

Proof When the channel is empty, the probability that request \mathbf{e}_i can be admitted is $\mathbf{Pr}(b_i \cdot t_i + V(t_i) \geq V(t + 1))$. We can rewrite $V(t)$ as

$$V(t) = b \sum_{t_i=1}^{T-t} (Y_{t_i} \cdot t_i) + \sum_{t_i=1}^{T-t} (Y_{t_i} V(t + t_i)),$$

where $b = \sum_{k=0}^{+\infty} \mathbf{Pr}(X_k) \cdot b_m(k)$ is a constant. So the probability whether request \mathbf{e}_i can be admitted or not is only related to the current time t and the required time duration t_i .

When k requests arrive at the same time, the probability that the bid value of one request is greater than the expected maximum bid value $b_m(k)$ is $\frac{1}{k+1}$. Given the distributions of bid values, required time durations, and the arrival rate, when k requests arrive at time t , we can find a $c(t_i)$ such that

$$\Pr(b_i \cdot t_i + V(t + t_i) \geq V(t + 1)) \geq \frac{c(t_i)}{k + 1}.$$

Note that without the specific distributions, we cannot give the exact form of $c(t_i)$; however, after the distributions given, $c(t_i)$ can be calculated.

We use A_1 to denote the event that a request could be admitted (the expected social revenue exceeded the reserved social revenue $V(t + 1)$) and A_0 to denote the event that no request could be admitted. When k requests arrive at time t , the probability that no requests could be admitted is

$$\Pr(A_0|X_k) \leq \prod_{i=1}^k \left(1 - \frac{c(i)}{k+1}\right) \leq \left(1 - \frac{c_m}{k+1}\right)^k,$$

where $c_m = \min_{t=1}^T c(t)$ is a constant. The probability that a request could be admitted is $\Pr(A_1|X_k) \geq 1 - \left(1 - \frac{c_m}{k+1}\right)^k$.

At time t , the expected probability that some request could be admitted is

$$\begin{aligned} \Pr(A_1) &= \sum_{k=1}^{+\infty} \Pr(X_k) \cdot \Pr(A_1|X_k) \geq \sum_{k=1}^{+\infty} \Pr(X_k) \cdot \left(1 - \left(1 - \frac{c_m}{k+1}\right)^k\right) \\ &= 1 - e^{-\frac{c_m + o(1)}{2}} \cdot \sum_{k=1}^{+\infty} \left(\frac{\lambda^k \cdot e^{-\lambda - \frac{(c_m + o(1))(k-1)}{2(k+1)}}}{k!} \right) \geq 1 - e^{-\frac{c_m + o(1)}{2}} \end{aligned}$$

which is at least a constant. \square

Theorem 18 *Mechanism \mathcal{M}_2 is $\Theta(1)$ -competitive with respect to offline Vickrey auction for both social and revenue efficiency.*

Proof According to Lemma 13, we know a request could be admitted with probability greater than a constant P , thus no request could be admitted with probability smaller than $1 - P$ at each time. Then, it is easy to show that in expectation, the channel is busy for $\frac{P_t}{P_t+1-P} T$ time slots where $t \geq 1$ is the expected time requirement of admitted request.

When the channel is busy, the social efficiency and revenue efficiency of \mathcal{M}_2 is no less than that of Vickrey auction. Then, we know in expectation, \mathcal{M}_2 is $\frac{P_t+1-P}{P_t}$ -competitive. \square

4.3 General Conflict Graph Model

In this section, we show that the protocols can be easily extended for networks where the conflict graph is growth-bounded by a polynomial function f , which implies the one-hop independent number of conflict graph H is $f(1)$.

4.3.1 Semi-arbitrary-arrival Case

For the semi-arbitrary-arrival case, we do not need to change the learning algorithm of the protocol. Here is the new decision algorithm for the extended protocol \mathcal{M}'_1 . We define $p_{\leq \tau}^{(s)}(\mathcal{R})$ as s -th highest bid among all requests which is received during time interval $[0, \tau]$ and proposed by secondary users in set \mathcal{R} .

The theoretical analysis of performance is similar to that of previous section. We omit the details due to space limit. The main difference is that when H has a one-hop independent number $f(1)$, with certain probability, the protocol may accept a request e_i with highest bid value during certain time interval and miss at most $f(1)b_i$ bid values. Therefore, compared with what we did in Sect. 4.1, there is an additional factor $f(1)$ in both social and revenue efficiency ratio.

4.3.2 Random-Arrival Case

For the random-arrival case, we have new decision algorithm $\mathcal{M}'_2(\mathcal{D})$ for the extended online mechanism \mathcal{M}'_2 as following. We still use $V(t)$ to denote the expected revenue of offline Vickrey auction from time t to T . When the conflict graph H is not a completed graph, more than one requests could be running at same time. Assume \mathcal{R} is a set of requests; we use $V'_{\mathcal{R}}(t)$ to denote the expected revenue after a request is admitted at time t , and all requests in \mathcal{R} are running before it is admitted. The expected revenue can be achieved by dynamic programming

Algorithm 8 Decision algorithm $\mathcal{M}'_1(\mathcal{D})$ for mechanism \mathcal{M}'_1

Input: Parameter τ learned by learning algorithm and requests that arrive during $[\delta T, T]$.

Output: Allocation and charging scheme.

- 1: All secondary users start a new *decision phase* initially.
 - 2: A request e_i will be admitted during a decision phase if:
 - 1) e_i does not conflict with any running requests in geometry region and the secondary user proposed e_i is in its *decision phase*.
 - 2) $\tau \leq t_i \leq 2\tau$.
 - 3) Request e_i arrives during a time interval $[s_i, s_i + \beta t_i]$ for a s_i such that $\alpha t_i \leq s_i \leq t_i$ where $\alpha < 1$ is a constant.
 - 4) Bid value $b_i \geq p_{\leq s_i}^{(1)}(\mathcal{R})$ where R is the set of secondary users in *decision phase* and whose requests will conflict with e_i in geometry region.
 - 3: When request e_i is admitted, *decision phases* of the secondary user who proposed e_i and those secondary users in \mathcal{R} will end. Start *service phases* for these secondary users. These *service phases* end when e_i is finished.
 - 4: New *decision phase* will start when either *service phase* ends or no request is admitted after $2(1 + \beta)\tau$ time slots. Time of each new *decision phase* is reset as 0.
 - 5: For each request e_i admitted, it will be charged $p_{\leq s_i}^{(1)}(\mathcal{R})$ per time slot; otherwise, it will be charged 0.
-

Algorithm 9 Decision algorithm $\mathcal{M}'_2(\mathcal{D})$ for mechanism \mathcal{M}'_2

Input: Probability distribution of bid values, required time durations and arrival rate λ , a set of requests \mathcal{R} at time t , and a set of running requests \mathcal{C} .

Output: Allocation and charging scheme

- 1: **while** \mathcal{R} is not empty **do**
- 2: Admit request \mathbf{e}_i such that $i = \arg \max V'_{\mathcal{C}}(t)$ where \mathbf{e}_i does not conflict with any request in \mathcal{C} .
- 3: If \mathbf{e}_i is a dummy request, then $\mathcal{R} = \emptyset$. Otherwise, (1) let $\mathcal{R} = \mathcal{R} - \mathcal{R}_i$ where \mathcal{R}_i is a set of requests which conflicts with \mathbf{e}_i in geometry region (including \mathbf{e}_i itself), and (2) $\mathcal{C} = \mathcal{C} \cup \{\mathbf{e}_i\}$.
- 4: For each request \mathbf{e}_i admitted, it will be charged $b_j t_j + V(t + t_j) - V(t + t_i)$ where \mathbf{e}_j is the request that makes the second highest $V'_{\mathcal{C}}(t)$ before \mathbf{e}_i is admitted; otherwise, it will be charged 0.

as described in Sect. 4.2. Details are omitted due to space limit. In the process of dynamic programming, we may need to find maximum weighted independent set. Finding maximum weighted independent set problem is *NP-hard* [12] and hard to approximate within $n^{1-o(1)}$ factor [29]. However, using the growth-bounded property, we could find a $1 + \epsilon$ approximation in polynomial time [24]. Algorithms 9 summarizes the method.

The theoretical analysis of performance is similar to that of previous Section. Again, compared with what we did in Sect. 4.2, there is an additional factor $f(1)$ in both social and revenue efficiency ratios.

4.4 Simulation Results

In this section, we conducted extensive simulations to study the performance of the online mechanisms \mathcal{M}_1 and \mathcal{M}_2 . In the simulations, we set the total time $T = 1,000$ time slots. We generate random requests with random bid values and time requirements. The arrival of these requests follows *Poissonian* distribution with arrival rate λ , which varies in the simulations to simulate lightly loaded or heavily loaded system. We omit the learning phase since we want to study the performance of the online mechanisms and the influence of learning is proven neglectable when the number of total time slots is large enough.

The bid value and time requirement of each request are either *uniformly* distributed or *normally* distributed. Here, we generate four different sets of requests. In set 1, the bid value of each request is uniformly distributed in $[0, 1]$, and time requirement of each request is uniformly drawn from all integers in $[1, 50]$; in set 2, the bid value of each request is uniformly distributed in $[0, 1]$, and time requirement of each request is uniformly drawn from all integers in $[1, 500]$; in set 3, the bid value of each request is normally distributed with mean value $\mu_{bid} = 0.5$ and standard deviation $\sigma_{bid} = 2$, and time requirement of each request is normally distributed with mean value $\mu_{time} = 25$ and standard deviation $\sigma_{time} = 3$; in set 4, the bid value of each request is normally distributed with mean value $\mu_{bid} = 0.5$ and standard deviation $\sigma_{bid} = 2$, and time requirement of each request is normally distributed

with mean value $\mu_{time} = 250$ and standard deviation $\sigma_{time} = 3$. The bid values are randomly matched with the time requirements.

4.4.1 Mechanism \mathcal{M}_1 for Semi-arbitrary-arrival Case

We set $\alpha = 1$ and $\beta = 1$ in the experiments. In Fig. 10a, b, we plot the competitive ratio for social and revenue efficiency when the arrival rate λ varies for different requests sets. As we stated in Theorems 13 and 14, the social efficiency ratio is no more than 37 % and the revenue efficiency ratio is no more than 66 %. Therefore, the performance of the mechanism \mathcal{M}_1 in the simulations, both social efficiency ratio and revenue efficiency ratio are almost 20–30 %, is close to the theoretical upper bound. We also observe that the competitive ratios increase when the system load increases since it is easier for \mathcal{M}_1 to admit a request when the system load is high, i.e., the competition among requests is high.

In Fig. 10c, we plot the spectrum utilization ratio of the method \mathcal{M}_1 . The spectrum utilization ratio is defined as the ratio between the busy time of the spectrum channel and the total time T . In all cases, the utilization ratio is no more than 50 %. According to mechanism \mathcal{M}_1 , a request with time requirement t_i is admitted while at least αt_i time slots are empty. Since we set $\alpha = 1$, then the utilization ratio is no more than 50 % which is corroborated by the simulation result.

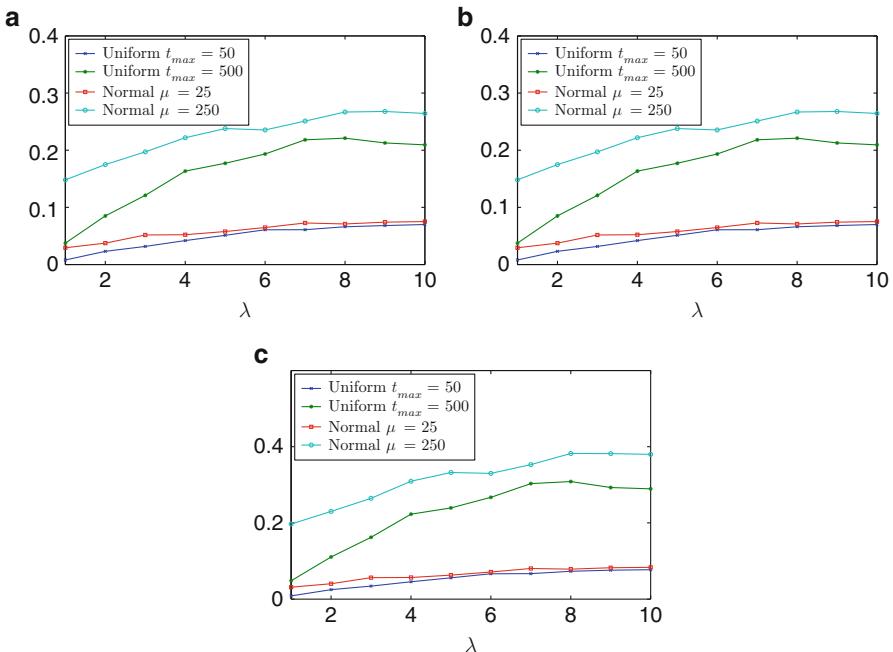


Fig. 10 The performance of online mechanism \mathcal{M}_1 . (a) Social efficiency ratio. (b) Revenue efficiency ratio. (c) Spectrum utilization ratio

4.4.2 Mechanism \mathcal{M}_2 for Random-Arrival Case

In Fig. 11a, b, we plot the competitive ratio for social and revenue efficiency when the arrival rate λ varies. Unsurprisingly, the performance of mechanism \mathcal{M}_2 is much better than that of \mathcal{M}_1 since \mathcal{M}_2 has more known information. We can see that the mechanism \mathcal{M}_2 even outperforms the average performance of Vickrey auction in all cases. The reason is that the protocol will choose the request maximizes the social efficiency and compare it with the average performance of Vickrey auction. In slightly loaded systems, the social efficiency and the revenue efficiency of the protocol are about 2–6 times of the performance by the Vickrey mechanism. The reason is that in slightly loaded system, the average performance of Vickrey auction is poorer, so the ratios are higher. Observe that the competitive ratios decrease when the arrival ratios increase. When the arrival ratio is big, the system load is high and the competition among requests is high. Then, it is more difficult to admit a request which outperforms the expected performance of Vickrey auction.

In Fig. 11c, we plot the spectrum utilization ratio of the method \mathcal{M}_2 . When the system is highly loaded, the spectrum utilization ratio of \mathcal{M}_2 is more than 95 %, while the spectrum utilization ratio is more than 75 % for lightly loaded system. When the system load is low, the competition among requests is low and thus it is harder to admit feasible request than that in a heavily loaded system. In all cases, mechanism \mathcal{M}_2 improves the efficiency ratios without sacrificing the spectrum utilization.

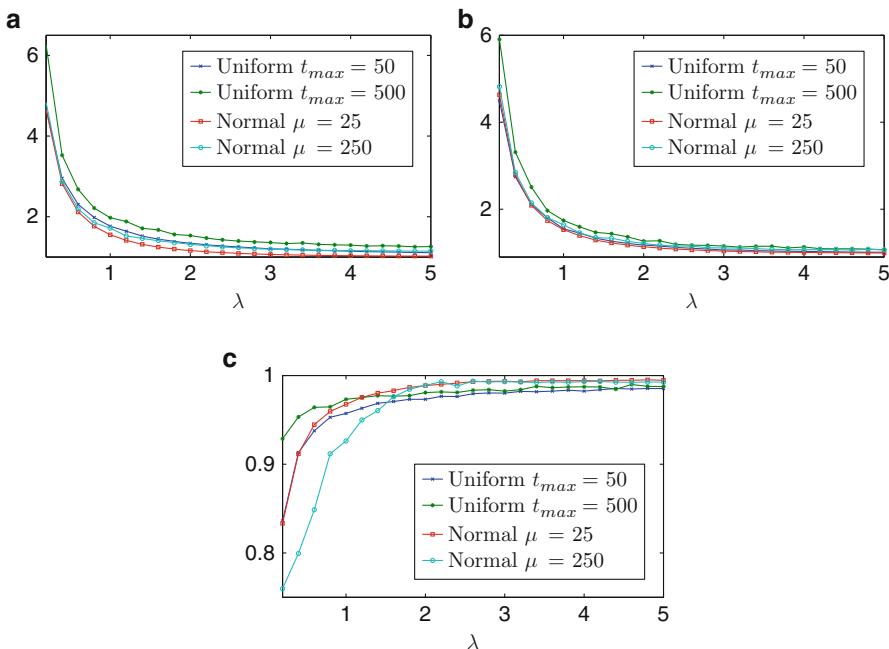


Fig. 11 The performance of online mechanism \mathcal{M}_2 . (a) Social efficiency ratio. (b) Revenue efficiency ratio. (c) Spectrum utilization ratio

5 Literature Reviews

The allocation of spectrums is essentially the combinatorial allocation problem, which has been well studied in the literature [2, 23]. Yuan et al. [40] introduced the concept of a time-spectrum block to model spectrum reservation in cognitive radio networks. Li et al. [25, 38] designed efficient methods for various dynamic spectrum assignment problems. They also showed how to design truthful mechanism based on those methods. Xu et al. [34] first studied online spectrum allocation when secondary users could bid arbitrarily. They designed efficient mechanisms that could thwart the selfish behavior of secondary users. In [35], Xu and Li designed protocols for spectrum allocation when requests are submitted in advance and the central authority only needs to make decisions within a certain time period. Zhou et al. [42] propose a truthful and efficient dynamic spectrum auction system to serve many small players. In [43], Zhou and Zheng designed truthful double spectrum auctions where multiple parties can trade spectrum based on their individual needs. In [31], Wang et al. designed an online version of truthful double auctions for spectrum allocation where the requests arrive in an online fashion. Ben-Porat et al. [6] gave a scheme scheduling decisions on the Cumulative Distribution Function (CDF). In [33], Wu and Tsang studied the distributed multichannel power allocation problem for the spectrum-sharing cognitive ratio networks. All these results are based on offline models.

In this work, we use the online model which is similar to the online job scheduling problems [15]. Various online scheduling problems focus on optimizing different objective functions. The most common objective function is *makespan*, which is the length of the schedule. The first proof of competitiveness of an online scheduling algorithm was given by Graham in 1966 [15]. This is one of the most basic problems in online scheduling. Suppose that we are given m identical machines, jobs arrive one by one, and no preemption is allowed. Graham [15] proved that greedy algorithm, which assigns a new job to the least loaded machine, is $2 - \frac{1}{m}$ -competitive. A number of results have been proved to improve the upper bounds [3, 10, 20] and lower bounds [19]. In 1992, Bartal et al. [3] gave an algorithm that is 1.986-competitive. Then, the bound was improved to 1.945 [20], to 1.923 [1], and finally to 1.9201 [10] which is the best upper bound known to date. And the best lower bound up to now for any deterministic algorithm is 1.88-competitive which was proposed by Rudin et al. [19]. Closing the gap between the best lower bound (1.88 [19]) and the upper bound (1.9201 [10]) is an open problem. Many authors [14, 27] also investigated the case where preemption is allowed without penalty. Phillips et al. [27] gave an $(8 + \epsilon)$ -approximation algorithm for preemptive single-machine scheduling problem. The approximation ratio was improved to 2 [13], and to 1.47 [14]. However, all these results did not consider the penalty of preemption. Bartal et al. [4] considered a version of online scheduling problem in which we pay penalty for rejecting jobs. This was improved later in [18] by Hoogeveen et al: they gave $(1 + \phi)$ -competitive algorithm, where ϕ is the golden ratio. Hoogeveen et al. [18] improved the ratio to 1.58. They assume that the penalty is job-dependent only and is not affected by the preemption time.

Above results focus on minimizing *makespan*. When jobs have deadlines, however, it is usually impossible to finish all jobs. Thus, another model aims to maximize the profit or number of completed jobs. There are different variants: preemption-restart, preemption-resume, and preemption-discard. In *preemption-restart* model, it is allowed to preempt a running job, and the preempted job has to be restarted again from the beginning. In *preemption-resume* model, it is allowed to preempt a running job, and the preempted job could resume from where it was preempted. And other models assume that a preempted job cannot be restarted or resumed. In 1991, Baruah et al. [5] proved that no online scheduling algorithm can make profit more than $\frac{1}{(1+\sqrt{D})^2}$ times the optimal. Koren et al. [22] gave an algorithm matching the lower bound in [5]. Woeginger [32] studied an online model of maximizing the profit of finished jobs where there is some relationship between the weight and length of job. He provided a four-competitive algorithm for tight deadline case and gave a matching lower bound. Hoogeveen et al. [17] gave a $\frac{1}{2}$ -competitive algorithm which maximizes the number of early jobs. They assume that preemption is allowed while no penalties will be charged. Chrobak et al. [8] gave a $\frac{2}{3}$ -competitive algorithm which maximizes the number of satisfied jobs that have uniform length in the *preemption-restart* model. Fung et al. [11] addressed a general model of nonuniform length and gave a $\Delta + 2\sqrt{\Delta} + 2$ -competitive algorithm. Zheng et al. [41] studied the *preemption-restart* model where a penalty is the weight of the preempted job. They gave a $3\Delta + o(\Delta)$ -competitive for $\Delta > 9$.

The work that is most similar to the work is a recent result by Constantin et al. [9]. They proposed and studied a simple model for auctioning ad slot reservations in advance. A seller will display a set of slots at some point T in the future. Until T , bidders arrive sequentially and place a bid on the slots they are interested in. The seller must decide immediately whether or not to grant a reservation. Their model allows the seller to cancel at any time any reservation made earlier with a penalty proportional to the bid value. The major differences between the model and their model are as follows. Their model considers online requests and *offline* services. The services (ad slots) start from a *fixed* time and last for one unit time. And the preemptions happen before services start. In the model, the services (spectrum usage) can start from *any* time, lasting for an *arbitrary* duration (subject to time bound Δ), and the preemptions happen after the spectrum usages are assigned. We also considered the spatial reuse of spectrum resources.

6 Conclusion

In this chapter, we studied online spectrum allocation for wireless networks where a set of secondary users will bid for leasing a spectrum channel for certain time duration in different locations. After the central authority received the bidding request, it has to make a decision within a certain time γ . For a number of variants, we reviewed efficient online scheduling algorithms and analytical proof of the constant competitive ratios of these methods. Especially, when γ is around

the maximum requested time duration Δ , the algorithm *TOFU* results in a profit that is almost optimum. For *TOFU* protocol, Xu and Li [36] showed that no user will bid lower than its willing payment under their mechanism *TOFU*. Under a simple assumption that the requests by secondary users arrive with the Poisson distribution and the willing payment per unit time slot is independent from the number of time slots required, Wang et al. [39] are able to prove that their *SALSA* protocols simultaneously approximately maximize the expected social efficiency and the expected revenue. They also analytically proved that every secondary user will maximize its expected profit if it proposed requests truthfully.

There are a number of interesting questions left for future research. It remains open to design a truthful mechanism for online spectrum auction when we do not know the distributions of user arrivals. It is also interesting to extend the *TOFU* mechanism to deal with different models, such as OFDM networks. It is also interesting to study the performance of the protocols when first price auction is used. In this case, secondary users may have tendency to lower their bids. Another challenge is to relax the wireless interference model used in both results. They assumed that the conflicts of spectrum usage among secondary users can be characterized by interference graphs. This assumption although is widely adopted in the literature and gives us some leverage in designing efficient protocols, it cannot perfectly reflect the interference in practice. Thus, it is important to extend the design of these protocols to network models with more complicated interference models.

Recommended Reading

1. S. Albers, Better bounds for online scheduling, in *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (ACM, New York, 1997), pp. 130–139
2. A. Archer, C. Papadimitriou, K. Talwar, E. Tardos, An approximate truthful mechanism for combinatorial auctions with single parameter agents. *Internet Math* **1**(2), 129–150 (2004)
3. Y. Bartal, A. Fiat, H. Karloff, R. Vohra, New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci.* **51**, 359–366 (1995)
4. Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall, L. Stougie, Multiprocessor scheduling with rejection, in *SODA* (Society for Industrial and Applied Mathematics, Philadelphia, 1996), pp. 95–103
5. S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, On-line scheduling in the presence of overload, in *FOCS* (IEEE Computer Society, Washington, DC, 1991), pp. 100–110
6. U. Ben-Porat, A. Bremler-Barr, H. Levy, On the exploitation of cdf based wireless scheduling, in IEEE, Rio de Janeiro (2009)
7. F.T. Bruss, A unified approach to a class of best choice problems with an unknown number of options. *Ann. Probab.* **12**(3), 882–889 (1984)
8. M. Chrobak, W. Jawor, J. Sgall, T. Tichý, Online scheduling of equal-length jobs: Randomization and restarts help. *SIAM J. Comput.* **36**(6), 1709–1728 (2007)
9. F. Constantin, J. Feldman, S. Muthukrishnan, M. Pal, An online mechanism for ad slot reservations with cancellations, in *SODA* (Society for Industrial and Applied Mathematics, Philadelphia, 2009), pp. 1265–1274
10. R. Fleischer, M. Wahl, On-line scheduling revisited. *J. Sched.* **3**, 343–353 (2000)

11. S.P. Fung, F.Y. Chin, C.K. Poon, Laxity helps in broadcast scheduling. *Theor. Comput. Sci.*, **370**, 251–264 (2005)
12. M. Garey, D. Johnson, *Computers and intractability: A guide to the theory of np-completeness* (Freeman, San Francisco, 1979)
13. M.X. Goemans, A supermodular relaxation for scheduling with release dates, in *Proceedings of the 5th International IPCO Conference on Integer Programming and Combinatorial Optimization* (Springer, London, 1996), pp. 288–300
14. M.X. Goemans, J.M. Wein, D.P. Williamson, A 1.47-approximation algorithm for a preemptive single-machine scheduling problem. *Oper. Res. Lett.* **26**(4), 149–154 (2004)
15. R. Graham, Bounds for certain multiprocessing anomalies. *Bell Syst. Tech. J.* **45**, 1563–1581 (1966)
16. M.T. Hajiaghayi, R. Kleinberg, D.C. Parkes, Adaptive limited-supply online auctions, in *EC* (ACM, New York, 2004), pp. 71–80
17. H. Hoogeveen, C.N. Potts, G.J. Woeginger, On-line scheduling on a single machine: maximizing the number of early jobs. *Oper. Res. Lett.* **27**(5), 193–197 (2000)
18. H. Hoogeveen, M. Skutella, G.J. Woeginger, Preemptive scheduling with rejection. In *Algorithms - ESA 2000* (2002), pp. 268–277
19. I.F.R. John, R. Chandrasekaran, Improved bounds for the online scheduling problem. *SIAM J. Comput.* **32**(3), 717–735 (2003)
20. D.R. Karger, S.J. Phillips, E. Torng, A better algorithm for an ancient scheduling problem, Society for Industrial and Applied Mathematics, Philadelphia in *SODA* (1994), pp. 132–140
21. S. Kern, S. Müller, N. Hansen, D. Büche, J. Ocenasek, P. Koumoutsakos, Learning probability distributions in continuous evolutionary algorithms—a comparative review. *Nat. Comput.* **3**(1), 77–112 (2004)
22. G. Koren, D. Shasha, Dover: An optimal on-line scheduling algorithm for overloaded uniprocessor real-time systems. *SIAM J. Comput.* **24**(2), 318–339 (1995)
23. D.J. Lehmann, L.I. O’Callaghan, Y. Shoham, Truth revelation in approximately efficient combinatorial auctions, in *ACM Conference on Electronic Commerce* (ACM, New York, 1999), pp. 96–102
24. X.-Y. Li, Y. Wang, Simple approximation algorithms and ptass for various problems in wireless ad hoc networks. *J. Parallel Distrib. Comput.* **66**(4), 515–530 (2006)
25. X.-Y. Li, P. Xu, S. Tang, X. Chu, Spectrum bidding in wireless networks and related, in *COCOON* (Springer, Berlin, 2008), pp. 558–567
26. R. Myerson, Optimal auction design. *Mathematics of operations research* **6**(1), 58 (1981)
27. C.A. Phillips, C. Stein, J. Wein, Scheduling jobs that arrive over time (extended abstract), in *WADS* (Springer, London, 1995), pp. 86–97
28. J.A. Stine, Spectrum management: The killer application of ad hoc and mesh networking, in *DySPAN*, Baltimore, 8–11 Nov 2005
29. L. Trevisan, Non-approximability results for optimization problems on bounded degree instances, in *STOC* (ACM, New York, 2001), pp. 453–461
30. V. Vapnik, A. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Appl.* **16**, 264–280 (1971)
31. S. Wang, P. Xu, X.-H. Xu, S. Tang, X.-Y. Li, X. Liu, Toda: Truthful online double auction for spectrum allocation, *IEEE DySpan*, IEEE, Singapore (2010)
32. G.J. Woeginger, On-line scheduling of jobs with fixed start and end times. *Theor. Comput. Sci.* **130**(1), 5–16 (1994)
33. Y. Wu, D.H.K. Tsang, Distributed multichannel power allocation algorithm for spectrum sharing cognitive radio networks, *IEEE INFOCOM*, IEEE, Rio de Janeiro (2009)
34. P. Xu, X. Li, Online market driven spectrum scheduling and auction. In *Proceedings of the 2009 ACM Workshop on Cognitive Radio Networks* (ACM, New York, 2009), pp. 49–54
35. P. Xu, X. Li, SOFA: Strategyproof online frequency allocation for multihop wireless networks, in *20th International Symposium on Algorithms and Computation (ISAAC)* (Springer, New York, 2009), p. 311

36. P. Xu, X. Li, TOFU: Semi-truthful online frequency allocation mechanism for wireless networks. *Netw. IEEE/ACM Trans.* **99**, 1 (2011)
37. P. Xu, X.-Y. Li, Oasis: Online algorithm for spectrum allocation in multihop wireless networks
38. P. Xu, X.-Y. Li, S. Tang, J. Zhao, Efficient and strategyproof spectrum allocations in multi-channel wireless networks. *IEEE Trans. Comput.* **60**(4), 580–593 (2011)
39. P. Xu, S. Wang, X. Li, SALSA: Strategyproof Online Spectrum Admissions for Wireless Networks. *IEEE Trans. Comput.* **59**(12), 1691–1702 (2010)
40. Y. Yuan, P. Bahl, R. Chandra, T. Moscibroda, Y. Wu, Allocating dynamic time-spectrum blocks in cognitive radio networks, in *MobiHoc* (ACM, New York, 2007), pp. 130–139
41. F. Zheng, Y. Xu, E. Zhang, On-line production order scheduling with preemption penalties. *J. Comb. Optim.* **13**, 189–204 (2007)
42. X. Zhou, S. Gandhi, S. Suri, H. Zheng, eBay in the Sky: strategy-proof wireless spectrum auctions, in *MobiCom* (ACM, New York, 2008), pp. 2–13
43. X. Zhou, H. Zheng, Trust: A general framework for truthful double spectrum auctions, in *IEEE INFOCOM* (IEEE, Rio de Janeiro, 2009)

Optimal Partitions

Frank K. Hwang and Uriel G. Rothblum*

Contents

1	Formulation and Classification of Partition Problems	2302
1.1	Sets of Partitions	2302
1.2	The Number of Characteristics Associated with each Partitioned Elements	2304
1.3	The Objective Function	2304
2	Preliminaries: Optimality of Extreme Points	2308
3	Single-Parameter: Polyhedral Approach	2313
4	Single-Parameter: Combinatorial Approach	2322
5	Multiparameter: Polyhedral Approach	2333
6	Multiparameter: Combinatorial Approach	2342
	Recommended Reading	2354

Abstract

This chapter provides a survey on the problem of partitioning points in the d -dimensional space with the goal of maximizing a given objective function; the survey is based on the forthcoming book (Hwang FK, Rothblum UG (2011b) Partitions: optimality and clustering. World Scientific). The first two sections describe the framework, terminology, and some background material that are useful for the analysis of the partition problem. The middle two sections study the case of $d = 1$, and the last two sections study the case of general d . Results

* Work of Uriel G. Rothblum was supported by The Israel Science Foundation (Grant 669/06).

F.K. Hwang
National Chiao Tung University (Retired), Hsinchu, People's Republic of China
e-mail: fkhwang@gmail.com

U.G. Rothblum
Faculty of Industrial Engineering and Management Science, Technion - Israel Institute of Technology, Haifa, Israel
e-mail: uri.rothblum@gmail.com

are presented without proofs, but with motivations, comments, and references. Complete proofs can also be found in the forthcoming book.

1 Formulation and Classification of Partition Problems

Consider a finite set \mathcal{N} of distinct positive integers (for most of the development $\mathcal{N} = \{1, \dots, |\mathcal{N}|\}$). A *partition* of \mathcal{N} is a finite collection of sets $\pi = (\pi_1, \dots, \pi_p)$ where π_1, \dots, π_p are pairwise disjoint nonempty sets whose union is \mathcal{N} . In this case, p is called the *size* of π , and the sets π_1, \dots, π_p are called the *parts* of π . Further, if n_1, \dots, n_p are the sizes of π_1, \dots, π_p , respectively, the vector (n_1, \dots, n_p) is called the *shape* of π ; of course, in this case, $\sum_{j=1}^p n_j = |\mathcal{N}|$. The prefix “ p -” or “ (n_1, \dots, n_p) -” is sometimes added to explicitly express the size or shape of a partition, referring to a p -*partition* of \mathcal{N} or to an (n_1, \dots, n_p) -*partition* of \mathcal{N} . Further, for brevity, the reference to the set \mathcal{N} as the partitioned set is frequently omitted, referring simply to *partitions*. In the forthcoming development, it is sometimes required that partitions’ parts are nonempty, while at other times, this requirement is relaxed.

Partition problems are (combinatorial) optimization problems in which a partition π is to be selected out of a given set Π so as to optimize (i.e., minimize or maximize) an objective function F that is defined over Π . Such problems are classified by (i) the *set of partitions* Π over which optimization takes place, (ii) the *number of characteristics* associated with each of the partitioned elements, and (iii) the *objective function* F that is defined over Π . It is next elaborated on each of these classifiers.

1.1 Sets of Partitions

At times, attention is restricted to the set of all partitions or to the set of all partitions whose size or shape satisfies prescribed restrictions; these situations are, respectively, referred to as *open*, *constrained-size*, and *constrained-shape sets of partitions*. Situations where the restrictions on the size or shape are expressed by prescribing a single element are referred to as *single-size* or *single-shape sets of partitions*, and situations where restrictions are expressed through lower and upper bounds on the size or shape are referred to as *bounded-size* or *bounded-shape sets of partitions*, respectively. All of the above classes of sets of partitions can be treated as special cases of a constrained-shape class. Their respective names simply emphasize the kind of constraints on the shapes. For example, the class of p -partitions collects those partitions with p (nonempty) parts, and the class of open partitions collects all partitions without restriction on the number of (nonempty) parts. Thus, constrained-shape class is the most general class. Still, the general framework of constrained-shape sets of partitions does not appear in the forthcoming development, and whenever constrained-shape partition problems are mentioned, all shapes have the same size; consequently, the terminology “constrained shape” is used for sets of

Table 1 Classification of sets of partitions

Open	Constrained-size	Constrained-shape (size given)
Bounded-size		Bounded-shape (size given)
Single-size		Single-shape

partitions with restricted shapes that have the same size (though formally, these are *single-size constrained-shape sets of partitions*). Sometimes, the above adjectives will be used casually to refer to partitions that belong to corresponding sets.

Let the (partitioned) set \mathcal{N} be given and let $n \equiv |\mathcal{N}|$. When a single-size set of partitions is considered, the prescribed single-size is given as a positive integer p . Given a positive integer p and a set Γ of positive integervectors (n_1, \dots, n_p) , each satisfying $\sum_{j=1}^p n_j = n$, Π^Γ will denote the corresponding constrained shape partitions, that is, all partitions with shape in Γ . In particular, if Γ consists of a single vector (n_1, \dots, n_p) , the notation $\Pi^{(n_1, \dots, n_p)}$ refers to (the single-shape set of partitions) Π^Γ . Also, if L and U are nonnegative integer p -vectors satisfying $L \leq U$ and $\sum_{j=1}^p L_j \leq n \leq \sum_{j=1}^p U_j$, $\Gamma^{(L,U)}$ will denote the set of nonnegative integervectors (n_1, \dots, n_p) satisfying $L_j \leq n_j \leq U_j$ for $j = 1, \dots, p$; in this case, the notation $\Pi^{(L,U)}$ is used for (the bounded-shape set of partitions) $\Pi^{\Gamma^{(L,U)}}$. (The restrictions on L and U assure that $\Gamma^{(L,U)}$ and $\Pi^{(L,U)}$ are nonempty.) Of course, single-size and single-shape sets of partitions are instances of bounded-shape sets obtained, respectively, by setting $L_j = 1$ and $U_j = |\mathcal{N}| - p + 1$ for all j or $L_j = n$ for all j . Similarly, open and single-size sets partitions are instances of bounded-size sets. The hierarchy of the classification of partitions is summarized in [Table 1](#).

A *multiset* is a group of elements where each element is allowed to have multiple occurrence. The formal notation of a multiset has double brackets, for example, $\{\{1, 1, 2, 2, 3\}\}$, or is given as a bracketed list of distinct elements with superscripts designating their duplications, for example, $\{1^2, 2^2, 3\}$. However, at times, (the abused notation of) single brackets is used, for example, $\{1, 1, 2, 2, 3\}$.

It is implicitly assumed in the above definitions that the parts of partitions are distinguishable. But, in some applications, the parts are indistinguishable and can be permuted without any restrictions. These situations are referred to as unlabeled partitions. Formally, an *unlabeled partition of \mathcal{N}* is a finite collection of sets $\pi = \{\pi_1, \dots, \pi_p\}$ where the π_j 's are as above. Again, p and the sets π_1, \dots, π_p are referred to as the *size* and the *parts of π* , respectively. Further, if n_1, \dots, n_p are the sizes of π_1, \dots, π_p , respectively, the *shape of π* is the multiset $\{\{n_1, \dots, n_p\}\}$; again, $\sum_{j=1}^p n_j = |\mathcal{N}|$ is required. In the literature, unlabeled partitions are commonly referred to as *allocations*. Herein, the term “partitions” is reserved to labeled ones; but, at times (when potential ambiguity may arise), there is reference to *labeled partitions*.

The classification to sets of labeled partitions applies to unlabeled ones; see [Table 1](#). Single-shape and bounded-shape sets of unlabeled partitions are defined, respectively, by a multiset $\{\{n_1, \dots, n_p\}\}$ or a multiset of pairs $\{\{(L_1, U_1), \dots, (L_p, U_p)\}\}$; the specification or the bounds on the sizes of the parts of partitions then hold for some labeling of the parts. Frequently, as parts of

unlabeled partitions are indistinguishable, sizes and bounds of unlabeled partitions are uniform, namely, all parts have the same size and a multiset of bounds $\{(L_1, U_1), \dots, (L_p, U_p)\}$ consists of p identical pairs.

1.2 The Number of Characteristics Associated with each Partitioned Elements

It is assumed throughout that each element i of the partitioned set \mathcal{N} is *associated* with a vector $A^i \in \mathbb{R}^d$ where d is a fixed positive integer (independent of i); the coordinates of A^i are referred to as *parameters* or *characteristics* associated with i . The n vectors A^i are part of the data for the problem and are given in the form of a real $d \times n$ matrix A . For a subset S of $\mathcal{N} = \{1, \dots, n\}$, A^S is the submatrix of A consisting of the columns of A indexed by S , ordered as in A . Also, “bars” over matrices are used to denote the multiset consisting of their columns, for example, a subset S of $\mathcal{N} = \{1, \dots, n\}$, \bar{A}^S is the set of columns of A^S , accounting for multiplicities.

1.3 The Objective Function

An objective function $F(\cdot)$ associates each (feasible) p -partition π with a value $F(\pi)$, and $F(\pi)$ is to be maximized (or minimized) over the given set of partitions. The value $F(\pi)$ of a partition π is assumed to depend on the parameters of the elements that are assigned to each part. More specifically, for each positive integer v , there is a column-symmetric function $h_v : \mathbb{R}^{d \times v} \rightarrow \mathbb{R}^d$, defined over multisets of v d -vectors; there are functions $g_j : \mathbb{R}^d \rightarrow \mathbb{R}^m$, $j = 1, \dots, p$, and a function $f_p : \mathbb{R}^{m \times p} \rightarrow \mathbb{R}$. Then the value $F(\pi)$ associated with partition π having shape (n_1, \dots, n_p) is given by

$$F(\pi) = f_p(g_1[h_{n_1}(\bar{A}^{\pi_1})], \dots, g_p[h_{n_p}(\bar{A}^{\pi_p})]) . \quad (1)$$

The functions h_{n_j} can, in a more general context, depend on the location within the variables of f_p , that is, on the index j ; also, the functions g_j may depend on n_j . In many common applications, each of the functions h_{n_j} is the summation function; in such cases, the problems are called *sum-partition problems*. When h_v or g_j is independent of the indexing parameter, the corresponding subscripts are dropped. Also, when referring to partitions of common size, the subscript “ p ” of f_p is dropped. Functions f_p are called *additive* when f_p is the sum function. It is also possible to consider partition problems where the domain of the functions h_{n_j} consists of ordered lists (and the functions h_{n_j} are not symmetric).

Some further notation is next introduced for a more concise form of sum-partition problems. For a $d \times n$ real matrix A and a p -partition $\pi = (\pi_1, \dots, \pi_p)$ of \mathcal{N} , the π -summation-matrix of A , denoted A_π , is given by

$$A_\pi \equiv \left[\sum_{i \in \pi_1} A^i, \dots, \sum_{i \in \pi_p} A^i \right] \in \mathbb{R}^{d \times p}, \quad (2)$$

where the empty sum is defined to be 0. When each of the g_j 's is the identity over \mathbb{R}^d , the objective function F associates with partition π the value $F(\pi)$ with the representation

$$F(\pi) = f_p(A_\pi). \quad (3)$$

(As was already mentioned, when the optimization over partitions concerns only partitions of fixed size p , the dependence of the functions f_p on p is suppressed).

Of particular interest is the case where all A^i 's have a common coordinate, say the first one, and it equals 1 for each A^i . In this case, row 1 of A_π is the shape of π . It follows that (3) allows for the objective function F to depend on the shape of partitions. Of course, for single-shape problems, the part-sizes (i.e., the coefficients of the shape) are fixed and can be viewed as parameters of the objective function.

In summary, the major classifiers of partition problems are:

- (1) *The family of partitions Π over which the function F (with representation as in (1) or (3)) is considered and optimized:* Using the classification of families of partitions provided in Table 1, there are *open*, *constrained-size*, *bounded-size*, *single-size*, *constrained-shape*, *bounded-shape*, and *single-shape partition problems*. In addition, *relaxed-size problems* refer to single-size problems which allow for empty parts. The description of the set of feasible partitions has to specify whether or not empty parts are allowed.
- (2) *The number of parameters associated with each of the partitioned elements:* The forthcoming development refers to *single-parameter problems*, *two-parameter problems*, and *multiparameter problems*.
- (3) *The objective (cost) function F as expressed by (1):* Adjectives like “sum-,” “max-,” or “mean-” of partition problems reflect properties of h_v while properties of f , like “linear,” “convex,” “Schur convex,” and “separable,” are explicitly referred to f_p , for example, sum-partition problems with f Schur convex.

A partition that maximizes/minimizes $F(\cdot)$ over a prescribed family of partitions Π is called *optimal over Π* .

For single-, bounded-, and constrained-shape families of partitions, the inclusion or exclusion of empty parts is implicit in the description of the set of feasible shapes – empty parts are prohibited when the set of feasible shapes consists only of positive vectors, whereas the inclusion of nonnegative shapes that are not (strictly) positive indicates that empty parts are allowed. In particular, results that apply to all single-, bounded-, or constrained-shape partition problems with empty parts allowed extend to corresponding problems with empty parts prohibited (by restricting attention to corresponding sets of shapes that consist only of positive vectors). Still, there are results in this chapter that apply to constrained-shape

problems that require the exclusion of empty parts. For sets of partitions that are determined by restrictions on size, for example, the case of single-size, the inclusion or exclusion of empty parts, must be made explicit as neither variant captures the other; see the results of the forthcoming Sect. 3 (summarized Table 4) about the optimality of monopolistic partitions for size problems with empty parts allowed and the optimality of extremal partitions for size problems with empty parts prohibited. Default positions in different parts of this chapter differ – at times they allow for empty parts, at times they exclude them, and at times they allow either way (to be determined through the specification of the set of feasible shapes).

The above classification refers to labeled partitions. Optimization problems over sets of unlabeled partitions can be embedded in the above framework, with Π and F being invariant under part permutation.

A major goal of this chapter is the identification of properties that are present in optimal partitions, thereby allowing one to restrict attention to corresponding subclasses of partitions. These properties are usually defined in terms of geometric/algebraic characteristics of the set of vectors associated with the indices assigned to the parts of the underlying partition. In particular, the focus is on properties that capture “clustering” of these vectors. The number of single-shape partitions is exponential in n (which implies the number of size partitions, open partitions, and other varieties of partitions are all exponential in n), so that it is not feasible to enumerate efficiently the partitions for selecting the optimal one. On the other hand, if a property has only polynomially many members, then it is possible to find an optimal partition by enumerating the subclasses of partitions having this property and comparing the values of the objective function.

Formally, a *property* Q of a partition is a unitary relation over sets of partitions, and it can be identified with the sets of partitions that satisfy it. Properties that are considered when studying multiparameter problems depend on the A^i ’s; as such, it seems natural to consider the partitioning of the A^i ’s rather than the partitioning of the underlying index set. But, possible equality among the A^i ’s often obscures the partition property. This situation is easily handled when the partitions are of the set of distinct indices and not of the A^i ’s.

A *type 1 result* for a partition problem is one that establishes a property for every optimal partition; a *type 2 result* is one that establishes a property for some optimal partition.

The next lemma, observed by Golany et al. [29], records a useful implication for the presence of properties in optimal solutions for single-shape/size and constrained-shape/size partition problems. For simplicity, attention is restricted to partitions whose size p is fixed.

Lemma 1 *Consider an objective function F over partitions, a set Γ of integer p -vectors whose coordinate-sum is n and a property Q of partitions such that each single-shape partition problem corresponding to a shape in Γ and the objective function F , Q is satisfied by some (every) optimal partition. Then, for each constrained-shape partition problem corresponding to a subset of Γ and the*

objective function F , Q is satisfied by some (every) optimal partition. Also, the above holds with “shape” replaced by “size.”

For single-parameter problems, the n elements in the single row of A are usually denoted $\theta^1, \dots, \theta^n$. In particular, for single-shape sum-partition problems, the π -summation matrix associated with a partition π is a p -vector, which is denoted θ_π and referred to as π -summation vector, that is,

$$\theta_\pi \equiv \left(\sum_{i \in \pi_1} \theta^i, \dots, \sum_{i \in \pi_p} \theta^i \right) \in \mathbb{R}^{1 \times p}. \quad (4)$$

Evidently, when $\theta^i = 1$ for each i , θ_π is the shape of π . For single-parameter problems, the partitioned elements can be renumbered so that θ^i is monotone in i , that is,

$$\theta^1 \leq \dots \leq \theta^n. \quad (5)$$

(Still, when analyzing the complexity of algorithm that solve partition problems, one should account for time required to sorting θ^i 's, if needed).

Much of the analysis focuses on partition problems with fixed size (single-shape, bounded-shape, constrained-shape, and single-size problems). When considering problems that allow for varying partition sizes, the objective function is, effectively, parameterized by an integer p representing the partition sizes; see (1). In particular, results about optimal partitions will then depend on some structure that expresses a connection between the objective function of p -partitions for different values of p . The most natural structure is the “*reduction assumption*” that asserts that F is invariant under the permutation/elimination of empty parts. For example, this is the case when f_p in (1) is the sum- (max/min)-function and the value assigned to empty parts is 0 ($-\infty/\infty$). A natural assumption that accompanies the “reduction assumption” is *symmetry* of F under part permutations.

Open problems that do not satisfy the reduction assumption can be difficult to analyze: They may have infinitely many feasible partitions, and there need not be an optimal one; this situation is demonstrated in the next example.

Example 1 Let $\mathcal{N} = \{1, 2\}$ and consider the open problem where empty sets are allowed and objective function is given by $F(\pi) = \sum_{j=1}^p j \left(\sum_{i \in \pi_j} i \right)$ for partitions π of size p . Then every single-size problem will assign both elements of \mathcal{N} to the part with the highest index. But, the open problem does not have an optimal partition as every p -partition is dominated by the best $(p+1)$ -partition. \square

When the “reduction assumption” is satisfied, each partition with p nonempty parts can, effectively, be viewed as (one of several) p' -partitions (with empty sets allowed) for any $p' \geq p$. In particular, upper bounded-size problems with upper bound p and open problem can be embedded in single-size problems with,

respectively, p or n as the prescribed size (and with empty parts allowed). The amount of computation of solving n -size problems may be high, but the reduction may be useful in establishing properties of optimal solutions. Further, in cases where an optimal solution for a single-size problem with empty parts excluded is simple, optimizing over the number of nonempty parts may be computationally tractable. When the reduction assumption is in effect, allowing for empty parts is a technical notion that helps us address bounded-size and open problems.

The following list provides a collection of applications of the optimal partition problem reported in the literature (number 11 has not yet appeared in the literature):

1. Assembly of systems (Derman et al. [17]; El-Newehbi et al. [22, 23]; Du [19]; Du and Hwang [20]; Malon [49]; Hwang et al. [41]; Hwang and Rothblum [36])
2. Group testing (Dorfman [18], Sobel and Groll [68], Gilstein [28], Pfeifer and Enis [59], Hwang et al. [41])
3. Circuit card library (Kodes [47]; Garey et al. [27])
4. Clustering (Fisher [24]; Boros and Hwang [7]; Golany et al. [29])
5. Abstraction of finite state machines (Oikonomou and Kain [58], Oikonomou [56, 57]).
6. Multischeduling (Mehta et al. [51] Rothkopf [62], Tanaev [69], Graham [31], Cody and Coffman [16], Chandra and Wang [13], Easton and Wong [21])
7. Cashe assignment (Gal et al. [26])
8. Blood analyzer problem (Nawijn [53])
9. Joint replenishment of inventory (Chakravarty et al. [11, 12], Goya [30], Silver [67], Nocturne [55], Chakravarty [10], A.K. Chakravarty 1983, Coordinated multi-product purchasing inventory decisions with group discounts, unpublished manuscript; A.K. Chakravarty, 1982b, Consecutiveness rule for inventory grouping with integer multiple constrained group-review periods, unpublished manuscript)
10. Statistical hypothesis testing (Neyman and Pearson [54])
11. Nearest neighbor assignment
12. Division of property (Granot and Rothblum [32])
13. Consolidating farm land (Brieden and Gritzmann [8]; Borgwardt et al. [5])

2 Preliminaries: Optimality of Extreme Points

This section presents sufficient conditions for the optimality of vertices, conditions that unify classical quasi-convexity and of Schur convexity. Throughout, standard definitions and facts about polytopes and convex sets are used; in particular, the *vertices* of a polytope are its *extreme points*.

Let C be a convex subset of \mathbb{R}^p and let $f : C \rightarrow \mathbb{R}$. The function f is (*strictly*) *quasi-convex along* a nonzero vector $d \in \mathbb{R}^p$, or briefly (*strictly*) d -*quasi-convex*, if the maximum of f over every line segment in C with direction d is attained (only) at one of the two endpoints of that line segment, possibly both. The function f is (*strictly*) *quasi-convex along* a set $D \subseteq \mathbb{R}^p$, or briefly (*strictly*) D -*quasi-convex*, if f is (*strictly*) quasi-convex along every nonzero vector $d \in D$. The function f

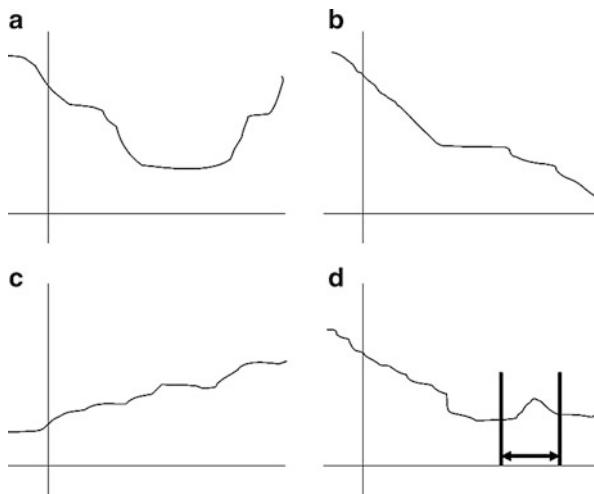


Fig. 1 Quasi-convexity

is (*strictly*) *quasi-convex* if it is (*strictly*) \mathbb{R}^p -quasi-convex. Reference to (*strict*) *directional quasi-convexity* is an abbreviation of (*strictly*) quasi-convexity along sets. Recall that a function f is (*strictly*) *convex* on C if for every distinct $x, y \in C$ and $0 < \alpha < 1$, $f[\alpha x + (1 - \alpha)y] \leq (<) \alpha f(x) + (1 - \alpha)f(y)$; of course, every (*strictly*) convex function is (*strictly*) quasi-convex.

The next lemma states a simple, but useful, characterization of directional quasi-convexity.

Lemma 2 *Let C be a convex subset of \mathbb{R}^p , $f : C \rightarrow \mathbb{R}$ and $d \in \mathbb{R}^p \setminus \{0\}$. Then the following are equivalent:*

- (a) *f is d -quasi-convex.*
- (b) *For each $x \in C$, the restriction of f to $C \cap \{x + \gamma d : \gamma \in \mathbb{R}\}$ (when viewed as a function in the parameter γ) cannot decrease after an increase.*

The behavior of a function, which is d -quasi-convex over line segments with direction d , can follow the patterns demonstrated in Figures (a), (b), or (c) of Fig. 1, but not (d) (where a line segment is identified over which the function does not attain a maximum at an endpoint).

Let C and D be subsets of \mathbb{R}^p where C is compact and convex. The set D is a *through-set* for C if for every point $x \in C$ which is not an extreme point of C , there is a line segment $L \subseteq C$ with direction in D such that x is in the relative interior of L . The following lemma identifies a necessary condition for through-sets. It also shows that this condition is sufficient for polytopes.

Lemma 3 *Let C be a convex set in \mathbb{R}^p . Then a subset D of \mathbb{R}^p is a through-set for C only if D contains a direction for each edge of C . Further, if C is a*

polytope, then every subset of \mathbb{R}^p that contains a direction for each edge of C is a through-set for C .

There are convex sets which have no edges, for example, the ℓ_2 -unit ball in \mathbb{R}^p . As the empty set is obviously not a through-set for any convex set of positive dimension, the sufficiency condition of [Lemma 2](#) does not generalize from polytopes to general convex sets.

Sufficient conditions for optimality of extreme points are next presented. The result was developed independently by Tardella [70] and Hwang and Rothblum [37]. It unifies the classical conditions of quasi-convexity and results about Schur convexity (see the paragraph following [Corollary 2](#)).

Theorem 1 *Let C be a compact convex set in \mathbb{R}^p , let E be the set of extreme points of C , and let D be a through-set for C , and let $f : C \rightarrow \mathbb{R}$ be D -quasi-convex. Then $\sup_C f = \sup_E f$; in particular, if f is continuous or if C is a polytope, f attains a maximum over C at one of C 's extreme points.*

Theorem 2 *Let C be a compact convex set in \mathbb{R}^p , let D be a through-set for C , and let $f : C \rightarrow \mathbb{R}$ be strictly quasi-convex along D . Then every maximizer of f over C is an extreme point of C ; in particular, if f is continuous or if C is a polytope, then such maximizers exist.*

Let C be a convex subset of \mathbb{R}^p and let $f : C \rightarrow \mathbb{R}$. The function f is (*strictly*) *edge-quasi-convex* on C if it is (*strictly*) quasi-convex along every (some) subset D of \mathbb{R}^p that contains a direction for each edge of C . Evidently, f is (*strictly*) edge-quasi-convex on C if and only if the maximum of f over any line segment in C which is parallel to one of C 's edges is attained at one of the two endpoints (and not at a point in the relative interior). Also, a (*strictly*) quasi-convex function is (*strictly*) edge-quasi-convex on every convex set over which it is defined.

Corollary 1 *Let P be a polytope and let $f : P \rightarrow \mathbb{R}$. If f is edge-quasi-convex on P , then f attains a maximum over P on one of P 's vertices. Further, if f is strictly edge-quasi-convex, then every maximizer of f over P is a vertex of P ; in particular, such a maximizer exists.*

The assumption in [Corollary 1](#) depends on the polytope P through its edge-directions, and the conclusion of the corollary specifically refers to P .

For $k \in \{1, \dots, p\}$, let e^k be the k -th *unit vector* in \mathbb{R}^p . Also, let $\Lambda \equiv \{(i, j) : i, j \in \{1, \dots, p\} \text{ and } i \neq j\}$, and for $(i, j) \in \Lambda$, let $e^{ij} \equiv e^i - e^j$. Let C be a convex subset of \mathbb{R}^p and $f : C \rightarrow \mathbb{R}$. The function f is called (*strictly*) *asymmetric Schur convex* if it is (*strictly*) quasi-convex along the set $\{e^{ij} : (i, j) \in \Lambda\}$. The use of the term “asymmetric” is to emphasize that these properties apply to functions which are not necessarily symmetric. Of course, every quasi-convex function, let alone a

convex function, is asymmetric Schur convex. (See the forthcoming definition of Schur convex functions).

The next lemma provides a characterization of asymmetric Schur convexity of continuously differentiable functions.

Lemma 4 *Let $C \subseteq \mathbb{R}^p$ be an open convex set of dimension p and let $f : C \rightarrow \mathbb{R}$ be continuously differentiable. Then f is asymmetric Schur convex on C if and only if*

$$\text{for all } x \in I^p \text{ and } i, j = 1, \dots, p, \left[\frac{\partial f}{\partial x_i} - \frac{\partial f}{\partial x_j} \right] [x + \gamma(e^i - e^j)] \quad (6)$$

cannot change sign from positive to negative as γ increases.

For a vector $x \in \mathbb{R}^n$ and $k = 1, \dots, n$, let $x_{[k]}$ be the k -th largest coordinate of x . A vector $a \in \mathbb{R}^p$ majorizes a vector $b \in \mathbb{R}^p$, written $a \succ b$, if

$$\sum_{i=1}^k a_{[i]} \geq \sum_{i=1}^k b_{[i]} \text{ for all } k = 1, \dots, p \quad (7)$$

and

$$\sum_{i=1}^p a_{[i]} = \sum_{i=1}^p b_{[i]}. \quad (8)$$

Evidently, (7) and (8) are, respectively, equivalent to

$$\max_{|I|=k} \sum_{i \in I} a_i \geq \max_{|I|=k} \sum_{i \in I} b_i \text{ for all } k = 1, \dots, p \quad (7a)$$

and

$$\sum_{i=1}^p a_i = \sum_{i=1}^p b_i. \quad (8a)$$

The vector a strictly majorizes b if a majorizes b , but b does not majorize a .

Let B be a convex subset of \mathbb{R}^p and let $f : B \rightarrow \mathbb{R}$. The function f is called *Schur convex on B* if

$$f(a) \geq f(b) \text{ for all } a, b \in B \text{ satisfying } a \succ b. \quad (9)$$

Condition (9) asserts that f is order-preserving with respect to the partial order majorization. The function f is called *strictly Schur convex* if it is Schur convex and (9) holds strictly whenever a strictly majorizes b .

Asymmetric Schur convexity is next related to classical Schur convexity. A subset B of \mathbb{R}^p is *symmetric* if B contains all vectors obtained by permuting

the coordinates of vectors in B . A real-valued function on a symmetric subset of \mathbb{R}^p is called *symmetric* if it is invariant under coordinate-permutations.

Theorem 3 *Let $I \subseteq \mathbb{R}$ be an open interval and $f : I^p \rightarrow \mathbb{R}$. Then the following are equivalent:*

- (a) *f is (strictly) Schur convex on I^p .*
- (b) *f is symmetric and (strictly) asymmetric Schur convex on I^p .*

As every (strictly) convex function is clearly (strictly) asymmetric Schur convex, **Theorem 3** implies that a symmetric (strictly) convex function is necessarily (strictly) Schur convex.

The next result relates asymmetric Schur convexity to edge quasi-convexity. Let \mathbb{R}_+ be the nonnegative real numbers.

Corollary 2 *Let $f : (\mathbb{R}_+)^p \rightarrow \mathbb{R}$. Then f is asymmetric Schur convex if and only if for every $\beta > 0$ the restriction of f to the polytope $\{x \in \mathbb{R}^p : x \geq 0, \sum_{i=1}^p x_i = \beta\}$ is edge-quasi-convex.*

Evidently, quasi-convex functions and Schur convex functions are asymmetric Schur convex. Thus, asymmetric Schur convexity unifies classical quasi-convexity and Schur convexity. The following example identifies functions that are asymmetric Schur convex but neither convex nor symmetric, let alone Schur convex.

Example 2 Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ be given by $f(x_1, x_2) = g(x_1 + x_2) + h(x_1 - x_2)$ with g arbitrary and h convex. To see that f is asymmetric Schur convex consider a line with direction $e^1 - e^2$, say $\{x + \gamma(e^1 - e^2) : \gamma \in \mathbb{R}\}$. It then follows that $f[x + \gamma(e^1 - e^2)] = g(x_1 + x_2) + h(x_1 - x_2 + 2\gamma)$, and this expression is clearly convex in γ . The function f need not be symmetric nor convex, for example, $f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2$. \square

Theorem 4 *Let P be a polytope all of whose edges have direction in $\{e^{ij} : (i, j) \in \Lambda\}$ and let $f : P \rightarrow \mathbb{R}$. If f is asymmetric Schur convex on P , then f attains a maximum over P at one of P 's vertices. Further, if f is strictly asymmetric Schur convex, then every maximizer of f over P is a vertex of P ; in particular, such a maximizer exists.*

Polytopes of the form $\{x \in \mathbb{R}^p : x \geq 0, \sum_{i=1}^p x_i = \beta\}$ with $\beta \geq 0$ clearly satisfy the assumption of **Theorem 4** about the direction of their edges. **Theorem 4** follows Hwang and Rothblum [37]. It streamlines the classical result about quasi-convexity being a sufficient condition for optimality of extreme points with a corresponding result about Schur convex functions. Interestingly, historically, these two conditions were considered distinct; in fact, Marshall and Olkin [50, p. 13] refer to the use of “convexity” in describing Schur convex functions as “inappropriate” and state that they adhere to the “historically accepted term ‘Schur convexity.’” **Figure 2** below provides a graphical illustration of the relationships between quasi-convexity, directional quasi-convexity, classical Schur convexity (for symmetric functions), and asymmetric Schur convexity.

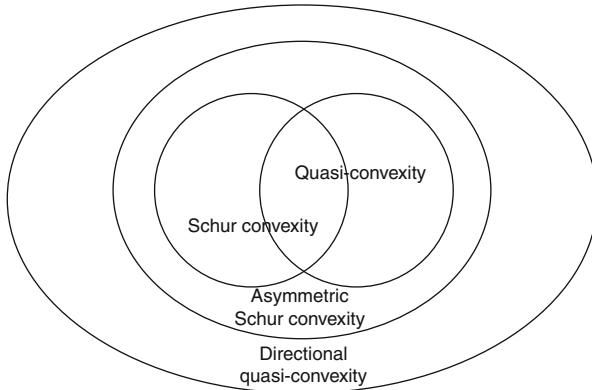


Fig. 2 Relationships among quasi-convexity, directional quasi-convexity, Schur convexity, and asymmetric Schur convexity

3 Single-Parameter: Polyhedral Approach

This section examines single-parameter partition problems. In particular, a class of polytopes, called *partition polytopes*, is introduced, and these polytopes are used to study partition problems.

Throughout this section and the next one, let (the partitioned set) $\mathcal{N} = \{1, \dots, n\}$, (the size of partitions) p and (the real numbers associated with the partitioned elements) $\theta^1, \dots, \theta^n$ be fixed. Without loss of generality, assume that the θ^i 's are ordered and satisfy

$$\theta^1 \leq \theta^2 \leq \dots \leq \theta^n ; \quad (10)$$

also, θ will stand for the vector $(\theta^1, \dots, \theta^n)$.

Recall from Sect. 1 that for a p -partition $\pi = (\pi_1, \dots, \pi_p)$ of \mathcal{N} , θ_π is given by

$$\theta_\pi \equiv (\theta_{\pi_1}, \dots, \theta_{\pi_p}) = \left(\sum_{i \in \pi_1} \theta^i, \dots, \sum_{i \in \pi_p} \theta^i \right) \in \mathbb{R}^p . \quad (11)$$

Given a set Π of ordered p -partitions, the Π -partition polytope is denoted $P^\Pi \equiv \text{conv} \{ \theta_\pi : \pi \in \Pi \} \subseteq \mathbb{R}^p$; when the set Π is either generic or clear from the context, the prefix “ $\Pi-$ ” is omitted. In forthcoming sections, partition problems are examined where the partitioned elements are associated with vectors rather than scalars; in the broader context, the above polytopes are called *single-parameter partition polytopes*.

The abbreviated notation P^Γ will be used for P^{Π^Γ} when Γ is a set of nonnegative integer p -vectors (n_1, \dots, n_p) satisfying $\sum_{j=1}^p n_j = n$, and such a polytope will be referred to as a *constrained-shape partition polytope*. Further, if Γ consists of a

single vector (n_1, \dots, n_p) , $P^{\Pi(n_1, \dots, n_p)}$ will stand for $P^{\Pi^{(n_1, \dots, n_p)}}$ and such a polytope will be referred to as a *single-shape partition polytope*; if L and U are nonnegative integer p -vectors satisfying $L \leq U$ and $\sum_{j=1}^p L_j \leq n \leq \sum_{j=1}^p U_j$, $P^{(L,U)}$ will stand for $P^{\Pi^{(L,U)}}$ and such a polytope will be referred to as a *bounded-shape partition polytope*; finally, if Π consists of all partitions of size p , P^P will stand for P^{Π^P} and such a polytope will be referred to as a *single-size partition polytope*.

The vertices of a Π -partition polytope P^Π have a representation θ_π with $\pi \in \Pi$. Thus, in seeking an optimal partition in Π when the objective function F has the representation $F(\pi) = f(\theta_\pi)$ for each $\pi \in \Pi$, it is useful to examine an extension of the functions f to P^Π (from $\{\theta_\pi : \pi \in \Pi\}$) and use results of Sect. 2 to determine conditions that suffice for this extension to attain a maximum over P^Π at a vertex.

Consider a single-shape problem corresponding to shape. If $n_j = 0$ for some $j = 1, \dots, p$, then for every partition $\pi \in \Pi^{(n_1, \dots, n_p)}$ π_j is empty and $(\theta_\pi)_j = 0$; so, part j is redundant. Consequently, such indices j can be eliminated, and there is no loss of generality by imposing the restriction that all n_j 's are positive. But, such a reduction needs not be possible when considering constrained-shape problems. The analysis of partition polytopes usually allows for empty parts (diverting from the default position that empty parts are prohibited).

The following example identifies an important class of partition polytopes.

Example 3 Consider the single-shape partition polytope defined by $(n_1, \dots, n_p) = (1, \dots, 1)$. In particular, in this case, $n = \sum_{j=1}^p 1 = p$. A partition π with shape $(1, \dots, 1)$ corresponds to a permutation of $(1, \dots, p)$, and for such a partition, θ_π is the p -vector obtained from the corresponding coordinate permutation of $(\theta^1, \dots, \theta^p)$. The partition polytope $P^{(1, \dots, 1)}$ is then the convex hull of these (permuted) vectors, and it is called the *generalized permutohedron* corresponding to $(\theta^1, \dots, \theta^p)$. The *standard permutohedron* is the generalized permutohedron with $\theta^i = i$ for $i = 1, \dots, p$. \square

Generalized permutohedra were first investigated by Schoute [63]; see also Ziegler [71, pp. 17–18 and 23].

A set of partitions Π is next associated with two additional polytopes; it will be shown that the definition of these polytopes provides useful representations for partition polytopes under general conditions. First, let C^Π be the solution set of the following system of linear inequalities:

$$\sum_{j \in I} x_j \geq \min \left\{ \sum_{j \in I} (\theta_\pi)_j : \pi \in \Pi \right\} \text{ for each } \emptyset \subset I \subseteq \{1, \dots, p\} \quad (12)$$

$$\sum_{j=1}^p x_j = \sum_{i=1}^n \theta^i; \quad (13)$$

evidently, C^Π is bounded and it is therefore a polytope.

An important property of partitions is next introduced; it will then be used to define the third polytope associated with a set of partitions. A partition is called *consecutive* if each of its parts consists of consecutive integers. Such a partition is determined by an order on its (nonempty) parts with a part preceding another when the indices of each element in the first part are smaller than each element in the second part. When the parts of a partition are nonempty, the order defining a consecutive partition is unique; further, a consecutive partition is uniquely determined by its order and its shape. In particular, for every nonnegative integer vector (n_1, \dots, n_p) , there exists a consecutive partition π with shape (n_1, \dots, n_p) . The third polytope associated with a set of partitions Π , denoted H^Π , is defined by $H^\Pi \equiv \text{conv}\{\theta_\pi : \pi \text{ is a consecutive partition in } \Pi\}$.

When $\Pi \equiv \Pi^{(n_1, \dots, n_p)}$ or $\Pi = \Pi^{(L, U)}$, superscripts (n_1, \dots, n_p) and (L, U) are used for $\Pi^{(n_1, \dots, n_p)}$ and $\Pi^{(L, U)}$, respectively, to index the polytope C and H , for example, $C^{(n_1, \dots, n_p)}$ and $H^{(L, U)}$. The next result relates the polytopes P^Π , C^Π , and H^Π for sets of partitions Π .

Lemma 5 *For every set Π of p -partitions, $H^\Pi \subseteq P^\Pi \subseteq C^\Pi$.*

In order to explore conditions under which the inclusions in Lemma 5 hold as equalities (yielding useful representations for partition polytopes), two polytopes associated with a real-valued functions on subsets are next introduced.

Let λ be a real-valued function on the subsets of $\{1, \dots, p\}$ with $\lambda(\emptyset) = 0$. A *permutation* (of $\{1, \dots, p\}$) is formally defined as an ordered collection of sets $\sigma = (\sigma_1, \dots, \sigma_p)$ where $\sigma_1, \dots, \sigma_p$ are singletons that partition $\{1, \dots, p\}$; given such a partition σ and $k \in \{1, \dots, p\}$, there is a unique index j with $\sigma_j = \{k\}$ – it is denoted $j_\sigma(k)$. Each permutation σ defines a vector $\lambda_\sigma \in \mathbb{R}^p$ whose k -th coordinate $(\lambda_\sigma)_k$ for $k = 1, \dots, p$ equals $\lambda(\cup_{t=1}^j \sigma_t) - \lambda(\cup_{t=1}^{j-1} \sigma_t)$ with $j = j_\sigma(k)$ (i.e., j as the unique index satisfying $\sigma_j = \{k\}$). The *permutation polytope corresponding to λ* , denoted H^λ , is the convex hull of the vectors λ_σ with σ ranging over all permutations of $\{1, \dots, p\}$. A second polytope associated with λ , denoted C^λ , is the solution set of the system of linear inequalities given by

$$\sum_{j \in I} x_j \geq \lambda(I) \text{ for each nonempty subset } I \text{ of } \{1, \dots, p\}, \quad (14)$$

$$\sum_{j=1}^p x_j = \lambda(\{1, \dots, p\}). \quad (15)$$

In the game theoretic literature, C^λ is referred to as the *core* of the p -person game defined by λ . Under this interpretation, $1, \dots, p$ are considered players, subsets of $\{1, \dots, p\}$ are called *coalitions*, $\lambda(I)$ for a coalition I is the *value* of I , and the vectors in C^λ represent allocations to the players with (14) assuring that the players in coalition I get *together* at least $\lambda(I)$ and (15) assuring that all players together get the value of the *grand coalition* $\{1, \dots, p\}$. These games and their cores were introduced by Morgenstern and Von Neuman [52] and explored in Shapley [66].

A real-valued function λ on the subsets of $\{1, \dots, p\}$ with $\lambda(\emptyset) = 0$ is called *supermodular* if for every pair I and J of subsets of $\{1, \dots, p\}$,

$$\lambda(I \cup J) + \lambda(I \cap J) \geq \lambda(I) + \lambda(J); \quad (16)$$

λ is called *strictly supermodular* if strict inequality holds whenever the two sets I and J are not ordered by set inclusion, that is, $I \not\subseteq J$ and $J \not\subseteq I$. Supermodular/submodular functions have extensive applications in optimization and polyhedral combinatorics.

The p standard unit vectors (in \mathbb{R}^p) will be denoted e^1, \dots, e^p . Parts (a) and (b) of the next theorem are due to Shapley [66], whereas part (c) is due to Hwang, Lee, and Rothblum [46].

Theorem 5 Suppose λ is supermodular on the subsets of $\{1, \dots, p\}$. Then:

- (a) $H^\lambda = C^\lambda$
- (b) For $v \in \mathbb{R}^p$, the following are equivalent:
 - (i) v is a vertex of H^λ
 - (ii) There is a permutation σ of $\{1, \dots, p\}$ with $v = \lambda_\sigma$;
in particular, H^λ has at most $p!$ vertices. Further, if λ is strictly supermodular, then the correspondence between vertices of H^λ and permutations of $\{1, \dots, p\}$ is one to one.
- (c) For distinct vertices v and v' of H^λ , the following are equivalent:
 - (i) $\text{conv}\{v, v'\}$ is an edge of H^λ
 - (ii) There exist permutations σ and σ' of $\{1, \dots, p\}$ such that $\{v, v'\} = \{\lambda_\sigma, \lambda_{\sigma'}\}$ and σ and σ' coincide on all but exactly two parts which are indexed by consecutive integers under both σ and σ' .

Further, if the above equivalent conditions hold and j and k are the elements in the noncoinciding parts of the permutations corresponding to v and v' , then $v - v'$ is a scalar multiple of $(e^j - e^k)$.

Theorem 5 will be used to provide conditions that suffice for having the inclusions in **Lemma 5** hold as equalities. For that purpose, for each set of partition Π , define the function θ_*^Π on the subsets of $\{1, \dots, p\}$ by setting

$$(\theta_*^\Pi)(I) \equiv \min \left\{ \sum_{j \in I} (\theta_\pi)_j : \pi \in \Pi \right\} \quad \text{for each } I \subseteq \{1, \dots, p\}; \quad (17)$$

in particular,

$$(\theta_*^\Pi)(\{1, \dots, p\}) = \sum_{i=1}^n \theta^i. \quad (18)$$

Trivially, $C^\Pi = C^{\theta_*^\Pi}$.

The set of partitions Π is called *complete* if for every permutation σ of $\{1, \dots, p\}$, there exists a partition π in Π with

$$\theta_\pi = (\theta_*^\Pi)_\sigma. \quad (19)$$

Note that for a permutation $\sigma = (\{j_1\}, \dots, \{j_p\})$ of $\{1, \dots, p\}$ and $t = 1, \dots, p$,

$$\sum_{s=1}^t [(\theta_*^\Pi)_\sigma]_{j_s} = (\theta_*^\Pi)(\{j_1, \dots, j_t\}) = \min_{\tau \in \Pi} \sum_{s=1}^t (\theta_\tau)_{j_s};$$

hence, $\theta_\pi = (\theta_*^\Pi)_\sigma$ for a partition π , if and only if

$$\sum_{s=1}^t (\theta_\pi)_{j_s} = \min_{\tau \in \Pi} \sum_{s=1}^t (\theta_\tau)_{j_s} \text{ for each } t = 1, \dots, p.$$

It is emphasized that completeness depends both on Π and on the θ^i 's.

The vector θ is called *one-sided*, if all θ^i 's are nonnegative or if all of them are nonpositive. Also, a constrained shape set of partitions is called *single symmetric shape* if the corresponding set of shapes consist of all coordinate-permutations of a single shape. The next result follows Hwang and Rothblum [39] in identifying several families of complete sets of partitions.

Theorem 6 *Every single-shape set of partitions is complete. If θ is one-sided, then every bounded-shape, single-size with empty parts allowed or prohibited, and single symmetric shape are complete.*

The following result is fundamental to the forthcoming development.

Theorem 7 *Let Π be a complete set of partitions. Then:*

- (a) θ_*^Π is supermodular.
- (b) $P^\Pi = C^\Pi = H^\Pi = H^{\theta_*^\Pi}$.
- (c) The direction of each edge of P^Π is the difference of two unit vectors.
- (d) v is a vertex of P^Π if and only if $v = (\theta_*^\Pi)_\sigma$ for some permutation σ of $\{1, \dots, p\}$, and in this case, $v = \theta_\pi$ for some consecutive partition π .
- (e) If no partition in Π has an empty set and the θ^i 's are distinct, then θ_*^Π is strictly supermodular, and for each vertex v , there is a unique permutation σ of $\{1, \dots, p\}$ satisfying $v = (\theta_*^\Pi)_\sigma$; in particular, P^Π has exactly $p!$ vertices.
- (f) If Π is constrained-shape, the θ^i 's are distinct, and either all θ^i 's are positive or they are all negative, then there is a unique partition π with $\theta_\pi = v$.

Theorem 7 shows that when the set of partitions Π is complete, the edge-directions of P^Π are differences of unit vectors, assuring (by [Theorem 4](#)) that asymmetric Schur convex functions over P^Π attain a maximum at a vertex. The next result states these conclusions formally, providing conditions for some (every) optimal partition to have its associated vector a vertex of the corresponding partition polytope.

Theorem 8 Suppose Π is a set of p -partitions and $f : P^\Pi \rightarrow \mathbb{R}$ where either f is quasi-convex, or Π is complete, and f is asymmetric Schur convex. Then:

- (a) A partition π^* is optimal if and only if θ_{π^*} maximizes f over P^Π .
- (b) There exists a partition π^* which is optimal over Π and has θ_{π^*} as a vertex of P^Π .
- (c) If f satisfies the corresponding property strictly on P^Π , then every partition π that is optimal over Π has θ_π as a vertex of P^Π .

Theorem 8 provides a tool for establishing properties of optimal partitions for a wide range of partition problems by identifying properties of partitions whose associated vectors are vertices of corresponding partition polytopes. Constrained shape problems are considered first, establishing the optimality of consecutive partitions.

Theorem 9 Suppose Π is a constrained-shape set of partitions and $f : P^\Pi \rightarrow \mathbb{R}$ is asymmetric Schur convex. Then:

- (a) There exists a partition which is optimal and consecutive; if furthermore either f is quasi-convex or Π is complete, then there exists a partition π which is optimal and consecutive and has θ_π as a vertex of P^Π .
- (b) If the θ^i 's are distinct and f is strictly asymmetric Schur convex, then every optimal partition is consecutive; if furthermore either f is strictly quasi-convex or Π is complete, then every optimal partition π is consecutive and has θ_π as a vertex of P^Π .

Jointly, [Theorems 8](#) and [9](#) present ten(!) results about optimal partitions – five providing conditions under which some optimal partition π is consecutive and/or has θ_π as a vertex of the partition polytope and five providing conditions under which every optimal partition has those properties. The “some” results are useful as they facilitate the restriction of the search for an optimal solution to subclasses of all partitions. When the set of partitions is complete, vertex enumeration is particularly efficient as the partition polytope is a permutation polytope having at most $p!$ vertices. The “some” results of [Theorems 8](#) and [9](#) are summarized in [Table 2](#). In this table (and the forthcoming ones in this section), “ Π^Γ ” under the “ Π ”-column means that Π is constrained-shape, “q-c” stands for quasi-convex, “(a)-S-c” stands for (asymmetric) Schur convex, an empty entry in the θ -column reflects no restrictions on θ , and a “+” in the “ P^Π perm”-column means P^Π is a permutation polytope. Other abbreviations are self-explanatory (e.g., “cons.” stands for “consecutive”).

Table 2 Properties of optimal partitions

Π	f	θ	Property of π^*	$\theta_{\pi^*} \in V(P^\Pi)$	P^Π perm	Ref
Complete	q-c			+		Theorem 8
	a-S-c			+	+	Theorems 8 and 7
Π^Γ	a-S-c		cons			Theorem 9
Π^Γ	q-c		cons	+		Theorem 9
$\Pi^\Gamma + \text{comp}$	a-S-c		cons	+	+	Theorems 9 and 7

Additional partition problems are next explored – their analysis requires additional definitions. A partition is *size-consecutive* (*reverse size-consecutive*) if it is consecutive and the larger elements are assigned to the bigger (smaller) parts. Also, a set of shapes is called *symmetric* if it contains every coordinate permutation of each of its members.

Theorem 10 Suppose Γ is a symmetric set of shapes, $\Pi = \Pi^\Gamma$, $\theta \geq 0$ ($\theta \leq 0$) and $f : P^\Pi \rightarrow \mathbb{R}$ is asymmetric Schur convex. Then:

- (a) There exists a partition which is optimal and size-consecutive (*reverse size-consecutive*); if furthermore either f is quasi-convex or Π is complete, then there exists a partition π which is optimal and size-consecutive (*reverse size-consecutive*), and has θ_π as a vertex of P^Π .
- (b) If the θ^i 's are distinct and f is strictly asymmetric Schur convex, then every optimal partition is size-consecutive (*reverse size-consecutive*); if furthermore either f is strictly quasi-convex or Π is complete, then every optimal partition π is size-consecutive (*reverse size-consecutive*) and has θ_π as a vertex of P^Π .

Theorem 11 Suppose Π is a constrained-shape set of partitions corresponding to a set of partitions Γ , $\theta \geq 0$ ($\theta \leq 0$) and $f : \mathbb{R}^P \rightarrow \mathbb{R}$ is Schur convex. Then:

- (a) There exists a partition which is optimal and size-consecutive (*reverse size-consecutive*); if furthermore Π is complete, then there exists a partition π which is optimal, size-consecutive (*reverse size-consecutive*), and has θ_π as a vertex of P^Π .
- (b) If the θ^i 's are distinct and f is strictly Schur convex, then every optimal partition is size-consecutive (*reverse size-consecutive*); if furthermore Π is complete, then every optimal partition π is size-consecutive (*reverse size-consecutive*) and has θ_π as a vertex of P^Π .

Theorem 11 does not include a variant that parallels the “quasi-convexity” of Theorem 10 as an alternative to completeness that allows one to augment (*reverse*) size-consecutiveness of optimal partitions with the property that θ_π is a vertex of P^Π . Situations where the last property can be used to accelerate the search for an optimal partition when Π is not complete are not known.

Table 2 is next augmented with the “some” results of Theorems 10 and 11 into Table 3.

Theorem 12 Suppose Π is a bounded-shape set of partitions, $f : P^\Pi \rightarrow \mathbb{R}$ is asymmetric Schur convex. Then (i) there exists a partition π which is optimal, consecutive, and has θ_π as a vertex of P^Π ; (ii) if the θ^i 's are distinct and f is strictly asymmetric Schur convex, then every optimal partition π is consecutive and has θ_π as a vertex of P^Π . Further, if f is Schur convex and $\theta \geq 0$ ($\theta \leq 0$), “consecutive” can be tightened to “size-consecutive (*reverse size-consecutive*).”

Of course, the conclusions of Theorem 12 apply to single-shape problems which are instances of bounded-shape problems.

Table 3 Properties of optimal partitions (continued)

$\Pi = \Pi^\Gamma$	f	θ	Properties of π^*	$\theta_{\pi^*} \in V(P^\Pi)$	P^Π perm	Ref.
Γ sym	a-S-c	≥ 0	s-cons			Theorem 10
	a-S-c	≤ 0	r-s-cons			Theorem 10
Γ sym	q-c	≥ 0	s-cons	+		Theorem 10
	q-c	≤ 0	r-s-cons	+		Theorem 10
Γ sym +complete	a-S-c	≥ 0	s-cons	+	+	Theorems 10 and 7
	a-S-c	≤ 0	r-s-cons	+	+	Theorems 10 and 7
	S-c	≥ 0	s-cons			Theorem 11
	S-c	≤ 0	r-s-cons			Theorem 11
Complete	S-c	≥ 0	s-cons	+	+	Theorems 11 and 7
Complete	S-c	≤ 0	r-s-cons	+	+	Theorems 11 and 7

A bounded-shape set of partitions is called *uniform bounded-shape* if both L and U are multiples of the vector $(1, \dots, 1)$ (i.e., the lower/upper bounds of all parts coincide).

Theorem 13 Suppose Π is a uniform bounded-shape set of partitions, $f : P^\Pi \rightarrow \mathbb{R}$ is asymmetric Schur convex and $\theta \geq 0$ ($\theta \leq 0$). Then (i) there exists a partition π which is optimal, size-consecutive (reverse size-consecutive), and has θ_π as a vertex of P^Π ; (ii) if the θ^i 's are distinct and f is strictly asymmetric Schur convex, then every optimal partition π is size-consecutive (reverse size-consecutive) and has θ_π as a vertex of P^Π .

An important instance of [Theorem 13](#) is the case where Π is single-symmetric shape.

The following additional properties of partitions are relevant only for size problem since the sets of shapes meeting the requirements of the definitions are severely restricted.

A (reverse) size-consecutive partition π with all parts, but one, consisting of a single element is called *(reverse) extremal*; if all parts, but one, are empty, π is called *monopolistic*. A consecutive partition is called *bipolar* if either of the following conditions holds: (i) It has two nonempty parts, one containing all indices i with $\theta^i < 0$ and the other containing all indices i with $\theta^i > 0$, and the two parts arbitrarily split the indices with $\theta^i = 0$, or alternatively (ii) if either $\theta \geq 0$ or $\theta \leq 0$ and the partition is monopolistic. Evidently, if all θ^i 's are positive or all of them are negative, a partition is bipolar if and only if it is monopolistic. Finally, the definition of the next property is lengthy. For $p > 1$, a consecutive partition π is called *polarized-extremal* if it has two distinguished parts, say $\pi_{j\oplus}$ and $\pi_{j\ominus}$, such that:

- (i) (a) If $i \in \pi_{j\oplus}$ and $u > i$, then $u \in \pi_{j\oplus}$.
(b) If $i \in \pi_{j\ominus}$ and $u < i$, then $u \in \pi_{j\ominus}$.
- (ii) (a) If $\pi_{j\oplus}$ is not a singleton, then $\pi_{j\oplus} \subseteq \{i \in \mathcal{N} : \theta^i \geq 0\}$.
(b) If $\pi_{j\ominus}$ is not a singleton, then $\pi_{j\ominus} \subseteq \{i \in \mathcal{N} : \theta^i \leq 0\}$.

- (iii) If $j \notin \{j_{\oplus}, j_{\ominus}\}$, then π_j is a singleton; further, if either $\pi_{j_{\oplus}}$ is not a singleton and $\theta^u = 0$ for some $u \in \pi_{j_{\oplus}}$, or $\pi_{j_{\ominus}}$ is not a singleton and $\theta^u = 0$ for some $u \in \pi_{j_{\ominus}}$, then the single element in π_j , say i , has $\theta^i = 0$.

When $\theta \geq 0$ ($\theta \leq 0$), every extremal (reverse-extremal) partition is polarized-extremal and when $\theta > 0$ ($\theta < 0$), a partition is extremal (reverse-extremal) if and only if it is polarized-extremal.

Example 4 Let $n = 7$ and $\theta = (-3, -2, -1, 0, 0, 1, 2)$. Then $(\{1, 2, 3\}, \{4\}, \{5, 6, 7\})$ and $(\{1, 2\}, \{3\}, \{4\}, \{5\}, \{6, 7\})$ are polarized-extremal partitions, but $(\{1, 2\}, \{3\}, \{4\}, \{5, 6, 7\})$ is not.

Theorem 14 Suppose Π is a single-size set of partitions with empty parts allowed, $f : P^{\Pi} \rightarrow \mathbb{R}$ is asymmetric Schur convex and $\theta \geq 0$ ($\theta \leq 0$). Then (i) there exists a partition π which is optimal, monopolistic, and has θ_{π} as a vertex of P^{Π} ; (ii) if the θ^i 's are distinct and f is strictly asymmetric Schur convex, then every optimal partition π is monopolistic and has θ_{π} as a vertex of P^{Π} .

Attention is next turned to the partition problems considered in [Theorem 14](#) without the restriction that the θ^i 's are one-sided.

Theorem 15 Suppose Π is a single-size set of partitions with empty parts prohibited and $f : P^{\Pi} \rightarrow \mathbb{R}$ is quasi-convex. Then (i) there exists a partition π which is optimal, polarized-extremal, and has θ_{π} as a vertex of P^{Π} ; (ii) if the θ^i 's are distinct and nonzero and f is strictly quasi-convex, then every optimal partition π is polarized extremal and has θ_{π} as a vertex of P^{Π} .

Table 4 Properties of optimal partitions (continued)

$\Pi = \Pi^{\Gamma}$	f	θ	Properties of π^*	$\theta_{\pi^*} \in V(P^{\Pi})$	P^{Π}	Ref
bdd-shape	a-S-c		consec	+		Theorem 12
bdd-shape	S-c	≥ 0	s-cons	+	+	Theorems 12, 6, and 7
	S-c	≤ 0	r-s-cons	+	+	Theorems 12, 6, and 7
Uniform	a-S-c	≥ 0	s-cons	+		Theorem 13
bdd-shape	a-S-c	≤ 0	r-s-cons	+		Theorem 13
Single	a-S-c	≥ 0	s-cons	+	+	Theorems 13, 6, and 7
sym-shape	a-S-c	≤ 0	r-s-cons	+	+	Theorems 13, 6, and 7
Single-size+	a-S-c	≥ 0	ext	+	+	Theorems 14, 6, and 7
no \emptyset parts	a-S-c	≤ 0	rev-ext	+	+	Theorems 14, 6, and 7
Single-size+ no \emptyset parts	q-c		polar-ext	+	+	Theorems 15, 6, and 7
Single-size+	a-S-c	≥ 0	monop	+	+	Theorems 16, 6, and 7
\emptyset parts ok	a-S-c	≤ 0	monop	+	+	Theorems 16, 6, and 7
Single-size+ \emptyset parts ok	a-S-c		Bipolar	+	+	Theorems 16, 6, and 7

When $\theta \leq 0$ or $\theta \geq 0$, the conclusions of [Theorem 15](#) follow from [Theorem 14](#).

Theorem 16 Suppose Π is a single-size set of partitions with empty parts allowed and $f : P^\Pi \rightarrow \mathbb{R}$ is asymmetric Schur convex. Then (i) there exists a partition π which is optimal, bipolar, and has θ_π as a vertex of P^Π ; (ii) if the θ^i 's are distinct and nonzero and f is strictly asymmetric Schur convex, then every optimal partition π is bipolar and has θ_π as a vertex of P^Π . Further, if $\theta \geq 0$ or $\theta \leq 0$, then “bipolar” in the above conclusions can be replaced by “monopolistic.”

[Sections 2](#) and [3](#) are next augmented with a summary of the “some” results of [Theorems 12–16](#) ([Table 4](#)).

4 Single-Parameter: Combinatorial Approach

The main goal in this section is to develop combinatorial tools for establishing that given a partition property Q and a set of partitions, some (every) partition in the set satisfies Q . Such results are useful for the analysis of partition problems when the set consists of all optimal partitions for a particular partition problem. The tool that is developed, referred to as *local sorting*, is to move from a partition in the given family that does not satisfy Q to a partition (in the family) that does by recursively rearranging (the elements in) a small number of parts at each step. Two conditions must be observed to guarantee that successive (local) sorting will produce a partition satisfying the given property Q have two parts: first, in each step, the sorting must produce a partition in the given set; and second, in order to avoid being trapped in a cycle, some statistic must decrease (or increase) monotonely as the sorting progresses. The first condition concerns the set of partitions, while the other concerns property Q . A useful approach of local sorting is the intuitive one which requires that the (sub)partition consisting of the sorted parts satisfies Q . These techniques will be applied to some specific optimization problems later, demonstrating the existence of optimal partitions with particular properties.

Some properties of partitions were introduced in [Sect. 3](#). These include (with abbreviations added in parenthesis) consecutiveness (C), size-consecutiveness (S), reverse size-consecutiveness (S^{-1}), extremalness (E), reverse extremalness (E^{-1}), monopolisticness (M_p), bipolar (BP), and polarized-extremalness (PE).

Let S and S' be two disjoint subsets of \mathcal{N} . Then S is said to *penetrate* S' , written $S \rightarrow S'$, if there exists a, c in S' and b in S such that $a < b < c$. Note that if $(S \cup S') \rightarrow S''$ then either $S \rightarrow S''$ or $S' \rightarrow S''$, but if $S \rightarrow (S' \cup S'')$, it is possible that neither $S \rightarrow S'$ nor $S \rightarrow S''$. In addition, the convention that each part penetrates itself will be adopted.

For a partition $\pi = \{\pi_1, \dots, \pi_p\}$ of a subset \mathcal{N}' of \mathcal{N} , define the digraph associated with π , denoted $G(\pi)$, as the digraph with p vertices representing the parts of π and edges (i, j) for $i, j \in \{1, \dots, p\}$ if π_i penetrates π_j . A partition π is called *noncrossing* (NC) if $\pi_i \rightarrow \pi_j$ implies $\pi_j \not\rightarrow \pi_i$. Noncrossing partitions were first studied in Kreweras [[48](#)].

Table 5 Properties of partitions

Noncrossing	NC	Consecutive	C
Nested	N	Nearly nested	$N_e N$
Size-consecutive	S	Reverse size-consecutive	S^{-1}
Extremal	E	Reverse extremal	E^{-1}
Polarized-extremal	PE	Order-nonpenetrating	O
Monopolistic	M_p	Bi-polar	BP

There are three special cases of noncrossingness that are of particular interest. The first concerns the consecutive partitions, introduced and studied in Sect. 3. A partition is called consecutive if and only if the penetration relation among its parts is an empty partial order, that is, no part penetrates any other part. The second instance of NC is *nestedness* (N); a p -partition π is called *nested* if the penetration relation among its parts is linear. Finally, a partition π is called *order-nonpenetrating* (O) if its parts can be indexed (ordered) such that for $i = 2, \dots, p$, $\pi_i \not\rightarrow \cup_{j=1}^{i-1} \pi_j$. Note that order-nonpenetrating was called *order-consecutive* by Chakravarty et al. [11]. Examples of partitions that satisfy the properties considered so far are next illustrated.

Example 5 Let $\mathcal{N} = \mathcal{N}' = \{1, \dots, 5\}$, $n = 5$ and $p = 3$. Then:

- (a) $\{\{2\}, \{4\}, \{1, 3, 5\}\}$ is noncrossing but, is not order-nonpenetrating.
- (b) $\{\{2\}, \{1, 3\}, \{4, 5\}\}$ is order-nonpenetrating, but neither nested nor consecutive.
- (c) $\{\{1\}, \{2, 3\}, \{4, 5\}\}$ is consecutive.
- (d) $\{\{3\}, \{2, 4\}, \{1, 5\}\}$ is nested. □

A noncrossing partition is called *nearly nested* ($N_e N$) if it is nested except for some parts of size 1.

Example 6 Let $\mathcal{N} = \{1, 2, 3, 4, 5, 6, 7\}$ and $p = 4$. Then the partition $\{\{1, 4, 7\}, \{2, 3\}, \{5\}, \{6\}\}$ is nearly nested but not order-nonpenetrating. □

Table 5 summarizes partition properties mentioned so far.

The following result records implications among the above properties of partitions. The relationship between C , N , NC , and O was determined in Hwang et al. [42].

Theorem 17 *The implications among the properties C , N , NC , O , $N_e N$, S , E , PE , M_p , and BP of partitions are represented precisely by the partial order illustrated in Fig. 3.*

Given a (single-parameter) partition problem, let n_+ , n_- , and n_0 denote the number of θ 's which are positive, negative, and equal 0, respectively. Following Hwang and Mallows [35] and [39], Table 6 provides the exact numbers of (single-)size partitions for the various properties listed in Table 5. Note that all the numbers are polynomial in n .

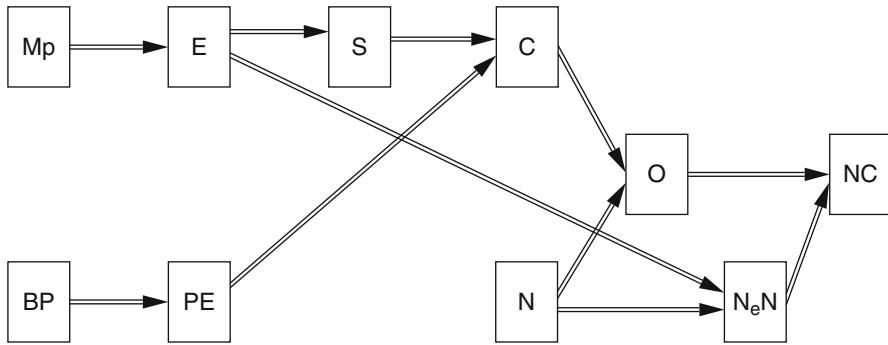


Fig. 3 Implications among $E, S, C, O, N, N_eN, NC, PE, M_p$, and BP

Table 6 The number of size partitions satisfying $NC, C, N, N_eN, S, S^{-1}, E, E^{-1}, PE, O, M_p$, and BP

Q	Number	Q	Number
NC	$\binom{n}{p-1} \binom{n}{p}$	C	$\binom{n-1}{p-1}$
N	$\frac{n}{\binom{n-1}{p-2}}$	N_eN	$\sum_{j=0}^{p-1} n_j \binom{n-j-2}{2p-2j-2}$
S	$\frac{n^{p-1}}{(p-1)p!}$	S^{-1}	$\frac{n^{p-1}}{(p-1)p!}$
E	1	E^{-1}	1
PE	$n_0 - p - 1$ if $n_0 \geq p - 2, n_+ > 0$ and $n_- > 0$ (different numbers in other cases)	O	$\sum_{j=0}^{p-1} \binom{n-1}{j} \binom{n-1-j}{2p-2j-2}$
M_p	1	BP	$n_0 + 1$

The focus will be on *local sorting* which changes only a prescribed number of parts. Local sorting requires that attention be given to subsets of the set of parts of a given partition which are then considered themselves as (sub)partitions. Let $\mathcal{N} = \{1, \dots, n\}$ and $\mathcal{E} = \{1, \dots, p\}$. To facilitate reference to the indices of these (sub)partitions, it will be useful to consider a framework where a *labeled partition* of a set $\mathcal{N}' \subseteq \mathcal{N}$ over an index set $\mathcal{E}' \subseteq \mathcal{E}$ is a set $\pi = \{(j, \pi_j) : j \in \mathcal{E}'\}$ where the π_j 's are disjoint subsets of \mathcal{N}' whose union is \mathcal{N}' . In particular, \mathcal{E}' , $|\mathcal{E}'|$, and $\{(j, |\pi_j|) : j \in \mathcal{E}'\}$ will be referred to as, respectively, the *support*, *size*, and *shape* of π . The notation $\pi = (\pi_1, \dots, \pi_p)$ for “(labeled) partitions,” used so far, captures the case where $\mathcal{N}' = \mathcal{N}$ and $\mathcal{E}' = \mathcal{E}$; its use will be continued in those cases. Given a labeled partition of \mathcal{N}' over \mathcal{E}' , say $\pi = \{(j, \pi_j) : j \in \mathcal{E}'\}$, and a subset J of \mathcal{E}' , $\pi_J \equiv \{(j, \pi_j) : j \in J\}$ will be called the *subpartition* of π corresponding to J ; in particular, π_J is a partition of $\cup_{j \in J} \pi_j$ over J (having support J , size $|J|$ and shape $\{(j, |\pi_j|) : j \in J\}$). The reference to the partitioned set and/or the support will be sometimes suppressed, referring to partition, subpartitions, etc. Also, a *k-partitions/subpartition* is a partition/subpartition of size k . Generically, “local” will refer to subpartitions of limited size.

Some definitions about properties of partitions that connect global satisfiability and local satisfiability are next introduced. Suppose Q is a property of partitions. Property Q is *hereditary* if every subpartition of a partition that satisfies Q must also satisfy Q . Also, for a positive integer k , property Q is k -*consistent* if for each $p \geq k$, a p -partition satisfies Q whenever all of its k -subpartitions satisfy Q . The *minimum consistency index* of a property of partitions is defined as the minimal positive integer k for which the property is k -consistent provided such an integer exists and as ∞ if Q is not k -consistent for any positive integer k .

The next lemma records an order for k -consistency.

Lemma 6 *Let k be a positive integer and let Q be a property of partitions such that a partition π satisfies Q if and only if every k -subpartition of π satisfies Q . Then Q is k' -consistent for all $k' > k$.*

Attention is next turned to transformations of partitions, where the image of a partition π consists of partitions obtained from π by sorting elements of subpartitions of π that do not satisfy Q to subpartitions that satisfy Q , while preserving the remaining parts of the partition.

For a partition π , let $\rho(\pi)$ denote its support and $v(\pi)$ denote its shape. Formally, three partition-transformations are defined for a given property Q of partitions and a given positive integer k . The transformations are $T_J^{Q,k,\text{open}}$, $T_J^{Q,k,\text{support}}$, and $T_J^{Q,k,\text{shape}}$, where for $J \subseteq \mathcal{E}$ and partition π , $T_J^{Q,k,t}(\pi) \equiv [T^{Q,k,t}]_J(\pi)$ is given by

$$T_J^{Q,k,\text{open}}(\pi) = \begin{cases} \{\pi' : \pi_{\rho(\pi) \setminus J} = \pi'_{\rho(\pi) \setminus J}, \\ \quad \cup_{j \in \rho(\pi)} \pi_j = \cup_{j \in \rho(\pi')} \pi'_j, \\ \quad \text{and } \pi'_{\rho(\pi') \setminus [\rho(\pi) \setminus J]} \in Q\} & \text{if } \pi_J \notin Q \text{ and } |J| = k \leq |\pi| \\ \{\pi' : \pi' \in Q\} & \text{if } \pi_J \notin Q \text{ and } |J| = k > |\pi| \\ \emptyset & \text{otherwise,} \end{cases}$$

$$T_J^{Q,k,\text{support}} = \left\{ \pi' \in T_J^{Q,k,\text{open}}(\pi) : \rho(\pi') = \rho(\pi) \right\}$$

and

$$T_J^{Q,k,\text{shape}}(\pi) = \left\{ \pi' \in T_J^{Q,k,\text{open}}(\pi) : v(\pi') = v(\pi) \right\}.$$

For $t \in \{\text{open, support, shape}\}$, partitions in $T^{Q,k,t}(\pi)$ are called (Q, k, t) -*sortings of π* .

Recall that $\mathcal{N} = \{1, \dots, n\}$ and $\mathcal{E} = \{1, \dots, p\}$ are assumed given. The natural universal domains for $T^{Q,k,\text{open}}$, $T^{Q,k,\text{support}}$, and $T^{Q,k,\text{shape}}$ are partitions without restriction, those with a prescribed support and those with prescribed shape, respectively (where no restriction is to be interpreted as $p = n$ and the support of the partitions is restricted to subsets of $\mathcal{E} = \{1, \dots, n\}$). To index these potential domains, define $X^{\text{open}} \equiv \{\text{open}\}$, $X^{\text{support}} \equiv \{\rho : \emptyset \neq \rho \subseteq \mathcal{E}\}$, and $X^{\text{shape}} \equiv \{v = \{(j, n_j) : j \in \rho\} : \rho \subseteq \mathcal{E}; \text{each } n_j \text{ is an integer and } \sum_{j \in \rho} n_j = |\mathcal{N}|\}\}.$

in particular, X^{support} and X^{shape} represent, respectively, all potential supports and shapes of subpartitions over \mathcal{E} . If $\widehat{\Pi}$ is the set of all partitions, then it will be denoted $\widehat{\Pi}^{\text{open}}$ (the support of partitions in $\widehat{\Pi}^{\text{open}}$ is in the power set of $\{1, \dots, n\}$). Further, the set of all partitions with a prescribed support ρ will be denoted $\widehat{\Pi}^\rho$, the set of all partitions with a prescribed shape v will be denoted $\widehat{\Pi}^v$, and finally, for positive integer p , write $\widehat{\Pi}^p$ for $\widehat{\Pi}^{\{1, \dots, p\}}$. If $\widehat{\Pi}$ is, respectively, $\widehat{\Pi}^{\text{open}}$, $\widehat{\Pi}^\rho$ for a particular ρ , or $\widehat{\Pi}^v$ for a particular v , a corresponding transformation on $\widehat{\Pi}$ is called *open*, *support-preserving*, or *shape-preserving*, respectively. The potential domains of the transformations $T^{Q,k,t}$ for given Q and $t \in \{\text{open, support, shape}\}$ are then the sets $\widehat{\Pi}^x$ with $x \in X^t$ (but degenerate instances are sometimes excluded).

For $t \in \{\text{open, support, shape}\}$ and $x \in X^t$, a property Q of partitions is *t-regular* for x , if for every $\mathcal{N}' \subseteq \mathcal{N}$ with $\Pi^x(\mathcal{N}') \neq \emptyset$, there is a partition in $\Pi^x(\mathcal{N}') \cap Q$ (with the intuitive interpretation of $\Pi^x(\mathcal{N}')$); Q is *t-regular* if it is *t-regular* for every $x \in X^t$.

Lemma 7 *Let Q be a t-regular property of partitions. Then for every $x \in X^t$, $\pi \in \widehat{\Pi}^x$ and $J \subseteq \mathcal{E}$ with $|J| = k$,*

$$[\pi_J \in Q] \Leftrightarrow [T_J^{Q,k,t}(\pi) = \emptyset]; \quad (20)$$

further, if Q is also *k-consistent* and *hereditary*, then

$$[\pi \in Q] \Leftrightarrow [T^{Q,k,t}(\pi) = \emptyset]. \quad (21)$$

Let k be a positive integer, let $t \in \{\text{open, support, shape}\}$, let Q be a property of partitions, and let Π^* be a set of partitions. Then:

- (i) Π^* is *Q-(strongly, k, t)-invariant* if for each $\pi \in \Pi^* \setminus Q$: for every subset J of \mathcal{E} having k elements and with $\pi_J \notin Q$, every π' in $(T_J^{Q,k,t})(\pi)$ satisfies $\pi' \in \Pi^*$.
- (ii) Π^* is *Q-(sort-specific, k, t)-invariant* if for each $\pi \in \Pi^* \setminus Q$: for every subset J of \mathcal{E} having k elements and with $\pi_J \notin Q$, some π' in $(T_J^{Q,k,t})(\pi)$ satisfies $\pi' \in \Pi^*$.
- (iii) Π^* is *Q-(part-specific, k, t)-invariant* if for each $\pi \in \Pi^* \setminus Q$: for some subset J of \mathcal{E} having k elements and with $\pi_J \notin Q$, every π' in $(T_J^{Q,k,t})(\pi)$ satisfies $\pi' \in \Pi^*$.
- (iv) Π^* is *Q-(weakly, k, t)-invariant* if for each $\pi \in \Pi^* \setminus Q$: for some subset J of \mathcal{E} having k elements and with $\pi_J \notin Q$, some π' in $(T_J^{Q,k,t})(\pi)$ satisfies $\pi' \in \Pi^*$.

When referring to *Q-(ℓ, k, t)-invariance*, the values of ℓ , k , and t are called *level*, *degree*, and *type* of the corresponding invariance. The level ℓ can take four possible values, forming two pairs – {strongly, weakly} and {part-specific, sort-specific}; if ℓ is a specific value, then and ℓ^{-1} is defined as the other value in the same pair. Also, type can take one of three possible values – “open,” “support,” and “shape” – and the possible values of degree are $k = 1, 2, \dots$. When a variable is not quantified

in forthcoming statements/results about (ℓ, k, t) , it means that the statement is valid for all choices of that variable.

A multivalued function on a set of partitions will be called a *statistic*. To compare values of a statistics, “ $<$ ” will stand for “ \leq and \neq .” Let k be a positive integer, let $t \in \{\text{open, support, shape}\}$, and let Q be a t -regular property of partitions. Then:

- (i) Q is (*strongly, k, t -sortable*) if for every $x \in X^t$, there is a statistic $s(\cdot)$ on $\widehat{\Pi}^x$ such that for every $\pi \in \Pi^x \setminus Q$ for every $J \subseteq \mathcal{E}$ having k elements and $\pi_J \notin Q$: every $\pi' \in (T_J^{Q,k,t})(\pi)$ satisfies $s(\pi') < s(\pi)$.
- (ii) Q is (*sort-specific, k, t -sortable*) if for every $x \in X^t$, there is a statistic $s(\cdot)$ on $\widehat{\Pi}^x$ such that for every $\pi \in \Pi^x \setminus Q$ and for every $J \subseteq \mathcal{E}$ having k elements and $\pi_J \notin Q$: some $\pi' \in (T_J^{Q,k,t})(\pi)$ satisfies $s(\pi') < s(\pi)$.
- (iii) Q is (*part-specific, k, t -sortable*) if for every $x \in X^t$, there is a statistic $s(\cdot)$ on $\widehat{\Pi}^x$ such that for every $\pi \in \Pi^x \setminus Q$, for some $J \subseteq \mathcal{E}$ having k elements and $\pi_J \notin Q$: every $\pi' \in (T_J^{Q,k,t})(\pi)$ satisfies $s(\pi') < s(\pi)$.
- (iv) Q is (*part-specific, k, t -sortable*) if for every $x \in X^t$, there is a statistic $s(\cdot)$ on $\widehat{\Pi}^x$ such that for every $\pi \in \Pi^x \setminus Q$, for some $J \subseteq \mathcal{E}$ having k elements and $\pi_J \notin Q$: some $\pi' \in (T_J^{Q,k,t})(\pi)$ satisfies $s(\pi') < s(\pi)$.

As for invariance, the variables ℓ , k , and t of (ℓ, k, t) -sortability are referred to as *level*, *degree*, and *type*. A comparison of the above definitions of sortability and historic ones is deferred till after [Theorem 19](#).

Theorem 18 *Let Q be a k -consistent and hereditary property of partitions, let $x \in X^t$, and let $\Pi^* \subseteq \widehat{\Pi}^x$. If Π^* is Q - (ℓ, k, t) -invariant and Q is (ℓ^{-1}, k, t) -sortable on $\widehat{\Pi}^x$, then for every partition $\pi \in \Pi^*$, there is a finite sequence $\pi^0 = \pi, \pi^1, \dots, \pi^s$ of partitions in Π^* with π^s satisfying Q ($s = 0$ when $\pi \in Q$). In particular, if Π^* is nonempty, then $\Pi^* \cap Q$ is nonempty.*

Suppose one wishes to prove that a property Q of partitions is not T -sortable. Using the definition of (ℓ, k, t) -sortability, it is then necessary to demonstrate that there is no statistic s having the property that all corresponding (Q, k, t) -sorting of a partition π in $\widehat{\Pi}$ which does not satisfy Q have a lower s -value than π . It is impossible to consider all potential statistics. But, the next corollary of [Theorem 18](#) and the following [Theorem 19](#) provide one with tools to establish non-sortability.

Corollary 3 *Let Q be a k -consistent and hereditary property of partitions and let $x \in X^t$. If some $\Pi^* \subseteq \widehat{\Pi}^x$ is Q - (ℓ, k, t) -invariant and does not satisfy Q , then Q is not (ℓ^{-1}, k, t) -sortable on $\widehat{\Pi}^x$.*

Theorem 19 *Let Q be a t -regular, k -consistent, and hereditary property of partitions. Then the following are equivalent:*

- (a) Q is (*strongly, k, t -sortable*).
- (b) For every $x \in X^t$, every set of partitions $\Pi^* \subseteq \widehat{\Pi}^x$ which is (*weakly, k, t -invariant*) satisfies Q .
- (c) There exist no $x \in X^t$ and finite sequence $\pi^0, \pi^1, \dots, \pi^{q-1}, \pi^q = \pi^0$ of partitions in $\widehat{\Pi}^x$ with $\pi^j \in (T^{Q,k,t})(\pi^{j-1}) \setminus Q$ for $j = 1, \dots, q$.
- (d) For every $x \in X^t$, $\Gamma(Q, t, x)$ is acyclic.

The historic evolution of sortability is next reviewed. The notion of sortability (as well as consistency and invariance, the latter under a different name) was first proposed by Hwang et al. [42]. However, this reference focused on sortability parameterized by (sort-specific, k , support). Chang et al. [15] extended the definition to the full (ℓ, k, t) range and gave sortability results in the full range to most properties listed in [Table 5](#). Both papers defined sortability by the conclusion of [Theorem 18](#), namely, a property Q is called (ℓ, k, t) -sortable if for every Q - (ℓ^{-1}, k, t) -invariant family Π contains a partition satisfying Q . While sortability results for particular properties relied on the construction of a corresponding decreasing statistic, the existence of a statistic was not part of the definition of sortability. [Theorem 18](#) demonstrates that the definition used herein is more demanding than the historic definition. Still, [Theorem 19](#) establishes an equivalence between the two definitions when $\ell = \text{strongly}$ (but, for other values of ℓ , “historic sortability” does not imply the “current sortability” as the implication $(b) \Rightarrow (a)$ of [Theorem 19](#) needs not hold. It is noted that the current definition of sortability explicitly requires consistency, whereas the historic definition does not; consequently, a result asserting that any (ℓ, k, t) -sortability implies k -consistency (e.g., Hwang et al. [42], Chang et al. [15]) does not appear in this chapter.

The next lemma summarizes some implications of (ℓ, k, t) -sortability of partition properties.

Lemma 8 *Let Q be a k -consistent, hereditary property of partitions. The various categories of sortability of Q satisfy the following implications:*

- (a) *(strongly, k, t) implies both (sort-specific, k, t) and (part-specific, k, t), and (sort-specific, k, t) and (part-specific, k, t) imply, each by itself, (weakly, k, t).*
- (b) *With $\ell \in \{\text{strongly, part-specific}\}$, if Q is support-regular, then (ℓ, k, open) implies $(\ell, k, \text{support})$, and if Q is shape-regular, then $(\ell, k, \text{support})$ implies (ℓ, k, shape) .*
- (c) *With $\ell \in \{\text{sort-specific, weakly}\}$, (ℓ, k, shape) implies $(\ell, k, \text{support})$, and $(\ell, k, \text{support})$ implies (ℓ, k, open) .*

The table below summarizes the implications among the various categories of sortability established in [Lemma 8](#) (using obvious abbreviations). The table should be viewed as a 2×6 matrix where each cell specifies a different pair (ℓ, t) , for example, cell $(1, 2)$, corresponds to $\ell = \text{strongly}$ and $t = \text{support}$. Implications represented by “ \Rightarrow^* ” require the corresponding regularity of Q , see part (b) of [Lemma 8 \(Table 7\)](#).

The (ℓ, k, t) -sortability of each property Q discussed in this section has been explicitly solved in the literature (see Hwang et al. [42], Chang et al. [15], and Hwang and Rothblum [39]).

It is next shown how the notions of consistency and sortability can be used to facilitate the search of an optimal partition in a family Π under a given objective function F . Let $\Pi_F^* \subseteq \Pi$ denote the set of partitions that maximize F over Π . Recall that there are two general types of results:

Table 7 Sortability-implications

$(st,k,op) \Rightarrow^* (st,k,supp)$	$\Rightarrow^* (st,k,shape)$	$\Rightarrow (s-s,k,shape)$	$\Rightarrow (s-s,k,supp)$	$\Rightarrow (s-s,k,op)$
\Downarrow	\Downarrow	\Downarrow	\Downarrow	\Downarrow
$(p-s,k,op) \Rightarrow^* (p-s,k,supp)$	$\Rightarrow^* (p-s,k,shape)$	$\Rightarrow (w,k,shape)$	$\Rightarrow (w,k,supp)$	$\Rightarrow (w,k,op)$

Type 1. Every optimal partition in Π satisfies a property Q , that is, $\Pi_F^* \subseteq Q$.

Type 2. There exists an optimal partition in Π satisfying a property Q , that is, $\Pi_F^* \cap Q$ is not empty.

Tools for establishing type 2 results are sortability arguments that show that starting from a partition in Π_F^* , local Q -sortings which preserve optimality will result in an optimal partition satisfying Q . Type 1 results are typically easier to prove, if true, by showing that the objective function of a partition not satisfying Q can be strictly improved. This last idea is next cast formally.

Theorem 20 Suppose Q is k -consistent and Π is a t -family for $t \in \{shape, support\}$. If every $\pi \in \Pi$ not satisfying Q has a k -subpartition which can be (Q, k, t) -sorted with F increasing, then every optimal partition in Π satisfies Q .

Consider objective functions of the form

$$F(\pi) = f(g_1[h(\theta_{\pi_1})], \dots, g_p[h(\theta_{\pi_p})]), \quad (22)$$

where $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is (monotone), convex, or Schur convex, the g_j 's are real-valued functions over the reals, and h is a real-valued function over finite subsets of the reals. Types 1 and 2 results will be established for such objective functions. In stating type 1 results, “strict conditions” will be imposed on f, g , and h that are strict (e.g., “nondecreasing” becomes “increasing” and “nonnegative” becomes “positive”) along with the requirement that the θ^i 's are distinct. Also, **Lemma 1** shows that for verifying the optimality of partitions that satisfy a particular property, it suffices to consider the “single” versions. Instances of (22) with h as the sum-function and g_j 's as the identity, that is, F expressed by $F(\pi) = f(\sum \theta_{\pi_1}, \dots, \sum \theta_{\pi_p})$, were studied in [Sect. 3](#). Results which are in a more general form are next recorded.

Theorem 21 Suppose $F(\cdot)$ is given by (22) where f is convex and one of the following holds:

- (a) f is nondecreasing, and each g_j is convex.
- (b) f is nonincreasing, and each g_j is concave.
- (c) Each g_j is linear (and no monotonicity assumption imposed on f).

Then every shape-problem has a consecutive optimal partition. Further, if $\theta \geq 0$ ($\theta \leq 0$), then every shape problem has a size-consecutive (reverse size-consecutive) optimal partition, every size-problem has an extremal (reverse extremal) optimal partition and every relaxed-size problem has a monopolistic optimal partition. Finally, under strict conditions that exclude f from being strictly convex under (a) and (b), the corresponding type 1 conclusions hold.

The case of single-shape-problems with $g_j(\sum \theta_{\pi_j}) = g(\theta_{\pi_j}, n_j)$ where (n_1, \dots, n_p) is the prescribed shape of π was first studied by Chakravarty et al. [11].

Recall the definition of majorization in Sect. 2 and the terminology used in that section. The notion of majorization is next extended to (two types of) weak majorization by dropping the requirement $\sum_{i=1}^p a_{[i]} = \sum_{i=1}^p b_{[i]}$ (condition (8)). Specifically, a p -vector a is said to *weakly submajorize* a p -vector b , written $a \succ_w b$ if

$$\sum_{i=1}^k a_{[i]} \geq \sum_{i=1}^k b_{[i]} \quad \text{for } k = 1, \dots, p; \quad (23)$$

a is said to *weakly supermajorize* b , written $a \succ^w b$ if

$$\sum_{i=k}^p a_{[i]} \leq \sum_{i=k}^p b_{[i]} \quad \text{for } k = 1, \dots, p. \quad (24)$$

Strict weak sub-/super-majorization will refer to situations where at least one of the corresponding inequalities is strict. The following lemma is standard (e.g., Marshall and Olkin [50]).

Lemma 9 Suppose $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is Schur convex and nondecreasing (nonincreasing) and a and b are vectors in \mathbb{R}^p with a weakly submajorizing (supermajorizing) b . Then $f(a) \geq f(b)$. Further, if the weak submajorization (supermajorizations) is strict and the function f is strictly Schur convex and increasing (decreasing), then $f(a) > f(b)$.

As majorization and weak sub/super-majorization are invariant over permutations of vectors, these relations can be applied to multisets of real numbers (with the interpretation that the relation holds for any representing vectors).

Lemma 10 Let f be Schur convex and nondecreasing (nonincreasing) and Q be (ℓ, k, t) -sortable with $\ell \in \{\text{strongly, sort-specific, part-specific, weak}\}$, $k = 2, 3, \dots$ and $t \in \{\text{shape, size}\}$. Suppose for any partition π not satisfying Q , there always exists an (ℓ^{-1}, k, t) - Q -sorting from π to π' such that, with K as the indices of the sorted parts, $\{\{g_j[h(\theta_{\pi'_j})] : j \in K\}\}_w \succ (^w \succ) \{\{g_j[h(\theta_{\pi_j})] : j \in K\}\}$. Then every t -problem has an optimal partition satisfying Q . Further, under strict conditions, every optimal partition satisfies Q . Finally, the above conclusions hold if f is just Schur convex and the weak majorization requirement is replaced by majorization.

Objective functions that are given by (22) with f Schur convex, g_j independent of j and h as the sum-function, are next addressed. The result modifies Theorem 21 which considered problems with f convex and g_j 's allowed to depend on j .

Theorem 22 Suppose

$$F(\pi) = f[g(\theta_{\pi_1}), \dots, g(\theta_{\pi_p})], \quad (25)$$

where f is Schur convex and either of the following holds:

- (a) f is nondecreasing, and g is convex and nondecreasing (nonincreasing).
- (b) f is nonincreasing, and g is concave and nondecreasing (nonincreasing).
- (c) g is linear and nondecreasing (nonincreasing) (and no monotonicity assumption imposed on f).

Then every shape-problem has a consecutive optimal partition. Further, if in addition $\theta \geq 0$ ($\theta \leq 0$), then every shape-problem has a size-consecutive (reverse size-consecutive) optimal partition, every size-problem has an extremal (reverse extremal) optimal partition, and every relaxed-size problem has a monopolistic optimal partition. Finally, under strict conditions, the corresponding type I conclusions hold.

The objective function of (25) with g as the identity and f Schur convex, but not necessarily monotone, was studied in Sect. 3. In fact, the conclusions of Theorem 22 for those cases were established in Theorem 11 for shape-problems and in Theorem 14 for size-problems (the latter with the Schur convexity of f relaxed to asymmetric Schur convexity).

In a clustering problem, one wishes to partition the points into clusters such that under some distance measure, points in the same cluster are close to each other compared to points in different clusters. It is natural to formulate the clustering problem into a minimization problem. To preserve this spirit, consider through the end of this section partition problems where $F(\pi)$ is to be minimized. The notion of “strict conditions,” will continue to refer to strict properties of functions appearing in the objective function, for parameters (the forthcoming w_j ’s) to be distinct and nonzero and, in addition, for θ^i ’s being distinct.

A general distance function (gdf) is a continuous symmetric function $D : \mathbb{R}^2 \rightarrow \mathbb{R}$ such that for $x' \geq x \geq y \geq y'$, $D(x', y) \geq D(x, y)$ and $D(x, y') \geq D(x, y)$ (the continuity requirement can be relaxed for much of the forthcoming results, but technical details are required). A gdf is strict if nondecreasing and nonincreasing are replaced by increasing and decreasing, respectively. A gdf is normalized if $D(x, x)$ is independent of x .

Boros and Hammer [6] considered the strict and normalized gdf $D(x, y) = |x - y|$ and established the following theorem.

Theorem 23 Suppose

$$F(\pi) = \sum_{j=1}^p \sum_{i,u \in \pi_j} |\theta^i - \theta^u|. \quad (26)$$

Then every size problem has a noncrossing optimal partition. Further, under strict conditions, the corresponding type I conclusions hold.

It is not known whether or not the conclusions of [Theorem 23](#) extend to shape-problems, except that the case of uniform shape was solved by Pfersky et al. [[60](#)].

Theorem 24 Suppose p divides n and consider the single shape-problem corresponding to $(n/p, \dots, n/p)$ with F defined by [\(26\)](#). Then there exists a consecutive optimal solution. Further, under strict conditions, a corresponding type 1 conclusion holds.

Assume the θ^i 's are given. For a given gdf D and a set $S \subseteq \mathcal{N}$, c is called a *centroid* of S with respect to D , or briefly the D -centroid of S , if $c \in \operatorname{argmin}_x \sum_{i \in S} D(\theta_i, x)$; continuity assures that every finite set has a centroid. When $D(x, y) = d(|x - y|)$ where $d(\cdot)$ is a continuous, nondecreasing function on the nonnegative reals with $d(0) = 0$, a d -centroid will refer to the corresponding D -centroid. If d is the identity, then the d -centroid is the *median*. Chang and Hwang [[14](#)] proved the type 1 results of the following theorem.

Theorem 25 Suppose

$$F(\pi) = \sum_{j=1}^p w_j \sum_{i \in \pi_j} d(|\theta^i - c_j|), \quad (27)$$

where $d(\cdot)$ is a nondecreasing function on the nonnegative reals with $\lg d$ concave and $d(0) = 0$, and for each j , $w_j > 0$ and c_j is a d -centroid of θ_{π_j} . Then every size problem has a noncrossing optimal partitions. Further, under strict conditions the corresponding type 1 conclusions hold.

Boros and Hwang [[7](#)] considered shape problems with the objective function given by [\(27\)](#) where d is the identity function (and c_j is a medium). They proved the following result.

Theorem 26 Suppose

$$F(\pi) = \sum_{j=1}^p w_j \sum_{i \in \pi_j} |\theta^i - m_j|, \quad (28)$$

where for each j , $w_j > 0$ and m_j is the median of π_j . Then every shape problem has a noncrossing optimal partition. Further, under strict conditions, the corresponding type 1 conclusions hold.

For uniform weights, the consecutive property can be preserved under more general distance functions.

Theorem 27 Suppose

$$F(\pi) = \sum_{j=1}^p \sum_{i \in \pi_j} D_i(\theta^i, c_j), \quad (29)$$

where for each i , D_i is a gdf and for each j , c_j is a centroid of π_j with respect to $\{D_i : i \in \pi_j\}$ (in the sense that $c_j \in \min \arg_x \sum_{i \in \pi_j} D_i(\theta^i, x)$). Then every size problem has a consecutive optimal partition. Further, under strict conditions, the corresponding type I conclusions hold.

Theorem 27 for $D_i(\theta^i, c_j) = w_i d(\theta^i - c_j)^2$ where w_i is a weight on element was proved by Fisher [24]. Hwang [34] extended the Fisher result to distance functions $D_i(\theta^i, c_j) = w_i D(\theta^i, c_j)$.

To obtain a shape-version of **Theorem 27**, uniform D_i 's that satisfy some further structure have to be assumed. Specifically, a gdf D is *submodular* if for every $x, x', y, y' \in \mathbb{R}$ with $x \leq x'$ and $y \leq y'$, $D(x', y') + D(x, y) \leq D(x', y) + D(x, y')$. The following result was proved by Hwang et al. [41].

Theorem 28 Suppose

$$F(\pi) = f \left(\sum_{i \in \pi_1} D(\theta^i, c_1), \dots, \sum_{i \in \pi_p} D(\theta^i, c_p) \right), \quad (30)$$

where D is a submodular gdf, f is Schur concave nondecreasing, and for each j , c_j is a D -centroid of π_j . Then every shape problem has a consecutive optimal partition. Further, under strict conditions, the corresponding type I conclusions hold.

5 Multiparameter: Polyhedral Approach

Sections [Sect. 3](#) and [Sect. 4](#) examined partition properties that facilitate a reduction of the size of the set of partitions through which one has to search for an optimal partition (for one-dimensional problems). This approach is continued in the current section and the next one but with $d \geq 1$ instead of $d = 1$. The properties that were explored in those sections are mostly geometric in nature, though the geometric characteristics may be in disguise for some of them. For example, the property “consecutive” was defined as “each part consists of consecutive integers,” but it can also be defined as “the convex hulls of the parts are pairwise disjoint.” This section and the next one continue the study of geometric properties of partitions but when the partitioned points are vectors in a high-dimensional space.

A unique feature of \mathbb{R}^1 vs. \mathbb{R}^d for $d > 1$ is that it has a natural order. In particular, when ordering the partitioned elements $\theta^1, \theta^2, \dots, \theta^n$ so that $\theta^1 \leq \theta^2 \leq \dots \leq \theta^n$, it is possible to express all relevant geometric properties of the partitioned elements in terms of the corresponding partition of the index set \mathcal{N} , except hiding the distinction whether two disjoint intervals of θ 's touch at the boundary (see next paragraph on this issue). This conversion allows us to focus on “index-partitions.” However, when the dimension d exceeds one, there is no natural ordering of the points which allows the conversion of geometric properties that depend on the formation of the points to properties of partitioned indices. Consequently, it is more natural to define partitions on the set of vectors and not on the set of indices.

There is still the question why the focus so far was on partitioning index sets in the case of single-parameter partition problems. The reason is that “index-partitions” have an advantage over “vector-partitions” in situations where the columns of A have duplicate vectors. As the indices are always distinct, such situations do not require any attention when partitioning the index set. But, when considering “vector-partitions,” the presence of duplicate points requires one to account for the duplication of each vector in each part, that is, the parts are multisets, rather than sets, and more importantly, the same vector may appear in different parts. Partitioning the index set in the case where $d = 1$ voided the need of dealing with multisets and allowed us to focus on the main development of the theory. In Sect. 5, the focus is to prove that there exists an extreme point of the partition polytope which corresponds to an optimal partition with certain geometric property. When A^i ’s are distinct, then this geometric property does not allow two disjoint parts to touch at boundary; when A^i ’s are not distinct, then the geometric property is weakened to allow such touching. But this distinction does not affect the role the partition polytopes play. Therefore, partition of the index set can be continued so as to take advantage of the convenience it provides. However, in Sect. 6, the emphasis is to study these two and other geometric properties in detail, not only in their natures, but also in their differences. In order to understand these fine points, the framework will then revert to partitioning the set A .

Throughout this section, it is assumed that $d \geq 1$ and $A^1, \dots, A^n \in \mathbb{R}^d$ are given. In the sum-partition problem, an objective function $F(\cdot)$ is maximized over a family of p -partitions Π where the objective value of a p -partition $\pi = (\pi_1, \dots, \pi_p)$ is given by $F(\pi) = f(A_\pi)$, with $A_\pi = \left(\sum_{i \in \pi_1} A^i, \dots, \sum_{i \in \pi_p} A^i \right) \in \mathbb{R}^{d \times p}$ and $f(\cdot)$ a real-valued function on $\mathbb{R}^{d \times p}$ (or a relevant subset thereof). The approach that is followed in the current section is to consider the problem of optimizing (an extension of) f over the partition polytope P_A^Π defined as the convex hull of $\{A_\pi : \pi \in \Pi\}$. When the function f is guaranteed to obtain a maximum over P_A^Π at a vertex, enumerating the vertices of P_A^Π and selecting a partition corresponding to a vertex that maximizes f will produce a solution to the partition problem. This approach is enhanced when the vertices are associated with partitions having properties that are present only in a “small number” of partitions.

The case that is considered has the function $f(\cdot)$ linear. It will be shown how linear programming can be used to solve the corresponding partition problem. Here, the approach is to obtain a representation of the problem as a projection of an optimization problem over a (generalized transportation) polytope which has a simple linear inequality representation. The resulting algorithm is polynomial in the number of partitioned vectors, their dimension, and the number of parts of the partitions. The approach follows Hwang, Onn, and Rothblum [44].

Superscripts will be used to denote columns of matrices, subscripts for rows, and double indices for elements, for example, U_t , U^j , and U_t^j . For matrices U and V of common dimension, say $d \times p$, the *inner product* of U and V is defined as the scalar $\langle U, V \rangle \equiv \sum_{t=1}^d \sum_{j=1}^p U_t^j V_t^j$. For matrices U , V , and W of dimension $d \times p$, $d \times q$, and $q \times p$, respectively,

$$\langle U, VW \rangle = \langle V^T U, W \rangle = \langle UW^T, V \rangle. \quad (31)$$

Let I be the $n \times n$ identity matrix. At times, the columns of I (the n standard unit vectors in \mathbb{R}^n) will be the vectors that are associated with the partitioned elements $1, \dots, n$. In such cases, for each p -partition π , $I_\pi \in \mathbb{R}^{n \times p}$ is the matrix associated with π , namely,

$$(I_\pi)_t^j \equiv \begin{cases} 1 & \text{if } t \in \pi_j \text{ and} \\ 0 & \text{otherwise.} \end{cases} \quad (32)$$

For a set of partitions Π , $P_I^\Pi = \text{conv}\{I_\pi : \pi \in \Pi\}$ will be referred to as a *generalized transportation polytope* (associated with Π) and to the corresponding partition problem as a *generalized transportation problem*. Classical transportation polytopes are obtained when Π consists of a single shape. The special attention given to this class of partition problems is justified by the following properties that will be established:

- (i) The correspondence $\pi \rightarrow I_\pi$ is a bijection; further, none of the matrices I_π are expressible as a convex combination of others.
- (ii) Explicit representing systems of linear inequalities are available for bounded-shape generalized transportation polytopes.
- (iii) Constrained-shape partition problems are reducible to generalized transportation problems.

These properties are next verified and then used to develop efficient computational methods for solving the (linear) partition problem.

Lemma 11 *Let Π be a set of p -partitions. Then the correspondence $\pi \rightarrow I_\pi$ is a bijection of Π onto the vertices of P_I^Π ; in particular, the vertices of P_I^Π are precisely the matrices $\{I_\pi : \pi \in \Pi\}$. Further, the inverse correspondence of vertices of P_I^Π onto the partitions of Π has vertex v corresponding to the partition π with $\pi_j = \{t : v_t^j = 1\}$ for $j = 1, \dots, p$.*

An important feature of the bijection $\pi \rightarrow I_\pi$ of Lemma 11 is the fact that it is constructive in both directions, namely, it is easy to determine I_π from π and, vice versa, π from I_π .

Linear-inequality representation for bounded-shape generalized transportation polytopes is next provided.

Theorem 29 *Let L and U be positive integer p -vectors satisfying $L \leq U$ and $\sum_{j=1}^p L_j \leq n \leq \sum_{j=1}^p U_j$, and let $\Pi \equiv \Pi^{(L,U)}$. Then P_I^Π is the solution set of the linear system:*

$$(a) \quad X_t^j \geq 0 \text{ for } t = 1, \dots, n \text{ and } j = 1, \dots, p. \quad (33)$$

$$(b) \quad \sum_{j=1}^p X_t^j = 1 \text{ for } t = 1, \dots, n.$$

$$(c) \quad L_j \leq \sum_{t=1}^n X_t^j \leq U_j \text{ for } j = 1, \dots, p.$$

When $L = U$, [Theorem 29](#) specializes to the following inequality description of classical transportation polytopes.

Corollary 4 *Let n_1, \dots, n_p be positive integers with $\sum_{j=1}^p n_j = n$ and let $\Pi \equiv \Pi^{(n_1, \dots, n_p)}$. Then P_I^Π is the solution set of the linear system:*

- (a) $X_t^j \geq 0$ for $t = 1, \dots, n$ and $j = 1, \dots, p$. (34)
- (b) $\sum_{j=1}^p X_t^j = 1$ for $t = 1, \dots, n$.
- (c) $\sum_{t=1}^n X_t^j = n_j$ for $j = 1, \dots, p$.

□

Theorem 30 *Let $A \in \mathbb{R}^{d \times n}$ and let Π be a set of p -partitions. Then the partition polytope P_A^Π is the image of the generalized transportation polytope P_I^Π under the linear function mapping $X \in P_I^\Pi \subseteq \mathbb{R}^{n \times p}$ into $AX \in \mathbb{R}^{d \times p}$. In particular, for every p -partition π , $A_\pi = AI_\pi$.*

The representation of partition polytopes as a projection of generalized transportation polytopes derived in [Theorem 30](#) yields the following test for vertices of constrained-shape partition polytopes:

Corollary 5 *Let $A \in \mathbb{R}^{d \times n}$, let Π be a set of p -partitions, and let $V \in P_A^\Pi$. Then V is a vertex of P_A^Π if and only if every representation $V = \frac{1}{2}A(X' + X'')$ with $X', X'' \in P_I^\Pi$ has $AX' = AX''$. □*

[Theorem 30](#) yields the following result that shows that partition problems over Π may be lifted to optimization problems over generalized transportation polytopes.

Corollary 6 *Let $A \in \mathbb{R}^{d \times n}$, Π be a set of p -partitions, $f : \mathbb{R}^{d \times p} \rightarrow \mathbb{R}$, and $F : \Pi \rightarrow \mathbb{R}$ with $F(\pi) = f(A_\pi)$ for each $\pi \in \Pi$. Consider the function $h : P_I^\Pi \rightarrow \mathbb{R}$ with $h(X) = f(AX)$ for each $X \in P_I^\Pi$. If π is a p -partition such that I_π maximizes h over $\{I_{\pi'} : \pi' \in \Pi\}$, then π maximizes $F(\cdot)$ over Π ; further, if f is linear with $f(X) = \langle C, X \rangle$ for every $X \in \mathbb{R}^{k \times p}$, then*

$$h(Z) = \langle C, AZ \rangle = \langle A^T C, Z \rangle \quad \text{for all } Z \in P_I^\Pi, \quad (35)$$

and a partition π with I_π maximizing h over $P_I^\Pi \supseteq \{I_{\pi'} : \pi' \in \Pi\}$ exists.

Let $A \in \mathbb{R}^{d \times n}$, $C \in \mathbb{R}^{d \times p}$ and let L and U be positive integer vectors satisfying $L \leq U$ and $\sum_{j=1}^p L_j \leq n \leq \sum_{j=1}^p U_j$, and $F : \Pi^{(L,U)} \rightarrow \mathbb{R}$ with $F(\pi) = \langle C, A_\pi \rangle$ for each $\pi \in \Pi^{(L,U)}$. The goal is to maximize $F(\cdot)$ over $\Pi \equiv \Pi^{(L,U)}$. Let $h : P_I^\Pi \rightarrow \mathbb{R}$ with $h(X) = \langle C, AX \rangle = \langle A^T C, X \rangle$ for each $X \in P_I^\Pi$. In view of [Corollary 6](#) and [Lemma 11](#), the problem of maximizing F over $\Pi^{(L,U)}$ reduces to finding a vertex of P_I^Π that maximizes the linear objective with coefficients $\{(A^T C)_t^j : t = 1, \dots, n \text{ and } j = 1, \dots, p\}$ over P_I^Π ; with $a \equiv \max\{\lg |A_t^i| : A_t^i \neq 0\}$ and $c \equiv \max\{\lg |C_i^j| : C_i^j \neq 0\}$, these coefficients are bounded by ke^{a+c} and are computable in time $O[npd(a+c)]$ (the availability of sophisticated fast algorithms for multiplying matrices is ignored); using the explicit representation of P_I^Π provided in [Theorem 29](#), the problem reduces to a structured linear program, and standard results show that an optimal vertex for P_I^Π can be identified in strongly polynomial time $O[(n+p)np + (n+p)^2u]$ where $u \equiv \max_i \lg(U_i - L_i) \leq n$ (see Ahuja and Orlin [1] or Ahuja, Orlin, and Tarjan [2]).

The above solution method applies (in fact, in a simplified form) to single-shape partition problems (with linear objective) and, consequently, to constrained-shape problems where the set of shapes is tractable. Thus, for a set of shapes Π whose size is polynomial in the parameters k, n , and p , a polynomial solution method by solving $|\Pi|$ linear programs (each having $L = U$ is available).

It is next shown that, with linear objective and special structure, a solution to single-shape partition problems can be obtained explicitly without addressing the linear programming problem over P_I^Π .

Theorem 31 *Let $A \in \mathbb{R}^{d \times n}$, $C \in \mathbb{R}^{d \times p}$, n_1, \dots, n_p be positive integers with $\sum_{j=1}^p n_j = n$, and $F : \Pi^{(n_1, \dots, n_p)} \rightarrow \mathbb{R}$ with $F(\pi) = \langle C, A_\pi \rangle$ for each $\pi \in \Pi^{(n_1, \dots, n_p)}$. If A and C satisfy*

$$(A^t - A^{t+1})^T (C^j - C^{j+1}) \geq 0 \text{ for } 1 \leq t < n \text{ and } 1 \leq j < p, \quad (36)$$

then the p -partition π with $\pi_j = \left\{ \sum_{u=1}^{j-1} n_u + 1, \dots, \sum_{u=1}^j n_u \right\}$ for $j = 1, \dots, p$ maximizes $F(\cdot)$ over $\Pi^{(n_1, \dots, n_p)}$; further, if the inequalities of (36) hold strictly, then π is a unique maximizer.

Standard arguments show that condition (36) is equivalent to (the seemingly stronger assertion):

$$(A^{t_1} - A^{t_2})^T (C^{j_1} - C^{j_2}) \geq 0 \text{ for } 1 \leq t_1 < t_2 \leq n \text{ and } 1 \leq j_1 < j_2 \leq p. \quad (37)$$

[Theorem 31](#) facilitates the identification of optimal partitions when specified permutations of the columns of A and C satisfy (36). This is always possible when $d = 1$ by sorting the two sets of scalars A^1, \dots, A^n and C^1, \dots, C^p to increasing sequences, thereby yielding an explicit solution to the partition problem.

Two properties of partitions – separability and almost separability – are next introduced. In particular, conditions are determined for these properties to be present

in partitions whose associated vector is a vertex of a constrained-shape partition polytope; results of Sect. 2 can then be used to deduce conditions which assure the existence of optimal partitions which are (almost) separable.

Two subsets Ω^1 and Ω^2 of \mathbb{R}^d are called *separable* if there exists a nonzero d -vector C such that

$$C^T u^1 > C^T u^2 \quad \text{for all } u^1 \in \Omega^1 \text{ and } u^2 \in \Omega^2. \quad (38)$$

Two subsets Ω^1 and Ω^2 of \mathbb{R}^d are called *almost separable* if there exists a nonzero d -vector C such that

$$C^T u^1 > C^T u^2 \quad \text{for all } u^1 \in \Omega^1 \text{ and } u^2 \in \Omega^2 \text{ with } u^1 \neq u^2. \quad (39)$$

In either of these cases, the vector C is referred to as a *separating vector*. Of course, separable sets are necessarily almost separable and disjoint and for disjoint sets almost separability and separability coincide. When two sets are not disjoint, almost separability (as defined by (39)) is the strongest possible form of separation by a linear functional, as the condition asserts strict separation of the values of the functional for all pairs of points at which the sets do not overlap. In particular, (39) implies that $\Omega^1 \cap \Omega^2$ contains at most a single point, for if u and w were distinct points u and w in the intersection, it would follow that $C^T u > C^T w$ and $C^T w > C^T u$.

The next lemma converts separability of finite sets to separability of their convex hulls, yielding as a corollary, a characterization of the former.

Lemma 12 *Let Ω^1 and Ω^2 be two finite sets in \mathbb{R}^d and let C be a nonzero vector in \mathbb{R}^d . Then:*

- (a) *$C^T u^1 > C^T u^2$ for all $u^1 \in \Omega^1$ and $u^2 \in \Omega^2$ if and only if $C^T w^1 > C^T w^2$ for all $w^1 \in \text{conv } \Omega^1$ and $w^2 \in \text{conv } \Omega^2$.*
- (b) *$C^T u^1 > C^T u^2$ for all $u^1 \in \Omega^1$ and $u^2 \in \Omega^2$ with $u^1 \neq u^2$ if and only if $C^T w^1 > C^T w^2$ for all $w^1 \in \text{conv } \Omega^1$ and $w^2 \in \text{conv } \Omega^2$ with $w^1 \neq w^2$.*

Call a partition $\pi = (\pi_1, \dots, \pi_p)$ *separable (almost separable)* if the sets $\{A^i : i \in \pi_j\}$ for $j = 1, \dots, p$ are pairwise separable (almost separable). The next result and its corollary establish (almost) separability of a partition π whose associated matrix A_π is a vertex of a corresponding constrained-shape partition polytope. Under the restriction that the columns of A are distinct, the results are due to Barnes et al. [4]. Some of the results about separable partitions for the general case appear in Hwang et al. [43] and the results about almost separable partitions are from Hwang and Rothblum [38, 40].

Theorem 32 *Let Γ be a nonempty set of positive integer p -vectors with coordinate-sum n , and let π be a partition in Π^Γ where A_π is a vertex of P_A^Γ . Then for some matrix $C \in \mathbb{R}^{d \times p}$ (with columns C^1, \dots, C^p)*

$$(C^r - C^s)^T u > (C^r - C^s)^T w \text{ for all distinct } r, s \in \{1, \dots, p\}, u \in \text{conv} \{A^i : i \in \pi_r\} \text{ and } w \in \text{conv} \{A^i : i \in \pi_s\} \text{ with } u \neq w. \quad (40)$$

Corollary 7 Let Γ be a nonempty set of positive integer p -vectors with coordinate-sum n , and let π be a partition in Π^Γ where A_π is a vertex of P_A^Γ . Then π is almost separable; further, if the columns of A are distinct, then π is separable.

The next result shows that when $d = 1$, the necessary condition for the matrix (in fact, vector for $d = 1$) associated with a partition to be a vertex is also sufficient.

Theorem 33 Let $A \in \mathbb{R}^{1 \times n}$, let n_1, \dots, n_p be positive integers with $\sum_{j=1}^p n_j = n$ and let π be a partition with shape (n_1, \dots, n_p) . Then π is almost separable if and only if A_π is a vertex of $P_A^{(n_1, \dots, n_p)}$ ($= P_A^\Gamma$ for $\Gamma = \{(n_1, \dots, n_p)\}$).

The next result establishes the optimality of (almost) separable partitions.

Theorem 34 Let Γ be a nonempty set of positive integer p -vectors with coordinate-sum n and let $f(\cdot)$ be an (edge-)quasi-convex function on the constrained-shape partition polytope P_A^Γ . Then there exists an optimal partition π which is almost separable and has A_π as a vertex of P_A^Γ . Further, if $f(\cdot)$ is strictly (edge-)quasi-convex, then every optimal partition π is almost separable and has A_π as a vertex of P_A^Γ . If A 's columns are distinct, the quantifier “almost” can be dropped in the above statements.

Theorem 34 shows that when considering constrained-shape partition problems with $f(\cdot)$ (edge-)quasi-convex, it suffices to consider partitions that are (almost) separable (and whose associated matrix is a vertex of the corresponding partition polytope).

Cone-separability, another property of partitions, it next introduced and studied. In particular, it will be shown that single-size partition problems have optimal partitions with this property and an enumeration algorithm for solving such problems will be developed. It is emphasized that the results require the assumption that empty parts are allowed. Of course, single-size problems with empty parts prohibited (allowed) are instances of constrained-shape problems corresponding to the set of all positive (nonnegative) shapes. In particular, problems of either type have optimal solutions that are (almost) separable, and enumerating (almost) separable partitions can be used to solve these problems. The following stronger conclusions do not apply to single-size problem when parts are required to nonempty.

Two subsets Ω^1 and Ω^2 of \mathbb{R}^d are called *cone-separable* if there exists a nonzero d -vector C such that

$$C^T u^1 > 0 > C^T u^2 \text{ for all } u^1 \in \Omega^1 \setminus \{0\} \text{ and } u^2 \in \Omega^2 \setminus \{0\} \quad (41)$$

(if either $\Omega^1 \setminus \{0\}$ or $\Omega^2 \setminus \{0\}$ is empty and the other set is not, then (41) is to be interpreted as a requirement just on the elements of the nonempty set).

A relation between cone-separability and (almost) separability is determined in the next lemma.

Lemma 13 Suppose Ω^1 and Ω^2 are subsets of \mathbb{R}^d that are cone-separable. Then Ω^1 and Ω^2 are almost separable with the vector C satisfying (41) as a separating vector; in this case, 0 is the only possible vector in $\Omega^1 \cap \Omega^2$. Further, if $0 \notin \Omega^1$ or $0 \notin \Omega^2$, then Ω^1 and Ω^2 are separable.

The next example demonstrates that (almost)-separability does not imply cone-separability.

Example 7 For $k = 1, 2$, let $\Omega^k = \{\binom{k}{0}, \binom{k}{2}\} \subseteq \mathbb{R}^2$. Then Ω^1 and Ω^2 are separable (with $C = \binom{1}{0}$ as a separating vector), but they are not cone-separable as a vector $C \in \mathbb{R}^2$ satisfies $C^T \binom{1}{0} > 0$ if and only if $C^T \binom{2}{0} > 0$. \square

Recall that the *conic hull* of a set $\Omega \subseteq \mathbb{R}^d$, denoted $\text{cone } \Omega$, is defined as the set of linear combinations $\sum_{t=1}^q \Gamma_t x^t$ with $\Gamma_t \geq 0$ and $x^t \in \Omega$ for $t = 1, \dots, q$ (with $\text{cone } \emptyset = \{0\}$). A set is a *cone* if it equals its conic hull. A cone is *pointed* if for no nonzero vector x , both x and $-x$ are in the cone. A fundamental result about cones (see Rockafellar [61], Schrijver [64], or Ziegler [71]) shows that a set is the conic hull of a finite set in \mathbb{R}^m if and only if it has a representation $\{x \in \mathbb{R}^m : Ax \leq 0\}$ with A as a real matrix having m columns; such a cone is called a *polyhedral cone*.

Lemma 14 Let Ω^1 and Ω^2 be two finite sets in \mathbb{R}^d and let C be a nonzero vector in \mathbb{R}^d . Then $C^T u^1 > 0 > C^T u^2$ for all $u^1 \in \Omega^1 \setminus \{0\}$ and $u^2 \in \Omega^2 \setminus \{0\}$ if and only if $C^T w^1 > 0 > C^T w^2$ for all $w^1 \in (\text{cone } \Omega^1) \setminus \{0\}$ and $w^2 \in (\text{cone } \Omega^2) \setminus \{0\}$.

The next result provides characterizations of cone-separability for polyhedral cones.

Lemma 15 Let Ω^1 and Ω^2 be two finite sets in \mathbb{R}^d . Then the following are equivalent:

- (a) Ω^1 and Ω^2 are cone-separable.
- (b) $\text{cone } \Omega^1$ and $\text{cone } \Omega^2$ are cone-separable.
- (c) $\text{cone } \Omega^1$ and $\text{cone } \Omega^2$ are almost separable.
- (d) $\text{cone } \Omega^1$ and $\text{cone } \Omega^2$ are pointed cones and $(\text{cone } \Omega^1) \cap (\text{cone } \Omega^2) = \{0\}$.

Attention is next turned to partition problems. Here, fixed-size problems with a relaxation of the assumption that partitions' parts are nonempty are considered. Formally, let $\widehat{\Pi}^p$ denote the set of p -partitions which allow empty parts. For $\pi \in \widehat{\Pi}^p$, A_π has the natural definition with the empty sum taken as zero. The partition polytope corresponding to $\widehat{\Pi}^p$ is defined by $P_A^{\widehat{\Pi}^p} \equiv \text{conv} \{A_\pi : \pi \in \widehat{\Pi}^p\}$. As in all sum-partition problems, it is assumed that the objective function F over $\widehat{\Pi}^p$ is given by $F(\pi) = f(A_\pi)$ with f as a real-valued function on $P_A^{\widehat{\Pi}^p}$.

Call a partition $\pi = (\pi_1, \dots, \pi_p)$ *cone-separable* if the sets $\{A^i : i \in \pi_j\}$ for $j = 1, \dots, p$ are pairwise cone-separable. Lemma 13 shows that cone-separability implies almost separability, with 0 as the only potential overlapping vector for any pair of parts; moreover, a cone-separable partition with at most one part containing indices that correspond to the 0 vector is separable. The next two theorems extend results of Barnes, Hoffman, and Rothblum [1992] (which considered the case where the A^i 's are distinct).

Theorem 35 *If $\pi \in \widehat{\Pi}^p$ and A_π is a vertex of $P_A^{\widehat{\Pi}^p}$, then for some matrix $C \in \mathbb{R}^{d \times p}$*

$$(C^r)^T u > (C^s)^T u \text{ for all distinct } r, s = 1, \dots, p \text{ and } u \in (\text{cone } \pi_r) \setminus \{0\}. \quad (42)$$

Theorem 36 *Let $f(\cdot)$ be an (edge-)quasi-convex on $P_A^{\widehat{\Pi}^p}$. Then there exists an optimal partition π which is both cone-separable and (almost) separable and has A_π as a vertex of $P_A^{\widehat{\Pi}^p}$. Further, if $f(\cdot)$ is strictly (edge-)quasi-convex, then every optimal partition π is cone-separable and has A_π as a vertex of $P_A^{\widehat{\Pi}^p}$.*

The next two results consider cone-separable partitions when $d = 1$ and $d = 2$.

Theorem 37 *Suppose $A \in \mathbb{R}^{1 \times n}$ has no zero column. If $p \geq 2$, then a p -partition π is cone-separable if and only if two of its parts are $\{i \in \mathcal{N} : A^i > 0\}$ and $\{i \in \mathcal{N} : A^i < 0\}$ (where either may be empty) and all other parts are empty. If $p = 1$, then the (only) 1-partition (\mathcal{N}) is cone-separable if and only if A contains either just positive elements or just negative elements.*

To study cone-separable partitions with $d = 2$, associate each $x \in \mathbb{R}^2 \setminus \{0\}$ with the *angular coordinate* of its polar-coordinate representation denoted $\phi(x)$ (measured in degrees). It is observed that for $\emptyset \neq C \subseteq \mathbb{R}^2 \setminus \{0\}$, $C \cup \{0\}$ is a pointed polyhedral cone if and only if for some $0 \leq \underline{\phi} \leq \bar{\phi}$: $\underline{\phi} < 360^\circ$, $\bar{\phi} < \underline{\phi} + 180^\circ$ and

$$C = \begin{cases} \{x \in \mathbb{R}^2 \setminus \{0\} : \underline{\phi} \leq \phi(x) \leq \bar{\phi}\} & \text{if } \bar{\phi} < 360^\circ \\ \{x \in \mathbb{R}^2 \setminus \{0\} : \underline{\phi} \leq \phi(x) < 360^\circ \text{ or } 0 \leq \phi(x) \leq \bar{\phi} - 360^\circ\} & \text{if } \bar{\phi} \geq 360^\circ. \end{cases}$$

Theorem 38 *Suppose $A \in \mathbb{R}^{2 \times n}$ has no zero column. A p -partition π is cone-separable if and only if for some $q \leq p$, there exist $0 \leq \underline{\phi}_1 \leq \bar{\phi}_1 < \underline{\phi}_2 \leq \bar{\phi}_2 < \dots < \underline{\phi}_q \leq \bar{\phi}_q < \underline{\phi}_1 + 360^\circ$ such that $\underline{\phi}_q < 360^\circ$, $\bar{\phi}_t - \underline{\phi}_t < 180^\circ$ for each $t = 1, \dots, q$, and the nonempty parts of π are*

$$\{i \in \mathcal{N} : \underline{\phi}_t \leq \phi(A^i) \leq \bar{\phi}_t\} \text{ for } t = 1, \dots, q-1$$

and

$$\begin{cases} \{i \in \mathcal{N} : \underline{\phi}_q \leq \phi(A^i) \leq \bar{\phi}_q\} & \text{if } \bar{\phi}_q < 360^\circ \\ \{i \in \mathcal{N} : \underline{\phi}_q \leq \phi(A^i) < 360^\circ \text{ or } 0 \leq \phi(A^i) \leq \bar{\phi}_q - 360^\circ\} & \text{if } \bar{\phi}_q \geq 360^\circ; \end{cases}$$

further, the $\underline{\phi}_t$'s and $\bar{\phi}_t$'s can be selected from $\{\phi(A^i) : i \in \mathcal{N}\}$.

6 Multiparameter: Combinatorial Approach

This section explores a combinatorial approach which can apply to partition problems that are more general than the sum-partition problems studied in Sect. 5. Here, vector partitions rather than index partitions are used – see the first three paragraphs of Sect. 5.

A finite subset Ω^1 of \mathbb{R}^d penetrates another finite subset Ω^2 of \mathbb{R}^d , written $\Omega^1 \rightarrow \Omega^2$, if $\Omega^1 \cap (\text{conv } \Omega^2) \neq \emptyset$; Ω^1 strictly penetrates Ω^2 if $\Omega^1 \cap (\text{ri}[\text{conv } \Omega^2]) \neq \emptyset$. Some implications about penetration that are true in \mathbb{R}^1 do not hold in \mathbb{R}^d with $d > 1$. For example, $\Omega^1 \rightarrow \Omega^2$ and $\Omega^2 \not\rightarrow \Omega^1$ no longer imply $\text{conv } \Omega^1 \subset \text{conv } \Omega^2$.

Properties of partitions of vectors in \mathbb{R}^d , with $d > 1$, are next introduced. The fact that some implications of penetration in \mathbb{R}^1 do not hold in \mathbb{R}^d for $d > 1$ implies that some properties of partitions of points in \mathbb{R}^1 that were defined in terms of penetration have more than one counterpart for partitions of points in \mathbb{R}^d .

A *sphere* in \mathbb{R}^d is defined as a set of the form $S = \{x \in \mathbb{R}^d : \|x - a\| \leq R\}$ for some $a \in \mathbb{R}^d$ and $R > 0$, where $\|\cdot\|$ stands for the Euclidean norm. The interior and boundary of such a sphere is given by $\text{int } S = \{x \in \mathbb{R}^d : \|x - a\| < R\}$ and $\text{bd } S = \{x \in \mathbb{R}^d : \|x - a\| = R\}$.

Consider a partition $\pi = (\pi_1, \dots, \pi_p)$ of a finite set of vectors in \mathbb{R}^d . The following properties of such partitions will be considered; the first three were introduced in Sect. 5.

- *Separable (S)* (also referred to as “disjoint” in the literature): For all distinct $r, s = 1, \dots, p$, π_r and π_s are separable, that is, $(\text{conv } \pi_r) \cap (\text{conv } \pi_s) = \emptyset$.
- *Almost separable (AS)*: For all distinct $r, s = 1, \dots, p$, π_r and π_s are almost separable, that is, $(\text{conv } \pi_r) \cap (\text{conv } \pi_s)$ contains at most a single point, and if the intersection contains a point, it is a vertex of both $\text{conv } \pi_r$ and $\text{conv } \pi_s$.
- *Cone-separable ($C_n S$)*: For all distinct $r, s = 1, \dots, p$, π_r and π_s are cone-separable, that is, $(\text{cone } \pi_r) \cap (\text{cone } \pi_s) = \{0\}$.
- *Sphere-separable (SS)*: For all distinct $r, s = 1, \dots, p$, there exists a sphere $S \subset \mathbb{R}^d$ such that either $(U, W) = (\pi_r, \pi_s)$ or $(U, W) = (\pi_s, \pi_r)$, satisfies $S \supseteq U$ and $S \cap W = \emptyset$.
- *Convex-separable ($C_v S$)*: For all distinct $r, s = 1, \dots, p$, there exists a convex set $S \subset \mathbb{R}^d$ such that either $(U, W) = (\pi_r, \pi_s)$ or $(U, W) = (\pi_s, \pi_r)$, satisfies $S \supseteq U$ and $S \cap W = \emptyset$. (Evidently, without loss of generality, it is possible to assume that $\dim S = d$.)
- *Almost sphere-separable (ASS)*: For all distinct $r, s = 1, \dots, p$, there exists a sphere $S \subset \mathbb{R}^d$ such that either $(U, W) = (\pi_r, \pi_s)$ or $(U, W) = (\pi_s, \pi_r)$, satisfies $S \supseteq U$ and $|S \cap W| \leq 1$ and if x is a single point in $S \cap W$, then $U \cap (\text{bd } S) = \{x\}$.
- *Nonpenetrating (NP)*: For all distinct $r, s = 1, \dots, p$, $\pi_r \not\rightarrow \text{conv } \pi_s$.
- *Noncrossing (NC)*: For all distinct $r, s = 1, \dots, p$, either $(\text{conv } \pi_r) \cap (\text{conv } \pi_s) = \emptyset$ or one convex hull is contained in the other with no member of the larger part penetrating the smaller part.

- *Acyclic (AC)*: There do not exist $q \geq 2$ parts of π , say $\pi_{i_1}, \dots, \pi_{i_q}$, such that $\pi_{i_1} \rightarrow \pi_{i_2} \rightarrow \dots \rightarrow \pi_{i_q} \rightarrow \pi_{i_1}$.
- *Monopolistic (M_p)*: One part has all elements. Note that if empty parts are prohibited, there is only one partition satisfying M_p and its size is 1, if empty parts are allowed it means that $p - 1$ parts are empty.
- *Nearly monopolistic ($N_e M_p$)*: At most one part of π has more than a single point.
- *Nearly cone-separable ($N_e C_n S$)*: For all distinct $r, s = 1, \dots, p$, π_r and π_s are either cone-separable or at least one of them is a singleton.

A property defined by penetration is called “weak” if the penetration allows touching at boundary. Also note that two subsets Ω^1 and Ω^2 of \mathbb{R}^d are separable if and only if there exists a (closed) half-space S such that $S \supseteq \Omega^1$ and $S \cap \Omega^2 = \emptyset$, and Ω^1 and Ω^2 are cone-separable if and only if there exists a (closed) cone S of dimension d such that $S \supseteq \Omega^1$ and $S \cap \Omega^2 = \{0\}$.

S and $C_n S$ were first considered in Barnes et al. [4]; $C_v S$ (called “noncrossing”) in Boros and Hwang [7]; AS in Hwang and Rothblum [38]; SS , NP (called “nested”), and AC (called “connected”) in Boros and Hammer [6]; NC (for $d = 1$) in Kreweras [48], and $N_e M_p$ in Pfersky et al. [60].

When $d = 1$, both S and NP reduce to consecutiveness, while $N_e C_n S$, NC , SS , AC , and $C_v S$ all reduce to noncrossingness. Further, when $d = 1$, partitions in $C_n S$ can have at most one part containing positive elements, at most one part containing negative elements, and some parts each consisting of only the 0-element (but may have multiple copies). For a set of points with k 0-elements, this set is not size(p)-regular for $p > k + 2$ and not shape-regular for almost all shapes. Hence, $C_n S$ is not studied for $d = 1$. For $d = 1$, when there are enough 0-elements, $C_n S$ reduces to bi-extremal, but $N_e M_p$ does not reduce to extremal since the singletons can be anywhere, not necessarily the smallest elements.

Theorem 39 *The relations among the properties of partitions that were introduced are represented in Fig. 4.*

Golany et al. [29] proved an unexpected relation between S and SS . Here, an “almost” version of the result is added. To state these results, the partitioned vectors are embedded in \mathbb{R}^{d+1} by defining

$$\widehat{A}^i \equiv \begin{pmatrix} A^i \\ \|A^i\|^2 \end{pmatrix} \in \mathbb{R}^{d+1} \text{ for } i = 1, \dots, n. \quad (43)$$

Next, for each p -partition $\pi = (\pi_1, \dots, \pi_p)$, let $\widehat{\pi}$ be the p -partition for the problem with partitioned vectors $\widehat{A}^1, \dots, \widehat{A}^n$ with $\widehat{\pi}_j = \{\widehat{A}^i : A^i \in \pi_j\}$ for $j = 1, \dots, p$; in particular,

$$\widehat{A}_{\pi} = \left[\sum_{A^i \in \pi_1} \widehat{A}^i, \dots, \sum_{A^i \in \pi_p} \widehat{A}^i \right] \in \mathbb{R}^{(d+1) \times p}. \quad (44)$$

Theorem 40 *A partition π is (almost) sphere-separable if and only if $\widehat{\pi}$ is (almost) separable when the partitioned vectors are $\widehat{A}^1, \dots, \widehat{A}^n$.*

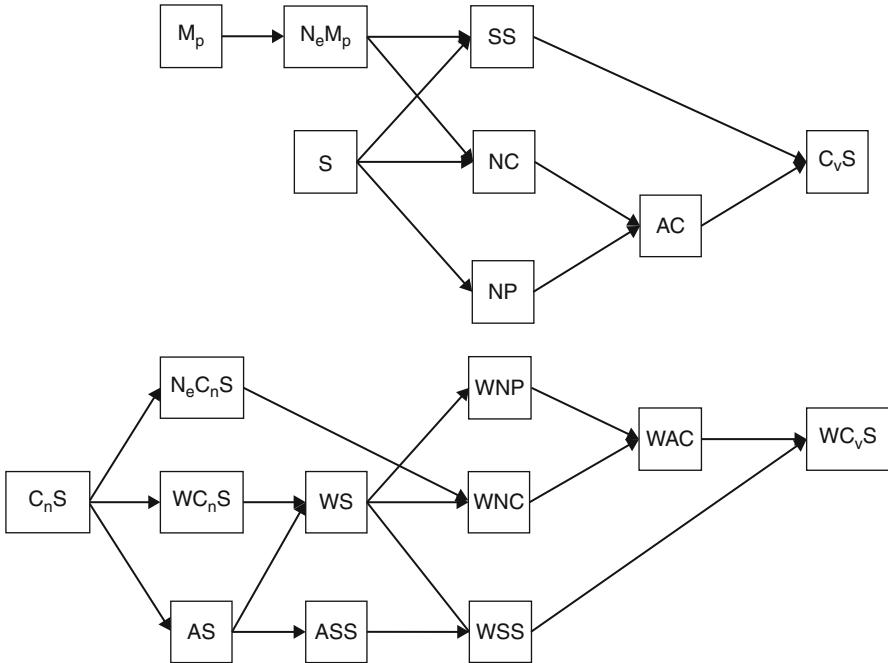


Fig. 4 Implications among properties of multiparameter partitions

The proofs of (Golany et al. [29]) of the equivalences of [Theorem 40](#) are constructive and show how to convert separating vectors that verify (almost) separability into spheres that verify (almost) sphere separability and vice versa.

The next result provides conditions that suffice for $(A)SS \Rightarrow (A)S$.

Lemma 16 *Assume that all the A^i 's lie on the boundary of a sphere. Then a partition is (almost) sphere-separable if and only if it is (almost) separable.*

Next, the number of Q -partitions for each property Q introduced earlier is bounded. Since vector-partitions are considered, the counts may depend on the partitioned vectors. For each $A \in \mathbb{R}^{d \times n}$, let $\#_Q^A(p)$ be the number of p -partitions of the columns of A that satisfy Q , allowing empty parts. The emphasis is to examine whether $\#_Q(n, p, d) \equiv \max_{A \in \mathbb{R}^{d \times n}} \#_Q^A(p)$ grows exponentially or polynomially in n , with $p \geq 2$ and d fixed (in the latter case, enumeration algorithms exist for all relevant properties Q , see [39]). Counting of a polynomial class is usually done by providing an upper bound of the form $O(n^m)$ for some positive m .

Every unlabeled p -partition corresponds to at most $p!$ labeled partitions (the option for identical parts implies that the bound needs not always be tight). Thus, the number of labeled p -partitions is bounded by $p!$ times the number of unlabeled p -partitions (for index-partitions, there are no identical parts, so the bound is

realized), a multiplier that is independent of n . Consequently, the $O(\cdot)$ order of the classes of labeled and unlabeled partitions with a given property coincide. Here, it is convenient to count the labeled classes. Also, empty parts will be allowed.

When columns of A are not distinct, a vector $x \in \mathbb{R}^d$ is referred to as a *multiple point* (of A) if it appears more than once among the columns of A ; the *multiplicity* of a multiple point x is the number of times x appears among the columns of A . As vector partitions are considered, two partitions are identified if their parts are assigned, respectively, the same number of copies of each multiple point. The next lemma records a standard combinatorial fact.

Lemma 17 *Suppose a point in \mathbb{R}^d has n copies. Then there are $\binom{n-1}{p-1}$ ways of splitting the point into p nonempty (labeled) parts and $\binom{n+p-1}{p-1}$ ways of splitting it into p parts when empty parts are allowed; in either case, the number is bounded by $O(n^{p-1})$.*

A set of points in \mathbb{R}^d is called *generic*, or in *general position*, if for $k = 1, \dots, d$ no $k + 1$ points of them lie in a $k - 1$ -dimensional hyperplane. A matrix is called *generic*, if the set of its columns is generic. Harding [33, Theorem 1] gave an exact count of the number of A -separable 2-partitions when A is generic. Harding's result is next presented in a more general context which corrects [33, Theorem 2]; see Hwang and Rothblum [38]. For $n, d \geq 1$, define the (n, d) -Harding number, denoted $H(n, d)$, by

$$H(n, d) \equiv \sum_{j=0}^d \binom{n-1}{j}, \quad (45)$$

(where the standard convention that $\binom{n}{0} = 1$ for each $n \geq 0$ and $\binom{n}{k} = 0$ if $k > n$ is used).

For $A \in \mathbb{R}^{d \times n}$ and $E \in \mathbb{R}^{d \times u}$, a p -partition π is (A, E) -separable if every pair of parts of π are separable by a hyperplane that contains the columns of E .

Theorem 41 *Suppose $A \in \mathbb{R}^{d \times n}$ and $E \in \mathbb{R}^{d \times u}$, where $0 \leq u \leq d$ and $[A, E]$ is generic. With empty parts allowed, the number of $(A|E)$ -separable 2-partitions is $2H(n, d-u) \leq 2^{d-u+1} \binom{n}{d-u}$. With empty parts prohibited, the number of $(A|E)$ -separable 2-partitions is $2H(n, d-u) - 2$ if either $u < d$ or if $u = d$ and neither side of the unique hyperplane spanned by the columns of E contains all of A 's columns. Finally, if empty parts are prohibited, $u = d$ and all columns of A are on one side of the unique hyperplane spanned by the columns of E , then the number of $(A|E)$ -separable is 0.*

Corollary 8 *Suppose $A \in \mathbb{R}^{d \times n}$ is generic. Then $\#_S^A(2) \leq 2H(n, d) = O(n^d)$.*

Hwang et al. [43] proved the equality $\#_S^A(2) = O(n^d)$ of Corollary 8 without using the Harding function. They also extended their result to arbitrary A in $\mathbb{R}^{d \times n}$ by perturbing A into A' which is generic and proved that the set of A -separable

2-partitions is contained in the set of A' -separable 2-partitions, thus bounding $\#_S^A(2)$ by $\#_{S'}^A(2)$ which equals $O(n^d)$ by Corollary 8. Finally, they gave a construction of each A -separable partition by merging a given set of $\binom{p}{2}$ 2-partitions π_{ij} of A , for all “ $1 \leq i < j \leq p$,” and showed that all A -separable p -partitions can be obtained from such constructions (by varying the given set of 2-partitions).

Theorem 42 For $A \in \mathbb{R}^{d \times n}$, $\#_S^A(p) \leq [\#_S^A(2)]^{\binom{p}{2}} = O(n^{d\binom{p}{2}})$. Also, $\#_S(n, p, d) = O(n^{d\binom{p}{2}})$.

Alon and Onn [3] proved that the $O(n^{d\binom{p}{2}})$ bound of Theorem 42 cannot be improved for either $p \geq 3$ or $d \geq 3$.

Using Theorem 40, the above results can be used to derive bounds for sphere-separability.

Theorem 43 Suppose $A \in \mathbb{R}^{d \times n}$, $p \geq 2$ and \bar{n} is the number of distinct columns of A . Then $\#_{SS}^A(p) \leq [\#_{SS}^A(2)]^{\binom{p}{2}} \leq [2H(\bar{n}, d + 1)]^{\binom{p}{2}} \leq [2^{d+1} \binom{\bar{n}}{d+1}]^{\binom{p}{2}}$. Also, $\#_{SS}(n, p, d) = O(n^{(d+1)\binom{p}{2}})$.

Given $A \in \mathbb{R}^{d \times n}$, Theorem 40 also shows that any enumeration method for separable partitions immediately yields a method for enumerating the A -sphere-separable partitions (with a complexity bound obtained by substituting $d + 1$ for d).

Hwang and Rothblum [39] obtained the following result.

Theorem 44 For $A \in \mathbb{R}^{d \times n}$ having no zero-column, $\#_{C_n S}^A(p) \leq [\#_{C_n S}^A(2)]^{\binom{p}{2}} \leq [2H(n, d - 1)]^{\binom{p}{2}} = O[n^{(d-1)\binom{p}{2}}]$. Also, $\#_{C_n S}(n, p, d) \leq [2H(n, d - 1)]^{\binom{p}{2}} = O(n^{(d-1)\binom{p}{2}})$.

Almost separable partitions are next considered, starting with 2-partitions. The result follows Hwang and Rothblum [40].

Lemma 18 For $A \in \mathbb{R}^{d \times n}$ having \widetilde{n} distinct columns, $\#_{AS}^A(2) \leq 2H(n, d) + (n - \widetilde{n})[2H(\widetilde{n} - 1, d - 1)] \leq 2H(n, d)$. Also, $\#_{AS}(n, 2, d) = \#_S(n, 2, d) = 2H(n, d)$.

Two difficulties prevent one from mimicking the perturbation argument and the merging argument used for separable partitions to obtain an extension of Lemma 18 to general p . First, using the same perturbation A to A' , the set of A -almost separable 2-partitions is no longer contained in the set of A' -separable partitions (though for generic A' , almost separable and separable coincide). Thus, one cannot transform a bound of the latter into a bound of the former. Second, the straightforward merging construction for separable partitions is not delicate enough to cover the almost separable case. Hwang and Rothblum [40] showed how to overcome these two problems. The first problem is solved by using the second conclusion of Lemma 18, that is, $\#_{AS}(n, 2, d) = 2H(n, d)$ for the need to bound $\#_{AS}^A(2)$. They also gave a much more elaborated construction to deliver the merging successfully. The resulting bound of $\#_{AS}(n, p, d)$ is given in the next theorem.

Table 8 Bounds of $\#_Q(n, p, d)$

Q	$\#_Q(n, p, d)$	Ref.
S	$O\left(n^d \binom{p}{2}\right)$	Theorem 42
SS	$O\left(n^{(d+1)\binom{p}{2}}\right)$	Theorem 43
$C_n S$	$O\left(n^{(d-1)\binom{p}{2}}\right)$	Theorem 44
AS	$O\left(n^d \binom{p}{2}\right)$	Theorem 45
ASS	$O\left(n^{(d+1)\binom{p}{2}}\right)$	Theorem 46

Theorem 45 $\#_{AS}(n, p, d) \leq [\#_{AS}(n, 2, d)]^{\binom{p}{2}} = O(n^{d\binom{p}{2}})$.

The following result is an immediate consequence of Theorems 40 and 45.

Theorem 46 $\#_{ASS}(n, p, d) = O(n^{(d+1)\binom{p}{2}})$.

The following Table 8 summarizes the bounds of $\#_Q(n, p, d)$ for those properties Q discussed above.

Several simple bounds on the number of partitions that satisfy other properties are next recorded. First, it is trivial that $\#_{MP}(n, p, d) = \binom{n}{p-1} = O(n^{p-1})$, consequently, $\#_{NeMP}(n, p, d) = p! \binom{n}{p-1}$. Also, a careful combinatorial argument yields $\#_{NeCnS}(n, p, d) = \sum_{q=0}^{p-1} \binom{n}{q} q! \#_{CnS}(n-q, p-q, d) = O(n^{(d-1)\binom{p}{2}+p-1})$. Finally, for $Q \in \{NP, AC, NC\}$, Hwang, Lee, Liu, and Rothblum [45] gave examples to show that even for $d = p = 2$, the number of Q -partitions is exponential. From the implication relation shown in Fig. 4, the number of CvS partitions must also be exponential. Finally, it is easily verified that all weak properties are exponential.

Consistency and sortability, as introduced in Sect. 4, refer to properties of index-partitions, without reference to the representation of the partitioned elements. However, p -partition or k -subpartition is said to satisfy Q , and then satisfaction of a part or parts of indices always implies the same satisfaction of the part or parts of the corresponding points. Further, the extension of the notion of satisfaction from a 1-dimensional point to a multi-dimensional point is straightforward. Hence, consistency and sortability can still be discussed for vector-partitions, and the approach of using Q -sortability to prove the existence of a Q -optimal partition is valid as it was for the single-parameter case.

The remainder of this section discusses geometric properties that are present in optimal partitions. Let $\{A^1, \dots, A^n\}$ be a set of n vectors in \mathbb{R}^d and let $F(\cdot)$ denote an objective function over partitions of these vectors. Throughout the remainder of this section, unless explicitly stated otherwise, $F(\cdot)$ is to be minimized in the partition problems that are considered (as is usually the case in “clustering problems”). Also, the results for size problems hold regardless of whether empty parts are allowed or not (for constrained-shape problems, prohibiting empty parts is captured by having the set of feasible shapes contain only positive integer vectors). However, if a result depends on allowing empty parts or on prohibiting such parts, then the condition will be stated explicitly.

Golany et al. [29] extended results of Barnes et al. [4] to the following (which extends [Theorem 34](#)):

Theorem 47 *Suppose*

$$F(\pi) = f(|\pi_1|, \dots, |\pi_p|, A_\pi), \quad (46)$$

where for each fixed set of values $(|\pi_1|, \dots, |\pi_p|)$, f is (edge-)quasi-concave on the corresponding single-shape partition polytope (in the $d \times p$ variables of A_π). Then every constrained-shape problem has an almost separable optimal partition. If furthermore, f is strictly (edge-)quasi-concave, then every optimal partition for a constrained-shape problem is almost separable.

Recall the use of “ $\widehat{\cdot}$ ” introduced in (43) and (44). The following result is reported in Golany et al. [29].

Corollary 9 *Suppose*

$$F(\pi) = f(|\pi_1|, \dots, |\pi_p|, \widehat{A}_\pi), \quad (47)$$

where for each fixed set of values $(|\pi_1|, \dots, |\pi_p|)$, f is (edge-)quasi-concave on the corresponding single-shape partition polytope (in the $(d+1) \times p$ variables of \widehat{A}_π). Then every constrained-shape problem has an almost sphere separable optimal partition. Further, if f is strictly (edge-)quasi-concave, then every optimal partition for a constrained-shape problem is almost sphere separable.

If the function f in [Corollary 9](#) is linear in the variables corresponding to \widehat{A}_π and in the shape-variable, the results of [Sect. 5](#) can be used; that is, bounded-shape partition problems with such objective functions can be solved by LP methods, applied in the enlarged $(d+2)$ -dimensional space where the partitioned vectors are $\binom{\widehat{A}^i}{1}$.

Following Golany et al. [29], [Corollary 9](#) is next used to address some clustering problems.

Theorem 48 *Let w_1, \dots, w_p be positive numbers. Suppose $F(\cdot)$ has either one of the following three expressions:*

- (i) $F(\pi) = \sum_{j=1}^p w_j \sum_{A^i, A^k \in \pi_j} \|A^i - A^k\|^2.$
- (ii) $F(\pi) = \sum_{j=1, \pi_j \neq \emptyset}^p w_j \sum_{A^i \in \pi_j} \|A^i - \bar{A}_j\|^2,$ where for $j = 1, \dots, p$,

$$\bar{A}_j = \frac{\sum_{A^k \in \pi_j} A^k}{|\pi_j|}.$$
- (iii) $F(\pi) = \sum_{j=1}^p w_j \sum_{A^i \in \pi_j} \|A^i - t_j\|^2,$ where t_1, \dots, t_p are prescribed d -vectors.

Then every constrained-shape problem has an almost sphere separable optimal partition.

The third objective function can be expressed as a linear function of \widehat{A}_π and of the parts' sizes. Consequently, the comment following [Corollary 9](#) applies and corresponding bounded-shape partition problems can be solved by LP methods.

Uniform versions of the objective functions that are listed in [Theorem 48](#) are next considered, where uniform means that the w_j 's are constant.

Theorem 49 Consider uniform versions of the objective functions that are listed in [Theorem 48](#). Then:

- (i) For a constrained-shape problem with constant part-sizes and with the first function, every optimal partition is almost separable.
- (ii) For a constrained-shape problem with the second function, every optimal partition is almost separable.
- (iii) For a constrained-shape problem with the third function, there exists an almost separable optimal partition.

Boros and Hammer [6] considered case (i) of [Theorem 49](#) and proved that every single size problem has a weakly separable optimal partition. [Theorem 49](#) strengthens their result in several ways: A stronger property is satisfied for every optimal partition, and the conclusion holds for a wider class of partition problems.

Attention is next turned to single-size problems. For a p -partition π and $i \in \mathcal{N}$, let $j_\pi(i)$ denote the index of the part of π that contains A^i (here, distinction must be made between multiple copies of the same vector).

Lemma 19 Let t_1, \dots, t_p be prescribed distinct vectors in \mathbb{R}^d . Then, allowing for empty parts, there is a separable partition π such that

$$\|A^i - t_{j_\pi(i)}\| \leq \|A^i - t_{j_{\pi'}(i)}\| \quad \text{for every partition } \pi' \text{ and } i \in \mathcal{N}. \quad (48)$$

Further, a partition π satisfies (48) if and only if each vector A^i is assigned to a part π_j for which $\|A^i - t_j\| = \min_{u=1,\dots,p} \|A^i - t_u\|$; each such partition is weakly separable.

Theorem 50 Let t_1, \dots, t_p be prescribed distinct vectors in \mathbb{R}^d and consider the single-size problem, allowing empty parts, which has

$$F(\pi) = f(\|A^1 - t_{j_\pi(1)}\|, \dots, \|A^n - t_{j_\pi(n)}\|), \quad (49)$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is nondecreasing. Then the partition π which assigns each vector A^i to the part π_j with the lowest index j among those for which $\|A^i - t_j\|$ is minimized is both optimal and separable. Further, if f is increasing, then a partition π is optimal if and only each vector A^i is assigned to a part π_j for which $\|A^i - t_j\| = \min_{u=1,\dots,p} \|A^i - t_u\|$, and every optimal partition is weakly separable.

Corollary 10 *The conclusions of Theorem 50 hold for the single-size problem, allowing empty parts, which has*

$$F(\pi) = \sum_{j=1, \pi_j \neq \emptyset}^p \sum_{A^i \in \pi_j} h(\|A^i - t_j\|), \quad (50)$$

where t_1, \dots, t_p be prescribed distinct vectors in \mathbb{R}^d where $h : \mathbb{R} \rightarrow \mathbb{R}$ is nondecreasing/increasing.

Theorem 50 and Corollary 10 specify a single separable partition that is uniformly optimal for all underlying partition problems, independently of the functions f and h that occur in (49) and (50).

The characterization of optimal partition of Theorem 50 (and Corollary 10) allows one to construct all optimal solutions by assigning each vector A^i to any part π_j for which t_j is the closest to A^i . More specifically, associate each vector $A^i \in \mathbb{R}^d$ with the vector $D^i \equiv (\|A^i - t_1\|, \dots, \|A^i - t_p\|)^T \in \mathbb{R}^p$. A partition is then optimal if and only if it assigns A^i to π_j with D_j^i being any minimal coordinate of D^i . The amount of effort needed to compute each vector D^i is $O(dp)$, so the total amount of effort to compute all of the D^i 's (and thereby solve the partition problem) is $O(dpn)$.

Since the boundary of the convex hull of two parts can contain points of both parts, the optimal partitions identified in Theorem 50 and Corollary 10 need not be almost separable, let alone separable. Still, it is noted that breaking ties by using any part-ranking will produce separable partitions.

The geometric figure of the partition of \mathbb{R}^d into p convex subsets S_1, \dots, S_p with S_j consisting of all points in \mathbb{R}^d closest to t_j (a point on a boundary is closest to all t_j whose S_j shares that boundary) is known in the literature as the Voronoi diagram. Efficient constructions of Voronoi diagrams are well studied in theoretical computer science (see Fortune [25] for a survey). In particular, hyperplanes representing a Voronoi diagram for p given points t_1, \dots, t_p in \mathbb{R}^d can be constructed in $O(d^{\frac{p+1}{2}})$ -time. For $d = 2$, Shamos and Hoey [65] showed that the Voronoi diagram can be constructed in $O(p \log p)$ -time.

A tool is next developed for using results about properties of optimal partitions of problems with objective functions that depend on prescribed vectors to results where the prescribed vectors are replaced by some minimizing vectors. Two new definitions are needed to present a formal result. A function $g : \mathbb{R}^d \rightarrow \mathbb{R}$ is *inverse-bounded* if for every $K \in \mathbb{R}$ $\{x \in \mathbb{R}^d : |g(x)| \leq K\}$ is a bounded set of \mathbb{R}^d . Evidently, an inverse-bounded function that is continuous is guaranteed to attain a minimum over \mathbb{R}^d . Also, let $P^*(\mathbb{R}^d)$ be the set of finite subsets of \mathbb{R}^d , that is, $P^*(\mathbb{R}^d) = \{\Omega \subset \mathbb{R}^d : |\Omega| \text{ is finite}\}$.

Theorem 51 *Let Q be a partition property, $f : \mathbb{R}^p \rightarrow \mathbb{R}$ be nondecreasing and $g : P^*(\mathbb{R}^d) \times \mathbb{R}^d \rightarrow \mathbb{R}$ with the property that for every $\Omega \in P^*(\mathbb{R}^d)$, $g(\Omega, \cdot)$ is inverse-bounded and continuous. Let Γ be a set of p -integer vectors with coordinate-sum n .*

Assume that for any $t_1, \dots, t_p \in \mathbb{R}^d$, some optimal partition of the constrained-shape problem corresponding to Γ and having objective function $F^{t_1, \dots, t_p}(\cdot)$, empty parts not allowed, given by

$$F^{t_1, \dots, t_p}(\pi) = f[g(\pi_1, t_1), \dots, g(\pi_p, t_p)] \text{ for each partition } \pi \quad (51)$$

satisfies Q . Then some optimal partition for the corresponding constrained-shape problem with objective function $F(\cdot)$ given by

$$F(\pi) = f \left[\min_{x_1 \in \mathbb{R}^d} g(\pi_1, x_1), \dots, \min_{x_p \in \mathbb{R}^d} g(\pi_p, x_p) \right] \text{ for each partition } \pi \quad (52)$$

satisfies Q . Further, the above holds with “every” replacing “some.”

Theorem 51 can be used to obtain **Theorem 49** (ii) from **Theorem 49** (iii) by noting that $\sum_{j=1}^p w_j \sum_{A^i \in \pi_j} \|A^i - t_j\|^2$ is minimized over (t_1, \dots, t_p) at $(\bar{A}_1, \dots, \bar{A}_p)$ (with $\bar{A}_j = \frac{\sum_{A^k \in \pi_j} A^k}{|\pi_j|}$ for $j = 1, \dots, p$).

The proof of **Theorem 51** in [39] can, in fact, be applied to a partition that is not optimal. Specifically, if π is any partition and t_1, \dots, t_p are minimizers of $g(\pi_1, \cdot), \dots, g(\pi_p, \cdot)$, respectively, then any partition $\pi^\#$ that satisfies $F^{t_1, \dots, t_p}(\pi^\#) \leq F^{t_1, \dots, t_p}(\pi)$ satisfies $F(\pi^\#) \leq F(\pi)$. In particular, any sorting method that is applicable to partition problems with objective function $F^{t_1, \dots, t_p}(\cdot)$ for fixed t_j 's which does not rely on a statistics depending on the t_j 's, applies to the partition problem with objective function $F(\cdot)$. This is the case for p -sortings for which no statistics is needed.

For $h : \mathbb{R} \rightarrow \mathbb{R}$, an h -centroid of a finite set $\Omega \subseteq \mathbb{R}^d$ is a minimizer of $\sum_{A^i \in \Omega} h(\|A^i - x\|)$ over $x \in \mathbb{R}^d$. When h is continuous, bounded from below and inverse-bounded, each finite set Ω has an h -centroid. (This definition extends the definition given in the paragraph preceding **Theorem 25** from $d = 1$ to $d > 1$.)

Theorem 52 Consider the single-size problem which has

$$F(\pi) = \sum_{j=1}^p \sum_{A^i \in \pi_j} h(\|A^i - c_j\|), \quad (53)$$

where $h : \mathbb{R} \rightarrow \mathbb{R}$ is nondecreasing, continuous, and inverse-bounded, and for $j = 1, \dots, p$, c_j is the h -centroid of π_j and where the sum over an empty set of $h(\cdot)$ -values in (53) is defined to be 0. Then there exists a separable optimal partition. Further, if h is increasing, then every optimal partition is weakly separable.

Corollary 11 Consider the partition problem of **Theorem 52**. Then there exists a separable optimal partition without empty parts. Further, if h is increasing, then every optimal partition has no empty parts.

[Theorem 52](#) asserts the existence of a separable optimal partition that is specific to the function h which occurs in (53). This type of result is weaker than the uniform optimality of a specific separable partition asserted in [Theorem 50](#).

The second paragraph following [Theorem 51](#) shows that the assignment of each vector to its closest t_j can be used to replace a partition π that is not separable with one that is and has improved F -value where $F(\cdot)$ is given by (53).

Partition problems that are next considered have objective functions that depend on the least radii of spheres that, respectively, include the parts. It is first shown that given a partition whose parts are included in prescribed spheres, there is a separable partition with the same property. The proof of this result follows an approach of Capoyleas, Rote, and Woeginger [9] who proved the case $d = 2$ (and claimed its extension to general d).

Lemma 20 *Let t_1, \dots, t_p be prescribed distinct vectors in \mathbb{R}^d . Allowing partitions to have empty parts, for each partition π , there exists a separable partition π' such that*

$$\max_{A^i \in \pi'_j} \|A^i - t_j\| \leq \max_{A^i \in \pi_j} \|A^i - t_j\| \quad \text{for } j = 1, \dots, p. \quad (54)$$

Further, if $r_j \equiv \max_{A^i \in \pi_j} \|A^i - t_j\|$ for $j = 1, \dots, p$ (i.e., r_j is the smallest radius of a sphere that is centered at t_j and contains π_j), then π' can be selected by the following rule: $A^i \in \pi'_j$ if j is the lowest index of those for which $\|A^i - t_j\|^2 - r_j^2 = \min_u [\|A^i - t_u\|^2 - r_u^2]$ (of course, one may use any a priori permutation of the part-indices).

Capoyleas, Rote, and Woeginger [9] did not mention a tie-breaking rule; as such, the “power diagram” they used to repartition the vectors leads to weak separability rather than strict separability (earlier results in this section show that the number of weakly separable partitions is not necessarily polynomial in the number of partitioned vectors).

The separating hyperplane between parts π'_j and π'_w of the partition π' constructed in [Lemma 20](#) is given by

$$\begin{aligned} H &\equiv \{x \in \mathbb{R}^d : \|x - t_j\|^2 - r_j^2 = \|x - t_w\|^2 - r_w^2\} \\ &= \{x \in \mathbb{R}^d : 2(t_w - t_j)^T x = \|t_w\|^2 - \|t_j\|^2 + r_j^2 - r_w^2\}. \end{aligned} \quad (55)$$

This hyperplane is called the *power-hyperplane* of the underlying spheres $\{x \in \mathbb{R}^d : \|x - t_s\| \leq r_s\}$ and $\{x \in \mathbb{R}^d : \|x - t_w\| \leq r_w\}$. Further, the construction of π' from π in the proof of [Lemma 20](#) is a sorting operation of all p -parts called *power-hyperplane sorting*. The use of p -sorting has the advantage that it achieves the goal in one step and therefore does not require a monotone statistics that shows iterative sortings will not cycle.

A natural variant of the use of power-hyperplane p -sorting to prove the conclusions of [Lemma 20](#) is to iteratively apply power-hyperplane 2-sorting (of pairs of parts). For this approach to work, that is to avoid the possibility of cycling, it

is necessary to identify a statistics over partitions that is reduced at each iteration. Unfortunately, it is not known that such a statistics exists. Still, power-hyperplane 2-sortings can be used to prove a weaker result. Specifically, consider $t_1, \dots, t_p \in \mathbb{R}^d$ and $r_1, \dots, r_p \in \mathbb{R}$. [Lemma 20](#) implies that power-hyperplane p -sorting will convert a partition π whose parts are included, respectively, in the spheres $\{x \in \mathbb{R}^d : \|x - t_j\| \leq r_j\}, j = 1, \dots, p$, into a separable partition having the same property. Here, one can use iterative power-hyperplane 2-sorting and assure that the process does not cycle because of strict lexicographic reduction of the statistics

$$s(\pi) \equiv \left(\sum_{j=1}^p \sum_{A^i \in \pi_j} [\|A^i - t_j\|^2 - r_j^2], -|\pi_1|, \dots, -|\pi_p| \right)$$

in each step. In particular, this proves that S is (sort-specific, 2, support)-sortable with “sort-specific” referring to the power-hyperplane method with fixed center and fixed radii.

Theorem 53 *Let t_1, \dots, t_p be prescribed distinct vectors in \mathbb{R}^d and consider the single-size problem, allowing empty parts, which has*

$$F(\pi) = f[\max_{A^i \in \pi_1} \|A^i - t_1\|, \dots, \max_{A^i \in \pi_p} \|A^i - t_p\|], \quad (56)$$

where $f : \mathbb{R}^p \rightarrow \mathbb{R}$ is nondecreasing. Let π be a given partition. Then the partition π' which satisfies the conclusions of [Lemma 20](#) is separable and satisfies $F(\pi') \leq F(\pi)$. In particular, the underlying single-size problem has a separable optimal partition.

[Theorem 53](#) provides a construction (appearing explicitly in the statement of [Lemma 20](#)) of a separable partition that improves on the objective function of a given partition π . This result can be cast as p -sortability result, as were the results of [Theorem 50](#) and [Corollary 10](#). The construction of [Theorem 53](#) is independent of the f appearing in (56), but unlike the construction of [Theorem 50](#) (and [Corollary 10](#)), it is specific to the underlying partition π .

For a bounded set $\Omega \subset \mathbb{R}^d$, let $r(\Omega)$ denote the minimum radius of a sphere that includes Ω ; in particular, when Ω is finite, $r(\Omega) = \inf_{x \in \mathbb{R}^d} [\sup_{y \in \Omega} \|x - y\|]$, and the “inf” and “sup” are attained (and can therefore be replaced by “min” and “max”). For a p -partition π , let $r(\pi) = (r(\pi_1), \dots, r(\pi_p))$. The next result considers partition problems with objective function that is determined by these partition characteristics; Capoyleas, Rote, and Woeginger [9] considered the case $d = 2$.

Theorem 54 *Consider the single-size problem which has $F(\pi) = f[r(\pi)]$ where $f : \mathbb{R}^p \rightarrow \mathbb{R}$ nondecreasing. Then some optimal partition is separable.*

A result of Pfersky et al. [60] is next generalized; their result is then deduced as a corollary.

Theorem 55 Consider the single-size problem, not allowing empty parts, which has

$$F(\pi) = \sum_{\substack{u,v=1 \\ u \neq v}}^p \sum_{A^i \in \pi_u} \sum_{A^k \in \pi_v} h(A^i, A^k), \quad (57)$$

where $h(\cdot, \cdot)$ is a metric. Then there exists a nearly monopolistic optimal partition. If h is strict, then every optimal partition is nearly monopolistic. Finally, when allowing empty parts, “nearly monopolistic” can be replaced by “monopolistic.”

The reason that there is no strict version for [Theorem 55](#) is that even if h is strict, when the A^i ’s are all the same point, then all p -partitions are optimal.

The Pfersky, Rudolf, and Woeginger result is next derived. It is stated in the original framework of a maximization problem (of course, multiplying the objective function by -1 converts a maximization problem into an equivalent minimization problem).

Corollary 12 Consider the single-size problem in which

$$F(\pi) = \sum_{j=1}^p \sum_{A^i, A^k \in \pi_j} \|A^i - A^k\|^2. \quad (58)$$

is to be maximized. Then there exists a nearly monopolistic optimal partition.

Recommended Reading

1. R.K. Ahuja, J.B. Orlin, A fast and simple algorithm for the maximum flow problem. *Oper. Res.* **37**, 748–759 (1989)
2. R.K. Ahuja, J.B. Orlin, R.E. Tarjan, Improved time bounds for the maximum flow problem. *SIAM J. Comput.* **18**, 939–954 (1989)
3. N. Alon, S. Onn, Separable partitions. *Discret Appl. Math.* **91**, 39–51 (1999)
4. E.R. Barnes, A.J. Hoffman, U.G. Rothblum, Optimal partitions having disjoint convex and conic hulls. *Math. Program.* **54**, 69–86 (1992)
5. S. Borgwardt, A. Brieden, P. Gritzmann, Constrained minimum-k-star clustering and its application to the consolidation of farmland. *Oper. Res.* (2009). Published Online First
6. E. Boros, P.L. Hammer, On clustering problems with connected optima in Euclidean spaces. *Discret. Math.* **75**, 81–88 (1989)
7. E. Boros, F.K. Hwang, Optimality of nested partitions and its application to cluster analysis. *SIAM J. Optim.* **6**, 1153–1162 (1996)
8. A. Brieden, P. Gritzmann, A quadratic optimization model for the consolidation of farmland by means of lend-lease agreements, in *Operations Research Proceedings 2003: Selected Papers of the International Conference on Operations Research (OR2003)*, ed. by D. Ahr, R. Fahrion, M. Oswald, G. Reinelt (Springer, Berlin, 2004), pp. 324–331
9. V. Capoyleas, G. Rote, G. Woeginger, Geometric clusterings. *J. Algorithms* **12**, 341–356 (1991)
10. A.K. Chakravarty, Multi-item inventory into groups. *J. Oper. Res. Soc. U. K.* **32**, 19–26 (1982)

11. A.K. Chakravarty, J.B. Orlin, U.G. Rothblum, A partitioning problem with additive objective with an application to optimal inventory grouping for joint replenishment. *Oper. Res.* **30**, 1018–1022 (1982)
12. A.K. Chakravarty, J.B. Orlin, U.G. Rothblum, Consecutive optimizers for a partitioning problem with applications to optimal inventory groupings for joint replenishment. *Oper. Res.* **33**, 820–834 (1985)
13. A.K. Chandra, C.K. Wong, Worst case analysis of a placement algorithm related to storage allocation. *SIAM J. Comput.* **4**, 249–263 (1975)
14. G.J. Chang, F.K. Hwang, Optimality of consecutive and nested tree partitions (1993, preprint) **30**, 75–80 (1997)
15. G.J. Chang, F.L. Chen, L.L. Huang, F.K. Hwang, S.T. Nuan, U.G. Rothblum, I.F. Sun, J.W. Wang, H.G. Yeh, Sortabilities of partition properties. *J. Comb. Optim.* **2**, 413–427 (1999)
16. R.A. Cody, E.G. Coffman Jr., Record allocation for minimizing expected retrieval costs on drum-like storage devices. *J. Assoc. Comput. Mach.* **23**, 103–115 (1976)
17. C. Derman, G.J. Lieberman, S.M. Ross, On optimal assembly of systems. *Nav. Res. Logist. Q.* **19**, 564–574 (1972)
18. R. Dorfman, The detection of defective members of large populations. *Ann. Math. Stat.* **14**, 436–440 (1943)
19. D.Z. Du, When is a monotonic grouping optimal? in *Reliability Theory and Applications*, ed. by S. Osaki, J. Cao (World Scientific, River Edge, 1987), pp. 66–76
20. D.Z. Du, F.K. Hwang, Optimal assembly of an s-stage k-out-of-n system. *SIAM J. Disc. Math.* **3**, 349–354 (1990)
21. B.C. Eaves, U.G. Rothblum, Relationships between properties of piecewise affine maps over ordered fields. *Linear Algebra Appl.* **132**, 1–63 (1990)
22. E. El-Newehi, F. Proschan, J. Sethuraman, Optimal allocation of components in parallel-series and series-parallel systems. *J. Appl. Probab.* **23**, 770–777 (1986)
23. E. El-Newehi, F. Proschan, J. Sethuraman, Optimal allocation assembly of systems using Schur functions and majorization. *Nav. Res. Logist. Q.* **34**, 705–712 (1987)
24. W.D. Fisher, On grouping for maximum homogeneity. *J. Am. Stat. Assoc.* **53**, 789–798 (1958)
25. S. Fortune, Voronoi diagrams and Delaunay triangulations, in *Computing in Euclidean Geometry*, 2nd edn., ed. by D.Z. Du, F.K. Hwang (World Scientific, Singapore/River Edge, 1995), pp. 193–233
26. S. Gail, Y. Hollander, A. Attai, Optimal mapping in direct mapped cache environments. *Math. Prog.* **63**, 371–387 (1994)
27. M.R. Garey, F.K. Hwang, D.S. Johnson, Algorithms for a set partitioning problem arising in the design of multipurpose units. *IEEE Trans. Comput.* **26**, 321–328 (1977)
28. C.Z. Gilstein, Optimal partitions of finite populations for Dorfman-type group testing. *J. Stat. Plan. Inference* **12**, 385–394 (1985)
29. B. Golany, F.K. Hwang, U.G. Rothblum, Sphere-separable partitions of multi-parameter elements. *Discret. Appl. Math.* **156**, 838–845 (2008)
30. S.K. Goyal, Determination of optimum packing frequency of items jointly replenished. *Manage. Sci.* **21**, 436–443 (1974)
31. R.L. Graham, Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.* **17**, 416–429 (1969)
32. D. Granot, U.G. Rothblum, The Pareto set of the partition bargaining game. *Games Econ. Behav.* **3**, 163–182 (1991)
33. E.F. Harding, The number of partitions of a set of n points in k dimensions induced by hyperplanes. *Proc. Edinb. Math. Soc.* **15**, 285–289 (1967)
34. F.K. Hwang, Optimal partitions. *J. Optim. Theory Appl.* **34**, 1–10 (1981)
35. F.K. Hwang, C.L. Mallows, Enumerating consecutive and nested partitions. *J. Comb. Theory A* **70**, 323–333 (1995)
36. F.K. Hwang, U.G. Rothblum, Optimality of monotone assemblies for coherent systems composed of series modules. *Oper. Res.* **42**, 709–720 (1994)

37. F.K. Hwang, U.G. Rothblum, Directional-quasi-convexity, asymmetric Schur-convexity and optimality of consecutive partitions. *Math. Oper. Res.* **21**, 540–554 (1996)
38. F.K. Hwang, U.G. Rothblum, On the number of separable partitions. *J. Combinatorial Optimization* **21**, 423–433 (2011)
39. F.K. Hwang, U.G. Rothblum, PARTITIONS Optimality and Cluster. Vol. 1: Single-Parameter, World Scientific, Singapore (2012)
40. F.K. Hwang, U.G. Rothblum, Almost separable partitions (2011, forthcoming)
41. F.K. Hwang, J. Sun, E.Y. Yao, Optimal set partitioning. *SIAM J. Algorithms Discret. Methods* **6**, 163–170 (1985)
42. F.K. Hwang, U.G. Rothblum, Y.-C. Yao, Localizing combinatorial properties of partitions. *Discret. Math.* **160**, 1–23 (1996)
43. F.K. Hwang, S. Onn, U.G. Rothblum, A polynomial time algorithm for shaped partition problems. *SIAM J. Optim.* **10**, 70–81 (1999)
44. F.K. Hwang, S. Onn, U.G. Rothblum, Linear shaped partition problems. *Oper. Res. Lett.* **26**, 159–163 (2000)
45. F.K. Hwang, J.S. Lee, Y.C. Liu, U.G. Rothblum, Sortability of vector partitions. *Discret. Math.* **263**, 129–142 (2003)
46. F.K. Hwang, J.S. Lee, U.G. Rothblum, Permutation polytopes corresponding to strongly supermodular functions. *Discret. Appl. Math.* **142**, 87–97 (2004)
47. U.R. Kodes, Partitioning and card selection, in *Design Automation of Digital Systems*, vol. 1, ed. by M.A. Breuer (Prentice Hall, Englewood Cliffs, 1972)
48. G. Kreweras, Sur le partitions non croisées d'un cycle. *Discret. Math.* **1**, 333–350 (1972)
49. D.M. Malon, When is greedy module assembly optimal. *Nav. Res. Logist. Q.* **37**, 847–854 (1990)
50. A.W. Marshall, I. Olkin, *Inequalities, Theory of Majorization and Its Applications* (Academic, New York, 1979)
51. S. Mehta, R. Chandrasekaran, H. Emmons, Order-preserving allocation of jobs to two machines. *Nav. Res. Logist. Q.* **21**, 361–374 (1974)
52. O. Morgenstern, J. Von Neumann, *Theory of Games and Economic Behavior* (Princeton University Press, Princeton, 1944) (A translation of a German 1928 text)
53. W.M. Nawijn, Optimizing the performance of a blood analyser: application of the set partitioning problem. *Eur. J. Oper. Res.* **36**, 167–173 (1988)
54. J. Neyman, E. Pearson, On the problem of the most efficient tests of statistical hypotheses. *Philos. Trans. R. Soc. Lond. A* **231**, 289–337 (1933)
55. D. Nocturne, Economic order frequency for several items jointly replenished. *Manage. Sci.* **19**, 1093–1096 (1973)
56. K.N. Oikonomou, Abstractions of finite-state machines optimal with respect to single undetectable output faults. *IEEE Trans. Comput.* **36**, 185–200 (1987)
57. K.N. Oikonomou, Abstractions of finite-state machines and immediately-detectable output faults. *IEEE Trans. Comput.* **41**, 325–338 (1992)
58. K.N. Oikonomou, R.Y. Kain, Abstractions for node-level passive fault detection in distributed systems. *IEEE Trans. Comput.* **C-32**, 543–550 (1983)
59. C.G. Pfeifer, P. Enis, Dorfman-type group testing for a modified binomial model. *J. Am. Stat. Assoc.* **73**, 588–592 (1978)
60. U. Pfersky, R. Rudolf, G. Woeginger, Some geometric clustering problems. *Nord. J. Comput.* **1**, 246–263 (1994)
61. T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, 1970)
62. M.H. Rothkopf, A note on allocating jobs to two machines. *Nav. Res. Logist. Q.* **22**, 829–830 (1975)
63. P.H. Schouten, Analytic treatment of the polytope regularly derived from the regular polytopes. *Verhandelingen der Koninklijke Akademie van Wetenschappen te Amsterdam* **11**(3), 87p. (1911). Johannes Müller, Amsterdam
64. A. Schrijver, *Theory of Linear and Integer Programming* (Wiley, New York, 1986)

65. M. Shamos, D. Hoey, Closest-point problems, in *Proceedings of the 16th Annual Symposium on Foundations of Computer Science* (1975), pp. 151–162
66. L.S. Shapley, Cores of convex games. *Int. J. Game Theory* **1**, 11–26 (1971)
67. E.A. Silver, A simple method of determining order quantities in joint replenishments under deterministic demands. *Manage. Sci.* **22**, 1351–1361 (1976)
68. M. Sobel, P.A. Groll, Group testing to eliminate efficient all detectives in a binomial sample. *Bell Syst. Tech. J.* **38**, 1179–1252 (1959)
69. V.S. Tanaev, Optimal subdivision of finite sets into subsets. *Akad. Nauk BSSR Dokl.* **23**, 26–28 (1979)
70. F. Tardella, On the equivalence between some discrete and continuous optimization problems. *Ann. Oper. Res.* **2**, 291–300 (1990)
71. G.M. Ziegler, *Graduate Texts in Mathematics. Lecture Notes on Polytopes* (Springer, New York, 1995)

Optimization in Multi-Channel Wireless Networks

Hongwei Du, Weili Wu, Lidong Wu, Kai Xing, Xuefei Zhang and
Deying Li

Contents

1	Introduction	2360
2	Multi-interface Assignment	2360
2.1	Bandwidth Allocation and NIC Assignment	2361
2.2	Efficient Solution	2363
3	Channel Assignment	2365
3.1	Minimum Interference	2366
3.2	Maximum Throughput	2368
3.3	Optimal Fairness	2371
4	Network Routing	2375
4.1	Multicasting	2375
4.2	Data Collection	2380
4.3	Data Aggregation	2382
	Cross-References	2386
	Recommended Reading	2387

H. Du (✉)

Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, People's Republic of China

e-mail: hwdu@hitsz.edu.cn

W. Wu • L. Wu • K. Xing • X. Zhang

Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

e-mail: weiliwu@utdallas.edu; lidong.wu@utdallas.edu; kai.xing@utdallas.edu;
wo0faye@gmail.com

D. Li

Department of Computer Science, Renmin University, Beijing, People's Republic of China

e-mail: deyingli@ruc.edu.cn

Abstract

Although most of research efforts are made on single-channel wireless networks, multichannel environment already exists in real world systems. In this chapter, we survey current research works in multi-channel wireless networks, especially in multihop wireless networks, such as wireless ad hoc, sensor, and mesh networks. We will focus on those optimization issues, to explore the hardness of extensions from single-channel to multichannel and valuable research topics. The chapter is divided into four parts: (1) introduction, (2) multi-interface assignment, (3) channel assignment, and (4) network routing. For each part, we give a detail explanation of the latest research in multichannel wireless networks.

1 Introduction

Although most of research efforts are made on single-channel wireless networks, multichannel environment already exists in real world systems such as 2.4 GHz and 5 GHz spectrum. To utilize those resource and to take advantage of this environment, many research works have made contributions to extend techniques from single-channel to multichannel environment and explore valuable new research problems. In this chapter, we survey those research works, especially on optimization issues.

2 Multi-interface Assignment

An interface is a network interface card equipped with a half-duplex radio transceiver, for example, a commodity IEEE 802.11 wireless card. Also, the definition of wireless network interface is a network card (NIC) connected to a radio-based computer network.

There are several advantages in using multi-interface instead of a single interface.

First, with multi-interface, a node can receive and transmit data in parallel so that the throughput on a multihop route can be increased, possibly doubled. Secondly, with multi-interface, the end-to-end latency can be reduced if different hops traversed are on different channels. Indeed, for each node with a single interface, if the interfaces of two nodes are on different channels, then they cannot communicate. Thus, when packets are traversing multihop paths, delaying may occur at each hop and hence the end-to-end latency can be increased.

A multichannel wireless mesh network (WMN) consists of a number of stationary wireless routers, forming a wireless backbone. Each router is equipped with multiple network interface cards (NICs) [45, 48, 55]. Each interface operates on a distinct frequency channel in the IEEE 802.11a/b/g bands. A logical topology is comprised of the sets of routers and logical links.

A problem for using a WMN as a backbone for large wireless access is the high bandwidth requirements, which cause capacity reduction. One solution to address this problem is to use multiple nonoverlapping channels on the interfering links to maximize the parallel transmission and throughput. Although the one-NIC

architecture is able to support simultaneous transmissions as long as the interfering links operate in different channels, its limit on the parallel use of multiple channels cannot be ignored. Since different NICs can operate on different channels without interference, multi-NICs for mesh network have made it possible for one node to access multiple channels at the same time.

Many researchers study on channel allocation issues. There are some exiting centralized and distributed algorithms to solve this problem. Marina et al. [41] and Tang et al. [51] proposed the joint channel allocation and topology control problem. Marina et al. [41] proposed a permanent channel allocation algorithm to solve the problem; this algorithm aims to minimize the interface among the links while maintaining the network connectivity. Tang et al. [51] proposed a permanently allocating frequency channels method to solve the problem; the method is to minimize the maximum number of interfering logical links within each neighborhood. The constraint is that the logical topology graph should be K-connected. In [45], the authors consider channel allocation and topology control as two separate but related problems and formulate the logical topology formation and interface assignment as a joint optimization problem. In [46] and [48], the authors assume that there is no system or hardware support to allow a radio interface to switch channels on a per-packet basis. They proposed a centralized joint channel assignment and multipath routing algorithm. This algorithm considers high load edges first. The routing algorithm uses both shortest path routing and randomized multipath routing algorithm. In [25, 27], they both assume that a radio interface is capable of switching channels rapidly and is supported by system software. In [25], the authors give a channel assignment and routing algorithm to characterize the capacity regions. In [27], the authors focus on how the capacity of multichannel wireless networks scale related to the number of radio interfaces and the number of channel as the number of nodes grow.

The backbone of the WMN is modeled as a directed graph $G = (V, E)$, where V represents the set of nodes in the network and E represents the set of directed links. Each node u ($u \in E$) may be equipped with a number of NICs, where the number is denoted as $I(u)$ ($I(u) \geq 1$). The total number of nonoverlapping channels is k , and the channel set is denoted as $C(|C| = k)$. Each NIC can be fixed on one channel during the entire transmission lifetime or can switch channels after several packets have been sent out. Two nodes can communicate with each other via wireless links when one pair of their NICs is tuned to be on the same channel.

2.1 Bandwidth Allocation and NIC Assignment

In this section, the bandwidth allocation and NIC assignment problem are addressed for the WMNs, which is shown in [55]. There are the following assumptions: (1) the topology of the WMN has been established; (2) each node u , due to physical constraints, can be equipped with at most $\delta(u)$ numbers of NICs and all NICs are fixed; and (3) each node u has a bandwidth requirement of $req(u)$. The objective of the problem is to max–min bandwidth allocation α_{\max} , that is, maximize guaranteed bandwidth for each node. This problem is practically interesting because it answers

the following question: If certain bandwidth allocation should be satisfied, what is the requirement to the NIC assignment?

Before the problem formulation given by Wang et al. [55] is formally introduced, some terms and constraints which will be used in the formulation are introduced. Suppose every node $u \in V$ has a bandwidth request as $req(u)$ and it is allocated with bandwidth of $g(u)$, since the allocated bandwidth should not be greater than the required bandwidth, there is the bandwidth constraint as

$$g(u) \leq req(u), \quad \forall u \in V \quad (1)$$

As in [55], for each node, the traffic gathered from local clients has to be relayed to at least one of its neighbors. The traffic flows can be transmitted over different channels. Let $f_{\text{out}}(u, i)$ denote node u 's traffic-out on channel i , and $f_{\text{in}}(u, i)$ the traffic-in on channel i . For a node u , every channel is assumed to have the same capacity, denoted as $c(u)$. In each channel, the traffic-in and traffic-out should not exceed the channel capacity, so there are the following channel capacity constraints:

$$f_{\text{in}}(u, i) + f_{\text{out}}(u, i) \leq c(u), \quad \forall u \in V, i \in C. \quad (2)$$

Since total traffic-in and traffic-out of node u over all channels should keep a balance, the flow balance constraints were also concluded in [55] as follows:

$$\sum_{i \in C} f_{\text{in}}(u, i) + g(u) = \sum_{i \in C} f_{\text{out}}(u, i), \quad \forall u \in V. \quad (3)$$

Another constraint (NIC constraint) was raised in this work [55], that is, to enable the network connectivity, each node must be equipped with at least one NIC. Since the system has total k available channels, different NICs in one node must operate on different channels; hence, there are at most k NICs on a single node. That is the NIC constraint by channels

$$1 \leq I(u) \leq k, \quad I(u) \in \text{Integer}, \quad \forall u \in V. \quad (4)$$

In the case of Fix-NIC [55], $R(u, i)$ is used to indicate whether there is a NIC operating on channel i , and $R(u, i) = 1$ indicates there is a NIC fixed on channel i ; otherwise, $R(u, i) = 0$. The relationship between $I(u)$ and $R(u, i)$ is

$$\sum_{i \in C} R(u, i) \leq I(u), \quad R(u, i) \in \{0, 1\}, \quad \forall u \in V. \quad (5)$$

For every node, a NIC can only transmit or receive from one neighbor at the same time [55]. It is

$$\frac{f_{\text{out}}(u, i)}{c(u)} + \frac{f_{\text{in}}(u, i)}{c(u)} \leq R(u, i), \quad \forall u \in V, i \in C. \quad (6)$$

The problem formally is stated as a linear programming problem. Given $G(V, E)$ and all parameters,

maximize α
subject to:

$$\begin{aligned}
 g(u) &\leq \text{req}(u), \quad \forall u \in V \\
 f_{\text{in}}(u, i) + f_{\text{out}}(u, i) &\leq c(u), \quad \forall u \in V, i \in C \\
 \sum_{i \in C} f_{\text{in}}(u, i) + g(u) &= \sum_{i \in C} f_{\text{out}}(u, i), \quad \forall u \in V \\
 1 \leq I(u) &\leq k, \quad I(u) \in \text{Integer}, \quad \forall u \in V \\
 \sum_{i \in C} R(u, i) &\leq I(u), \quad R(u, i) \in 0, 1, \quad \forall u \in V \\
 \frac{f_{\text{out}}(u, i)}{c(u)} + \frac{f_{\text{in}}(u, i)}{c(u)} &\leq R(u, i), \quad \forall u \in V, i \in C \\
 g(u) &\geq \alpha, \quad u \in V. \tag{7}
 \end{aligned}$$

2.2 Efficient Solution

As the number of nodes in the mesh network increases, the cost to compute the optimal solutions by LP will increase dramatically due to the problem complexity. Hence, the LP solution is only suitable for off-line decisions for building or analyzing the static WMN architecture. If the system would have to recalculate the allocation and assignment strategy online, according to the topology change or new traffic requirement in a large WMN, finding the optimal solution is not practical. In [55], the authors discussed the feasible solutions case by case.

2.2.1 Assign NICs to Satisfy a Bandwidth Allocation

Given a feasible bandwidth allocation $g(u)$ for each node u , the first question is how to calculate the number of NICs ($I(u)$) in order to satisfy the bandwidth requirement. First, the case of Switch-NIC is considered, in which one NIC can switch among different channels. If the channels are not fully utilized around, it can always switch to a less used channel to alleviate the impact of interference, and all NICs in the WMN can work cooperatively. So the NIC is able to fully consume the capacity of $c(u)$. The only constraint is the capacity and the number of NICs. Hence, the number of NICs for node u can be approximated as

$$I_{\text{switch}}(u) = \left\lceil \frac{f(u)}{c(u)} \right\rceil. \tag{8}$$

Here, $f(u)$ represents the total traffic of node u . When taking Fix-NICs into consideration, the proper number of NICs depends much more on the interference

around. For instance, if nearby nodes have heavy traffic and every channel is equally utilized, then the transmission load by using just the same number of Fix-NICs cannot be accomplished in the case of Switch-NICs. Here, a compensation factor $\beta(u)$ to reflect the interference impact is introduced. Let $f_{\text{int}}(u)$ represent all the traffic flows that may conflict with the in and out traffic of node u ; then $\beta(u)$ is defined as

$$\beta(u) = \frac{f_{\text{int}}(u)}{k \cdot c(u)}, \quad k = |C|. \quad (9)$$

So the number of NICs required can be approximated as

$$I_{\text{fix}}(u) = \left\lceil (1 + \beta(u)) \cdot \frac{f(u)}{c(u)} \right\rceil. \quad (10)$$

These equations in [55] can be used to calculate $I(u)$ for each node.

2.2.2 Optimize Bandwidth Allocation with a Given NIC Assignment

For this subproblem, [55] showed the solutions for two cases:

(1) Case 1: Switch-NIC

If each node is assigned a number of Switch-NICs, the algorithm to find a feasible bandwidth allocation for all nodes is presented in [55]. Here, consider two allocation strategies: equally allocating the bandwidth to each node or allocating the bandwidth proportionally to the request. In both scenarios, the algorithm calculates the maximum value. First, an algorithm is proposed to estimate the max-min bandwidth allocation α_{\max} under the assumption that the bandwidth is equally allocated to each node and the bandwidth is α . For a node u , the total available capacity is $I(u) \cdot c(u)$, and the total traffic in and out should not exceed this capacity. So there must have

$$f(u) \leq I(u) \cdot c(u), \quad u \in V. \quad (11)$$

According to the aggregate traffic model, the total traffic-in can be deduced and traffic-out of a node u is $f_{\text{in}}(u) = \alpha \cdot |E|$ and $f_{\text{out}}(u) = \alpha \cdot |E|$. So the total traffic of u is

$$f(u) = \alpha \cdot 2 \cdot |V|. \quad (12)$$

According to Eqs. (11) and (12), there are as follows:

$$\alpha \cdot 2 \cdot |V| \leq I(u) \cdot c(u), \quad u \in V \quad (13)$$

or

$$\alpha \leq \frac{I(u) \cdot c(u)}{2 \cdot |V|}, \quad u \in V. \quad (14)$$

Equation (14) [55] gives an upper bound to the bandwidth allocation for each node. Now another scenario is considered: allocating bandwidth proportionally according to the request $req(u)$. So every node u will have different bandwidth

allocation as $\lambda \cdot req(u)$. Similar to the deduction above, there are

$$f(u) = \lambda \cdot 2 \cdot \sum_{u \in V} req(u) \quad (15)$$

or

$$\lambda \leq \frac{I(u) \cdot c(u)}{2 \cdot \sum_{u \in V} req(u)}, \quad u \in V. \quad (16)$$

The optimal α and λ can be achieved by minimizing Eqs. (14) and (16) [55].

(2) Case 2: Fix-NIC

When the NIC cannot switch channel freely, the total available capacity for each node u is less than $I(u) \cdot c(u)$. As [55] mentioned before, $\beta(u)$ is introduced to represent the impact of interference. Here, the total available capacity is $\frac{I(u) \cdot c(u)}{1 + \beta(u)}$. So the bandwidth allocation $g(u)$ for Fix-NIC is $1/(1 + \beta(u))$ of the bandwidth allocated for Switch-NIC. According to Eqs. (14) and (16), the maximum bandwidth allocation for each node with equal allocation strategy and the maximum proportion value are presented in [55] as the following:

$$\alpha \leq \frac{I(u) \cdot c(u)/(1 + \beta(u))}{2 \cdot |V|}, \quad u \in V \quad (17)$$

$$\lambda \leq \frac{I(u) \cdot c(u)/(1 + \beta(u))}{2 \cdot \sum_{u \in V} req(u)}, \quad u \in V. \quad (18)$$

3 Channel Assignment

Wireless interference significantly impacts the performance in multihop networks. This is because most of the current multihop networks use just a single channel to communicate so that neighboring nodes interfere with each other. The IEEE 802.11 standards provide several orthogonal channels that can be used for wireless networks. Usually, in the networks working in infrastructure mode, neighboring access points operate on orthogonal channels to eliminate interference. However, this kind of design does not apply to most of the multihop networks for some reasons. The most important reason is that most of the commodity mobile devices have only one network interface card, and they cannot operate simultaneously using multichannel.

Recently, wireless mesh architecture is more and more used to provide high-bandwidth wireless network services. But since this architecture is breaking long distances into a series of shorter hops, the interference problem still exists. Recently, many studies begin to focus on using multichannels in WMNs, and some studies have presented channel assignment algorithms to improve the network performance.

Some of the channel assignment algorithms [1, 40, 46, 48, 52] change the channels whenever there are significant changes to traffic loads or network topology.

These channel assignment algorithms are relatively static. But if the network topology or the traffic load changes frequently, the performance of the network would significantly drop because of the delay of the channel switching.

Some other algorithms [5, 49, 59] are designed to change the channel on the interface very frequently. These algorithms use the current state of the medium to determine the channel for each packet. However, an important condition for these algorithms is that the channel switch delay of the devices in the network should be small enough (switching in a packet or a few packets). And this condition eliminates most of the off-the-shelf hardware.

3.1 Minimum Interference

3.1.1 Optimization Problem Description

Recent studies of minimum interference in multichannel wireless networks provide channel assignment algorithms optimizing this problem. In these studies, the main issue is how to design the channel assignment algorithms. The main goal of the algorithms is to use the limited channels and the network interface in each node to minimize the interference.

In [50], the authors present the problem of static assignment of channels to link in the networks with multi-radio nodes. The objective of the channel assignment is to minimize the overall network interference. This channel assignment algorithm is derived from a graph coloring problem. The authors assume a binary interference model, in which there are two links that either interfere or not. Though the interference level between two links depends on the traffic on the links, the paper assumes uniform traffic on all links first and then generalizes the algorithms to nonuniform traffic.

3.1.2 Mathematical Formulation

The communication graph of the network is modeled as an undirected graph over the set of nodes N . Each node $i \in N$ denotes a router that has R_i radio interfaces. There is a total of K channels available in the network. Let $\mathcal{K} = \{1, 2, \dots, K\}$ denote the set of K available channels.

Let the conflict graph G_c [22] represent the interference between nodes. A conflict graph $G_c(V_c, E_c)$ is an undirected graph in which

$$V_c = \{l_{ij} \mid (i, j) \text{ is a communication link}\}.$$

And a conflict edge (l_{ij}, l_{ab}) is used to represent that two communication links (i, j) and (a, b) interfere with each other if they are on the same channel.

Then the channel assignment problem converts to a coloring problem, which is to color the vertices V_c of the conflict graph G_c using K colors without violating the interface constraint and minimizing the number of monochromatic edges in G_c . To solve this problem, a function $f : V_c \rightarrow \mathcal{K}$ needs to be computed to minimize the overall network interference $I(f)$. The following constraint satisfies the interface constraint:

$$\forall i \in N, |\{k | f(e) = k \text{ for some } e \in E(i)\}| \leq R_i$$

where $E(i) = \{l_{ij} \in V_c\}$, that is $E(i)$ is a set of vertices in V_c that denote the communication links incident on node i and the network interference $I(f)$ is

$$I(f) = |\{(u, v) \in E_c | f(u) = f(v)\}|.$$

The network interference is the number of monochromatic edges in the colored conflict graph.

3.1.3 An Existing Solution

In this section, a channel assignment algorithm presented in [50] is introduced. This algorithm, which is based on tabu search [19] technique, consists of two phases. In the first phase, a tabu search-based technique is used to find a solution f without considering the interface constraint. Then the violations of interface constraint are removed to get a feasible channel assignment function f in the second phase.

First Phase: First, each vertex in V_c is given a random color in K . And this assignment is taken as the initial solution f_0 , then a sequence of solutions $f_0, f_1, f_2, \dots, f_j, \dots$ is created to get a solution with minimum network interference. The next solution f_{j+1} in the j th iteration is created.

The j th Iteration: Start by making a certain number (say, r) of random neighboring solutions of f_j . A random neighboring solution of f_j can be made by picking a random vertex u and assigning a different random color in $(K - \{f_j(u)\})$ to it. Therefore, a neighboring solution of f_j is different from f_j in the color assignment of only one vertex. Such randomly generated neighboring solutions of f_j is made a solution set. Then pick the neighboring solution which has the lowest network interference as the next solution f_{j+1} from the set. In order to avoid local minima, $I(f_{j+1})$ cannot be required to be less than $I(f_j)$.

Tabu List: A tabu list [19] τ is maintained in order to avoid reassigning the same color to a vertex more than once. For example, if f_{j+1} is created by assigning a new color to a vertex u , then $(u, f_j(u))$ is put into the tabu list τ . According to the tabu list, the solutions, which assign the color k to vertex u if (u, k) is already in τ , are discarded.

Termination: When the iterations pass the allowed maximum number (say, i_{\max} without any improvement in $I(f_{\text{best}})$, the first phase terminates. In this algorithm, let i_{\max} be $|V_c|$. The network interference $I(f)$ is an integer, and it is lower than $(|V_c|)^2$. Let r denote the number of random neighboring solutions, and d denote the maximum degree of a vertex in G_c . At each iteration, the value of $I(f_{\text{best}})$ decreases by 1 or the first phase terminates. Therefore, the time complexity of the first phase is $O(r d |V_c|^3)$.

Second Phase: After the first phase, a solution of channel assignment is got. However, the number of channels on a node may be larger than the number of

interface cards on a node. The goal of the second phase is to remove the violations by using the following “merge” procedure. Assume a solution f has been got in the first phase. Based on this solution on the conflict graph G_c , the communication graph G is used in the second phase. After the first phase, each edge in the communication graph is colored.

To begin with, a node which the violation is maximum for the merge operation is chosen. Let i be the node picked, two colors k_1 and k_2 incident on i are chosen. Then the color of all k_1 -colored links is changed to k_2 . To avoid creating more interface constraint violations at other nodes, all the k_1 -colored links which are neighboring (two or more links are incident on a common node) are iteratively changed to the link just changed. This procedure ensures that for any node j , either all or none of the k_1 -colored links incident on j are changed to color k_2 .

The steps above can reduce the number of distinct colors incident on i by one and does not increase the number of violations on other nodes. Therefore, this procedure can be repeated to eliminate all the interface constraint violations on the communication graph. However, this procedure will increase the interference in the network. Therefore, those two colors k_1 and k_2 are picked for the merge operation that cause the least increase of network interference.

Simulations in [50] show that this algorithm effectively reduces the interference of the network and therefore significantly improves the performance.

3.2 Maximum Throughput

3.2.1 Optimization Problem Description

Recent studies present multichannel multihop network models to optimize the throughput. In these studies, how to design the channel assignment algorithms is the core problem because the number of channels available and the number of the interfaces in each node are limited. So it is needed to design channel assignment algorithms to get the maximum throughput in the proposed networks. In [3], the authors present a fixed channel assignment algorithm for multi-radio multichannel mesh networks. In [4], the authors formulate the joint channel assignment and routing problem mathematically into a Linear Programming (LP) problem. The LP takes into account the interference constraints, the number of channels in the network, and the number of radios available at each router. Then the authors use this formulation to develop a solution for our problem that optimizes the overall network throughput. The optimization algorithm in [4] will be introduced in the following sections.

3.2.2 Problem Description and System Modeling

In [4], a multichannel multihop infrastructure WMN is used as a network model. The WMN consists of some static wireless mesh routers and some mobile terminal clients. These wireless routers form a multihop wireless backbone. The backbone network relays the traffic from and to the clients. Some routers are the gateway to the wired Internet.

To simplify the problem, it is assumed that there is only outgoing traffic to the wired Internet and $l(u)$ represents the outgoing traffic. Then the mathematical formulation is given in [4].

3.2.3 Mathematical Formulation

The given backbone of the wireless network is modeled as a graph (V, E) . V is the set of all routers. Each edge $e \in E$ denotes a link between two routers in V . For that at least $\lambda l(u)$ of throughput can be routed to the Internet, it requires to find the maximized λ . Let t denote the destination node which connects directly to the Internet.

Some definitions are given as follows:

1. There are K channels available.
2. $I(e) \subset E$ denotes the set of edges that e interferes with.
3. Each node $u \in V$ has $I(u)$ network interface cards.
4. Each node has a total demand $l(u)$ from all the connected users.

To get $\lambda l(u)$ throughput for each node u , to compute the following is necessary:

1. A network flow that consists of edges $e = (u, v)$ and values $f(e(i))$, ($1 \leq i \leq K$) where $f(e(i))$ denotes the rate of the traffic which is transmitted from node u to node v using channel i .
2. A feasible channel assignment $F(u)$, which is an ordered set where the i th interface of u operates on the i th channel in $F(u)$, such that, whenever $f(e(i)) > 0$, $i \in F(u) \cap F(v)$. This case is called that edge e is using channel i to communicate.
3. A feasible schedule S which decides the set of edge channel pair (e, i) (i.e., edge e is using channel i) scheduled at time slot τ , for $\tau = 1, 2, \dots, T$ where T is the period of the schedule.

If the edges of no two edge pairs $(e_1, i), (e_2, i)$ scheduled in the a time slot for the same channel i interfere with each other ($(e_1 \notin I(e_2) \text{ and } e_2 \notin I(e_1))$), the schedule is called as the feasible schedule. Let $c(e)$ denote the maximum rate at which router u can communicate with router v in one hop on a same channel. And let (R_T) denote the interference range, which is assumed to be q times a transmission range R_T for some small fixed constant q .

Then the following constraints are got:

The Link Congestion Constraint: Any valid edge flows without interference must satisfy for channel 1(or other equivalent channels):

$$\frac{f(e(1))}{c(e)} + \sum_{e' \in I(e)} \frac{f(e'(1))}{c(e')} \leq c(q). \quad (19)$$

Link Schedulability Constraint: If the edge flows satisfy for channel 1 (or other equivalent channels) the following link schedulability constraint, then an interference-free edge communication schedule can be found:

$$\frac{f(e(1))}{c(e)} + \sum_{e' \in I(e)} \frac{f(e'(1))}{c(e')} \leq 1. \quad (20)$$

$f(e)$ is defined as the rate at which traffic is transmitted by node u for node v on the only available channel. These values of $f(e)$ form the edge flows for a max-flow on a flow graph H . And H is constructed by the following steps. First, a source node s and sink node t are introduced into the graph G . And every gateway node $v \in V$ has an unlimited capacity directed edge to the sink node t . Node s has a directed edge to every node $v \in V$ with a capacity of $l(v)$. Let the set E_s denote these edges and H be the required flow graph with s and t . The flow on every edge $e \in E$ in H is $f(e)$, and every $e \in E_s$ in H has a flow equal to its capacity. According to the definition, the flow is a valid network flow, and it is a max-flow on H for the given source and sink nodes.

Flow Constraints: Let $V^H(E^H)$ be the set of nodes (edges) in H and $f(e)(c(e))$ denote the flow (capacity) on an edge e of H . At any node $v \in V^H$ other than s or t , the total incoming flow is equal to the total outgoing flow:

$$\sum_{(u,v) \in E^H} f((u,v)) = \sum_{(v,u) \in E^H} f((v,u)), \forall v \in V^H - \{s, t\}. \quad (21)$$

A flow is required on each outgoing edge for the largest possible λ . This result in the constraint

$$f((s, s_v)) = \lambda l(v), \forall v \in V. \quad (22)$$

Also, it is ensured that no capacities are violated:

$$f(e) \leq c(e), \forall e \in E^H. \quad (23)$$

Node Radio Constraints: Let $\alpha(e, i) = \frac{f(e(i))}{c(e)}$ for any $e \in E$ and channel i . It comes to the following linear constraints for each node $v \in V$.

$$\sum_{1 \leq i \leq K} \sum_{e=(u,v) \in E} \frac{f(e(i))}{c(e)} + \sum_{1 \leq i \leq K} \sum_{e=(v,u) \in E} \frac{f(e(i))}{c(e)} \leq I(v). \quad (24)$$

Link Congestion Constraints: The link congestion constraint using link flows is changed:

$$\frac{f(e(i))}{c(e)} + \sum_{e' \in I(e)} \frac{f(e'(i))}{c(e')} \leq c(q), \forall e \in E, 1 \leq i \leq k. \quad (25)$$

Objective Function:

$$\max \lambda \quad (26)$$

The LP above involves channel assignment, routing, and link scheduling constraints. However, all the constraints we listed are necessary conditions. This means if a solution satisfies these constraints, it may still not be a feasible solution. This LP also yields a lower bound on the λ value, which can be used to establish the guaranteed worst performance case.

In [4], the authors present a channel assignment algorithm used to adjust the flow on the flow graph (routing changes) to ensure a feasible channel assignment. This flow adjustment also keeps minimum interference for each channel. The authors also provide algorithms for routing and link scheduling to form a complete solution of the optimization issue.

3.3 Optimal Fairness

In above sections, channel assignments in multichannel multi-radio WMNs have been studied, with the objective of minimum interference and maximum throughput. Now the fairness channel assignment scheme to optimally achieve Nash equilibrium (NE) solution with focus on fairness consideration to equalize the throughput of flows will be discussed.

3.3.1 Optimization Problem Description

Multichannel WMNs often consist of mesh clients, mesh routers, and gateways [2]. A mesh topology is made up of the sets of routers and logical links. To increase the capability of WMNs, each node should be equipped with multiple radios interfaces. Achieving fairness channel assignment of communications in WMNs is important. In some instances, fairness is even more crucial than system-wide throughout optimization [8]. A Nash equilibrium with perfect fairness among all players is provided. And it is proved by using this fair channel assignment; when system converges to a constant status by Nash equilibrium, all game players obtain the equal throughput.

The fairness problem was pointed out by Bharghavan et al., which was caused mainly because of hidden terminal problem as well as the backoff scheme used in the DFWMAC protocol [6, 21]. In this subsection, a system model is firstly formulated and presented, with some concepts of game theory and a game model in channel assignment. Then a Nash equilibrium solution is provided with perfect fairness and abundant proofs is given.

3.3.2 System Model and Concepts

In this section, the model which is going to be studied will formally be defined. Let K be the number of radio transmitters which is manufactured by each user in a single collision domain. Suppose that each user participates and only participates in one such communication. Let N be the set of communicating pairs and C be the channel set. We suppose that the same channels have the same conditions. And let T_c denote the total throughput of channel c and σ_c denote the throughput of each

radio on channel c . So σ_c can be assumed as a decreasing function of the number of radios on channel c [8].

The following definitions are given in [60] about game theory:

1. n is the number of all players.
2. S_i be a strategy set for player i .
3. S is the set of strategy profiles and $S = S_1 \times S_2 \dots \times S_n$.
4. f is the payoff function and $f = (f_1(x), \dots, f_n(x))$.
5. x_{-i}^* is a strategy profile of all players except for player i .

Then for each player $i \in 1, \dots, n$, if strategy x_i resulting in strategy profile $x = (x_1, \dots, x_n)$, then player i obtains payoff $f_i(x)$. Note that the payoff is decided by the strategy profile chosen.

Definition 1 Nash equilibrium: A strategy profile $x^* \in S$ is a Nash equilibrium (NE) if no unilateral deviation in strategy by any single player is profitable for that player; then

$$\forall i, x_i \in S_i, x_i \neq x_i^* : f_i(x_i^*, x_{-i}^*) \geq f_i(x_i, x_{-i}^*). \quad (27)$$

Definition 2 Perfect fairness [8]: In the channel assignment game, the strategy profile s is perfectly fair, if $\forall i, j \in N$,

$$f_i(x) \geq f_j(x), \quad \text{and} \quad \sum_{c \in C} r_{i,c} = \sum_{c \in C} r_{j,c}. \quad (28)$$

3.3.3 Mathematical Formulation

People who use network nowadays need to pay for their connection and communication with other people in the network. Therefore, it is fairly to assume that charging to players for the usage of the channels to transmit packets is necessary. More significantly, a Nash equilibrium with perfect fairness can be completed by evaluating and designing carefully the amount of payment what players should give on the basis of the usage method of the channels. Here there is an assumption that a certain kind of virtual currency exists in the system and it demonstrates the payment for using channels. Every player pays to the system administrator some virtual currency on the basis of the channels which has chosen to use. Therefore, let $p_{i,c}$ represent the payment of player i for using channel c ; a definition for $p_{i,c}$ is given as

$$p_{i,c} = r_{i,c} \left| k_{i,c} - a - \left\lceil \frac{1}{c+1} - \frac{1}{b+1} \right\rceil \right| \quad (29)$$

in which $a = \left\lfloor \frac{K}{|C|} \right\rfloor$, $b = K \bmod |C|$.

Therefore, the total utility of player i is defined as

$$u_i = \sum_{c \in C} r_{i,c} - \sum_{c \in C} p_{i,c}. \quad (30)$$

A strategy profile x^* is defined as for every player $i \in N$,

$$x_i^* = \{x_{i,1}^*, x_{i,2}^*, \dots, x_{i,|C|}^*\} = \{\underbrace{a+1, \dots, a+1}_b, \underbrace{a, \dots, a}_{|C|-b}\}. \quad (31)$$

Note that $(a+1)b + a(|C|-b) = K$, which is the total number of radios that each player has.

Theorem 1 *In the channel assignment game described before, x^* achieves a Nash equilibrium.*

In order to clearly describe the theorem, the following definition for any player i [8] is given:

$$k_{i,c}^* = \begin{cases} a+1 & \text{if } 1 \leq c \leq b \\ a & \text{if } b+1 \leq c \leq |C| \end{cases} \quad (32)$$

Note that for each channel $c \in C$, the identification number of that channel c is also used by us. Given that $b > 0, c > 0$,

$$\left\lceil \frac{1}{c+1} - \frac{1}{b+1} \right\rceil = \begin{cases} 1 & \text{if } 1 \leq c \leq b \\ 0 & \text{if } b+1 \leq c \leq |C| \end{cases}. \quad (33)$$

From previous two equations, for $\forall c \in C$, there is

$$\left| k_{i,c}^* - a - \left\lceil \frac{1}{c+1} - \frac{1}{b+1} \right\rceil \right| = 0. \quad (34)$$

so for $x_i^*, \sum_{c \in C} p_{i,c} = 0$. Therefore,

$$f_i(x_i^*, x_{-i}^*) = \sum_{c \in C} r_{i,c}. \quad (35)$$

Now the strategies other than x_i^* will be taken into consideration. $\forall x_i \neq x_i^*$, let C'_i denote the channel set on which the radio assignment of player i deviates from x_i^* . Since $x_i \neq x_i^*, \exists c \in C, k_{i,c} \neq k_{i,c}^*$. So $|C'_i| > 0$.

Next, it will be proved that for player i , $\forall c \in C'_i, p_{i,c} \geq r_{i,c}$. Since $k_{i,c}$ is not a negative integer and $k_{i,c} \neq k_{i,c}^*, \forall c \in C'_i$,

$$\left| k_{i,c} - a - \left\lceil \frac{1}{c+1} - \frac{1}{b+1} \right\rceil \right| \geq 1. \quad (36)$$

From the previous (1) and (8) equations, it can be obtained that $\forall c \in C'_i, p_{i,c} \geq r_{i,c}$.

Eventually, there is

$$\begin{aligned}
& f_i(x_i^*, x_{-i}^*) - f_i(x_i, x_{-i}^*) \\
&= \sum_{c \in C} r_{i,c}^* - \left(\sum_{c \notin C'_i} r_{i,c} - \sum_{c \notin C'_i} p_{i,c} + \sum_{c \in C'_i} r_{i,c} - \sum_{c \in C'_i} p_{i,c} \right) \\
&\geq \sum_{c \in C} r_{i,c}^* - \left(\sum_{c \notin C'_i} r_{i,c} - 0 + \sum_{c \in C'_i} r_{i,c} - \sum_{c \in C'_i} r_{i,c} \right) \\
&= \sum_{c \in C} r_{i,c}^* - \sum_{c \notin C'_i} r_{i,c} \\
&= \sum_{c \in C} r_{i,c}^* - \sum_{c \notin C'_i} r_{i,c}^* \\
&= \sum_{c \in C'_i} r_{i,c}^* \geq 0.
\end{aligned}$$

Hence, x^* is a Nash equilibrium.

Next, it will be demonstrated that S^* is perfectly fair as well.

Theorem 2 S^* reaches perfect fairness.

From $r_{i,c} = k_{i,c} \Sigma(k_c)$, it can be observed that throughput of player i on channel c depends on $k_{i,c}$ and the sum of the radios number on the channel c , k . It is easy to see that in x_i^* , the total number of radios on channel $c(k_c^*)$ is

$$k_c^* = \sum_{i \in N} k_{i,c}^* = \begin{cases} (a+1)|N| & \text{if } 1 \leq c \leq b \\ a|N| & \text{if } b+1 \leq c \leq |C| \end{cases} \quad (37)$$

where $|N|$ is the number of players. Given (7) and (9), $\forall i \in N$, there is

$$\begin{aligned}
f_i^* &= \sum_{c \in C} r_{i,c} = \sum_{c \in C} k_{i,c}^* \sigma(k_c^*) \\
&= \sum_{1 \leq c \leq b} (a+1) \sigma((a+1)|N|) + \sum_{b+1 \leq c \leq |C|} a \sigma(a|N|) \\
&= b(a+1) \sigma((a+1)|N|) + (|C| - b)a \sigma(a|N|)
\end{aligned}$$

Therefore, the value of f_i^* is independent for i . Therefore, $\forall i, j \in C$, $f_i^* = f_j^*$, and $\sum_{c \in C} r_{i,c} = \sum_{c \in C} r_{j,c}$.

Theorem 2 describes the strategy profile s^* , which achieves a Nash equilibrium. Moreover, the theorem also guarantees the perfect fairness in the system. That means each player can obtain the same total throughput such as perfect fairness when the system converges to a stable state. For all positive values of K and $|C|$, **Theorems 1** and **2** hold true.

3.3.4 Conclusion

In this section, a Nash equilibrium (NE) approach with perfect fair channel assignment in the selfish WMNs was formulated. The fairness is measured by the equal total throughput of all players no matter how many players are in the network.

Using this approach we validated the proposed result, compared it against Max–min fairness channel assignment.

4 Network Routing

4.1 Multicasting

4.1.1 Optimization Problem Description

Limitations on single radio and single channel have explored the attention to apply multichannel technologies in wireless networks. Channel assignment in multi-radio wireless networks apparently mitigates the interference and increases the network capacity. Since channel assignment has an impact on the link bandwidth which affects to the choosing of transmission links without interference, consequently, it also has relation with network routing to satisfy the traffic demand. The routing mechanisms in multichannel wireless networks should be well suited for the use with the proposed channel assignment strategy.

Previous works [16, 47] show the advantage of employing multiple nonoverlapping channels to overcome the interference problems. Several research efforts have been made on channel assignment methods [16, 49] and routing algorithms [14, 47] for multi-radio and multichannel (MR-MC) WMNs. The *minimum cost multicast tree* (MCMT) problem in (MR-MC) wireless networks [32] is discussed. Considering the dynamic traffic model, multicast routing (session) requests arrive one by one without any prior knowledge.

Ruiz and Gomez-Skarmeta [58] have proved that the MCMT problem in WMN is NP-hard. In [29], a heuristic algorithm has been proposed to build up a multicast structure with channels which are assigned to the nodes in the networks to avoid interferences. But this work could only support for one single multicast session with predetermined traffic pattern. In this section, the MCMT problem in MR-MC WMNs with dynamic traffic model is discussed. Assumed that the channel assignment is given and static, an Integer Linear Programming (ILP) model is presented to optimally solve the MCMT problem.

4.1.2 System Model and Concepts

A static MR-MC WMN is considered where each node in the network behaves like a TAP [28, 30]. Some nodes in the networks serve as gateway nodes which could directly access to the Internet. Each node is equipped with one or multiple radios and nonoverlapping channels. The set of available channels is set to Q , and the channel assignment is assumed as given and static. The network is represented as a directed multi-graph $G = (V, E)$ where V is the set of nodes and E is the set of edges. Each edge in E is denoted by $(i, j)_k$, representing a directed edge from node i to j operating on channel k . Assuming that each radio has a common transmission and interference range, a pair of directed edges $(i, j)_k$ and $(j, i)_k$ exists in E if the nodes i and j are in each other's transmission range and have a radio tuned to channel k .

4.1.3 Mathematical Formulation

Given instances, $V(T, k)$ denotes the set of transmitting nodes on channel k in multicast tree T . The multicast tree total cost is

$$\text{cost}(T) = \sum_{k \in Q} |V(T, k)|$$

where Q is the set of available channels.

The MCMT problem is formulated as follows: Given an (G, s, M) , where $G = (V, E)$ is a network graph, $s \in V$ is a source node, and $M \subseteq V$ is a set of destinations. The problem is how to find a minimum cost multicast tree in G that is rooted at s and spans all destinations in M .

An Integer Linear Programming (ILP) model to solve the MCMT problem optimally is proposed. Given an instance (G, s, M) of the MCMT problem, the network topology and channel assignment are described as

$$\{X_{i-j,k} : \forall i, j \in V, i \neq j, k \in Q\}$$

where $C_{i-j,k}$ equals to one if a directed edge $(i, j)_k$ exists in G , and zero otherwise.

A multicast tree solution can be represented by a set of binary variable, $\{X_{i-j,k} : \forall i, j, i \neq j, k \in Q\}$ where $X_{i-j,k}$ equals to one if the multicast tree includes a directed edge $(i, j)_k$, and zero otherwise. If the directed edge $(i, j)_k$ includes in the multicast tree, then it must be in the network G . Thus, the following constraints are obtained:

$$X_{i-j,k} \leq C_{i-j,k}, \forall i, j \in V, i \neq j, \forall k \in Q.$$

The network flow model [31] is used to guarantee that the resulting multicast tree indeed spans all the destinations in M . Let F_{i-j} denote the aggregate amount of energy consume from node i to j on any channel. From the network flow model, the following constraints can be obtained:

1. Source node s sends $|M|$ units of energy to the network:

$$\sum_{j \in V, j \neq s} F_{s-j} = |M|.$$

2. No input flows to the source node:

$$\sum_{j \in V, j \neq s} F_{j-s} = 0.$$

3. Each destination takes out one unit of energy flow from the passing energy flow, while non-destination nodes do not cost energy from the flow:

$$\sum_{j \in V, j \neq i} F_{j-i} - \sum_{j \in V, j \neq i} F_{i-j} = 1, \forall i \in M$$

$$\sum_{j \in V, j \neq i} F_{j-i} - \sum_{j \in V, j \neq i} F_{i-j} = 0, \forall i \in V, i \notin M, i \neq s.$$

4. Only when an edge between nodes i and j is included in the multicast tree it is possible that $F_{i,j} > 0$:

$$|M| \cdot \sum_{k \in Q} X_{i-j,k} \geq F_{i-j}, \forall i, j \in V, i \neq j.$$

The equation in list 4 relates the $X_{i-j,k}$ variables to the flow variables F_{i-j} . Note that the optimization procedure excludes the case that two edges (operating on different channels) between the same two nodes are both included in the multicast tree. To achieve the minimize cost of multicast tree, a set of variables is introduced:

$$\{Y_{i,k} : \forall i \in V, k \in Q\}.$$

$Y_{i,k}$ equals to one if node i is a transmitting node on channel k in the resulting multicast tree, and zero otherwise. Thus, a set of constraints relate to $Y_{i,k}$ to the $X_{i-j,k}$ can be obtained:

$$Y_{i,k} \geq X_{i-j,k}, \forall i, j \in V, i \neq j, \forall k \in Q$$

Therefore, the objective function is

$$\text{minimize } \sum_{k \in Q} \sum_{i \in V} Y_{i,k}$$

where the multicast tree cost is the total number of transmitting nodes in the tree over all channels. The ILP formulation for the MCMT problem is summarized below.

$$\text{minimize } \sum_{k \in Q} \sum_{i \in V} Y_{i,k}$$

subject to

$$X_{i-j,k} \leq C_{i-j,k}; \forall i, j \in V, i \neq j, \forall k \in Q$$

$$\sum_{j \in V, j \neq s} F_{s-j} = |M|;$$

$$\sum_{j \in V, j \neq s} F_{j-s} = 0;$$

$$\sum_{j \in V, j \neq i} F_{j-i} - \sum_{j \in V, j \neq i} F_{i-j} = 1, \forall i \in M;$$

$$\sum_{j \in V, j \neq i} F_{j-i} - \sum_{j \in V, j \neq i} F_{i-j} = 0, \forall i \in V, i \notin M, i \neq s;$$

$$\begin{aligned}
|M| \cdot \sum_{k \in Q} X_{i-j,k} &\geq F_{i-j}, \forall i, j \in V, i \neq j; \\
Y_{i,k} &\geq X_{i-j,k}, \forall i, j \in V, i \neq j, \forall k \in Q; \\
X_{i-j,k} &\in \{0, 1\}; \forall i, j \in V, i \neq j, \forall k \in Q \\
F_{i-j} &\geq 0; \forall i, j \in V \\
Y_{i,k} &\in \{0, 1\}; \forall i \in V, \forall k \in Q.
\end{aligned}$$

4.1.4 Solution

Next, a polynomial-time approximation algorithm, called Wireless Closest Terminal Branching (WCTB), is proposed for the MCMT problem. The main idea of the WCTB is that:

1. For each step, one new branch is added to the current multicast tree to reach a new destination that is closest to the source node.
2. The cost of each link in the networks initials as one. The crux is that when a new branch is included into the multicast tree, the cost of links may become zero due to the broadcast advantage.
3. It is easy to implement based on the Dijkstra's algorithm.

Theorem 3 *The WCTB approximation ratio is no more than $(2 \cdot D_{max})$, that is,*

$$\frac{\text{cost}(T_{\text{WCTB}})}{\text{cost}(T_{\text{opt}})} \leq 2 \cdot D_{max}$$

where T_{opt} denote the optimal multicast tree and T_{WCTB} denote the multicast tree generated by WCTB. And $\text{cost}(T_{\text{WCTB}})$ and $\text{cost}(T_{\text{opt}})$ denote the tree costs produced by the WCTB and the optimal tree, respectively.

Given $G = (V, E; s, M)$, let t_1 denote the source node and t_{i+1} denote the i th destination in M . Let $\text{dist}(u, v)$ denote the shortest distance in terms of hop count from node u to v in G . The proof of **Theorem 3** needs two lemmas.

Lemma 1 *Suppose p_1, p_2, \dots, p_k is a permutation of $1, 2, \dots, k$, then for every $i, 2 \leq i \leq k$, there exists a unique critical pair (p_{j-1}, p_j) such that $\min(p_{j-1}, p_j) < i \leq \max(p_{j-1}, p_j), 2 \leq j \leq k$.*

Lemma 2 *For any instance $(G = (V, E), s, M)$ of the MCMT problem,*

$$\text{cost}(T_{\text{opt}}) \geq \frac{\sum_{i=1}^{|M|} \text{dist}(t_{p_i}, t_{p_{i+1}})}{2 \cdot D_{max}}$$

where $\{t_{p_1}, t_{p_2}, \dots, t_{p_{|M|+1}}\}$ is a permutation of $\{t_1, t_2, \dots, t_{|M|+1}\}$ obtained by a preorder traversal of T_{opt} starting from the source node s .

Both lemmas proof can be found in [32]. From the above lemmas, for $2 \leq i \leq |M| + 1$, there is

$$\sum_{i=2}^{|M|+1} dist(n_i, n'_i) = \sum_{j=2}^{|M|+1} dist(m_{p_{j-1}, m_{p_j}}).$$

In the algorithm WCTB, to reach a new destination, say m_i , a minimum cost path from s is integrated. The cost of the minimum cost path is $cost(s, m_i) = cost(m_1, m_i)$; therefore,

$$cost(T_{WCTB}) = \sum_{i=2}^{|M|+1} cost(m_1, m_i).$$

Furthermore, since the link cost of edges in the current multicast tree T in WCTB is zero, $cost(m_1, m_i)$ is no more than the distance between m_i and any destination that is included in the current multicast tree, and thus no more than the distance between n_i^{earlier} and $n_i^{\text{no-earlier}}$, that is, for $2 \leq i \leq |M| + 1$,

$$cost(m_1, m_i) \leq dist(n_i^{\text{earlier}}, n_i^{\text{no-earlier}}).$$

Combining the above proof equations,

$$cost(T_{WCTB}) \leq \sum_{j=2}^{|M|+1} dist(m_{p_{j-1}}, m_{p_j})$$

is obtained. Accordingly, from Lemma 2 and above equation, there is

$$cost(T_{\text{opt}}) \geq \frac{\sum_{i=1}^{|M|} dist(t_{p_i}, t_{p_{i+1}})}{2 \cdot D_{\max}} \geq \frac{cost(T_{WCTB})}{2 \cdot D_{\max}}.$$

Thus,

$$\frac{cost(T_{WCTB})}{cost(T_{\text{opt}})} \leq 2 \cdot D_{\max}.$$

4.1.5 Conclusion

In this section, MCMT problem is considered in MR-MC WMNs. The problem is in concern of dynamic traffic model. An ILP model for solving this problem optimally has been provided. The polynomial-time near-optimal algorithm, WCTB, is proposed to solve the problem efficiently.

4.2 Data Collection

Multichannel wireless sensor networks for supporting a wide range of applications such as environmental monitoring, military surveillance, and traffic control have been drawn attention to researchers in the current wireless research area. Such kind of networks can be used to collect sensory information from the real world. In order to collect data, there are two types of data gathering in multichannel wireless sensor networks. One is data collection [11, 38, 63] which gathering data from the source nodes and send them to the base station. Different from data collection, data aggregation [15, 24, 33, 57] rather than sending individual data from sensor nodes to base station is that multiple data are aggregated as a data to be forwarded by the sensor network. In data collection [23], the problem of collecting all the sensing information from the sensors at a particular time is defined as snapshot data collection. Continuous data collection is represented as the collection of multiple continuous snapshot.

The network capacity [18] is one of the network measurement for the network performance evaluation, while wireless sensor networks suffer from the interference problem during data transmission. In [23], the data receiving rate at the sink node is used to measure the data collection capacity. Extensive works have been studied in distinct communication tasks such as broadcast capacity [36], multicast capacity [34, 35], unicast capacity [44], and snapshot data collection capacity [11, 38, 63]. Most of the previous works are based on single radio and single channel in wireless sensor networks which the sensor node is only working on the half-duplex mode. In such sensor network environment, data transmission suffers from the radio conflict [12, 27] and channel interference [7, 13] issues. In this section, a dual radio multichannel is applied which dramatically will reduce the interference. The problem considering in this section is based on the protocol interference model [54]. Two algorithms are presented which investigate the snapshot and continuous data collection problem in dual radio multichannel wireless sensor networks.

4.2.1 Network Model and Concepts

The network is represented as a graph $G = (V, E)$, where V is the set of sensor nodes and E is the set of links in V . The network is composed of n sensor nodes and one sink node. Each sensor is equipped with two radios and each radio's transmission radius is fixed to 1 and the interference range is defined as ρ , $\rho > 1$. In protocol interference model, a receiving node v successfully receives the data from the transmitting node u only if $\|u - v\| \leq 1$ and furthermore any other node s which satisfies $\|s - v\| \leq \rho$ will not simultaneously transmit data packet over the same channel with u . Each radio has K nonoverlapping channels as $\omega_1, \omega_2, \omega_3, \dots, \omega_K$. Each sensor can transmit at a rate of W bits/second over a channel [11]. It is assumed that all the transmissions are synchronized and the size of a time slot is $t = b/W$ seconds, where b bits is the size of all the packet transmission in the network. The capacity γ of data collection is the ratio between the size of the data successfully received by the sink and the time τ used to collect these data. The capacity to collect

N snapshots of data is $\gamma = \frac{Nnb}{\tau}$, which is truly the data rate at the sink. A connected dominating set based on the mechanism used in [57] is built for constructing a routing tree. Vertex coloring terminologies have been derived in [43, 57].

4.2.2 Scheduling Algorithm for Snapshot Data Collection

In this section, a novel scheduling algorithm is proposed with a tighter lower capacity bound and order-optimal. Since at any time slot, the sink can receive data from at most one sensor, the upper bound of the snapshot data collection capacity is W [11].

A new multipath scheduling algorithm based on the previous built routing tree in [57] is proposed [23]. First how to schedule a single path and extend it to the schedule of multipath in the routing tree is discussed. For simplicity, the concept of round, which indicates a period of time consisting of multiple continuous time slots, is introduced. The path is denoted by P , and P_0 (resp., P_e) denotes the set of links on P whose heads (resp., tails) are dominators and whose tails have at least one packet to be transmitted. Scheduling P has the following two steps.

Step 1. In an odd round, schedule every link in P_0 once, that is assign a dedicated channel and one dedicated time slot to each link in P_0 .

Step 2. In an even round, schedule every link in P_e once, that is assign a dedicated channel and one dedicated time slot to each link in P_e .

Then the multipath scheduling on the routing tree is discussed. The basic idea of the multipath scheduling algorithm is as follows:

Step 1. For the nonintersecting paths without wireless interference, schedule them by the single path scheduling algorithm concurrently.

Step 2. For the nonintersecting paths with wireless interference, schedule them according to a certain order. The path with the smallest subscript has the highest priority.

Step 3. For the intersecting paths, schedule them according to a certain order, for example, if P_i is the one of these intersecting paths with the smallest subscript, schedule P_i until there is no packet having to be transmitted on the sub-path and then continue to schedule the next path.

Theorem 4 *The capacity γ at the sink of T of the multipath scheduling algorithm is at least*

$$\frac{W}{\frac{3.63}{K} \rho^2 + o(\rho)},$$

which is order-optimal.

The proof can be found in [23]. Compared with the recent result in [11], the proposed algorithm has a lower bound of $\frac{W}{8\rho^2}$.

4.2.3 Scheduling Algorithm for Continuous Data Collection

In this section, a novel pipeline scheduling algorithm base on the compressive data gathering (CDG) [38] in dual radio multichannel wireless sensor networks

is presented. From the benefit brought by CDG, the continuous data collection problem with the pipeline technique [11] can be addressed, different from the multipath scheduling algorithm, in which a sensor sends one packet over a link in one time slot. A sensor sends M packets for a snapshot employing CDG. A concept of a super time slot (STS) is introduced which consists of M time slots. In a STS, a sensor can send M packets over a channel for a snapshot. For the links working simultaneously, channels and STSs are assigned using the similar way of the multipath scheduling algorithm.

Theorem 5 *For a long-run continuous data collection, the lower bound of the achievable asymptotic network capacity of the pipeline scheduling algorithm is*

$$\frac{nW}{12M\left(\frac{3.63}{K}\rho^2 + o(\rho)\right)}$$

when $\Delta_e \leq 12$ or

$$\frac{nW}{M\Delta_e\left(\frac{3.63}{K}\rho^2 + o(\rho)\right)}$$

when $\Delta_e > 12$.

Proof detail is explained in [23].

4.2.4 Conclusion

Motivated from the no research work studying the continuous data collection before [23], the problem in the dual radio multichannel wireless sensor networks is investigated. Two algorithms in snapshot and continuous data collection are proposed. The snapshot data collection algorithm can achieve the network capacity of at least $\frac{W}{\frac{3.63}{K}\rho^2 + o(\rho)}$, and the continuous data collection achieves asymptotic network capacity of $\frac{nW}{12M\left(\frac{3.63}{K}\rho^2 + o(\rho)\right)}$ when $\Delta_e \leq 12$ or $\frac{nW}{M\Delta_e\left(\frac{3.63}{K}\rho^2 + o(\rho)\right)}$ when $\Delta_e > 12$. The simulation results show that the proposed algorithms' network capacity outperforms the existing works.

4.3 Data Aggregation

Recent advancement of radio technology has enabled the sensor devices which are capable of sensing, computation, and communication setting in a multichannel environment. A multichannel wireless sensor network (MWSN) communication does not rely on any infrastructure but hop-by-hop sensor communication. To collect the sensory information such as temperature, medical, and surveillance data in this type of network, an efficient and robust data collection mechanism is in need. Data aggregation is one of the data gathering paradigm to perform in-network fusion of data packets with small number and size of data transmission. The goal of data aggregation is to achieve the minimum interference and latency in the network.

Lots of works focus on distinct communication tasks with the objective of minimum latency. Many existing works have been studied in the minimum latency interference-aware broadcast scheduling extensively [9, 39, 56]. In [9], Chen et al. studied the minimum latency interference-aware broadcast scheduling considering the transmission range and interference range. Mahjourian et al. [39] extended a much more consideration of carrier sensing range in the minimum latency conflict-aware broadcast scheduling. Wan et al. [56] investigated the minimum latency scheduling for group communications (broadcast, aggregation, gathering, and gossiping) in multichannel multihop wireless networks. Data aggregation shows out to be more effective in data gathering in MWSN. Previous research considers the conflict-free data aggregation issues [10, 20, 26, 57, 61] in single-channel wireless sensor networks. The works proposed in [10, 20, 61] are only considered the nodes' transmission range, and all nodes in the existing networks have data packet sent to the base station. Wan et al. [57] studied the minimum data aggregation latency problem in a more general situation. An algorithm with latency bound $\beta_{\rho+1}(15R + \Delta - 4)$ was proposed in [57], when the interference range is ρ times of the transmission range. Krishnamachari et al. [26] viewed data aggregation from another aspect. They considered the case where there was a subset of nodes whose data need to be sent to the base station and regarded aggregating these data as a way to save energy. The work [62] is the only one existing for data aggregation considering the interference range and carrier sensing range of the wireless nodes.

From the above existing works, this section addresses the more generic minimum latency conflict-free data aggregation scheduling problem in multi-channel multihop wireless sensor networks [37]. Given an instance in which the network consists of a set of sensor nodes and a base station, the discussed problem is formally defined as to find a schedule such that data from a set of sensors can be successfully transmitted to the base station without any wireless conflict and the latency of accomplishing data transmission is minimized.

4.3.1 Conflicts and Conflict-Free Communication

In this section, the conflict-free communication model is formulated. Two parallel data transmissions can be successfully transmitted only if all the following types of conflicts are avoided. Those transmissions which avoid all the types of conflicts are called conflict-free transmissions:

1. Collision at receiver: If there are two or more nodes in node u 's transmission range to transmit using the same channel k , node u cannot correctly receive messages transmitted by its neighbors using channel k .
2. Interference at receiver: If there is a node in node u 's interference range to transmit using channel k , node u cannot correctly receive the messages from its neighbors transmitting using channel k .
3. Contention at sender: If there is a node in node u 's carrier sensing range transmitting using channel k , node u cannot correctly transmit messages to its neighbors using channel k .

4.3.2 System Model and Problem Definition

The instance network consists of n sensors and a base station. Each sensor is equipped with a RF transceiver with omnidirectional antennae to send or receive data. There are $\lambda \geq 1$ available nonoverlapping channels for node transmitting and receiving data. The transmission range, interference range, and carrier sensing range which roughly represented as disks are considered during the data transmission.

The mathematical formulation of the communication network is illustrated as a unit disk graph (UDG) $G = (V, E)$. Each node has the network factor of $r, \alpha r, \alpha \geq 1, \beta r, \beta \geq 1$ which indicate the transmission range, interference range, and carrier sensing range, respectively. An edge exists between node u and node v if and only if $d(u, v) \leq r$, where $d(u, v)$ is the Euclidean distance between u and v . There exists an edge between node u and the base station b if and only if $d(u, b) \leq r$.

The problem is then formulated as follows: Given instances of a UDG $G = (V, E)$ along with a base station $b \in V$, a subset $S \subseteq V - \{b\}$, and parameters $\lambda, r, \alpha, \beta$, the considering problem is how to find a conflict-free many-to-one data aggregation schedule with latency minimized.

4.3.3 Data Aggregation Scheduling Algorithms

In this section, the data aggregation scheduling algorithm will be proposed and analyzed. The algorithm will mainly contain two phases. The first phase is to construct an initial many-to-one data aggregation tree. The second phase is to derive an aggregation schedule from the initial data aggregation tree.

4.3.4 Many-to-One Data Aggregation Tree Construction

The main idea of the initial data aggregation tree is to firstly take a breath-first search of some nodes which are not taken action in the many-to-one schedule with minimum latency. After that, a many-to-one tree is constructed based on the subgraph induced by these nodes. The breadth-first search (BFS) tree is rooted at the base station denoted by T_{BFS} for the network $G = (V, E)$ and eliminates the nodes from G and T_{BFS} which are not on the paths from any node $u \in S$ to the base station in T_{BFS} . Note that the nodes eliminated will not join in the schedule.

The node elimination could be done as follows. (1) Let $A = V - \{b\}$ and $P(u)$ be the set of all the nodes on the path from node u to the base station in T_{BFS} . Then the nodes in $A - \bigcup_{u \in S} P(u)$ need to be eliminated. (2) After eliminating the nodes, the obtained network and breadth-first search tree are denoted by G_p and $T_{\text{BFS}, p}$, respectively.

The fact is that the subgraph G_p of G and the subgraph $T_{\text{BFS}, p}$ of T_{BFS} are induced by $\bigcup_{u \in S} P(u) \cup \{b\}$.

Based on the subgraphs G_p and $T_{\text{BFS}, p}$, we construct a many-to-one data aggregation tree. After pruning the network, we construct a maximal independent set (MIS) on the pruned network according to the breadth-first search ordering. Then

we transform the MIS to minimum cover problem and compute the minimum cover to obtain the initial data aggregation tree. The data aggregation tree does not prevent any conflicts. More detailed explanation is in [37].

4.3.5 Many-to-One Data Aggregation Scheduling

The assignment of many-to-one data aggregation scheduling is designed by node coloring. The fact in node coloring is that any graph G is able to produce a proper node coloring in G using at most $1 + \delta^*(G)$ colors. Smallest-degree-last ordering of nodes is applied to assign proper color to each node in particular order [42].

The following useful lemmas are given to construct a conflict-free schedule.

Lemma 3 *Two parallel transmissions “ $t_1 \xrightarrow{k_1} r_1$ ” and “ $t_2 \xrightarrow{k_2} r_2$ ” are conflict-free ($t_1 \neq t_2$ and $r_1 \neq r_2$), if and only if at least one of the following two conditions are satisfied:*

1. $d(t_1, r_2) > \alpha r$; $d(t_2, r_1) > \alpha r$; $d(t_1, t_2) > \beta r$
2. $k_1 \neq k_2$

The following lemma shows the sufficient conditions for avoiding the three types of conflict.

Lemma 4 ([39]) *In order for two parallel transmissions “ $t_1 \xrightarrow{k} r_1$ ” and “ $t_2 \xrightarrow{k} r_2$ ” using the same channel k to be conflict-free according to the network model, it is sufficient to have $d(t_1, t_2) > \max(\alpha + 1, \beta)r \vee d(r_1, r_2) > (\max(\alpha, \beta) + 2)r$.*

Two conflict graphs which are denoted by GC_t and GC_r according to Lemma 4 are constructed. The λ channels are denoted by integers $0, 1, \dots, \lambda - 1$. There are two cases. (1) When $\lambda = 1$, there is only one available channel. The only needed thing is to schedule the transmitting time slot of each node. After coloring the conflict graph, if two nodes have the same color, that is, their distance is bigger than $\max(\alpha + 1, \beta)r$ (or $(\max(\alpha, \beta) + 2)r$), then the two nodes can be scheduled to transmit (receive) at the same time slot. The nodes which have distinct colors will not be scheduled to transmit (receive) at the same slot. (2) When $\lambda > 1$, the nodes with distinct colors can schedule at the same time slot when assigning different channels to them. Thus, it is needed to schedule the transmitting time slot and the channel of each node. Fortunately, a uniform method is found to deal with the two situations.

Two sub-schedules are derived to obtain conflict-free scheduling [37]. Basing on the two sub-schedules, the conflict-free data aggregation schedule is constructed; the details can be found in [37].

4.3.6 Performance Analysis

In this section, the approximation ratio of the proposed algorithm, which is for the minimum latency conflict-free many-to-one data aggregation scheduling problem,

is analyzed. The performance of proposed algorithm is compared against the trivial lower bound R , where R is the depth of the T_{BFS_p} . A known theorem [17] is used to prove Lemma 5.

Theorem 6 ([17]) Suppose that C is a compact convex set and U is a set of points with mutual distances at least one, then

$$|U \cap C| \leq \frac{2\text{area}(C)}{\sqrt{3}} + \frac{\text{peri}(C)}{2} + 1,$$

where $\text{area}(C)$ and $\text{peri}(C)$ are the area and perimeter of C , respectively.

Lemma 5 Let $G_p = (V_p, E_p)$, $S \subseteq V_p$ be independent set in G_p , the inductivities of $GC_l[S]$ and $GC_r[S]$ have an upper bound of $\lfloor \frac{\pi}{\sqrt{3}}h^2 + (\frac{\pi}{2} + 1)h \rfloor$ and $\lfloor \frac{\pi}{\sqrt{3}}k^2 + (\frac{\pi}{2} + 1)k \rfloor$, respectively, where $h = \max(\alpha + 1, \beta)$ and $k = \max(\alpha, \beta) + 2$.

Lemma 6 ([57]) The following statements are true:

1. $|C_0| \leq 12$.
2. For each $1 \leq i \leq R_U - 1$, each dominator in U_i is adjacent to at most 11 connectors in C_i .
3. For each $0 \leq i \leq R_U - 1$, each connector in C_i is adjacent to at most 4 dominators in U_{i+1} .

Theorem 7 The algorithm has a latency upper bound $(\lceil \frac{a}{\lambda} \rceil + 11\lceil \frac{b}{\lambda} \rceil)R + (\Delta - 23)\lceil \frac{b}{\lambda} \rceil - \lceil \frac{a}{\lambda} \rceil + 12$, where $a = \lfloor \frac{\pi}{\sqrt{3}}h^2 + (\frac{\pi}{2} + 1)h \rfloor + 1$, $b = \lfloor \frac{\pi}{\sqrt{3}}k^2 + (\frac{\pi}{2} + 1)k \rfloor + 1$, Δ and R are the maximum degree of pruned network G_p and the depth of the pruned BFS tree T_{BFS_p} , respectively.

Corollary 1 For $\lambda = 1$, the algorithm has a latency upper bound $(a + 11b)R + (\Delta - 23)b - a + 12$ and thus it has a nearly constant $(a + 11b)$ ratio.

4.3.7 Conclusion

In this section, the minimum latency conflict-free many-to-one data aggregation scheduling problem in multichannel multihop wireless networks is addressed. Nearly a constant $(\lceil \frac{a}{\lambda} \rceil + 11\lceil \frac{b}{\lambda} \rceil)$ -ratio algorithm for the problem is introduced.

Cross-References

► [Energy Efficiency in Wireless Networks](#)

Recommended Reading

1. A. Adya, P. Bahl, J. Padhye, A. Wolman, L. Zhou, A multi-radio unification protocol for IEEE 802.11 wireless networks, in *Proc. of Broadnets*, 2004, pp. 344–354
2. I.F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: a survey. *Comput. Network* **47**(4), 445–487 (2005)
3. H.M.K. Alazemi, A. Das, R. Vijaykumar, S. Roy, Fixed channel assignment algorithm for multi-radio multi-channel mesh networks. *Wireless Comm. Mobile Comput.* **8**(6), 811–828 (2008)
4. M. Alicherry, R. Bhatia, L. (Erran) Li, Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks, in *Proc. of Mobicom'05*, 2005, pp. 58–72
5. P. Bahl, R. Chandra, J. Dunagan, SSCH: slotted seeded channel hopping for capacity improvement in IEEE 802.11 ad hoc wireless networks, in *MOBICOM*, 2004
6. V. Bharghavan, A. Demers, S. Shenker, L. Zhang, MACAW: a media access protocol for wireless LANs, in *Proceedings of ACM SIGCOMM*, 1994
7. V. Bhandari, N.H. Vaidya, Connectivity and capacity of multi-channel wireless networks with channel switching constraints, in *Infocom*, 2007
8. T. Chen, S. Zhong, Perfectly fair channel assignment in non-cooperative multi-radio multi-channel wireless networks. *Comput. Comm.* **32**(6, 27), 1058–1061 (2009)
9. Z. Chen, C. Qiao, J. Xu, T. Lee, A constant approximation algorithm for interference aware broadcast in wireless networks, in *INFOCOM*, 2007, pp. 740–748
10. X. Chen, X. Hu, J. Zhu, Minimum data aggregation time problem in wireless sensor networks, in *MSN*, 2005, pp. 133–142
11. S. Chen, S. Tang, M. Huang, Y. Wang, Capacity of data collection in arbitrary wireless sensor networks, in *Infocom*, 2010
12. W. Cheng, X. Chen, T. Znati, X. Lu, Z. Lu, The complexity of channel scheduling in multi-radio multi-channel wireless networks, in *Infocom*, 2009
13. H. Dai, K. Ng, R.C.-W. Wong, M. Wu, On the capacity of multi- channel wireless networks using directional antennas, in *Infocom*, 2008
14. A.K. Das, H.M.K. Alazemi, R. Vijayakumar, S. Roy, Optimization models for fixed channel assignment in WMNs with multiple radios, in *IEEE SECON*, 2005
15. H. Du, X. Hu, X. Jia, Energy efficient routing and scheduling for real-time data aggregation in WSNs. *Comput. Comm.* **29**(17), 3527–3535 (2006)
16. V. Gambiroza, B. Sadeghi, E.W. Knightly, End-to-end performance and fairness in multihop wireless backhaul networks, in *ACM MOBICOM*, 2004
17. H. Groemer, Über die Einlagerung von Kreisen in einen konvexen Bereich. *Math. Z.* **73**, 285–294 (1960)
18. P. Gupta, P.R. Kumar, The capacity of wireless networks. *IEEE Trans. Inform. Theor.* **46**(2), 388–404 (2000)
19. A. Hertz, D. de Werra, Using tabu search techniques for graph coloring. *Computing* **39**(4), 345–351 (1987)
20. S.C.-H. Huang, P.-J. Wan, C.T. Vu, Y.-S. Li, Frances Yao, Nearly constant approximation for data aggregation scheduling in wireless sensor networks, in *IEEE INFOCOM*, 2007
21. X.L. Huang, B. Bensaou, On max-min fairness and scheduling in wireless ad-hoc networks. Analytical Framework and Implementation, in *ACM*, 2001
22. K. Jain, J. Padhye, V.N. Padmanabhan, L. Qiu, Impact of interference on multi-hop wireless network performance. *Mobile Comput. Network.* 66–80 (2003)
23. S. Ji, Y. Li, X. Jia, Capacity of dual-radio multi-channel wireless sensor networks for continuous data collection, in *IEEE INFOCOM*, 2011, pp. 1062–1070
24. C. Joo, J. Choi, N.B. Shroff, Delay performance of scheduling with data aggregation in wireless sensor networks, in *INFOCOM*, 2010
25. M. Kodialam, T. Nandagopal, Characterizing the capacity region in multi-radio and multi-channel mesh network, in *Proc. ACM MOBICOM*, 2005

26. B. Krishnamachari, D. Estrin, S.B. Wicker, The impact of data aggregation in wireless sensor networks, in *ICDCSW*, 2002, pp. 575–578
27. P. Kyasanur, N.H. Vaidya, Capacity of multi-channel wireless networks: impact of number of channels and interfaces, in *MOBICOM*, pp. 43–57, 2005
28. J.-F. Lee, W. Liao, M.-C. Chen, An incentive-based fairness mechanism for multi-hop wireless backhaul networks with selfish nodes. *IEEE Trans. Wireless Comm.* **2**, 697–704 (2008)
29. T. Liu, W. Liao, On routing in multi-channel wireless mesh networks: challenges and solutions. *IEEE Network* **22**(1), 13–18 (2008)
30. T. Liu, W. Liao, Location-dependent throughput and delay in wireless mesh networks. *IEEE Trans. Veh. Technol.* **2**, 1188–1198 (2008)
31. T. Liu, W. Liao, Interplay of network topology and channel assignment in multi-radio multi-rate multi-channel wireless mesh networks, in *IEEE GLOBECOM*, 2008
32. T. Liu, W. Liao, Multicast routing in multi-radio multi-channel wireless mesh networks. *IEEE Trans. Wireless Comm.* **9**(10), 3031–3039 (2010)
33. H. Liu, Z. Liu, H. Du, D. Li, X. Lu, Minimum latency data aggregation in wireless sensor network with directional antenna, in *COCOA*, 2011, pp. 207–221
34. S. Li, Y. Liu, X.-Y. Li, Capacity of large scale wireless networks under Gaussian channel model, in *Mobicom*, 2008
35. X.-Y. Li, S.-J. Tang, O. Frieder, Multicast capacity for large scale wireless ad hoc networks, in *Mobicom*, 2007
36. X.-Y. Li, J. Zhao, Y.W. Wu, S.J. Tang, X.H. Xu, X.F. Mao, Broadcast capacity for wireless ad hoc networks, in *MASS*, 2008
37. D. Li, Q. Zhu, H. Du, W. Wu, H. Chen, W. Chen, “Conflict-free many-to-one data aggregation scheduling in multi-channel multi-hop wireless sensor networks”, in *ICC*, 2011
38. C. Luo, F. Wu, J. Sun, C.W. Chen, Compressive data gathering for large-scale wireless sensor networks, in *MobiCom*, 2009
39. R. Mahjourian, Feng Chen, R. Tiwari, My Thai, H. Zhai, Y. Fang, An approximation algorithm for conflict-aware broadcast scheduling in wireless ad hoc networks, in *MOBIHOC*, 2008, pp. 331–340
40. M.K. Marina, S. Das, A topology control approach to channel assignment in multi-radio wireless mesh networks, in *Broadnets*, 2005
41. M.K. Marina, Samir R. Das, A. Subramanian, A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. *Comput. Network* **54**(2), 241–256 (2010)
42. D.W. Matula, L.L. Beck, Smallest-last ordering and clustering and graph coloring algorithms, in *J. ACM*, 1983
43. D.W. Matula, L.L. Beck, Smallest-last ordering and clustering and graph coloring algorithms. *J. Assoc. Comput. Mach.* **30**(3), 417–427 (1983)
44. U. Niesen, P. Gupta, D. Shah, On capacity scaling in arbitrary wireless networks. *IEEE Trans. Inform. Theor.* **55**(9), 3959–3982 (2009)
45. A. Rad, W.S. Wong, Logical topology design and interface assignment for multi-channel wireless mesh networks. in *GLOBECOM*, 2006
46. A. Raniwala, T.-C. Chiueh, Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network, in *Proc. IEEE INFOCOM*, 2005
47. A. Raniwala, T.-C. Chiueh, Architecture and algorithms for an IEEE 802.11-based multi-channel WMN, in *IEEE INFOCOM*, 2005
48. A. Raniwala, K. Gopalan, T. Chiueh, Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks. *Mobile Comput. Comm. Rev.* **8**(2), 50–65 (2004)
49. J. So, N. Vaidya, Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver, in *ACM MOBICOM*, 2004
50. A.P. Subramanian, H. Gupta, S. Das, J. Cao, Minimum interference channel assignment in multiradio wireless mesh networks. *IEEE Trans. Mobile Comput.* **7**(12), 1459–1473 (2008)
51. J. Tang, G. Xue, W. Zhang, Interference-aware topology control and QoS routing in multi-channel wireless mesh networks, in *MobiHoc*, 2005, pp. 68–77

52. J. Tand, G. Xue, W. Zhang, Interference-aware topology control and QoS routing in multi-channel wireless mesh networks, in *MOBIHOC*, 2005
53. M. Thorup, U. Zwick, Compact routing schemes, in *SPAA*, 2001, pp. 1–10
54. P.-J. Wan, Z. Wang, H. Du, S.C.-H. Huang, Z. Wan, First-fit scheduling for beaconing in multihop wireless networks, in *INFOCOM*, 2010, pp. 2205–2212
55. J. Wang, H. Li, W. Jia, L. Huang, J. Li, Interface assignment and bandwidth allocation for multi-channel wireless mesh networks. *Comput. Comm.* **31**(17), 3995–4004 (2008)
56. P.J. Wan, Z. Wang, Z.Y. Wan, Scott C.-H. Huang, H. Liu, Minimum-latency schedulings for group communications in multi-channel multihop wireless networks, in *WASA*, 2009
57. P.-J. Wan, Scott C.-H. Huang, L.X. Wang, Z.Y. Wan, X.H. Jia, Minimum-latency aggregation scheduling in multihop wireless networks, in *ACM MOBIHOC*, 2009
58. J.E. Wieselthier, G.D. Nguyen, A. Ephremides, Energy-efficient broadcast and multicast trees in wireless networks. *Mobile Network Appl.* 481–492 (2002)
59. S. Wu, C. Lin, Y. Tseng, J. Sheu, A new multichannel MAC protocol with on-demand channel assignment for multi-hop mobile ad hoc networks, in *ISSPAN*, 2000
60. Wikipedia: <http://en.wikipedia.org/wiki/Nashequilibrium>. *Wikipedia*, 2011
61. X.H. Xu, S.G. Wang, X.F. Mao, S.J. Tang, X.Y. Li, An improved approximation algorithm for data aggregation in multi-hop wireless sensor networks, in *ACM MOBIHOC*, 2009
62. Q.H. Zhu, D.Y. Li, Approximation for a scheduling problem with application in wireless networks. *Sci. China Math.* **53**(6), 1407–1662 (2010)
63. X. Zhu, B. Tang, H. Gupta, Delay efficient data gathering in sensor networks, in *MSN*, 2005

Optimization Problems in Data Broadcasting

Yan Shi, Weili Wu, Jiaofei Zhong, Zaixin Lu and Xiaofeng Gao

Contents

1	Introduction	2392
2	Scheduling Optimization for Pure-Pushed Data Broadcasting	2394
2.1	Scheduling Optimization for Single-Channel Data Broadcasting	2395
2.2	Scheduling Optimization for Multichannel Data Broadcasting	2398
2.3	Scheduling Optimization for Dependent Data Broadcasting	2402
3	Optimizations for On-Demand Data Broadcasting	2406
3.1	Basic Online Scheduling Algorithms for On-Demand Data Broadcast	2408
3.2	$R \times W$ Scheduling for Large-Scale On-Demand Data Broadcast	2408
3.3	Scheduling Time-Critical On-Demand Data Broadcast	2412
3.4	Scheduling Preempting On-Demand Broadcast for Heterogeneous Data	2415
4	Data Retrieval Optimization in Data Broadcast Systems	2417
5	Indexing Techniques for Data Broadcast Optimization	2420
5.1	Indexing in Single-Channel Broadcasting Environment	2420
5.2	Indexing in Multichannel Broadcasting Environment	2435
6	Coverage Optimization for Data Broadcast	2440
6.1	Mathematical Modeling and Problem Definition	2441

Y. Shi (✉)

Department of Computer Science and Software Engineering, University of Wisconsin -
Platteville, Platteville, WI, USA

e-mail: shiy@uwplatt.edu

W. Wu • Z. Lu

Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA
e-mail: weiliwu@utdallas.edu; zaixinlu@utdallas.edu

J. Zhong

Department of Mathematics and Computer Science, University of Central Missouri, Warrensburg,
MO, USA

e-mail: zhong@ucmo.edu

X. Gao

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai,
People's Republic of China
e-mail: gao-xf@cs.sjtu.edu.cn

6.2 Classification and Reduction.....	2442
6.3 Cell Broadcasting Scheme to Support Multilingual Service.....	2444
6.4 Coverage Prediction and Optimization for Digital Broadcast.....	2445
6.5 Data Delivery Optimization in Multi-system Heterogeneous Overlayed Wireless Network.....	2447
6.6 Summary.....	2449
7 Conclusion.....	2449
Recommended Reading.....	2450

Abstract

Broadcasting is an efficient way to disseminate information to a large number of users. With the fast development of wireless communication techniques, wireless data broadcasting has become a popular data dissemination method. The major performance concerns for any wireless data broadcast systems are response time and energy consumption. Various optimization methods for different aspects of data broadcasting systems have been developed in order to improve the broadcast system performance. This chapter gives a complete overview of different optimization issues in data broadcasting. In this chapter, optimization problems and corresponding algorithms and solutions for data scheduling, indexing, retrieval, and coverage are discussed in various types of data broadcast systems, including single-channel, multichannel, push-based, and on-demand data broadcasting.

1 Introduction

Data broadcasting is not a new concept. The term has been used for more than 30 years. Traditionally, data broadcasting refers to the ability to send data other than video/audio through broadcast television/radio transport [14, 54]. A best-known example is captioning. Takenaga categorized this type of data broadcasting into independent data broadcasting and linked/supplementary data broadcasting. Independent data broadcasting includes only data, such as weather reports or stock information, while linked/supplementary data broadcasting is for the purpose of better facilitating regular television programs and can sometimes be interactive.

Traditional data broadcasting reached its peak during the middle and late 1990s due to the digitalization of broadcasting and the massive use of home networks. A lot of data broadcasting services became available, for example, on-screen TV guide, SkyLife service in South Korea, and Teletext service in United Kingdom. Most of these services, however, still use traditional broadcast media (cable or television satellites) for data transportation.

In recent years, the boosting of wireless technology and mobile communications (GPRS, 3G, 4G, etc.) has given data broadcasting a completely new meaning. In a wireless communication environment, data broadcasting means disseminating data to a large group of candidate clients in a broadcast manner. The beauty of wireless data broadcast is its scalability, flexibility, and bandwidth efficiency.

In wireless data broadcast systems, data are broadcasted from a base station on some channels of particular frequencies within a transmission range. Clients in this

range can access the channels and download the data. Several regulatively located base stations may work together to cover a certain area, each taking charge for its local cell named *microcell* (usually known as hexagon, quadrangle, or triangle, depending on the geometric layout of base stations). Clients can move freely among cells. In contrast to point-to-point data delivery, data broadcast allows a mass number of clients to simultaneously access the data and thus is more scalable. Broadcast technique also fits the asymmetric structures of most wireless systems because clients simply “listen” to the broadcast channels, filter, and pick the data they need, which alleviates the uplink channel load. Therefore, the system can make use of most bandwidth as downlink for data dissemination. Moreover, within the coverage range of a base station, no matter how many clients need to download the data, the network behavior remains the same.

In general, there are three types of data broadcast: push-based, pull-based, and hybrid. Push-based [1] data broadcast uses all bandwidth for broadcast. It is usually used to disseminate public information such as stock price, news reports, or traffic information. One benefit of pure push-based broadcast is that it makes full use of the network bandwidth for data dissemination. It also has very little computational requirement for the server because the broadcast program can be organized off-line. On the other hand, since there is no uplink in a push-based broadcast system, it is very hard to dynamically adjust the broadcast program to changes of clients’ requirement. Hence, push-based data broadcast is suitable for environments where the data and client access patterns are almost static.

Pull-based data broadcast is also known as “on-demand” data broadcast. Compared to push-based data broadcast, it has an additional uplink through which clients can send their requests to the server. Requests are queued in the server. Instead of organizing a broadcast program off-line and then broadcast it periodically, the server will use some online algorithm to decide the broadcast content based on the service queue in real time. In such a way, the broadcast content will usually reflect the clients’ real requirements and can also be adjusted dynamically according to changes of requests. Therefore, on-demand data broadcast can be used in environments where the data and client access patterns change frequently. However, handling client requests and making broadcast decision in real time means a heavy work load for the server and often requires high computational capacity.

Hybrid data broadcast takes advantage of both sides. It combines push- and-pull based mechanisms. Different hybrid systems have different integration methods. Some systems use push-based scheme to broadcast data with static access patterns and on-demand scheme for data with dynamic access patterns, while some other systems broadcast most popular data in a pure push way and handle not-so-popular data individually.

Response time is a major concern for all kinds of data broadcast systems. The definition of response time is not exactly the same for different systems. In general, *response time* for a request is the time interval between the moment a request is generated (or the client first tunes in a channel) to the moment the requested information is completely downloaded. In push-based systems, it is also called *access time*. There are many factors that can affect response time. From the network point of view, an intuitive way is to increase the throughput of the broadcast. Using more

channels for data broadcast can naturally increase the overall throughput. From the server side, different ways of scheduling the broadcast data may result in different response time, even with the same sets of requests. There have been intensive research efforts on scheduling algorithm design in order to optimize response time. [Sections 2 and 3](#) will introduce different scheduling algorithms for different push-based and on-demand data broadcast systems. From the client side, when requesting multiple items, the downloading sequence can also make a difference. [Section 4](#) will discuss several algorithms trying to optimize the retrieval scheduling.

In wireless data broadcast systems, since most clients are mobile devices operating on battery power, energy conservation becomes another important performance concern. A mobile device usually operates on two modes: the active mode and the doze mode. The energy consumed in the active mode can be 100 times of that in the doze mode [42]. Based on this observation, *tuning time*, which is the total time when a client is in the active mode, is defined as a performance criterium to evaluate energy efficiency of a system. In order to optimize energy consumption of a broadcast system, index is introduced. With the help of index information, clients do not need to stay active all the time to wait for requested data. Instead, they can change to the doze mode while waiting and be active again and tune in the broadcast channel right before the requested data arrive. This can significantly reduce the tuning time of clients. [Section 5](#) will talk in detail about dominating index methods for data broadcast in existing literature.

Another optimization issue related to data broadcast is the coverage problem. As discussed before, data are broadcasted from a base station. The range that a base station can cover is determined by the physical feature of both the base station and the terrain. Given a relative wide area, how to locate base stations in order to cover the whole area is an important optimization problem. Based on different objectives such as minimizing financial cost or enhancing reliability, the coverage problem can be formulated differently. In general, most of them are NP-hard and can be converted into combinatorial optimization problems. [Section 6](#) will give the formal model of the coverage problem and introduce different variations of coverage optimization for data broadcast.

2 Scheduling Optimization for Pure-Pushed Data Broadcasting

In a pure-pushed broadcast network, there are two distinct sets of entities: servers and clients. The servers at the base station repeatedly disseminate a fixed set of data items via wireless channels, while the clients listen to the channel, wait for their desired data items, and download them when they are coming. An example is shown in [Fig. 1](#). In general, the data items are scheduled over time, and the clients do not have to send requests to the server, but wait for the requested items to be broadcasted. Informally, the scheduling optimization problem for data broadcast is that given a set of data items, find a schedule for deciding the broadcast time for each data item of the set such that the expected average access time of the requests is minimized.

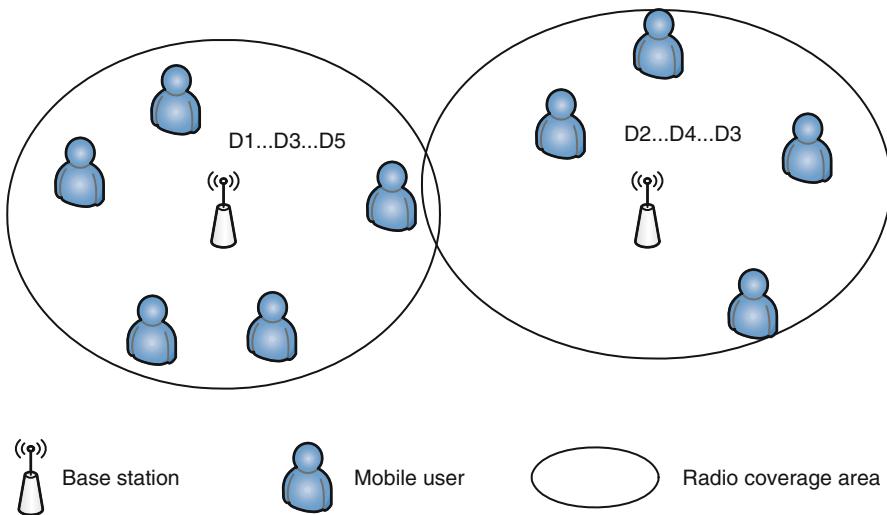


Fig. 1 Broadcast wireless network

In the past two decades, several variants for the scheduling problem of purepushed data broadcast have been discussed, and lots of works have been done. They can be roughly classified into three kinds: scheduling optimization for single-channel data broadcasting, scheduling optimization for multi-channel data broadcasting, and scheduling optimization for dependent data broadcasting. In Sect. 2.1 the works that focus on single channel are presented. Besides developing the single-channel data broadcast, the multichannel data broadcast is also an important area and recently more works focus on this issue. In Sect. 2.2 the research for scheduling multichannel data broadcast is introduced. In addition it is much more likely that a client query a set of data instead of only one data item. Therefore, some works recently discuss about broadcasting dependent data, whose access sequence is pre-ordered. How to schedule dependent data broadcast is discussed in Sect. 2.3.

2.1 Scheduling Optimization for Single-Channel Data Broadcasting

In the early stage of wireless data broadcast research, researchers first study on scheduling data on single broadcast channel. In 1995 Acharya et al. first proposed the scheduling problem for data broadcast [2]. A new technique called broadcast disks for data broadcast is introduced in that paper. In order to make an initial study of the scheduling problem for data broadcast, they defined many assumptions for the broadcasting environments. The assumptions include the following: (1) the data items of the broadcast program remains static; (2) each data item is read only;

(3) clients download the data items of interest from the broadcast server on demand; and (4) there is no upstream communications channels, which means that the clients don't have to send request to servers to get data. Based on these four assumptions, they presented two problems which need to be solved when designing data broadcast programs: the first one is that given a set of data items and the access probabilities for those data items by all the users, how to generate a broadcast program to satisfy as much as possible the needs of the users; the second one is that given the server with a fixed data broadcast schedule, how to manage the data items in cache for the individual users to maximize their own performances.

Furthermore, three types of broadcast programs are demonstrated in that paper. They are flat data broadcast, skewed data broadcast, and regular data broadcast. Examples for the three types of data broadcast programs are shown in Fig. 2. The first program is a flat data broadcast. It is easy to see that the average expected access time for all the data items on a flat broadcast program, regardless of their access frequencies to the clients, is the same for all the items and it equals to half a broadcast period. If considering the access frequencies, the server can also overbalance some data items in a broadcast cycle. Such a program can enhance broadcasting of the popular items. For the second and third programs, both of them broadcast $data_1$ twice as often as $data_2$ and $data_3$. The second program is called a skewed data broadcast, in which the data item $data_1$ is clustered together to be broadcasted, while, the third program is called regular data broadcast, where the same data items are separated by different data items with same intervals. Because the structure design of the third one is similar to the disk and memory design, they refer to regular data broadcast as multi-disk broadcasting.

Based on the idea of broadcast disk, they developed an efficient algorithm for scheduling data broadcast given a set of uniform-length data items and their access pattern. The algorithm has the following steps:

1. Sort the data items in descending order of their popularity.
2. Classify all data items into small subsets, and each subset contains data items with same or similar access frequencies. They call each subset a broadcast disk.
3. Choose an appropriate integer as the broadcast frequency for each of the disks.
4. Calculate the least common multiple (LCM) of all the frequencies; then, split each $disk_i$ into $\frac{LCM}{freq_i}$ small sub-disks.
5. These sub-disks are called chunks and denoted by C_{ij} . i, j refers to the j th chunk in $disk_i$.
6. Let N_i denote the number of chunks in $disk_i$.
7. Let M denote the number of disks.
8. Finally the broadcasting program is generated by [Algorithm 1](#).

The basic idea of the algorithm is to partition the data items into different kinds of disks, and the data items stored in smaller disks are broadcasted more frequently than the data items stored in larger disks. Thus, the data broadcast corresponds to the traditional notion of a memory hierarchy. They also pointed out that automatic determination of the parameters in the algorithm is an interesting optimization problem. Another topic they mentioned is the local cache management for clients.

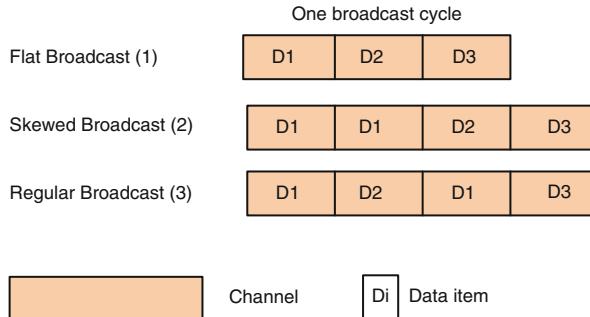


Fig. 2 Three broadcast programs

Algorithm 1 [2]

```

1:  $h \leftarrow \max N_i$ ;
2: for  $i = 0$  to  $h - 1$  do
3:   for  $j = 1$  to  $M$  do
4:      $k \leftarrow i \pmod{N_j}$ ;
5:     Broadcast  $C_{j,k}$ .
6:   end for
7: end for

```

Unlike traditional operating system, in the broadcast systems, a client should cache those data items for which the local probability of access (P) is greater than the frequency of broadcast (X). Based on this idea, they proposed two coast-based data replacement strategies. One is PIX (P Inverse X) which uses the ratio between the access frequency of a data item and its frequency of broadcast as the cost. It is an optimal policy but not easy to implement. The second one LIX (LRU-style) is an approximation for PIX.

In terms of single-channel data broadcast, some researchers concentrate on uniform-length data broadcast; others focus on nonuniform-length data broadcast. Ammar and Wong studied periodic data broadcast with uniform length and proposed a greedy algorithm for it in [7]. Moreover they defined the average response time for nonperiodic schedules; they also analyzed the properties of the optimal schedules and proved that the optimal schedule for periodic data broadcasting is also optimal for arbitrary data broadcasting in [8]. In that paper they presented two algorithms for solving the arbitrary data broadcast scheduling problem by using the works in [36]. The first one is a finite-time algorithm which gives an optimal schedule, and the second one is a heuristic algorithm which gives a nonoptimal schedule. However, they did not provide any performance ratio analysis for the heuristic algorithm in that paper. The similar problem is also pursued by Anily et al. in [9], in which they obtained a 2.5-approximation algorithm for the problem. Later Barnoy et al. developed a 2-approximation algorithm and proved that the heuristic algorithm proposed in [8] has approximation ratio 9/8 in [11].

In terms of nonuniform-length data broadcast, in [57, 58] Vaidya et al. investigated heuristic algorithms for it and showed some experimental results. Kenyon and Schabanel studied approximation algorithms for similar problems and gave theoretical analysis for the problem in both single-channel and multichannel environments, respectively, in [41]. The main results in [41] with respect to the case of single-channel and nonuniform-length data are that they proved the data scheduling problem with nonuniform length is NP-hard, and developed a randomized approximation algorithm for this problem. They first proved that there exists an optimal schedule for the problem which is periodic if broadcast costs are not considered; they also showed even in that case, the problem is NP-hard too. Then, they presented a randomized polynomial approximation algorithm for the problem. Its approximation performance ratio is 3. This is the first paper finding a constant-factor approximation algorithm for nonuniform-length data broadcasting scheduling.

2.2 Scheduling Optimization for Multichannel Data Broadcasting

Besides developing the scheduling mechanisms for single-channel data broadcast, multichannel data broadcast is also an important research area. Recently lots of works have been done for this problem. The multichannel broadcast can be achieved by applying techniques such as frequency-division multiplexing or time-division multiplexing. Consequently multichannel broadcast becomes feasible and can significantly reduce the request response time.

Ardizzone et al. discussed the scheduling problem of data broadcast over multiple channels in [10]. Like other works, they also defined many constraints for the problem such as the data items are allowed to be skewed allocated to different channels but are flat scheduled per channel. The objective is to minimize the average access time for all the requests. The assumptions they defined in that paper for the multichannel data broadcasting environment are as follows: a database $D = \{d_1, d_2, \dots, d_N\}$, which has N data items, is broadcasted by a set of K channels. The bandwidth for each channel is the same. Each data item d_i has a probability p_i and a length z_i . The access frequency of item d_i being requested by clients is denoted by p_i , and the time needed to download the data item d_i is represented by z_i . They also assume that the access frequencies for the items never change and the length z_i for data item d_i is subject to an integer number. It is obvious that $\sum_{i=1}^N p_i = 1$. When the lengths of all the data lengths are the same, it is called uniform-length multichannel data broadcasting, and the z_i for $1 \leq i \leq N$ is one time slot for the uniform-length data scheduling problem. Informally the target of this problem is to partition the N data items into K groups G_1, \dots, G_K such that the average access latency is minimized. The group G_j denotes the set of data items assigned to the j th channel. So there is a integer N_j to represent the size of G_j ,

while the sum of its item lengths is denoted by Z_j . It is clear that $Z_j = \sum_{d_i \in G_j} z_i$. Note that, since the data items in G_j are cyclically broadcasted according to a flat schedule, Z_j is the schedule period on the j th channel. $z_i = 1$ for $1 \leq i \leq N$ for the problem with uniform-length data items, $Z_j = N_j$ for $1 \leq j \leq K$. Assume an item is assigned to the j th channel and clients start to listen at any time slot with the same probability, the client average waiting time for receiving the item equals to half of the period, which is $Z_j/2$. Suppose with the help of indexes the clients know in advance the locations of data items on all the channels; then, the average expected delay (AED) for all the data items over all the channels can be computed by (1):

$$AED = \frac{1}{2} \sum_{j=1}^K \left(Z_j \sum_{d_i \in G_j} p_i \right). \quad (1)$$

To study the uniform-length data scheduling problem for multichannel data broadcasting, [Theorem 1](#) and some notations were proposed in [69].

Theorem 1 ([69]) *Let d_1, d_2, \dots, d_N be N items with access probabilities $p_i \leq p_N$. Then, there exists an optimal solution for partitioning the data items into K groups G_1, \dots, G_K , where $K \leq N$ and each group consists of consecutive elements.*

Let d_1, d_2, \dots, d_N be N data items which are sorted in the descending order of the access probabilities. Partition them in K subsets. Then, an optimal partition $\text{OPT}_{N,K} = (B_1, B_2, \dots, B_{K-1})$ is called leftmost optimal if for any other optimal solution $(B'_1, B'_2, \dots, B'_{K-1})$, B_{K-1} in $\text{OPT}_{N,K}$ is not bigger than B'_{K-1} in that solution, where B_i is the index of the last data item that belongs to group G_i and it is similar for B'_i .

In addition, an optimal solution $(B_1, B_2, \dots, B_{K-1})$ is called strict left-most optimal solution if (B_1, B_2, \dots, B_i) for every $i < K$ is a leftmost optimal solution. In the rest of this section, the leftmost optimal solution is denoted as $L_{N,K}$, and the strict leftmost optimal solution is denoted as $SL_{N,K}$.

In the paper they proposed a dynamic programming (DP) algorithm for the multichannel data scheduling problem. The algorithm computes only the leftmost optimal solution for subproblems and then finds a strict leftmost optimal solution. The most important property of using the DP algorithm is that given the items which are sorted by nonincreasing probabilities, if an optimal solution for the first $N - 1$ data items has been found, then to find an optimal solution for N data items, it only has to check the area larger than B'_{K-1} to find the final border B_{K-1} for $SL_{N,K}$, where B'_{K-1} is the border in the optimal solution for $N - 1$ data items. Such a property can be applied to solve the problems of increasing sizes to reduce the time complexity. If $L_{l,k}$ and $L_{r,k}$ are obtained for some $1 \leq l \leq r \leq N$, then it is clear that B_{k-1}^c is between B_{k-1}^l and B_{k-1}^r , for any $l \leq c \leq r$. If $C_{i,h}$ denotes the

cost of assigning consecutive items d_i, d_{i+1}, \dots, d_h to one of the channels, then $C_{i,h} = \frac{h-i+1}{2} \sum_{q=i}^h p_q$. The recurrences of the DP algorithm can be represent as Eqs. (2) and (3). Equation (3) is derived by choosing $c = \lceil \frac{l+r}{2} \rceil$ in Eq. (2):

$$\text{opt}_{c,k} = \min_{l \in \{B_k^l - 1, \dots, B_k^r\}} \text{opt}_{l,k-1} + C_{l+1,c}. \quad (2)$$

$$\text{opt}_{\lceil \frac{l+r}{2} \rceil, k} = \min_{l \in \{B_k^l - 1, \dots, B_k^r\}} \text{opt}_{l,k-1} + C_{l+1, \lceil \frac{l+r}{2} \rceil}. \quad (3)$$

Based on these two recurrences, it is easy to prove that the uniform-length multichannel data scheduling problem can be solved in $O(KN \log N)$ time by the DP algorithm, where K is the number of channels and N is the number of data items.

Besides the DP algorithm, they also proposed a faster $O(N \log N)$ time algorithm for the uniform-length multi-channel data scheduling problem with the number of channels less than 4. If the data items are already sorted, it requires only $O(N)$ time. The algorithm is based on the special case which has only two channels. When $K = 2$, adding a new item with lowest probability to an optimal solution for the subproblem can be done in constant time. This property can also be applied to find the optimal solution for the case $K = 3$. Actually, the solution for $K = 3$ can be obtained by combining the solutions for $K = 2$ and $K = 1$ in Eqs. (4) and (5):

$$\text{opt}_{N,2} = \min_{l \in \{B_1^{N-l}, B_1^{N-1} + 1\}} C_{1,l} + C_{l+1,N}. \quad (4)$$

$$\text{opt}_{N,3} = \min_{l \in \{1, \dots, N\}} \text{opt}_{1,l} + C_{l+1,N}. \quad (5)$$

For the case that data items have nonuniform lengths, they proved the problem is NP-hard even when $K = 2$. A pseudo-polynomial algorithm is presented with time complexity $O(NZ)$ where Z is the sum of the data lengths and N is the number of data items. They proved the problem is strong NP-hard for arbitrary K . An algorithm is proposed with time exponential to the maximum length of data items. Therefore, the time complexity of this algorithm is only acceptable for small data items. For larger data items, they proposed a heuristic algorithm which is a modified local search algorithm. Its time complexity is $O(N(K + \log N))$, and if the items are already sorted, the time complexity becomes $O(KN)$.

In the aspect of independent data scheduling, the queries issued by clients are assumed to only request one single item at one time. There are a lot of research that investigate how to minimize the average expected response time for multichannel data broadcasting. A brief summary is provided as follows. Yee et al. presented two algorithms to minimize the average expected access delay by partitioning data among multiple channels [69]. They continued their work in this filed to take into consideration the hopping cost in [70]. A near-optimal greedy algorithm for data scheduling of multichannel was proposed by Zheng et al. [73]. In addition, Prabhakara et al. suggested a new system model for multilevel and multichannel data broadcast with air-cache design to improve the server performance [48].

The flat algorithm proposed in [56] adopts a simple allocation algorithm, equally allocating the items to each channel. However, the expected response time for the most popular item is the same as that for a data item not frequently asked by clients. Hsu et al. proposed the VF^k algorithm, which skews the amount of data allocated to each channel [24]. The greedy algorithm in [69], which will be explained in detail later, can achieve a good performance when compared to the DP algorithm, and its time complexity is much better. Although all the algorithms have the same or similar objective function, they assign different weights to complexity and performance.

The greedy algorithm is an efficient algorithm which is often applied to achieve an optimal or near-optimal performance in scheduling problems. In [69], the authors assumed that each data item has a unit length and each channel has the same bandwidth. They also assumed that N_i items were cyclically broadcasted via channel C_i , so the expected delay w_j of receiving d_j on C_i is $\frac{N_i}{2}$. Given these assumptions, the multichannel average expected delay (MCAED) can be derived by Eq. (6), where the access probability of data item d_i is denoted as p_i , and K is the number of channels:

$$\text{MCAED} = \sum_{i=1}^K \sum_{d_j \in C_i} w_j p_j. \quad (6)$$

The greedy algorithm is to allocate data items to K channels in a way that minimizes the MCAED. A simple allocation technique which is easy to implement, called *flat design*, allocates the same number of items to each channel. In the flat design, $N_i = \frac{N}{K}$, $1 \leq i \leq K$, and it is obvious that $\text{MCAED} = \frac{N}{2K}$. In [48], the authors discussed *skewed design*, where items are placed on varying sized channels, depending on their access probabilities. Skewed design has been shown to be better than flat design at reducing the MCAED [2].

The principle of the greedy algorithm proposed in [69] is to find the split points among all the partitions that reduces the total cost of MCAED most significantly. This process is terminated when there are K partitions. In detail, the algorithm is shown in [Algorithm 3](#), where SCAED denotes the single-channel average expected delay for a channel containing data items from d_i to d_j , and it is computed by Eq. (7):

$$\text{SCAED} = C_{i,j} = \frac{j-i+1}{2} \sum_{q=i}^j p_q, (j \geq i, 1 \leq i, j \leq N). \quad (7)$$

The greedy algorithm runs very quickly, in which k denotes the number of partitions. It has the time complexity of $O((N + K) \log K)$, which makes it fast and suitable for practical use. In [69], the authors also discussed the case that the broadcast environment is changed during broadcasting by changing the number of channels as well as adding new data items. They described how a broadcast server generated by the greedy algorithm can easily adjust to changes of the number of channels. They also investigated how to adjust the broadcast with new data items added. Actually adding a channel requires only one more iteration in the greedy algorithm, and removing a channel just requires combines the last two partitions

Algorithm 2 Greedy [69]

Input: a set of N items sorted by probabilities;
Output: K partitions;
 $k \leftarrow 1$;
while $k < K$ **do**
 Let $point_k$ be the split point that most reduces SCAED for each partition P_k ;
 Find the point $point_j$ be the one in $point_1, \dots, point_k$ that most reduces MCAED;
 Split the partition P_j at $point_j$;
 $k \leftarrow k + 1$;
end while

together. It is also possible that items may change their access frequencies since some new data items may be inserted into the broadcast. The new data item is added to the correct position in the broadcast, and the access probabilities are recalculated so that it still holds $\sum p_i = 1$. The good thing is that the scheduler only needs to update split points where necessary. In other words many old split results may still be reused. Although this technique has to recompute the entire solution in the worst case, its efficiency is much better on average.

2.3 Scheduling Optimization for Dependent Data Broadcasting

It is much more likely that a client query requests a set of data items at one time instead of only asking for one data item from the broadcast server. Therefore, besides the independent data broadcast, dependent data scheduling is more useful for developing broadcasting programs in practice. Some works focus on this area, and they usually have the following common assumptions:

1. Data is broadcasted over single or multiple channels.
2. A query requests more than one data item at a time.
3. The queries with multiple data items have different access frequencies.
4. Data items are not necessarily of same sizes.

In [19], Chung et al. discussed the problem of scheduling dependent data to minimize the total access time of queries in wireless data broadcast systems. A query requests a subset of all data items with a query probability. They defined a measure called *query distance* (QD), which represents the coherence degree of the data set requested by a query, and then proposed the problem of *wireless data placement*. The authors also proved that the problem is NP-hard by transforming it from a *optimal linear arrangement* problem. A greedy scheduling algorithm was designed for this problem, in which the QD of queries with higher frequencies is guaranteed not to be changed when minimizing the QD for the queries with lower access probabilities.

In this chapter, the data placement problem for wireless broadcasting is to find a broadcast schedule such that the total access time (TAT) is minimized. The TAT can be computed by Eq. (8), where Q is the set of all the queries q_1, q_2, \dots, q_N , S is

a schedule of solution, and $\text{AT}^{\text{avg}}(q_i, S)$ is the average access time of the query q_i based on S :

$$\text{TAT}(S) = \sum_{q_i \in Q} \text{AT}^{\text{avg}}(q_i, S) \times freq(q_i). \quad (8)$$

The access time of a query for dependent data broadcast not only depends on the starting time of the query but also depends on the data items in the query set. It may spend the whole broadcast cycle for a single query. Thus, in [19], for a given query q_i , if $P(x)$ and $F(x)$ are the probability and the access time of the query q_i issued at time x , respectively, then the estimated average access time of q_i can be computed by the following equation, where k is the factor converting the sizes of data items into time lengths:

$$\begin{aligned} \int_0^B p(x) \cdot F(x) dx &= \frac{1}{B} \int_0^B F(x) dx \\ &= \frac{1}{B} \sum_{j=1}^n \int_0^{t_j} F(y) dy \\ &= \frac{1}{B} \sum_{j=1}^n \left[\int_0^{\text{size}(d_j)} F(y) dy + \int_{\text{size}(d_j)}^{t_j} F(y) dy \right] \\ &= \frac{1}{B} \sum_{j=1}^n \left[\int_0^{\text{size}(d_j)} kB dy + \int_{\text{size}(d_j)}^{t_j} k(B - y + \text{size}(d_j)) dy \right] \\ &= k \left(B - \frac{1}{2B} \sum_{j=1}^n k(t_j - \text{size}(d_j))^2 \right). \end{aligned}$$

The measurement of average access time defined above is too difficult to apply when developing data scheduling methods for wireless broadcasting. Therefore, Chung et al. defined a new metric, called query distance (QD), which indicates the largest distance between any two consecutive data items in a query. Let the total query distance denoted by $\text{TQD}(S)$, which is defined as $\sum_{q_i \in Q} QD(q_i, S) \times freq(q_i)$. They showed that given a set of data items D and a set of queries Q , the problem of finding a broadcast schedule S such that $\text{TQD}(S)$ is minimized is NP-complete. Thus, they proposed a data placement method based on a greedy algorithm for the dependent data scheduling problem.

Definition 1 Suppose a query q_i with data items d_1, d_2, \dots, d_n in a schedule S , and δ_j is the interval between d_j and d_{j+1} in the schedule S . Then, the QD of q_i in S is defined as $\max(\delta_k)$, $k = 1, 2, \dots, n$.

The algorithm schedules the broadcast program by inserting the set of data items requested by each query based on a greedy strategy. That is, the query data set with higher access frequencies is guaranteed not to be changed when minimizing QD of the queries with lower access probabilities. For this purpose five basic operations are defined in [19] as follows:

1. “Move data to the Right of a Fragment (MRF).”
2. “Move data to the Left of a Fragment (MLF).”
3. “Move data to the Right in a Segment (MRS).”
4. “Move data to the Left in a Segment (MLS).”
5. “Move Data Closest (MDC).”

Each of the five operations may be used for add new data items of a query into a given schedule. The operations MRF, MRS, MLF, and MLS move a set of data item to the most right end of the given schedule or most left end of the given schedule. They are used for minimizing the QD of the newly added data items of a query. The operation MDC is used for minimizing the QD when all data items for a given query are already included in the current schedule. The operation MDC moves the data items for a query as close as possible.

The proposed algorithm constructs a broadcast schedule by adding data items of each query in a greedy manner after sorting the queries based on their frequencies. Thereafter, many approaches have been proposed for minimizing the average expected delay for dependent data broadcast, and some of them further investigated the dependent data scheduling problem over multiple broadcasting channels. To generate a dependent data broadcast schedule in a multichannel environment, the server should partition the data items on multiple groups based on the collection of the data items for each query and their access frequencies. In order to retrieve a set of data items, a client may switch from one channel to another to download the data items until all the items requested in the query list are downloaded. The requested data items in a query can be retrieved in an arbitrary order. Moreover, if more than one queried item appear in different channels simultaneously, only one of them can be downloaded at one time. The other items should be retrieved when they are coming again. In a dependent data broadcasting program, one data item may also be requested by multiple queries. A brief summary of relative works is provided as follows.

Hurson et al., tried to minimize the overall access time, and proposed an allocation algorithm of dependent data based on some heuristics for multichannel broadcast systems in [32]. Lee et al. introduced an algorithm for queries with multiple data items in mobile environments in [44]. After that, Huang and Chen proposed a scheme based on a generic algorithm to handle the similar problem in [28]; and they explored the problem of broadcasting dependent data for ordered query over multiple broadcast channels in [27]. An ordered query means that the access sequence for the given query data set is preordered. Hung et al. studied the same problem and proposed a greedy algorithm called PBA, standing for *placement-based allocation*, in [31]. This algorithm will be explained in detail next.

Huang et al. first proposed a genetic algorithms called GA in [28] for the multichannel dependent data scheduling problem. The GA algorithm has been proved by experimental results to be a good approach; however, there are still

some weaknesses of the GA algorithm which may result in poor performance. For example, genetic algorithms will be affected by the initial setting. The initial setting is generated randomly, so it is not guaranteed to result in a good result constantly. To overcome such drawback, Hung et al. proposed a deterministic algorithm called PBA to schedule the dependent data items onto multiple channels. In PBA, the broadcast cycle length is assumed to be the same for all channels. Since each query contains multiple items in a dependent data broadcasting environment, the main policy of the algorithm PBA is to avoid the occurrence of the conflicting items in a broadcast cycle, which are the situations that multiple data items in a query are located in the same time slot on different channels. In order to explain the algorithm formally, two definitions in [31] need to be introduced.

Definition 2 Given a query q_i , t_{cycle_i} is the number of complete broadcast cycles needed for a mobile user to download the items in q_i . The refined query q'_i denotes the set of remaining items after the $cycle_i$ cycles.

Definition 3 Given a refined query q'_i , let $T_{Startup}(q'_i)$ be the time duration since the query q'_i starts until its first data item appears. Its retrieval time, denoted by $T_{Retr}(q'_i)$, is the time duration since the first data item appears until all the data items in q'_i are downloaded. The access time of a refined query q'_i , $T_{Access}(q'_i)$ is the sum of the start-up time and the retrieval time.

Assume all data items are of the same size. Let s and B represent the size of data item and the bandwidth of each channel, respectively. The access time of a query q_i , denoted by $T_{Access}(q_i)$, equals to $T_{Access}(q'_i) + cycle_i \cdot length_c$, where $length_c$ is the time length of a broadcast cycle. The average access time of the query set Q is denoted as $T_{Access}(Q)$.

It is obvious that a query will wait for multiple broadcast cycles to download all the data items required if there are many conflicting data items. The conflicting data items are in the same time placement on different broadcasting channels, so clients cannot download them in one cycle. Hence, the problem of generating a broadcast program equals to the problem of allocating the items of each query into L groups such that the number of conflicting items is minimized and each group contains items no more than K data items, where K is the number of available channels. The algorithm PBA in detail is outlined in [Algorithm 3](#).

To allocate the data items, the algorithm PBA first sorts the queries in descending order Q according to their access probabilities. A query with higher access probability has higher priority to be scheduled. After that, a broadcast program P , which contains L sets of data items, is constructed. Each data item set D_i contains the data items allocated in the i th time slot, but not the i th channel. Obviously if a query q_j is processed, the data items requested by q_j cannot be processed again. To prevent one data item be allocated more than once, they partition the data items in q_i into two disjoint subsets, q_i^+ and q_i^- , which store the already allocated and remaining items, respectively. The algorithm PBA terminates when all the items in D are allocated in P . A query with high access probability is processed earlier than the lower one. Therefore, it is a greedy approximation algorithm.

Algorithm 3 PBA [31]

Input: the query profile Q , the broadcast database D , and the number of channels K ;

Output: the broadcast program P ;

```

1: Sort elements in  $Q$  according to the access probability in descending order;
2: Let  $L \leftarrow \lceil \frac{\text{size}(D)}{L} \rceil$ ;
3: for each  $q_i \in Q$  do
4:   Unmark all item sets;
5:   Find  $q_i^+$  and  $q_i^-$  which store the scheduled and unscheduled items in  $q_i$ , respectively;
6:   Mark the item sets which contain  $K$  items and the item sets which contain the items in  $q_i^+$ ;
7:   if all item sets are marked then
8:      $curr \leftarrow \arg_i \min \text{size}(D_i)$  for all  $D_i$ ,  $\text{size}(D_i) < K$ ;
9:   else
10:     $curr \leftarrow \arg_i \min \text{size}(D_i)$  for all unmarked  $D_i$ ;
11:   end if
12:   for each item  $d_j \in q_i^-$  do
13:     Insert  $d_j$  into  $D_{curr}$ ;
14:     mark  $D_{curr}$ ;
15:   if all item sets are marked then
16:     Select next from  $D_i$ ,  $\text{size}(D_i) < K$  such that  $DIST(curr, next)$  is minimized;
17:   else
18:     Select  $next$  from the unmarked item sets such that  $DIST(curr, next)$  is minimized
19:   end if
20:   Set  $curr = next$ ;
21: end for
22: end for

```

3 Optimizations for On-Demand Data Broadcasting

In pure push-based data broadcast systems, data access patterns are usually assumed to be static in a certain period. Data are organized into broadcast programs according to their access patterns. The programs will then be broadcasted in cycles and updated periodically. In such systems, individual clients are passive listeners, which means they have no direct influence over the content of a broadcast program. This network structure brings up several issues. Since there is no feedback of clients in the form of requests, it is very difficult for pure push-based systems to get accurate data access patterns. Moreover, in circumstances where the client request pattern changes frequently, the static access pattern assumption will not be true anymore. This may result in longer system response time. In the worst case, the program may not be able to satisfy client requests anymore.

“Pull-based” data broadcast is a different data broadcast method which does not have the aforementioned drawbacks of push-based data broadcast method. It is also referred to as “on-demand” data broadcast [13, 63]. A typical on-demand data broadcast system structure is shown in Fig. 3.

As in Fig. 3, the network connection in an on-demand data broadcast system consists of two types of links: uplink and downlink. Clients send requests through uplink to the server. These requests are queued up at the server upon arrival.

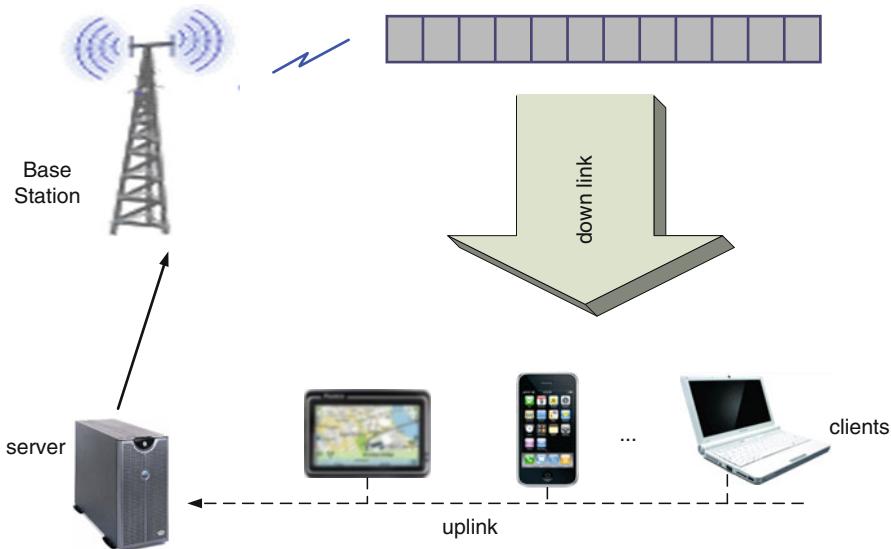


Fig. 3 A typical on-demand data broadcast system structure

The broadcast channel(s) serves as the downlink to broadcast data to clients. The server continuously puts data items onto the broadcast channel(s) according to the queued requests and removes corresponding requests from the queue. Clients will listen to the channel(s) and download their requested data when they arrive. This data dissemination process shows the influence of clients' requests over the choice of broadcasted data. On-demand data broadcast can dynamically adjust broadcast programs according to the client requests and thus is suitable for situations when data access patterns change frequently.

A good online scheduling algorithm to choose data items for broadcast is critical for the good performance of an on-demand data broadcast system. Similar as push-based broadcasting, the main objective of scheduling on-demand data broadcast is also to reduce the response time. Different from push-based broadcasting, the *response time* for a request in an on-demand broadcast system is the time interval between the time the client sends the request through uplink and the time the client downloads the data from broadcast channels.

There have been a lot of literature on scheduling on-demand data broadcast. This section will focus on different on-demand data scheduling algorithms for different communication environments. Section 3.1 introduces several basic scheduling algorithms. Section 3.2 presents the $R \times W$ algorithm which can handle large-scale databases. Section 3.3 gives a scheduling algorithm for time-critical requests with deadlines. Section 3.4 discusses how to schedule heterogeneous data with preempting technique. All on-demand scheduling algorithms discussed in Sect. 3 are based on single broadcast channel.

3.1 Basic Online Scheduling Algorithms for On-Demand Data Broadcast

Earliest studies on online scheduling for on-demand broadcast can be traced back to the late 1980s. In [21, 62], the following algorithms were proposed:

- *First come, first served (FCFS)*: always broadcast the data item of the request that arrives at the earliest time. Ignore a new request if the same request already exists in the queue.
- *Most requests first (MRF)*: always broadcast the data item of the maximum number of requests in the queue.
- *Longest wait first (LWF)*: always broadcast the data item that the sum of waiting times of all its requests in the queue is the longest.

Intuitively, FCFS emphasizes fairness and gives preference to “cold” items, while MRF considers data popularity and prefers “hot” items. LWF tries to balance the waiting time of all requests and also prevents possible “starving” requests. Simulation results in the original papers show that FCFS results in poor average response time when the data access distribution is nonuniform, and LWF performs better than FCFS and MRF in general. A more theoretical analysis of these algorithms were presented in [47], which gives performance ratios of FCFS and MRF.

Assume there are m data items in the database for broadcasting and n requests. Each request contains only one data item. All items are of same length. An *instance* is a data item on the broadcast channel. For any instance, let $\sum_{i=1}^n w_i(\bullet)$ represent the total response time of the solution given by \bullet scheduling algorithm. OPT represents the optimal solution. [47] proved the following theorems:

Theorem 2 $\sum_{i=1}^n w_i(\text{FCFS}) \leq m \sum_{i=1}^n w_i(\text{OPT})$.

Theorem 3 *There exists at least one instance that satisfies $\sum_{i=1}^n w_i(\text{FCFS}) = m \sum_{i=1}^n w_i(\text{OPT})$.*

Theorem 4 $\sum_{i=1}^n w_i(\text{MRF}) \leq m \sum_{i=1}^n w_i(\text{OPT})$.

Theorem 5 *There exists at least one instance that satisfies $\sum_{i=1}^n w_i(\text{MRF}) = m \sum_{i=1}^n w_i(\text{OPT})$.*

3.2 $R \times W$ Scheduling for Large-Scale On-Demand Data Broadcast

When the database size is getting large, simply considering average or total response time as the performance criteria for an on-demand data broadcast system is obviously not enough. In [4, 5], Aksoy and Franklin categorized the performance criteria for on-demand scheduling algorithms as follows:

1. Responsiveness
 - (a) *Average waiting time*: the average amount of time from the moment that a client request arrives at the server to the moment the requested item is broadcast.
 - (b) *Worst case waiting time*: the maximum amount of time that any client request will have to wait in the service queue to be satisfied.
2. Scheduling overhead
 - (a) *Request processing*: the overhead of the server deciding whether to add an entry to the service queue for the requested item and/or updating the queue
 - (b) *Scheduling decisions*: the overhead to choose an item to broadcast at the beginning of each broadcast time slot.
3. Robustness: whether the scheduling algorithm can adjust to the changes of the workload or environment.

Based on the observations of FCFS, MRF, and LWF's performances with respect to the above criteria, Aksoy and Franklin developed a scheduling algorithm named $R \times W$ in [4, 5]. The intuition is to combine the benefits of FCFS and MRF in a way that ensures scalability while preserving low overhead. They first proposed an exhaustive algorithm and then improved it with a pruning technique to reduce the search space. To further reduce the scheduling overhead for large-scale databases, they developed an approximate, parameterized version of $R \times W$ algorithm at the expense of possible suboptimal broadcast decisions.

3.2.1 The Exhaustive $R \times W$ Algorithm

The idea of $R \times W$ algorithm is to broadcast an item either because it is popular or because it has at least one request that has been waiting for a long time. Instead of a straightforward service queue, it maintains a service table that has an entry for each data item with outstanding requests. The structure of the service table is shown in Fig. 4.

- Each entry in the service table consists of the following elements:
- PID : identifier of the data item in the database
 - $IstArrv$: the arrival time of the earliest outstanding request for this item
 - $Prev\ in\ W\text{-List}$: a pointer to the previous entry in the W-List
 - $Next\ in\ W\text{-List}$: a pointer to the next entry in the W-List
 - R : the total number of outstanding requests for this item
 - $Prev\ in\ R\text{-List}$: a pointer to the previous entry in the R-List
 - $Next\ in\ R\text{-List}$: a pointer to the next entry in the R-List

The service table is hashed on PID . W represents the longest waiting time of all requests for a particular data item. W can be computed as $W = \text{clock} - IstArrv$, where clock is the current time. All time used in the rest of Sect. 3 is represented in broadcast ticks, that is, the number of items broadcasted so far. The algorithm uses two sorted lists threaded through the service table: W-List in descending order of W and R-List in descending order of R . An R-Index is introduced to reduce the maintenance overhead of the R-List. For each distinct R value, there is a pointer to the first entry in the R-List of the same R value.

The algorithm can be divided into two steps: request process and broadcast decision. The procedures of these two steps are described in Algorithms 4 and 5.

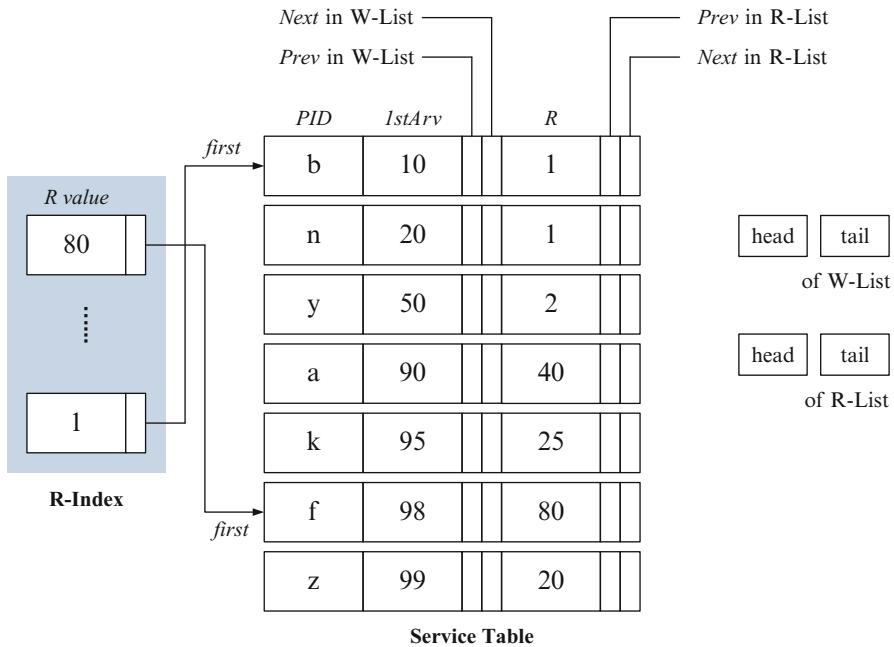


Fig. 4 The service table structure of $R \times W$ algorithm

Algorithm 4 Request process of exhaustive $R \times W$

Input: a request of item with identifier *PID*;

Output: updated service table;

- 1: perform a hash lookup on *PID* in the service table.
 - 2: **if** an entry exists **then**
 - 3: increment the *R* value of the entry.
 - 4: **else**
 - 5: create a new entry with *R* = 1 and *IstArrv* = current time.
 - 6: **end if**
-

Algorithm 5 Broadcast decision of exhaustive $R \times W$

Input: service table;

Output: updated service table;

- 1: find the entry with the maximum $R \times W$ value in the service table. e
 - 2: broadcast the corresponding data item.
 - 3: remove this entry from the service table.
-

A theoretical analysis of the $R \times W$ algorithm in [5] concluded that for an arbitrary data item b , the expected waiting time is

$$W_b = \frac{W_0 + \sum_{i=b+1}^{N-1} \rho_i \sqrt{\frac{p_i}{p_b}} + \sum_{i=0}^{b-1} \rho_i W_i}{1 - \sum_{i=0}^{b-1} \rho_i \left[1 - \sqrt{\frac{p_b}{p_i}} \right]}. \quad (9)$$

In Eq. (9), W_0 is the mandatory wait for the completion of an already initiated page broadcast; ρ_i is the utilization of the bandwidth to broadcast item i ; and p_i is the access probability of item i .

3.2.2 $R \times W$ Algorithm with Pruning

The scheduling overhead of the exhaustive $R \times W$ algorithm is $O(N)$, where N is the total number of data items to broadcast. As N increases, the search space of [Algorithm 5](#) is getting larger, which consequently increases the overhead. Aksoy and Franklin proposed a pruning technique for the $R \times W$ algorithm in [5].

The pruning technique uses the “sorted” feature of the R-List and W-List to reduce the search space when looking for the maximum $R \times W$ value. Suppose the examination starts at the top entry of the R-List. MAX is used to record the current maximum $R \times W$ value. Since the R-List is in descending order, the next entry in the list R' is smaller than R . Naturally, for any unexamined entries with $R \times W$ value greater than MAX, their W value must satisfy

$$W > \frac{\text{MAX}}{R'}.$$

Therefore, the search range of the W-List can be truncated by

$$\text{limit(1stArv)} = \text{clock} - \frac{\text{MAX}}{R'}.$$

That is, no entry with 1stArv greater than limit(1stArv) in the W-List will have $R \times W$ larger than MAX. Similarly, if the examination starts at the top entry of the W-List, the search range of the R-List can also be restricted by

$$\text{limit}(R) = \frac{\text{MAX}}{W'},$$

where W' is the next entry in the W-List.

Based on this observation, the scheduling algorithm starts examining the top entry of the R-List and keeps on alternating between the two lists. It can be described in [Algorithm 6](#).

This pruning technique is effective in reducing the search space. Experiment results in [5] showed that there is a 73 % reduction in entries examined to find the maximum $R \times W$ for their Zipf-based workload.

Algorithm 6 Broadcast decision of $R \times W$ with pruning

Input: service table, R-List and W-List;

Output: updated service table;

- 1: examine the top entry of the R-List; set MAX and $limit(1stArv)$.
 - 2: **while** neither $limit(1stArv)$ nor $limit(R)$ is reached **do**
 - 3: examine the next entry of the previously examined entry in the other list.
 - 4: Set MAX = $R \times W$ if $R \times W > MAX$.
 - 5: update $limit(1stArv)$ and $limit(R)$.
 - 6: **end while**
 - 7: broadcast the corresponding data item.
 - 8: remove this entry from the service table.
-

3.2.3 The Approximate $R \times W$ Algorithm

Although the pruning technique discussed in Sect. 3.2.2 can help reducing the search space, the reduction may not be enough to confine the scheduling overhead within a reasonable range when the database scale is really large. Therefore, an approximating, parameterized variation of the $R \times W$ algorithm was proposed in [5] to further reduce the search space at the expense of possible suboptimal scheduling decisions.

In the approximating algorithm, instead of finding the maximum $R \times W$ value, it chooses the first item encountered with $R \times W \geq \alpha \times threshold$. α is the parameter of the algorithm, which is greater than 0. $threshold$ is the self-adaptive average, whose value can be computed as

$$threshold(t+1) = \frac{threshold(t) + last\ R \times W(t)}{2}$$

$threshold(0)$ is set to 1. $last\ R \times W(t)$ is the $R \times W$ value of the item broadcasted at time t .

α is parameter of the algorithm that can be adjusted. It determines the trade-off between responsiveness and scheduling overhead. The smaller α is the less entries need to be examined, but the more likely the scheduling decision is not optimal. In such cases, the scheduling overhead is reduced but the responsiveness may be jeopardized.

3.3 Scheduling Time-Critical On-Demand Data Broadcast

The work [4,5] by Aksoy and Franklin and many other literature on on-demand data broadcasting use the responsiveness performance criteria as discussed in Sect. 3.2, which aim at minimizing the average or worst waiting time. This implies an assumption that once a client generates and sends a request, it will wait until the request is satisfied. However, this assumption is not always true. In some circumstances, clients may lose patience and drop the request if they have waited

for too long. In order to take the time constraint into consideration for time-critical requests, the concept of deadline is introduced [39, 40, 66].

In a time-critical on-demand data broadcast system, each request has an additional parameter *deadline*. An outstanding request will be dropped after its deadline is reached. Motivated by the EDF (earliest deadline first) scheduling algorithm in unicast and the MRF algorithm in broadcast, Xu et al. proposed a scheduling algorithm for time-critical on-demand data broadcast in [66]. The objective is to satisfy as many requests as possible. A new performance criterion is defined to measure this objective:

$$\text{request drop rate} = \frac{\text{number of requests missing their deadlines}}{\text{total number of requests}}.$$

The idea of the algorithm can be summarized into two rules:

1. If two data items have the same number of outstanding requests, broadcast the one with a closer deadline.
2. If two data items have the same deadline, broadcast the one with more outstanding requests.

The first rule gives priority to *urgent* requests, while the second rule gives preference to requests that are more *productive*. Both rules are for the sake of reducing the request drop rates.

The algorithm proposed by Xu et al. is named as SIN- α (slack time inverse number of outstanding requests). For each data item with at least one outstanding request, there is a $\sin.\alpha$ value attached to it:

$$\sin.\alpha = \frac{\text{slack}}{\text{num}^\alpha} = \frac{\text{1stDeadline} - \text{clock}}{\text{num}^\alpha},$$

where slack represents the remaining time before the earliest deadline of all requests for this item, and num is the total number of requests for this item. $\alpha \geq 0$ is a parameter determining the relative importance of productivity over urgency in the broadcast decision. The larger α is the more influential the number of outstanding requests in computing $\sin.\alpha$.

An exhaustive version of SIN- α is to compute the $\sin.\alpha$ value of all items in the service queue at each broadcast tick and broadcast the item with minimum $\sin.\alpha$ value. The complexity of this implementation is $O(N)$, where N is the total number of data items with outstanding requests.

Inspired by the pruning technique discussed in Sect. 3.2.2, Xu et al. applied a similar pruning method during the searching for the minimum $\sin.\alpha$. Two double-linked sorted lists are maintained: S-List and N-List. Each data item has one entry in both lists. S-List maintains each item's associated earliest request deadline in ascending order, while N-List contains the number of outstanding requests for each item in descending order. Similar as Algorithm 6, the scheduling algorithm starts examining the top entry of the N-List and keeps on alternating between the two lists.

Two limits are also maintained to reduce the search space for minimum $\sin.\alpha$: $limit(N)$ for the N-List and $limit(1stDeadline)$ for the S-List.

During the searching, let MIN denote the current minimum $\sin.\alpha$ value. If an entry in the S-List has just been checked, then any unexamined item with $\sin.\alpha < \text{MIN}$ should have its num value exceeding

$$limit(N) = \left(\frac{NextS}{\text{MIN}} \right)^{\frac{1}{\alpha}},$$

where $NextS$ is the slack of the next entry in the S-List. On the other hand, if an entry in the N-List has just been examined, then the 1stDeadline of any unexamined item should be less than

$$limit(1stDeadline) = (NextN)^\alpha \cdot \text{MIN} + \text{clock},$$

where $NextN$ is the num of the next entry in the N-List.

In [66], Xu et al. also gave a thorough analysis of the theoretical bound of request drop rate when the request arrival rate is approaching infinity. The request drop rate for item i arriving in interval $[0, \tau]$ is

$$\eta(\tau) = \frac{1}{\tau} \int_0^\tau F(\tau - t)dt, \quad (10)$$

where $F(t)$ is the probability that a relative deadline is shorter than t . Relative deadline refers to the time interval between current time and the deadline.

Xu et al. proved that the request drop rate of an item is minimized when instances of the item are broadcasted with equal intervals. Therefore, assume the spacing of any item i is uniform, denoted as s_i . The total request drop rate can be computed as

$$\eta(s_1, \dots, s_N) = \sum_{i=1}^N \frac{\lambda_i}{\lambda} \eta(s_i) = \sum_{i=1}^N \frac{p_i}{s_i} \frac{1}{\tau} \int_0^\tau F(\tau - t)dt, \quad (11)$$

where λ_i is the request rate of item i and λ is the total request rate. Their ratio p_i is exactly the access probability of item i .

If a broadcast tick is l , it means $\frac{l}{s_i}$ of the bandwidth is used to broadcast item i , which means

$$\sum_{i=1}^N \frac{l}{s_i} = 1. \quad (12)$$

Xu et al. proposed the following theorems about the lower bound of request drop rate as computed in Eq. (11) under three different relative deadline distributions.

Theorem 6 *Under exponential distribution of relative deadlines with a mean value of M , the total request drop rate is minimized when (s_1, \dots, s_N) satisfies*

$$p_i \left(1 - \frac{s_i}{M} e^{-\frac{s_i}{M}} - e^{-\frac{s_i}{M}} \right) = K, (i = 1, \dots, N).$$

Theorem 7 Under uniform distribution of relative deadlines between 0 and L , the minimum total request drop rate is given by

$$\begin{aligned} & \min_{\{I, f | 0 \leq I \leq N, f \geq \frac{IL}{L}, f > 1 - \frac{L}{L}\}} \left\{ \frac{1}{2L\{\frac{f}{l} - \frac{I-m}{L}\}} \left(\sum_{i=1}^m \sqrt{p_i} \right)^2 + \sum_{i=m+1}^I \frac{p_i}{2} \right. \\ & \quad \left. + p_{I+1} \left(1 - \frac{L}{2} \cdot \frac{1-f}{l} \right) + \sum_{i=I+2}^N p_i \right\}, \end{aligned}$$

where

$$m = \max \left\{ x \mid \frac{\sum_{j=1}^x \sqrt{p_j}}{(\frac{f}{l} - \frac{I-x}{L}) \sqrt{p_x}} \leq L \right\}.$$

Theorem 8 Under fixed relative deadlines of C , the total request drop rate is minimized when (s_1, \dots, s_N) satisfies

$$s = \begin{cases} C & i \leq \lfloor y \rfloor, \\ \frac{C}{y - \lfloor y \rfloor} & i = \lfloor y \rfloor + 1, \\ \infty & i > \lfloor y \rfloor + 1, \end{cases}$$

where $y = \frac{C}{l}$. The minimum request drop rate is

$$\eta(s_1^*, \dots, s_N^*) = p_{\lfloor y \rfloor + 1}(1 - y + \lfloor y \rfloor) + \sum_{i=\lfloor y \rfloor + 2}^N p_i.$$

3.4 Scheduling Preempting On-Demand Broadcast for Heterogeneous Data

All scheduling algorithms discussed in Sect. 3 so far have one common assumption: the size of all data items are equal. In many real applications of on-demand broadcast, data of requests are heterogeneous, which means that data items can be of different sizes. In [3], Acharya and Muthukrishnan first brought up the problem of scheduling on-demand broadcast for heterogeneous data and proposed the usage of preemption during scheduling.

Preemption is usually used on heterogeneous workload in order to prevent backlog of pending requests when servicing a long job. In a broadcast environment, preemption means the server can abort the current broadcasting item and broadcast items for more valuable requests. Two models allowing preemption have been

proposed: the *restart* model in [40] and the *resume* model in [39]. However, the treatments of the aborted broadcast in these two models are different:

- Restart: the server can restart a preempted broadcast from the beginning
- Resume: the server can resume a preempted broadcast from the point of preemption

Ting [55] designed a scheduling algorithm named *balance* for on-demand broadcast utilizing preemption, under the heterogeneous data assumption. The design is based on the restart model. Given a set of data items, assume the length of the shortest item is 1 and the length of the longest item is Δ . For any item P , $l(P)$ is the time needed to completely broadcast P . A *broadcast*, *schedule*, and *request* are defined respectively as following:

Definition 4 A *broadcast* $B = (P, t)$ is to broadcast item P at time t .

Definition 5 A *schedule* is a sequence of broadcasts $S = \langle (P_1, t_1), \dots, (P_m, t_m) \rangle$, where $t_1 < \dots < t_m$.

For any broadcast (P_i, t_i) , it is a *complete* broadcast if $t_i + l(P_i) < t_{i+1}$; otherwise, it is preempted by (P_{i+1}, t_{i+1}) , and (P_{i+1}, t_{i+1}) is a *preempting* broadcast.

Definition 6 A *request* r is a four-tuple (P, a, d, v) , where P is the requested item, a is its arrival time, d is its deadline, and v is its value.

Given a sequence of requests σ , the scheduling problem for time-critic on-demand data broadcast is to find a schedule S with maximum profit. Let $R_{S,\sigma}(P, t)$ denote the set of outstanding requests of P at time t . The profit $\rho(S, \sigma)$ of a schedule S can be computed as

$$\rho(S, \sigma) = \sum_{(P,t) \in S, (P,t) \text{ is complete}} v_{S,\sigma}(P, t), \quad (13)$$

where $v_{S,\sigma}(P, t)$ is the value of P at time t with respect to (S, σ) , which equals to $\sum_{r \in R_{S,\sigma}(P,t)} r.v$.

The motivation of *balance* algorithm is to reach a good balance between good-profit broadcasts and short-length broadcasts. Good-profit broadcasts are those whose completion will increase the profit of the schedule, while short-length broadcasts are those whose completion will reduce the total time needed for the schedule. All complete broadcasts are considered to be good-profit broadcasts.

Except for the preemption, *balance* has a “greedy”-like strategy. The scheduling process can be illustrated in [Algorithm 7](#).

The preemption in *balance* follows a relatively complicated set of rules in order to achieve better performance ratio. The procedure is described in [Algorithm 8](#).

In his paper [55], Ting also derived an upper bound and lower bound on the performance ratio of *balance*. Suppose S is the schedule generated by *balance*:

Algorithm 7 Balance

```

1: while there are still unsatisfied requests do
2:   find the item of highest total value and broadcast it;
3:   check for possible preempting;
4:   remove the requests of the completely broadcasted item from the service queue.
5: end while

```

Algorithm 8 Preemption

Input: $S = \langle B_1, \dots, B_{gp}, \dots, B_{last} \rangle$, $t \in (t_{gp}, t_{gp} + l(P_{gp}))$, $\lambda = \frac{3\Delta}{\log \Delta}$.

```

1: if  $\exists P$ , s.t.  $v_{S,\sigma}(P, t) > \sqrt{\Delta} v_{S,\sigma}(P_{gp}, t_{gp})$  then
2:   preempt  $B_{last}$  and broadcast  $P$ .
3: else
4:    $\Gamma = \{Q | t + l(Q) < t_{last} + l(P_{last}) \wedge v_{S,\sigma}(Q, t) > \left( \frac{2^{\lfloor \frac{t_c - t_{gp}}{\lambda} \rfloor}}{\Delta^{1/3}} \right) v_{S,\sigma}(P_{gp}, t_{gp}) \}$ .
5:   if  $\Gamma \neq \emptyset$  then
6:      $Q' = \operatorname{argmin}_{Q \in \Gamma} \{l(Q)\}$ .
7:     preempt  $B_{last}$  and broadcast  $Q'$ .
8:   end if
9: end if

```

Theorem 9 $\rho(\text{OPT}, \sigma) \leq \left(\frac{6\Delta}{\log \Delta} + O(\Delta^{5/6}) \right) \rho(S, \sigma)$.

Theorem 10 $\rho(\text{OPT}, \sigma) \geq \left(\frac{\Delta}{2 \ln \Delta} - 1 \right) \rho(S, \sigma)$.

[Theorem 9](#) shows that *balance* breaks the Δ -barrier for all previous schedulers without preemption. This demonstrates the effectiveness of introducing preemption into scheduling heterogeneous data. The combination of [Theorems 9](#) and [10](#) also proves that *balance* is optimal within a constant factor.

4 Data Retrieval Optimization in Data Broadcast Systems

Scheduling data onto broadcast channels is a key optimization research issue on the server side in data broadcast. Similarly, scheduling the data retrieval from broadcast channels for a mobile client is also an important optimization problem. When there is only one broadcast channel, or the client request contains single data item, the data retrieval scheduling problem is usually trivial. With the fast development of mobile communication technology, multichannel data broadcast is gaining more popularity because the utilization of multiple channels can dramatically enlarge the bandwidth for the broadcast downlink. In a multichannel broadcast environment, how to schedule the downloading of multiple data items is the focus of this section.

In an indexed multichannel data broadcast system, the data retrieval for a client is usually composed of three steps:

1. Initial probing: the client tunes in a broadcast channel
2. Searching: the client follows the guide of embedded index information on broadcast channels and gets the locations of requested data.
3. Downloading: the client tunes in the channel and downloads all requested data.

All literature on data retrieval scheduling so far consider only the downloading phase, that is, assume locations and arrival time of all requested data are known. Same as data broadcast scheduling, response time is also a primary concern when scheduling the data retrieval process for a client. Energy conservation is another performance criterion for data retrieval scheduling because of the limit battery lives of mobile clients. Therefore, the data retrieval scheduling problem can be summarized as follows:

Definition 7 (Data retrieval scheduling sroblem) suppose there is a request for multiple data items. Given the channel locations and arrival time of all requested data items, how to scheduling the downloading process so that

- The overall response time and/or
 - The energy consumption
- of the client can be minimized.

The reason why a good retrieval schedule can reduce the response time is straightforward, while why it can reduce energy consumption is not so obvious. Hurson et al. [33] pointed out that switching among broadcast channels during retrieving data also consumes energy: in general, on switching costs about 10 % of one unit time active mode power consumption of a mobile client. Therefore, the energy conservation objective can be converted to minimizing the total number of switchings during the data downloading.

One important issue that needs to be considered during data retrieval scheduling is *conflicts*. When a client just finished downloading one item at time t on channel A , it is impossible to start downloading another item at time t on channel $B \neq A$. This is because switching from one channel to another also takes time. Although it is usually faster than one broadcast tick, it is not neglectable. Hence, when a client just downloaded one item at time t on channel A , it can only start downloading at time $t + 1$ from any channel other than A . An illustration of conflicts can be found in Fig. 5.

Hurson et al. [33] first proposed the problem of downloading multiple data items from multiple broadcast channels with single retrieving process. They also explained the concepts of conflict during retrieval. In their work, they treated the two objectives, minimizing access time and minimizing number of switchings, separately, and proposed a heuristic algorithm for each of them.

With the development of antenna technique, adding MIMO antenna in mobile devices is already feasible. As a consequence, it is possible for clients to use multiple processes to retrieve data in parallel. Shi et al. discussed the retrieval scheduling problem for such a communication environment in [50], which will be introduced in the rest of this section.

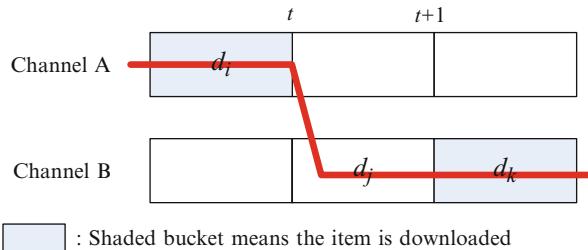


Fig. 5 An illustration of conflicts

Instead of treating the two objectives separately, Shi et al. defined a new cost function that combines the two objectives together:

Definition 8 The *cost* of a parallel data retrieving schedule is evaluated as $c = \alpha \cdot AT + \beta \cdot S$, where AT is the access time of the data retrieving process from the starting time (initial probing and searching processes are not included); S is the number of switches among channels during the retrieving process.

α and β are adjustable parameters which can reflect the user's preference. If the priority for the user is to get the requested data faster, α should be increased; if energy conservation is more important, β should be increased.

The retrieval scheduling only considers the downloading phase, so only channels with requested data on them are of interest, defined as *requested channels*. Suppose the number of requested channels are K , and the number of available retrieving processes for a client is M . When $K \leq M$, it is easy to prove that the optimal schedule is to assign one process to download all requested data on each requested channel. When $K > M$, the problem becomes nontrivial.

Shi et al. designed two greedy heuristic scheduling algorithm in the case of $K > M$. The first algorithm is *least switch*. The idea is to guarantee minimum number of switching during downloading. [Algorithm 9](#) describes the scheduling process. Note that in *least switch*, every time a process finishes downloading one channel, the scheduler will assign a most costly channel to it to continue downloading. This is because the overall access time is determined by the process which takes longest time to finish its task, so balancing the workload among processes is important in order to reduce the overall access time.

The other algorithm is called *best first*. It is a greedy algorithm in that the scheduler always assigns to a process the item which costs least for a process to download. [Algorithm 10](#) demonstrates the scheduling process.

Best first can be divided into three phases: initial assignment, best-first assignment, and channel assignment. As the request list is shrinking during the downloading process, the number of requested channels is also getting smaller. When $K = M$, it is optimal to assign one channel to one process, so the best-first assignment will stop.

Algorithm 9 Least switch

```

1: Assign  $M$  requested channels to  $M$  retrieving processes;  $K = K - M$ ;
2: while  $K > 0$  do
3:   whenever a process finishes downloading, assign to it the most costly channel to download;
4:    $K --$ ;
5: end while

```

Algorithm 10 Best-first

```

1: assign  $M$  requested items with least cost to download to the  $M$  retrieving processes;
2: remove these item from the request list and update  $K$ ;
3: while  $K > M$  do
4:   whenever a process finishes downloading, assign to it an item which costs least to download;
5:   remove this item from the request list and update  $K$ ;
6: end while
7: assign remaining  $M$  requested channels to the  $M$  processes.

```

5 Indexing Techniques for Data Broadcast Optimization

Utilizing indices into the broadcasting program is another way of optimizing data broadcast problem by means of reducing the tuning time so as to improve the energy efficiency of a wireless data [16] broadcast system. An index is a specific data structure with pointers storing the location information that directs to the data items. Due to the nature of data broadcasting, such location information of an index usually indicates the time offset of the target data. Once a client gets this time offset, it could be aware of the total waiting time before the target data item is broadcasted on the broadcasting channel. Then, the client could turn into doze mode (or sleep) to save some energy and then tune back to the broadcasting channel before the data item appears.

Indexing techniques could dramatically reduce the average tuning time for clients within a wireless data broadcasting system. However, if indices are inserted between data items, then the total length of a broadcasting program will increase, which may cause a longer access time. Therefore, when discussing about an indexing scheme, researchers will always consider the balance or trade-off between tuning time and access latency [37]. Since different indexing schemes have different searching efficiencies, how to optimize the tuning time while maintaining acceptable access latency is a critical issue in wireless data broadcasting systems.

5.1 Indexing in Single-Channel Broadcasting Environment

There are a number of works trying to apply traditional disk-based indexing approaches onto air indexing by converting location information or physical address into time offset. A lot of existing air indexing schemes, such as distributed index,

hash-based scheme, and exponential index, were designed for flat broadcast on single-channel broadcasting environment, in which all data items are broadcasted under the same frequency [34, 35, 65, 67]. Imielinski et al. [34] were the first to discuss air indexing schemes, and they proposed flexible index. Furthermore, they customized B⁺-tree index in [35] and proposed (1, m) index as well as distributed index. Distributed index was extended by many other researchers to fit different system requirements. For example, Hsu et al. [24] modified distributed index to deal with data under nonuniform access frequencies. Moreover, Lee et al. [45] discussed a signature-based approach for information filtering in wireless data broadcasting. Later, Hu et al. [25] designed a hybrid indexing scheme combining both distributed index and signature-based index. Xu et al. [67] gave an idea of exponential index that shares links in different search trees and allows clients to start searching at any index node, which is error resilient due to its distributed structure.

More indexing schemes are further developed to deal with skewed access patterns on single-channel broadcasting environment. For instance, Huffman tree is a skewed index tree which takes into consideration the access probability of each data item, where the more popular data is stored at a higher level of the tree which has a shorter path from the root node. Thus, it minimizes the average tuning time in the search process. However, almost all existing works discussed Huffman tree on multichannel environment [17, 52]. It is hard to compare the performance of Huffman-tree index with other single-channel indexing methods, because when it comes to multichannel data broadcast, how to allocate index and data will produce heavy impact on the performance of the indexing techniques. A certain allocation method could be helpful to specific indexing structure, but at the same time it might reduce the efficiency of another indexing scheme. Recently, Zhong et al. [74] proposed an alphabetic Huffman-tree distributed index in the single-channel data broadcast environment, which minimizes both average access time and average tuning time. The experiments show that it outperforms the B⁺-tree distributed indexing approach. Besides Huffman-tree index, Imielinski et al. [34] presented hash-based scheme. Another work by Yao et al. [68] proposed MHash scheme to facilitate skewed data access probabilities and reduce access latency.

5.1.1 Flexible Index

As Imielinski et al. [34] proposed, flexible index simply divides the data file into p segments and provides indices to help the clients “navigate,” so as to reduce the tuning time. The parameter p makes this scheme *flexible* since it can be adjusted to allow users choosing between either good tuning time or good access time.

In flexible index, data records are sorted in ascending order. All the data buckets are divided into p groups called data segments, numbered from 1 to p , the first bucket of which stores the *control index* and possibly the data records if space allows. Control index is a binary index used for locating the data bucket which contains the given key.

Figure 6 shows an example of flexible indexing with 66 data buckets. Let $p = 9$, with the length of eight buckets for all data segments except the last one. The first bucket of data segment contains the index. Three examples of such buckets are

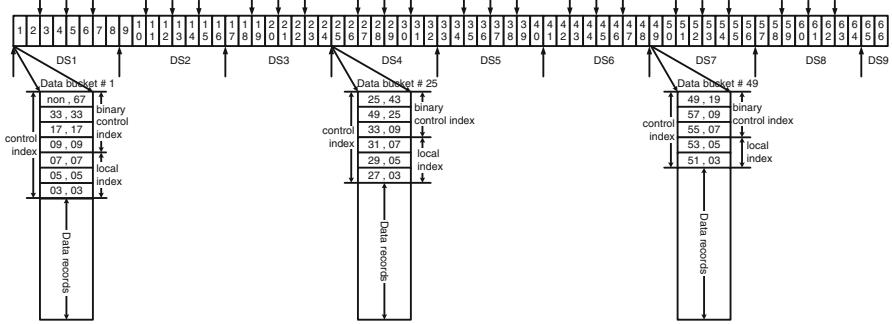


Fig. 6 An example of flexible index

shown in [Fig. 6](#). Each index entry is a pair of the key value and the *offset* (*offset* means the relative distance from this current bucket to that pointed bucket), which tells the clients when to tune in again to find the requested data.

For instance, as shown in [Fig. 6](#), the index of the data bucket #25 indicates that for all key values smaller than 25, the client should tune to the bucket #1 of the next *bcast* that is 43 buckets away (in this case the client missed the key and has to wait for the next broadcast). If the key is greater than 49, the client has to tune in again to bucket #49 (which the offset # = 25 points to). That data bucket will provide another index to help the client continue searching. Not all data buckets contain the index, and that the index information is distributed among different data buckets.

The control index is divided into two parts: the *binary control index* and the *local index*. The *binary control index* has $\lceil \log_2 i \rceil$ tuples, where i is the number of data segments in front of (including) this one. The k th tuple contains the key of the first data item of the $\lfloor \log_2 i / 2^{k-1} \rfloor + 1$ th data segment and an offset to the first data bucket of that data segment. The *local index* has m tuples. Parameter m depends on how many tuples a bucket can hold, the access time desired, etc. The local index further partitions a data segment into $m + 1$ data subsegments D_1, D_2, \dots, D_{m+1} , where the k th tuple contains the key of the first data item of D_{m+1-k} and an offset to the first data bucket of D_{m+1-k} .

Following is the access protocol of flexible indexing scheme [34] for a record with key K .

1. Initial probe, get the offset to the next data segment, and go to doze mode.
2. Tune in again to the designated data segment. (1) If the search key K is less than the first tuple's first field in the binary control index, then go to doze mode for the offset time (as the second field indicates) and repeat step 2; else (2) search the rest entries of the binary control index to check if K is greater than or equal to each tuple's first field. If so, follow the pointer of the first such tuple and repeat step 2; else (3) search the local index and check if K is greater than or equal to each tuple's first field. If so, follow the pointer of the first one of such tuples, tune in at the designated bucket and go to step 3.

3. Search the following $L_data_segment/(m + 1)$ buckets and sequentially locate K, where $L_data_segment$ indicates the length of a data segment.

In general, as shown in [34], the average tuning time using the flexible indexing technique is $(\sum \lceil \log_2 i \rceil + Data / (2 \times (m + 1))) / p$, $\forall 1 \leq i \leq p$, and Data denotes the size of the entire broadcast. The average access time using flexible indexing is $0.5 \times (File + p \times (\lceil \log_2 p \rceil + m) / n) \times (1 + p) / p$.

Later on, Yu et al. [71] extended the flexible indexing method with another parameter, the index base r , and proposed a parameterized flexible indexing scheme (PFIInd), which can be used to tune the trade-off between access efficiency and energy conservation based on the specific requirements of clients.

5.1.2 (1, m) Index and Distributed Index

Imielinski et al. [35] also developed a method for efficient multiplexing of data file with clustering index, called distributed index, which becomes one popular air indexing approach and was extended by many other researchers to meet different system requirements. Before introducing distributed index, this section starts with a simple index allocation method called (1, m) index.

(1, m) Index. (1, m) indexing scheme is an index allocation approach that broadcasts each index m times during one broadcasting cycle, in front of every fraction $(1/m)$ of the file, as illustrated in Fig. 7.

The first bucket of each index segment contains a tuple. The first field of this tuple is the attribute value of the last record in one *bcast*, and the second field is the offset to the next *bcast*. This tuple is used for guiding clients to the next *bcast*, if they missed the required record in the current *bcast*. Here is the access protocol of (1, m) indexing scheme [35] for the request with attribute value K :

1. Initial probe: tune into the broadcast channel, and get the pointer to the next index segment.
2. Go to doze mode and tune back to the broadcast channel to get the index segment.
3. Follow the index to determine when the data bucket of the first record with attribute value K will be broadcasted, and then follow the pointers to direct successive probes (client may doze between two successive probes).
4. Tune in again when the first record with attribute K is broadcasted and download all the records with attribute K .

Data denotes the average file size, and Index indicates the total number of buckets in the index tree. Let C denote the coarseness of the attribute. As shown in [35], the tuning time of (1, m) indexing method is $1 + k + C$. The access latency of (1, m) indexing method is

$$\frac{1}{2} \times \left((m + 1) \times Index + \left(\frac{1}{m} + 1 \right) \times Data \right) + C. \quad (14)$$

Optimum m . In order to minimize the latency for the (1, m) indexing, there is a formula to compute the optimum m (denoted as m^*), which is obtained by

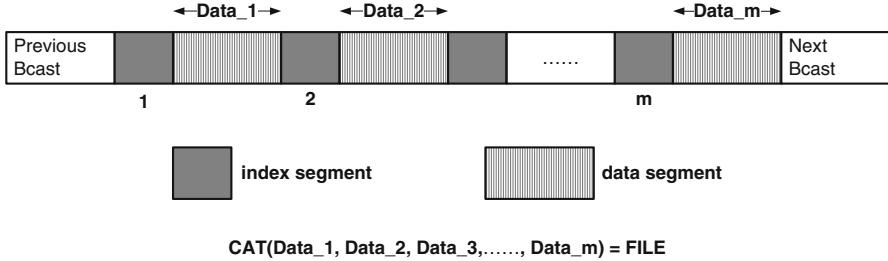


Fig. 7 An example of $(1, m)$ indexing

differentiating the formula of access latency with respect to m , equating it to zero and solving for m .

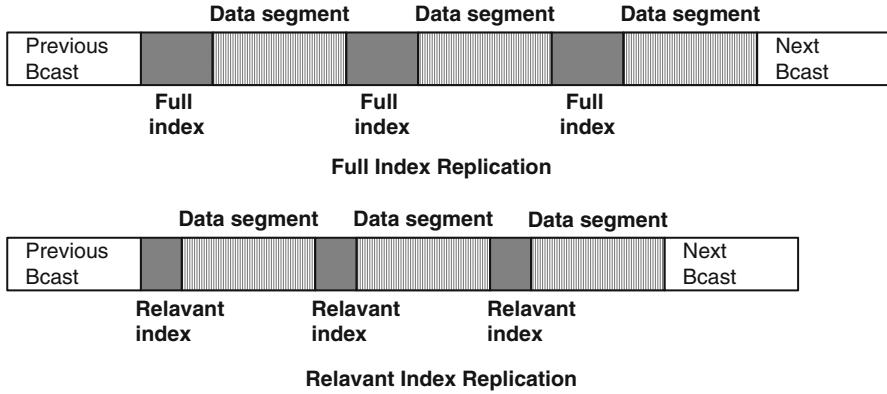
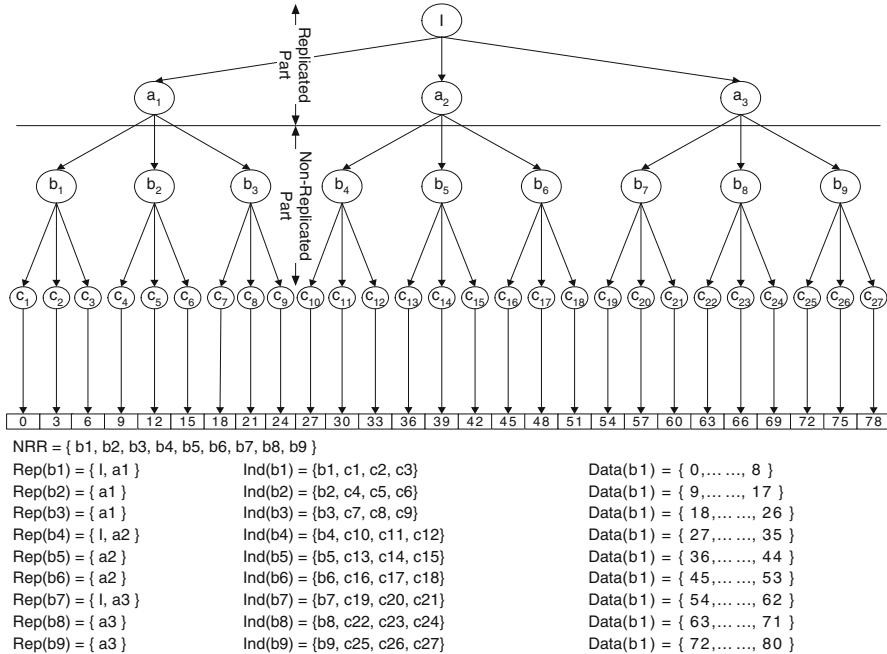
$$m^* = \sqrt{\frac{\text{Data}}{\text{Index}}}. \quad (15)$$

Therefore, just divide the file into m^* data segments and broadcast each segment after each index segment on that channel.

Distributed Index. $(1, m)$ indexing scheme could be improved by cutting down the replication times of an index. Distributed index is such approach that index is partially replicated, since the portion of index that indexes the following data segment is not necessary to be replicated. Figure 8 shows the difference between full index replication as in $(1, m)$ indexing scheme and the relevant index replication as in distributed index [35].

Distributed index takes an index tree T and subdivides it into two parts: the replicated part and the non-replicated part. Replicated part consists of the top r levels of T , and non-replicated part contains the bottom $(k - r)$ levels. Index nodes on the $(r + 1)$ th level are *non-replicated roots*, denoted by NRR. Any index subtree rooted at non-replicated root in NRR appears only once in one *bcast*, in front of the data segments it indexes. On the other hand, the replication times of each index node in the replicated part are equal to the number of its children.

As in [35], I indicates the root of T . B represents an index bucket in NRR, and B_i means the i th index bucket in NRR. $\text{Path}(C, B)$ is the path sequence from index bucket C to (excluding) index bucket B. $\text{Data}(B)$ shows the set of data buckets indexed by B. $\text{Ind}(B)$ indicates the part of T below (including) B. $\text{LCA}(B_i, B_k)$ represents the least common ancestor of B_i and B_k . $\text{NRR} = B_1, B_2, \dots, B_t$, and $\text{Rep}(B_1) = \text{Path}(I, B_1)$, where B_1 is the first bucket in NRR. $\text{Rep}(B_i) = \text{Path}(\text{LCA}(B_{i-1}, B_i), B_i)$ for $i = 2, \dots, t$. Thus, $\text{Rep}(B)$ indicates the *replicated part* of the path from the root to index segment B, while $\text{Ind}(B)$ refers to the *non-replicated part*. Figure 9 shows the values of Rep, Ind, and Data for each index buckets in NRR. The *bcast* contains a sequence of triples: $\langle \text{Rep}(B), \text{Ind}(B), \text{Data}(B) \rangle \forall B \in \text{NRR}$.

**Fig. 8** Index distribution**Fig. 9** An example of the index tree for distributed index

Control index is stored in those indices of replicated part. Suppose P_1, P_2, \dots, P_r is the sequence of buckets in $\text{Path}(I, B)$. $\text{Last}(P_i)$ indicates the attribute value in the last record indexed by P_i . $\text{NEXT}_B(i)$ represents the offset to the next occurrence of P_i (Fig. 10). Let l be the attribute value in the last record before B, and let begin be the offset to the next bcast. Control index in P_i have the following i tuples [35]:

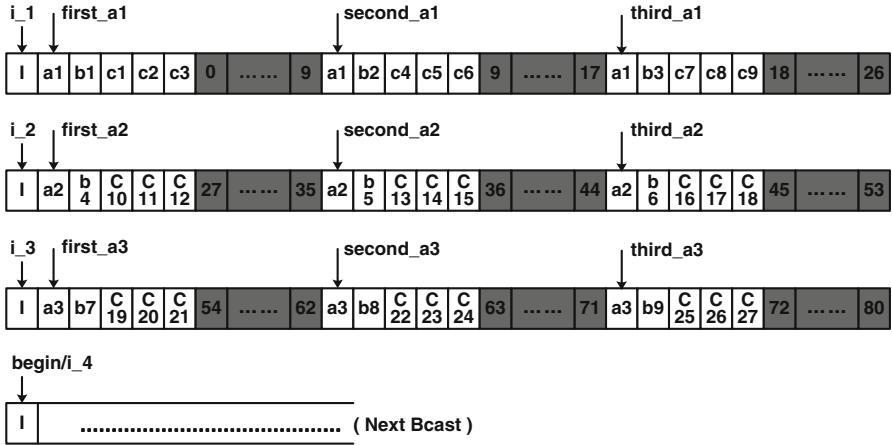


Fig. 10 An example of distributed index

```
[I, begin]
[Last(P2), NEXTB(1)]
[Last(P3), NEXTB(2)]
...
[Last(Pi), NEXTB(i - 1)]
```

An example of the control index is illustrated in Fig. 11. Control index of bucket P_i is used in this way: for the required data with key value of K , if $K < 1$, then the client is directed to the next *bcast*. If $K \geq 1$, then $(K > \text{Last}(P_j))$ is checked to find the smallest j . If $j \leq i$, then follow $\text{NEXT}_B(j - 1)$ to the next occurrence of P_i ; otherwise search the rest of the index in bucket P_i .

Here is the access protocol [35] for a data record with key value K :

1. Initial probe: tune into the *bcast*, and get the pointer to the next control index.
2. Go to the designated bucket with control index. Based on K and the control index, determine whether to do the following:
 - (a) Wait for the next *bcast* and go to step 3.
 - (b) Look for the appropriate higher level index, following one of the “NEXT” pointers, and go to step 3.
3. Follow the designated index and pointers to check when the first data bucket with key value K will be broadcasted, and doze between two successive probes.
4. Tune in again to download all the data with key value K .

The tuning time of distributed indexing is bounded by $2 + k + C$ as shown in [35]. The access time of distributed index is

$$\frac{1}{2} \times \left(\frac{\text{Index} - \text{Index}[r]}{\text{Level}[r + 1]} + \frac{\text{Data}}{\text{Level}[r + 1]} + \text{Data} + \text{Index} + \Delta\text{Index}_r \right) + C. \quad (16)$$

Control Index at first_a1	NON, NON 26, i_2	Control Index at first_a3	53, begin 80, i_4
Control Index at second_a1	8, begin 26, i_2	Control Index at second_a3	62, begin 80, i_4
Control Index at third_a1	17, begin 26, i_2	Control Index at third_a3	71, begin 80, i_4
Control Index at first_a2	26, begin 53, i_3	Control Index at i_2	26, begin
Control Index at second_a2	35, begin 53, i_3	Control Index at i_3	53, begin
Control Index at third_a2	44, begin 53, i_3		

Fig. 11 An example of control index

Optimizing the Number of Replicated Levels. The authors proposed that optimizing the number of r is related to minimizing the access time. Choose r such that the following expression is minimal:

$$\Delta \text{Index}_r + \left(\frac{\text{Index} - \text{Index}[r]}{\text{Level}[r+1]} + \frac{\text{Data}}{\text{Level}[r+1]} \right). \quad (17)$$

Evaluate the above expression (Eq. 17) for a given index tree with r ranging from 1 to the number of levels in the tree. The value of r which results in the minimal value for (Eq. 17) is the optimal number of replicated levels.

5.1.3 Huffman-Tree Index

Among traditional indexing structures, alphabetic Huffman tree is another important tree-based indexing technique, which has been applied to the wireless broadcast environment ever since the last decades. It is an efficient indexing technique because it takes into consideration the access probabilities of data items when constructing the Huffman tree. The popular data with higher access probability resides closer to the root in Huffman tree, which helps clients reduce search time when traversing from the root. However, almost all existing Huffman-tree indexing methods were discussed under multichannel broadcasting environment, which makes it hard to compare the performance between Huffman tree and other single-channel air indexing methods. Recently, Zhong et al. [74] proposed a novel Huffman-tree-based distributed indexing scheme (HTD) on single-channel broadcasting environment. They applied the feature of distributed indexing onto Huffman-tree indexing, which is an innovative idea that has not been considered before. The experiment results show that under a uniform communication environment, HTD outperforms B⁺-tree distributed indexing (BTD) and can replace BTD in most existing wireless data broadcast systems.

Data Item Key	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Frequency	23	4	12	10	17	31	15	21	29	19	7	12	16	14	20	48

Fig. 12 An example data set of Huffman-tree distributed indexing

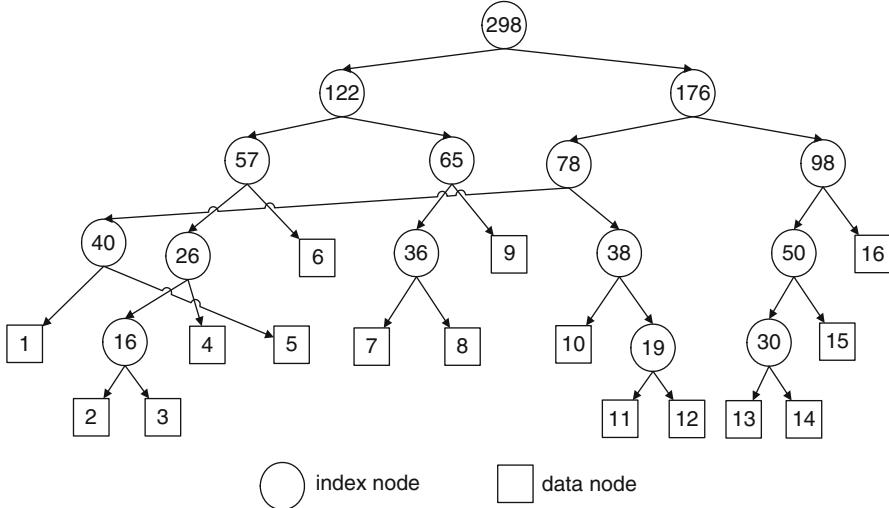


Fig. 13 The first step of constructing T_0

Here is the construction of Huffman-tree-based distributed indexing scheme (HTD). The structure of index bucket and data bucket of HTD is almost the same as that of BTD. The first stage is to construct a k -ary alphabetic Huffman tree as in [52]. Figure 12 gives an example of the data set and the corresponding access frequencies. The construction process of alphabetic Huffman tree is illustrated in Figs. 13 and 14.

Stage 1: choose data nodes d_i, d_j to merge when:

1. There are no data nodes between d_i and d_j .
2. The sum of their frequencies is the minimum.
3. They are the leftmost pair of nodes among all candidates.

Create a new index node d'_i with frequency equal to the sum of d_i 's and d_j 's. Replace d_i and d_j with d'_i in the construction sequence. This first stage produces tree T_0 without alphabetic ordering of the data nodes. Figure 13 gives an example, where the frequencies of each index node are shown inside the circle as index key values.

Stage 2: record the level of each data node of T_0 as L_i for data d_i , while the root node is level 1. After this, rearrange pointers from bottom to the root, such that for each level the leftmost two nodes have the same parent, and then the next two, so on, and so forth. Generate an alphabetic Huffman tree T like this, while keeping the level of each node in T_0 , as shown in Fig. 14.

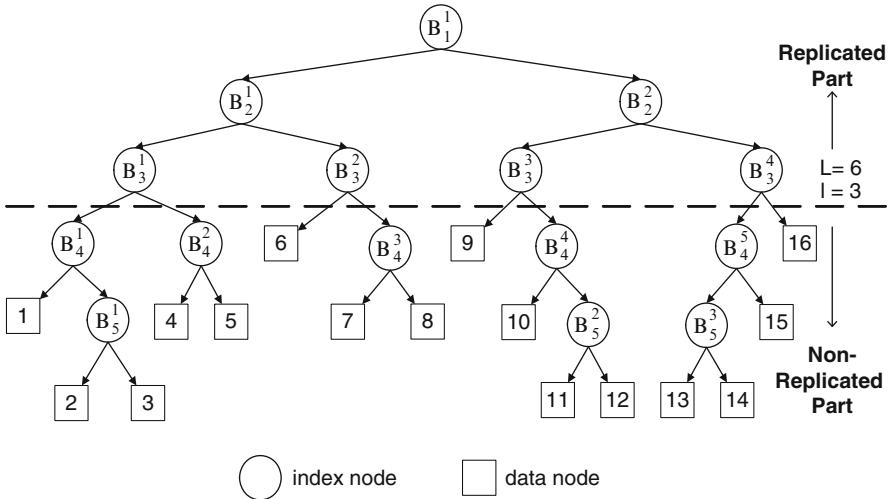


Fig. 14 The final Huffman-tree T

To construct k -ary Huffman tree, just extend this algorithm by merging at most k nodes in stage 1 and combining up to k nodes in stage 2.

After generating the alphabetic Huffman tree T , cut it at level l , and perform a distributed traversal as in distributed indexing method. The index nodes above cutting level l are *control index*, and index nodes below l are *search index*. Next, append control tables onto control index in the same way as distributed indexing.

The major differences between Huffman-tree and B^+ -tree indexing are that (1) the positions of leaf nodes are different and that (2) the subtree sizes below l are different. There might be data items above l in a Huffman tree, depending on which level is chosen for l , because data items are not restricted to reside at bottom level of Huffman tree. Also the sizes of subtrees below l may vary a lot in Huffman tree, but for B^+ -tree the subtrees have similar sizes.

The final broadcast sequence \mathbb{B} of this example is shown in Fig. 15. Control table contains a number of entries, each of which is a tuple of *key value* and *offset*. Here is the access protocol of Huffman-tree-based distributed indexing: when searching in a control table, client compares the *key value* of the first entry in control table with the request key value K :

- If request key K is less than or equal to the first key in control table, wait until the root of the next bcast.
- If request key is greater, go on to compare the next entry in control table.
 - Turn to doze mode for *offset* time if request key K is not greater than the next key in control table.
 - Otherwise, continue comparing with the rest entries of control table until finding such an entry, or go to default bucket (the next index) if not found.

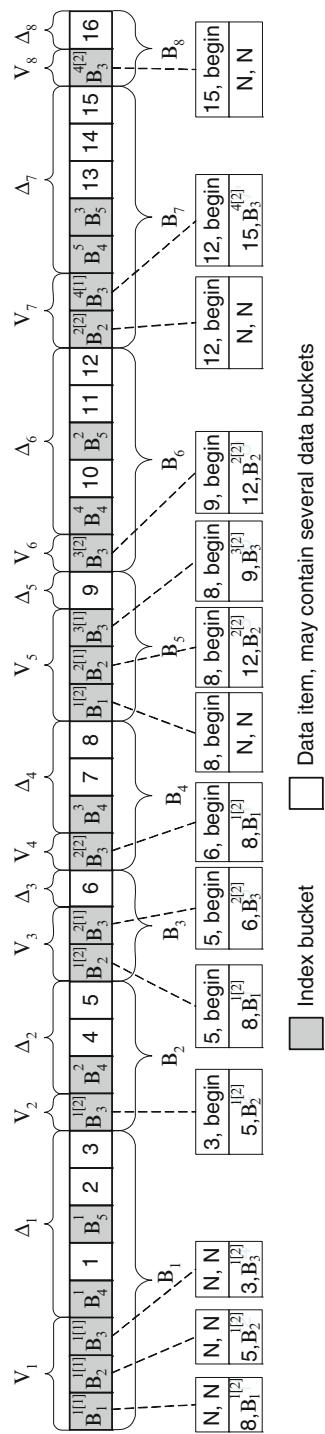


Fig. 15 The broadcast sequence of Huffman-tree distributed indexing

As proven in [74], the expected average access time of alphabetic Huffman-tree-based distributed indexing scheme is

$$E(AL) = \frac{1}{\|\mathbb{B}\|} \sum_{i=1}^R \left(\sum_{w=1}^{R-2} \left(\frac{v_i + \delta_i}{2} + \sum_{j=i+1}^{i+w-1} (v_j + \delta_j) + v_{i+w} + \frac{\delta_{i+w}}{2} \right) P_{(i+w)\%R} (v_i + \delta_i) + \left(\frac{v_i + \delta_i}{2} \right) P_i v_i + \sum_{i=1}^R (v_i + \delta_i) P_i \delta_i \right). \quad (18)$$

The expected average tuning time for alphabetic Huffman-tree-based distributed indexing scheme is

$$E(TT) = \frac{2 \sum_{i=1}^R v_i + (2+r)|D| + 3 \sum_{i=1}^R \delta_i}{r \|\mathbb{B}\|} + \sum_{i=1}^{|D|} \left(\frac{l}{2r} + \frac{1}{r}(L_i - l) + s_i \right) p_i. \quad (19)$$

5.1.4 Exponential Index

Xu et al. [65] proposed the *exponential index*, which facilitates replication by sharing links in different search trees to minimize storage overhead and allows clients to start searching at any index node. Exponential index can be adjusted to optimize the access time with the tuning time bounded by a given limit, and vice versa. The nature of distributed structure makes it easy to handle link errors. When dealing with data updates, exponential index works well, especially for applications where the non-key values may change but the search key values do not change often.

Exponential index was created based on clustered broadcasting, which means that data items are broadcasted in ascending order of their attribute values. Figure 16 shows an example of exponential index, that a server broadcasts stock information periodically. Each bucket stores one stock item in its data part and some index information in its index table. The index table contains four entries, each of them is a tuple $\{distInt, maxKey\}$, where distInt indicates the distance range, whose size grows exponentially, and maxKey is the maximum key value of these buckets. The first entry of index table shows the next bucket. For each $i > 1$, the i th entry indicates the bucket segments 2^{i-1} to $2^i - 1$ away of totally 2^{i-1} buckets. Thus, the $(i-1)$ th and i th entries' maxKey values give the range of buckets indexed by the i th entry. Here in this example the sizes of the indexed segments exponentially increase by a base of 2.

Exponential index has a linear and distributed structure, which enables searching from any index bucket. An index link is shared by different index buckets. In addition, by adjusting the exponential base, the indexing overhead can be controlled. Exponential index could be generalized by changing the index base to any value $r \geq 1$. The i th entry of an index table describes the maximum key value of buckets that are $\lfloor \sum_{j=1}^{i-2} r^j + 1 \rfloor = \lfloor \frac{r^{i-1}-1}{r-1} + 1 \rfloor$ to $\lfloor \sum_{j=1}^{i-1} r^j \rfloor = \lfloor \frac{r^i-1}{r-1} \rfloor$ away.

To reduce the indexing overhead, each group of I buckets are merged into a *data chunk*, and the exponential index is built for each chunk. Therefore, the size and the number of index tables are decreased. Furthermore, index links are constructed

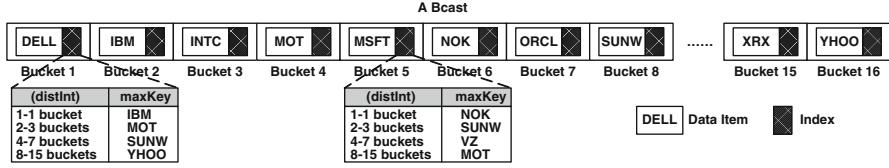


Fig. 16 An example of exponential index

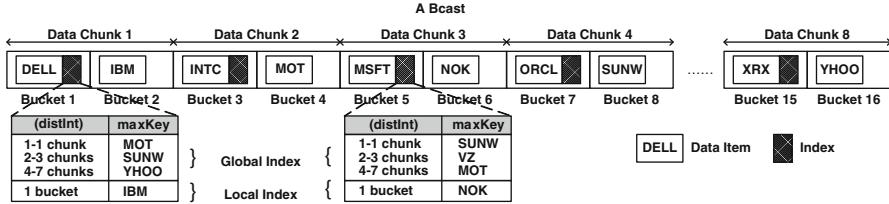


Fig. 17 An example of generalized exponential index

for all buckets within each chunk, called *local index*. Those index entries directing to other data chunks are named *global index*. An example of such generalized exponential index is shown in Fig. 17, where the index base r and the chunk size I are both set to 2. Algorithm 11 presents the searching algorithm of exponential index [65].

Two major parameters for exponential index are index base r and chunk size I , which offer the flexibility between access time and tuning time. Generally, decreasing index base r leads to the increasing number of index entries and indexing overhead, while tuning time decreasing with r . If the chunk size I is larger, the tuning time becomes less, but the initial probing time might be longer.

As shown in [65], the initial probing time plus half of the length of *bcast* makes the average access latency:

$$E(d) = \frac{I}{2} + \frac{IC}{2}. \quad (20)$$

The average tuning time is

$$E(t) = \frac{I-1}{I} + \frac{B(I-1)}{B(I-1) + B'} + \frac{1}{C} \sum_{l=0}^{C-1} t(l). \quad (21)$$

Here $t(l)$ is the tuning time for a chunk that is l chunks away [65]:

$$t(l) = \begin{cases} 1, & \text{if } l = 0; \\ t(l-x) + 1, & \text{if } l > 0, \end{cases} \quad (22)$$

where x is the maximum value in the set of $\{1, 2, \lfloor r+2 \rfloor, \dots, \lfloor \frac{r^{nc}-1}{r-1} \rfloor + 1\}$ that is less than or equal to l .

Algorithm 11 Index searching algorithm for exponential index

```

1: wait for the first bucket of next chunk to be broadcasted;
2: for each data item in the bucket do
3:   if it is the requested data then
4:     stop searching and finish the query;
5:   end if
6: end for
7: check local index in the index table;
8: if the requested data is within the key range of the  $i$ th entry then
9:   go to doze mode until the  $i$ th bucket appears, retrieve data and finish the query;
10: end if
11: check global index in the index table;
12: if the requested data is within the key range of the  $i$ th entry then
13:   go to doze mode for  $(\lfloor \frac{r^{i-1}-1}{r-1} + 1 \rfloor \cdot I - 1)$  buckets until the first bucket of the  $(\lfloor \frac{r^{i-1}-1}{r-1} + 1 \rfloor)$ th chunk appears, and repeat this search process;
14: end if

```

5.1.5 Hashing Schemes

Hashing-based schemes do not require additional index to be broadcasted together with the data. Hashing parameters are included in the *control part* of data buckets, which helps in reducing access latency and tuning time. Control part for the first N buckets contains the following:

1. Hash function $h(K)$
2. Shift: the pointer to a bucket with key K that $h(K) = \text{address(Bucket)}$

In most cases the hashing function is not perfect, and there might be collisions. The colliding records will be stored immediately after the bucket assigned to them, pushing the rest of the file “down,” and create offsets for those buckets. Imielinski et al. [34] proposed two hashing protocols: hashing A and hashing B.

The access protocol of hashing A [34] for record with key K is like this:

1. Initial probe, read the control data from the current bucket, and calculate $h(K)$.
2. If *Current bucket #* < $h(K)$, then go to *doze mode* and tune in again to slot $h(K)$; Else wait till the next *bcast*, and repeat protocol again.
3. At $h(K)$, get the *Shift* value, go to *doze mode* and tune in again after *Shift* number of buckets.
4. Listen until either the record with key K is found (success) or with key L ($h(L) \neq h(K)$) is found (failure).

[Figure 18a](#) is an example of searching key $K = 15$ in the file, hashed Modulo 4. Initial probe is at the second bucket. Follow the hashing function, client first goes to bucket four which may contain key $K = 15$ if there is no overflow. However, the keys are shifted. Client reads the shift value, goes to doze mode, and tunes in again later. It must look at all the overflow buckets of that bucket to find the key $K = 15$. [Figure 18b](#) shows the cases when client needs to wait for the next bcast, which may occur due to either the data miss or the directory miss.

Hashing A was further extended to hashing B, which significantly reduced the directory miss by a minor modification of the hashing function. Let d be the size of the minimum overflow chain. The hashing function h was modified as follows [34]:

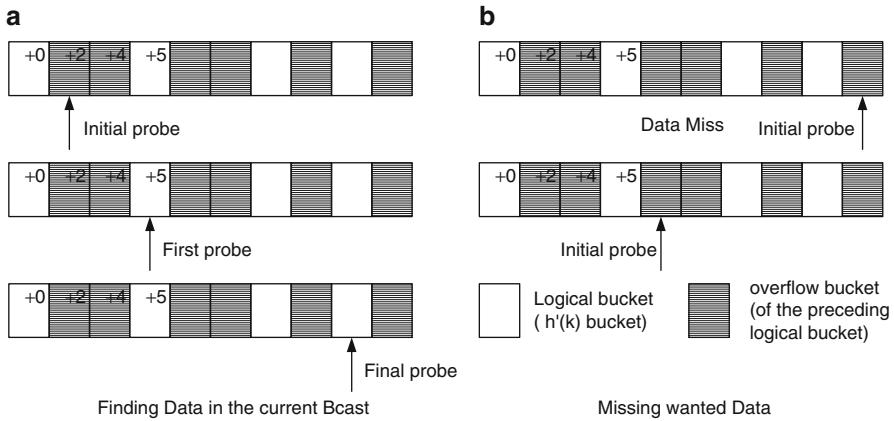


Fig. 18 An example of Hashing A (the numbers in the *upper right* corner of some buckets indicate the shift value in the control part)

$$h'(K) = \begin{cases} h(K), & \text{if } h(K) = 1. \\ (h(K) - 1)(1 + \min_overflow) + 1, & \text{if } h(K) > 1. \end{cases}$$

The new hashing function h' takes into consideration the shift caused by overflow. Figure 19 shows the improvement of hashing B. The reduction is substantial if the sizes of the overflow chains do not differ much. When the sizes of overflow chain are same for all buckets, h' becomes the perfect hashing function. However, perfect hashing function does not guarantee the minimal access latency.

Let $Displacement(h, K)$ denote the distance between the physical bucket where K resides ($Physical_bucket(k)$) and the designated bucket for k ($h'(k)$): $Displacement(h, K) = Physical_bucket(K) - h'(k)$. As presented in [34], the expected access time of hash B is obtained by calculating the access time for each key K and then averaging it out. The expected access time “per key” is combined with two factors:

1. Data miss occurs if the initial probe is between the designated bucket and the physical bucket. The client has to wait an extra *bcast*. Thus,

$$(Displacement(h, K)/Data(h)) \times (Data(h) + 1/2 \times Displacement(h, K)).$$

2. If the initial probe occurs outside the displacement area, it has to wait on average between the $Displacement(h, K)$ and the file size. Then,

$$(1 - (Displacement(h, K)/Data(h))) \times (Data(h) + Displacement(h, K))/2.$$

The sum of expected access latency per key divided by total number of keys becomes the expected access latency. For tuning time, the more logical buckets, the lower the tuning time. When the number of logical buckets grows, the tuning time could go down to 3 [34].

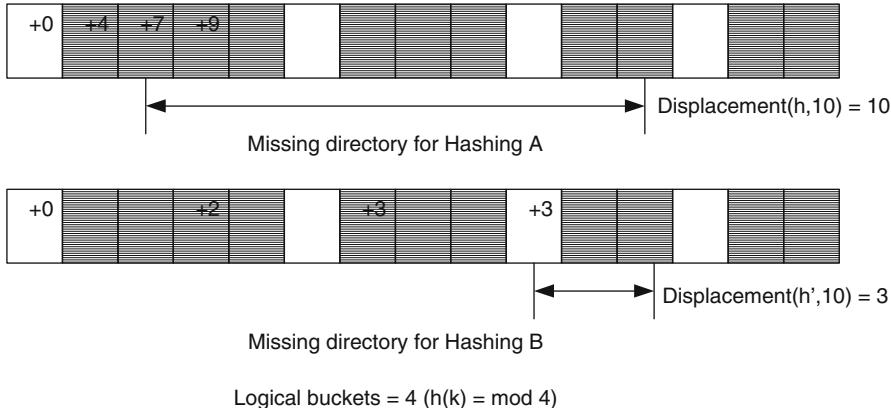


Fig. 19 Displacement comparison

Later, Yao et al. [68] proposed MHash, which optimizes both tuning time and access latency. MHash uses a two-argument hash function $H(k, l)$, which allows each data to be mapped to a number of slots. Popular items are broadcast more frequently, which enables non-flat broadcast to reduce access time. The authors refined MHash by investigating the design of hash function, and analyzed the optimal bandwidth allocation for MHash indexing. They generated hash functions that produce broadcast schedules without unoccupied slots. Such a hole-free hash function was constructed by injecting an offset. MHash reduces access latency and tuning time, since it enables non-flat broadcast, properly spacing the instances of each data, and optimally allocates the bandwidth among data items. Under skewed access distribution, MHash achieves the access latency that is close to the optimal broadcast scheduling.

5.2 Indexing in Multichannel Broadcasting Environment

Multichannel broadcasting techniques enable further reduction of access time by partitioning data onto multiple channels. Several works [10, 15, 48, 49, 69] deal with efficient data allocation for multichannel data broadcast systems. Furthermore, various indexing techniques and index allocation methods have been developed for multichannel data broadcast. Due to the feature of multichannel broadcasting, how to efficiently allocate index has attracted more attentions.

There exist two popular categories of index allocation techniques. One approach is to interleave index with data on multiple channels. Examples of such approach include [6, 46, 70]. In [6], Amarmend et al. derived external indices from the scheduled data to determine which channel broadcasts the required data, and allocate them over upper channels. They also extended the exponential index as the local index for each channel. Lo et al. [46] used a k -ary search tree as the

index tree and discussed the optimal data and index allocation problem on multiple broadcast channels. In [70], indices are striped over multiple channels in such a way that index levels are sequentially broadcasted in sub-strips at certain frequencies. Hsu et al. [24] extended the distributed indexing scheme on multiple channels. Vijayalakshmi et al. [59] presented a hashing scheme for information access through multiple channels in which hash functions are used to index broadcast information across multiple channels.

Another approach is to assign certain channels as designated index channels and others as data channels. Jung et al. [38] gave a tree-structured index allocation method with replication over multiple broadcast channels. Their allocation method designed index blocks based on the number of index channels, which contain null elements. Waluyo et al. [60] proposed a global indexing scheme in which each index channel has a different part of the entire index tree while preserving its overall structure, with replicated tree nodes among index channels. Wang and Chen [61] proposed an index allocation method named TMBT for multichannel data broadcast, which creates a virtual distributed index for each data channel and multiplexes them on the index channel.

So far, almost all literature on multichannel data broadcast have the assumption that all channels are synchronized, that is, all channels have the same throughput and they begin and finish a broadcast cycle at the same time.

Shivakumar et al. [52] discussed the construction of Huffman tree that is similar to Huffman code construction. Chen et al. [17] proposed algorithms for constructing the skewed Huffman tree. However, there exists a problem that the clients may fail to find the desired data item by traversing the Huffman tree. Another kind of Huffman tree proposed in [26], called alphabetic Huffman tree, serves as a binary search tree, which is further extended to k -ary search tree in [52], so that a tree node will fit in a wireless packet of any size by adjusting the fanout of the tree. In [51], Shivakumar and Venkatasubramanian proposed an unbalanced index tree allocation method based on alphabetic Huffman tree, which partitioned the multiple channels into data channels and index channels. They assigned one index channel to each level of the index tree, which might be inefficient when the number of index channels is less than the depth of the index tree.

Jung et al. [38] studied an efficient alphabetic Huffman-tree index allocation method for broadcasting data with skewed access frequencies over multiple physical channels. The rest of Sect. 5.2 will introduce their method in detail.

To cope with the two major problems of previously proposed indexing allocation methods that (1) require equal size for both index and data and (2) the performance degrades when the number of given physical channels is not enough, the proposed tree-structured index allocation method minimizes the average access time by broadcasting the hot data and their indices more frequently than the less hot data and their indexes over the dedicated index and data channels.

Here are the assumptions in [38]. Let DC_i denote the i th data channel. Let $L(v)$ denote node v 's level number in an alphabetic Huffman tree. IL_1 is the first ordered list that consists of the index sequence along the path from the root to the data in DC_1 . Then, IL_i ($2 \leq i \leq$ the number of data channels) denotes the i th ordered

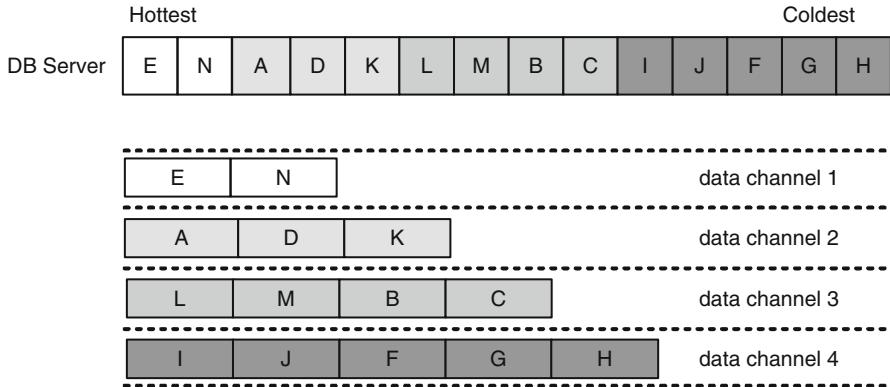


Fig. 20 An example of data broadcast schedule

list, excluding the index nodes in IL_j for $j = 1, \dots, i - 1$. Each ordered list $IL_i = \langle I_1, I_2, I_3, \dots, I_q \rangle$ satisfies two requirements:

1. $L(I_1) \leq L(I_2) \leq L(I_3) \leq \dots \leq L(I_q)$
2. When $L(I_j) = L(I_{j+1})$, index node I_j is on the left of I_{j+1} in the index tree.

Indexes in IL_i point to the hotter (more popular) data than indices in IL_{i+1} , since data at DC_i are hotter than data at DC_{i+1} . Thus, to minimize the access latency, indices in IL_i are broadcasted as many times as the corresponding data in DC_i . Let DX_i denote the number of data items in channel DC_i . LCM means the least common multiple of $DX_1, DX_2, \dots, DX_{ND}$, where ND is the number of data channels. Then, the repetition frequency that IL_i has to be broadcasted is $RF_i = \text{LCM}/DX_i$ [38].

An example of an alphabetic Huffman tree with data nodes {A, B, C, D, E, F, G, H, I, J, K, L, M, N} of Fig. 20 is shown in Fig. 21. The set of index nodes is $\{1, 2, 3, \dots, 11, 12, 13\}$. The hottest data are E and N, and the oldest data are F and G, since they reside at the highest and lowest level of the tree, respectively.

Figure 22 shows the examples of IL_i and RF_i from that Huffman tree of Fig. 21. Here, for data nodes (A, D, K) in channel DC_2 , the corresponding ordered list of index is $IL_2 = \langle 4, 5, 6, 7 \rangle$. $RF_2 = 20$ means the indices in IL_2 will be repeated for 20 times in one index broadcast cycle.

Given $RF_1, RF_2, RF_3, \dots, RF_{ND}$, the normalized repetition frequencies are $NRF_1 = RF_1/RF_{ND}, NRF_2 = RF_2/RF_{ND}, \dots, NRF_{ND} = RF_{ND}/RF_{ND}$ [38]. And the value of NRF_i is rounded off.

Figure 23 illustrates the index blocks for the index ordered lists $\langle 1, 2, 3 \rangle, \langle 4, 5, 6, 7 \rangle, \langle 8, 9 \rangle$, and $\langle 10, 11, 12, 13 \rangle$ of the Huffman tree shown in Fig. 21. Since the number of index channels is 2, the size of each member of index block should be 2, and each member's element is not related to each other with an ancestor-descendent relationship in the tree. Figure 24 shows the index allocation method on 2 index channels for the index blocks of Fig. 23. This index allocation method could also be extended to k channels.

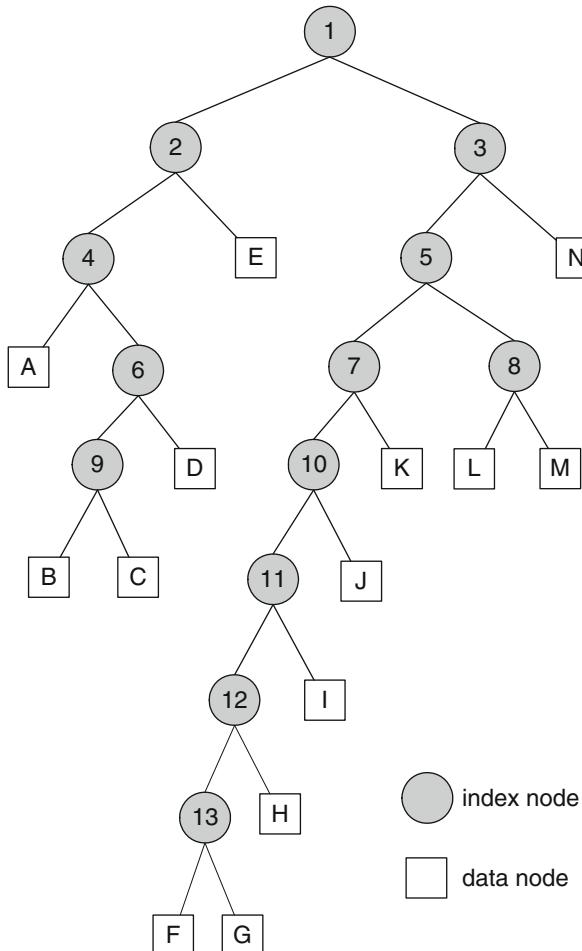


Fig. 21 An example of alphabetic Huffman tree

4 Data Channels						
E	N					IL ₁ :<1, 2, 3> RF ₁ = 60/2=30
A	D	K				IL ₂ :<4, 5, 6, 7> RF ₂ = 60/3=20
L	M	B	C			IL ₃ :<8, 9> RF ₃ = 60/4=15
I	J	F	G	H		IL ₄ :<10, 11, 12, 13> RF ₄ = 60/5=12

Fig. 22 The index list and repetition frequencies

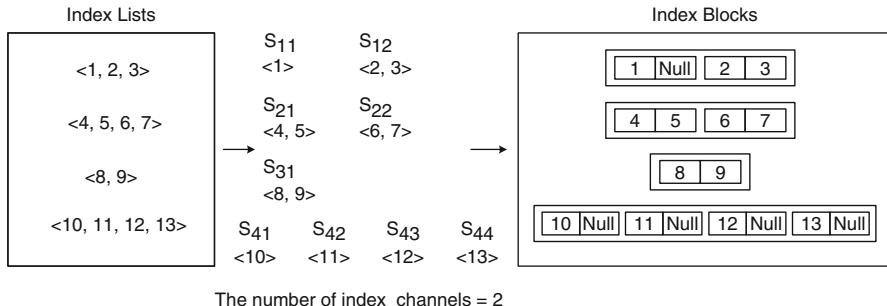


Fig. 23 The example of index blocks

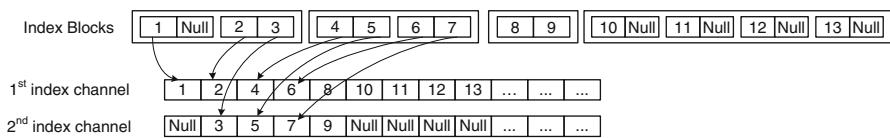


Fig. 24 Allocation of indices on two index channels

[Figure 25](#) gives an example of broadcasting with two index channels and four data channels in the system. The gray boxes represent the index nodes pointing to indices, while the white boxes with letters represent the index nodes pointing to data. Assume that data node size is ten times larger than index node size and that a client requests data E which is broadcasted on data channel 1. The client first tunes into index channel 1 to get the root of the tree, and then traverses the index tree to reach the index node pointing to data E. Assume that when $t = 14$, the index node 2 is being broadcasted over index channel 1, which has the information of data E. With the location of E, the client goes to doze mode until E arrives, and then downloads data E.

In [38], the average access time of alphabetic Huffman-tree indexing allocation method for multichannel environment is as follows:

$$\begin{aligned}
 AveAT = & \frac{ISize}{2 \cdot c_i \cdot NRF_1} \sum_{j=1}^{c_d} NRF_j \cdot |IL_j| + \frac{ISize}{c_i} \sum_{j=1}^{c_d} \left[\left\{ \sum_{k=1}^{j-1} |IL_k| + \frac{|IL_j|}{2} \right. \right. \\
 & \left. \left. + \frac{1}{NRF_j} \cdot \sum_{k=1}^{j-1} |IL_k| \cdot (NRF_k - NRF_j) \right\} \cdot \frac{NRF_j}{NRF} \right] \\
 & + \frac{DSize}{2} \sum_{j=1}^{c_d} \frac{NRF_j}{NRF} \cdot n_j. \tag{23}
 \end{aligned}$$

As shown in [38], given c physical channels, the optimal number of index channels c_i and the number of data channels c_d can be determined by the above Eq. (23).

	t=0	t=10	t=20	t=30	
index channel 1	... E A D L J I H F M G M M G M ...	E A D L J I H F M G M M G M	E A D L J I H F M G M M G M	E A D L M ...	
index channel 2	... N K B C N K B C N K B C ...	N K B C N K B C	N K B C N K B C	N K N K B C ... C ...	
data channel 1	... E ...	N	E	...	
data channel 2	... A ...	D	K	A	
data channel 3	... L ...	M	B	C	
data channel 4	... I ...	J	F	G	

Fig. 25 The final *bcast* over index channels and data channels

Just compute a set of *AveATs* for all possible values of c_i and c_d , and then choose the pair with minimal *AveAT*. Since the time complexity of computing *AveAT* is $O((c_d)^2)$ and $c_d < c$, then the time complexity of this method is $O(c^3)$. Although the optimal split might need to be recomputed with the changes of data set, it is not a big issue since the time complexity $O(c^3)$ is low.

6 Coverage Optimization for Data Broadcast

Coverage optimization is a special topic for data broadcasting problem in wireless communications. Usually, this topic deals with data broadcast and data transmission problems in wireless ZigBee networks [20, 53], wireless/mobile Ad-Hoc networks [64], cellular networks [75], heterogeneous overlayed wireless networks [30], and other types of data broadcast services. Most of the coverage problems are discussing whether the data broadcast service provider guarantees reliable coverage for a certain area or region and how to make the broadcast system more energy efficient, time efficient, or fault tolerant (more reliable). As described in previous sections, the researches for data broadcast are cross-disciplinary works among multimedia technology [43], wireless networking [22], data engineering [18], and other related fields [29]. Accordingly, majority of the studies on coverage optimization belong to wireless networking. This type of problems can be converted into combinatorial optimization problem and most of them are NP-hard problems reducible to known NP problems, which cannot be solved in polynomial time unless $P = NP$ [23]. Therefore, to solve this type of problems, people usually design approximation algorithms and heuristic approaches to provide the near-optimal solutions.

This section will briefly introduce several existing problems and optimization methods of coverage optimization for data broadcast in wireless communications. The organization of this section is shown as follows: Sect. 6.1 will describe the mathematical modeling and problem definition of coverage optimization, Sect. 6.2 will introduce the classification and reduction of this problem, while Sects. 6.3–6.5 will illustrate different variants of coverage optimization for data

broadcast and its corresponding solutions. Finally, Sect. 6.6 will give a summary for coverage optimization and some discussions for the future directions.

6.1 Mathematical Modeling and Problem Definition

To formally define coverage optimization problems for data broadcast, the problem formulation and reduction have to be introduced. More specifically, as shown in Fig. 26, in a typical data broadcast system, one or more base stations (BSs) (also called “nodes” in combinatorial geometry) will continually broadcast information among a certain area/region. Within this area/region, there are hundreds/thousands of fixed-line or mobile clients that have the requirements for these information. As a consequence, how to economically, efficiently, and reliably cover the whole area such that each client can successfully receive the required messages/data streams becomes an important issue for all service providers.

In mathematical description, given an area A in Euclidean space, a group of nodes $N = \{n_1, \dots, n_k\}$, each node n_i covers a certain area described as $V(n_i)$. Each node can also communicate with other nodes in this area. In A , there are $T = \{t_1, \dots, t_m\}$ clients. In different mathematical models, $V(n_i)$ has different representations. If considering signal interference like interrupt, collision, and contention $V(n_i)$ usually denotes a circle whose center is n_i . If each n_i has the same capability, then the whole graph becomes a unit disk graph (UDG) otherwise the graph is a disk graph (DG). Figures 27 and 28 are examples of UDG and DG correspondingly. If not considering signal interference, then $V(n_i)$ usually represents a Voronoi division with the following definition:

$$V(n_i) = \bigcap_{1 \leq j \leq k, j \neq i} \{p : |p - n_i| \leq |p - n_j|\}.$$

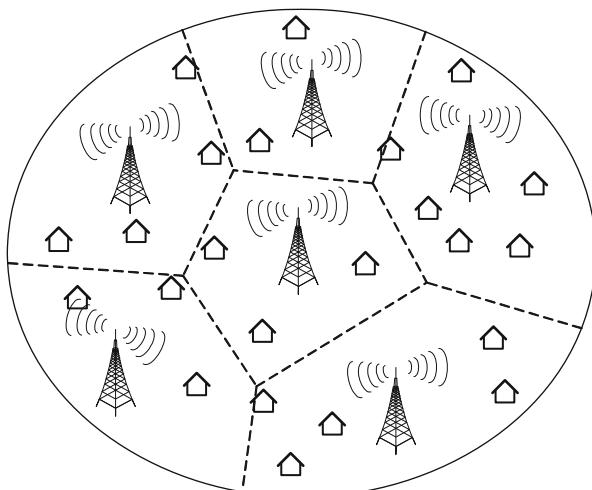


Fig. 26 An example of coverage for data broadcast

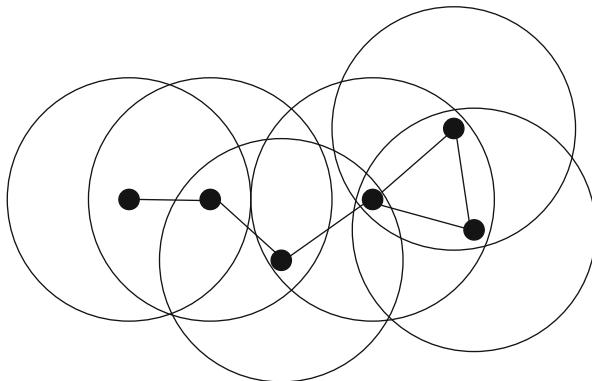


Fig. 27 Unit disk graph model for wireless network

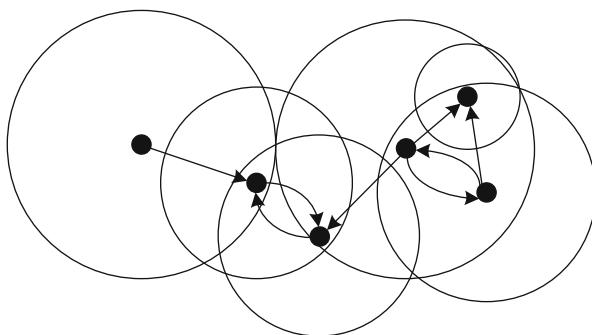


Fig. 28 Disk graph model for wireless network

It means that each BS n_i only charges for its domain $V(n_i)$. Such graph is named as Voronoi graph (VG). An easy example can be seen from Fig. 26, in which the dashed lines represent the boundary of covered area $V(n_i)$ for each n_i in this case.

The problems of coverage optimization are usually defined according to the above given factors. They can be reduced as a minimization-optimization problem with a coverage constraint and some cost constraint together. The next subsection will discuss the classification and reduction of coverage problem.

6.2 Classification and Reduction

There are two types of coverage problems. If not considering the data transmission among nodes, the object of coverage optimization is as follows:

1. Cover the whole area A using the minimum cost (defined as *region coverage*).
2. Cover the target set T using minimum cost (defined as *target coverage*).

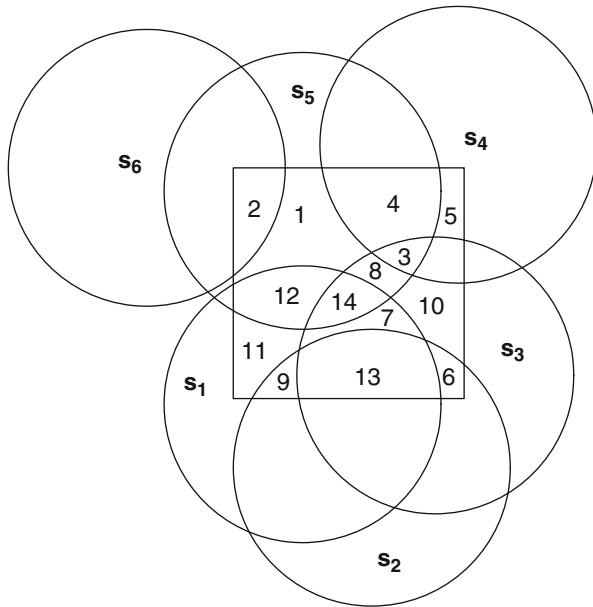


Fig. 29 An example to reduce region coverage to target coverage

Here the cost can be defined as a customized function, according to different requirements in different scenarios.

The region coverage problem can be simply converted to the target coverage problem in UDG or DG [72]. [Figure 29](#) is an example to illustrate this reduction. The square in [Fig. 29](#) is the potential region for broadcasting. For each point in this area, let A denote the set of nodes that can cover this point. Partition the area into different parts, each with different node coverage sets. As a consequence, area coverage problem can be reduced to target coverage problem when considering each part as a target. This problem can further reduced to a set cover problem. Consider each small division as a target mark it with a number from 1 to 14, and then insert covered targets into each set of sensors. Say, $S_1 = \{7, 9, 11, 12, 13, 14\}$, $S_2 = \{6, 9, 13\}$, $S_3 = \{3, 6, 7, 8, 10, 13, 14\}$, $S_4 = \{3, 4, 5\}$, $S_5 = \{2, 1, 3, 4, 8, 12, 14\}$, and $S_6 = \{2\}$. The coverage problem can be reduced to the problem of finding a minimum set cover from S_i to cover $T = \{1, \dots, 14\}$.

In VG, target coverage is widely studied, since due to the definition of Voronoi division, area A has already been divided into different parts, and each part is dominated by some node.

If further considering the data transmission among nodes, then the coverage problems are more complicated. It will either involve the traditional broadcast tree optimization, or deal with selecting a subset of broadcasting sources to reduce the

energy consumption and latency cost. Moreover, based on different requirements from real world applications, there are additional constraints or benefits for coverage optimizations, for example the multilingual message broadcast service, the TV data streaming requirements, the multichannel data broadcast, and the multi-input multi-output (MIMO) antenna technologies. The following subsections will introduce specific coverage optimization problems according to various system models and various client requirements.

6.3 Cell Broadcasting Scheme to Support Multilingual Service

This subsection will describe a coverage problem for cell broadcast to support multi-lingual service [75]. It solves a coverage problem named EEML-CB, which is described as follows:

Definition 9 The *energy-efficient multilingual cell broadcasting (EEML-CB) problem* can be described as

Given: $\mathbf{C} = \{c_1, \dots, c_n\}$ as client set, $\mathbf{BS} = bs_1, \dots, bs_m$ as set of broadcast stations, $\mathbf{H} = \{h_1, \dots, h_n\}$ as the index of the BS communicating with c_i , $\mathbf{L} = \{l_1, \dots, l_k\}$ as language set, $\mathbf{CL} = \{cl_i^q\}$ as client acceptable language set, \mathbf{W} as language weight set, R_{\max} as the maximum broadcasting radius, and A as target area. For each bs_j , assign an efficient strategy S_j , containing several couples as $\langle l_q, r_q^j \rangle$, which means that bs_j should broadcast language l_q with distance r_q^j .

Find: A minimum energy consumption strategy $\mathbf{S} = \{s_j\}, (0 \leq j \leq m)$ for every BS, such that each client c_i can receive at least one copy of the required messages with its acceptable languages.

In short, this topic deals with a target coverage problem for each BS specifically. Within $V(n_i)$ for each node, there locates a group of customers with different language requirements. Sending messages in different language will result different weights. And the object of this topic is trying to assign a minimum energy-cost schedule for the BS, such that all the clients within its domain can receive as least one copy of message in its acceptable language set.

The authors first proved that EEML-CB is a NP-hard problem, by reducing it to an instance of set cover problem, and then provide the inter linear programming (ILP) as follows:

$$\begin{aligned} \min & \sum_{j=1}^m \sum_{q=1}^k E_j^q \\ \text{s.t. } & \sum_{q=1}^k x_i^q \geq 1 && \forall c_i; \\ & x_i^q \leq cl_i^q && \forall c_i, l_q; \\ & A \cdot (d_i^{h_i})^2 \cdot w_q \cdot x_i^q \leq E_{h_i}^q, && \forall c_i, l_q; \\ & d_i^{h_i} \cdot x_i^q \leq r_{h_i}^q, && \forall c_i, l_q. \end{aligned}$$

Algorithm 12 Smart multi-lingual cell broadcasting (SMCB)**Input:** $C, BS, H, L, CL, W, r_{\max}$ and A .**Output:** S and E .

- 1: For each bs_j , select its client set C_j .
- 2: Find the farthest client $c_i \in C_j$ with its least weight language l_q . Assign $r_j^q = d_i^j$. Mark this client.
- 3: Mark clients whose acceptable language set includes l_q .
- 4: Repeat steps 2–3 until no client left unmarked in C_j .

The objective of this ILP model is to minimize the total energy consumption for a BS. The first two constraints guarantee that each client receives at least one language from its accepted language set. The third constraint calculates the energy consumption for each language, while the last one guarantees that for each given language, the energy is large enough for the BS to cover all target clients.

The ILP is acceptable in small-scale networks. However, when network size grows, the time consumption of ILP grows exponentially. Therefore, the authors designed another greedy algorithm shown as follows:

The main idea of this algorithm is to find the farthest client for a BS, broadcast with its least weight acceptable language l_q , and then remove all clients accepting l_q . Repeat this greedy step until no client is left without getting a copy of the content.

Finally, they evaluated the performance of their designs by numerical experiments. The ILP was solved by CPLEX 7.0 for small-scale network, while the greedy algorithm was tested by simulation for large-scale network, with 5 base stations, 100,000 clients, and 50 languages. The results proved the efficient of their design. Their heuristic saves more than 50 % of the energy consumed by traditional broadcasting scheme without any optimizations.

6.4 Coverage Prediction and Optimization for Digital Broadcast

Digital terrestrial broadcasting services (e.g., DVB-T and DAB) are proving a big success in Europe, as improved service quality, low setup costs, and increased content offered by the broadcasters attract more viewers and listeners to these new platforms. In order to reach their target subscriber levels, digital operators not only face the challenge of making their content attractive but also need to know more accurately the quality of terrestrial coverage they can provide to their potential customers. DVB/DAB requires higher prediction accuracy than traditional analogue networks because digital services are planned with tighter margins on the signal strength and interference. This subsection will introduce new prediction models that offer higher accuracy and which can be used to optimize digital broadcasting networks [12].

Accordingly, broadcasters for DVB/DAB services need optimize their new digital networks to provide the following:

- Accurate modeling of the impact of new digital services on analogue availability.
- Prediction of the coverage that can be provided by a new digital network.

- Selection of channels to best accommodate digital services.
- Creation of efficient and effective networks to achieve migration from analogue.
- Optimization of digital networks to maximize the number of customers.

To meet these requirements, the authors provided a high-accuracy coverage prediction models to estimate the maximum coverage of their service, and optimize the following parameters for their network to deliver the best possible coverage.

- Selection of site locations.
- Antenna heights.
- Antenna types and orientations.
- Channel selection.

This model can provide higher accuracy than traditional semiempirical models, yet can be computed in a reasonable timescale with a lower requirement on data accuracy than full 3D deterministic solvers. It relies on three important elements:

1. *High-resolution data*, representing the locations and heights of terrain and buildings in built-up areas, and medium-resolution data for more open, rural areas. They are usually collected by *stereo aerial photography*, *laser interferometry*, or *satellite collection*. The authors for this model require only a raster of terrain heights, avoiding the need to process the data into a vector form that represents the full geometry of the buildings.
2. *Propagation prediction algorithms*, to account for the improved data. The main mechanism for predicting the attenuation of signals across obstructions such as buildings and hills is multiple diffraction. The authors use *slope diffraction* to produce solutions that are as accurate as much more complex techniques in short run times. If radio wave propagation occurs over a clearly defined path (the great-circle path) between the transmitter and receiver, then it is necessary to account for the finite width of buildings and the possibility of off-path propagation around building edges.
3. *Optimized model*, to be implemented efficiently in order to make the most of available computing resources. It also includes (a) the extraction of suitable path profiles to suit the requirements of the model (b) post processing of the prediction results to include the effects of antenna radiation patterns, and (c) computation of the availability, based on the relevant planning margins.

The authors also provided example predictions on behalf of a digital broadcaster. The measurements covered a large area consisting of 6 km^2 , served by a transmitter approximately 30 km away. Field-strength measurements on four channels were made at a height of 10 m using a vehicle equipped with a pump-up mast. An average of 20 measurements per square kilometer were made, resulting in a total set of 468 measurements. In order to assist with the prediction process, the authors have also developed a unique automated network planning and optimization service, known as *SmartPlan*, and have used it successfully in real networks. Their results prove that this model has consistently performed well and has proved to be very effective in (a) reducing the standard deviation of the prediction error, and (b) significantly increasing the hit rate. It has already been used in real situations, either to address optimization issues in existing networks or to choose the optimum parameters for a new network.

6.5 Data Delivery Optimization in Multi-system Heterogeneous Overlayed Wireless Network

This subsection will employ data broadcast on multisystem heterogeneous overlayed wireless networks, due to the development of 4G networks [30]. The authors provided detailed two-step construction for data broadcast system and proved the efficiency of their algorithms by simulations. The system model can be described as follows:

Consider a multisystem heterogeneous overlayed network $N = \{N_1, N_2, \dots, N_{|N|}\}$ consisting of $|N|$ subnetworks, and suppose these subnetworks are ordered by the sizes of their coverage areas in ascending order. Each N_i allocates C_i channels to provide the data broadcast service, and the bandwidth of each channel is B_i . The data program contains $D = \{D_1, D_2, \dots, D_{|D|}\}$ data items. Each D_i has data size s_i and access probability p_i . For instance, Fig. 30a is a general system model of multisystem heterogeneous overlayed networks, while Fig. 30b is a practical example with regional-area subnetwork, metropolitan-area subnetwork, and campus-area subnetwork.

According to the phenomenon that wireless networks with larger coverage areas are usually of higher connection fee and of lower bandwidth than those with smaller coverage areas, the authors want to build a communication system with lowest cost such that all the subscribers can receive the required messages with minimum access time within the corresponding coverage area.

The main idea for data broadcast system in this type of wireless network is as follows: they divided broadcast data program into several subsets D_{cut_i} for subnetwork N_i , such that the largest service area will broadcast the whole data set, while other subnetworks will be viewed as supplements to reduce the minimum overall access time. Each subnetwork will only broadcast a subset of D correspondingly. As a consequence, the problem of broadcast program generation on heterogeneous overlayed wireless networks can be formulated as follows:

Definition 10 (Data broadcast in multi-coverage network) Given a multisystem heterogeneous wireless network $N = \{N_1, N_2, \dots, N_{|N|}\}$, the number of allocated channels C_i in each subnetwork N_i , the number of data items, and the access probabilities and sizes of all data items, for each subnetwork N_i , determine:

1. Which data items will be broadcast in subnetwork N_i (i.e., a proper cutting configuration).
2. How these data items are broadcast in subnetwork N_i (i.e., one proper broadcast program for each subnetwork),

such that the overall average access time for subscribers within the service area will be minimized.

To solve this problem, the authors firstly designed an algorithm named *bruteforce* to enumerate and select the best cutting configuration and the corresponding broadcast programs for all subnetworks. When evaluating one cutting configuration, they generate one broadcast program for each subnetwork by executing algorithm

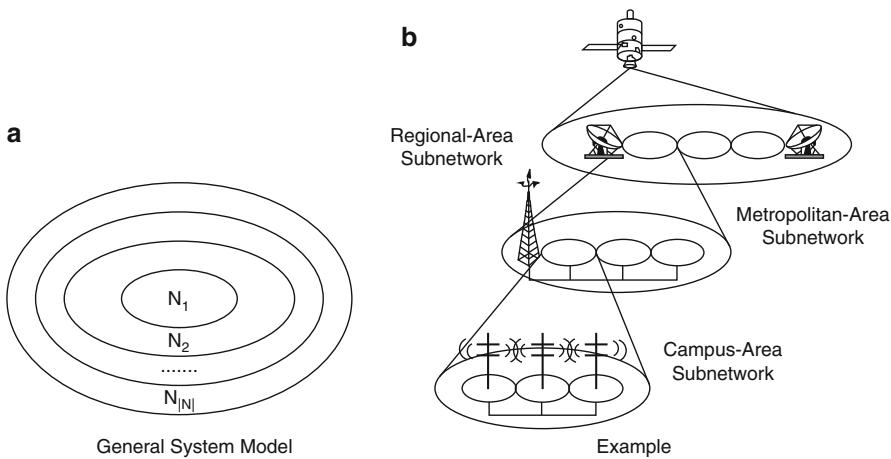


Fig. 30 Multisystem heterogeneous wireless networks

BPG-Single (a single-system network algorithm for data allocation) once so that the average access time of the subnetwork is minimized. Due to the high time complexity of this exhausted search, they next provided a heuristic algorithm named *layeredcutting* to determine a suboptimal cutting configuration and the corresponding broadcast programs for all subnetworks.

Layeredcutting is a two-phase algorithm consisting of inter-network data allocation phase and intra-network data allocation phase. The objective of inter-network data allocation phase is to determine a proper cutting configuration so that the overall average access time of the whole multisystem network is minimized. Then, in intra-network data allocation phase, layeredcutting will generate one broadcast program for each subnetwork according to the resultant cutting configuration. It is an iterative algorithm which merges several subnetworks and determines the value of one cutting point in each iteration. The threshold is the probability estimation of each subnetwork for each data item.

In each iteration, inter-network-data-allocation uses a divide-and-conquer procedure named *test-and-prune* to determine cut_i , and then returns all cut_i group, while intra-network-data-allocation applies execute algorithm BPG-Single on subnetwork N_j and returns broadcast programs of all subnetworks, which become the final broadcast program.

The authors proved that the time complexity of layered-cutting is $O(|N| \times (\log |D| + O(\text{BPG}_{\text{Single}})))$. They also illustrated their heuristics by simulation with 1,000 data items, following a Zipf distribution with parameter θ . There are $|N|$ subnetworks, and the rate between the sizes of the service areas of subnetwork N_i and subnetwork N_{i+1} is equal to 0.8. Finally, according to their simulation results, algorithm layered-cutting is able to efficiently generate broadcast programs of high quality for a multisystem heterogeneous overlayed wireless network and cover the whole service area.

6.6 Summary

To summarize, this section describes the coverage optimization for data broadcast in wireless communications. Coverage problems are introduced in different types of networks like wireless ZigBee networks, wireless/mobile ad hoc networks, cellular networks, and heterogeneous overlayed wireless networks. Firstly, the formal definition of coverage optimization is presented, including the prerequisite factors, the classification, relation among various types of coverage problems, and many possible variants. The researches of coverage optimization belong to wireless networking field. These problems can be reduced to some known NP problem and cannot be easily solved in polynomial time unless $P = NP$. Next, several coverage problems are discussed in details, including the problem formulation, the construction of approximation or heuristic algorithms, and the performance of the solutions.

Future study of coverage optimization could get benefit from the latest technologies, especially the new ideas and technologies in the 4G wireless networks. Moreover, researchers can combine knowledge from different disciplines and think about the requirements from the server side and client side wisely. Even for the same number of BS's with the same capacity, different clients/service providers may have different requirements about the coverage issues. Clients always want to receive their requests quickly and correctly, while servers always hope to provide a reliable and efficient service. Therefore, the coverage problem is always a critical issue for data broadcast system, which makes coverage optimization an active and interesting research topic in both theoretical and practical fields.

7 Conclusion

This chapter discusses various optimization problems in all aspects of data broadcasting. Scheduling is a big topic in data broadcast. In order to reduce response time, push-based broadcast systems need optimized off-line data scheduling (or allocation) algorithms to organize broadcast programs; on-demand broadcast systems need optimized and efficient online scheduling algorithms to decide broadcast contents in real time, while clients may also need scheduling algorithms to decide the downloading sequence of multiple data items. Energy optimization in wireless data broadcast systems relies on good index schemes. Index design should be customized based on different features of broadcast data. Coverage optimization is also an important research area of data broadcast.

Within this big picture, it also introduces different techniques during algorithm design to handle different features of data, communication environment, and user type. For example, data can be homogeneous or heterogeneous; broadcast can be on single channel or on multiple channels; request can be for single data item or multiple data items; and clients may have deadlines on responses to their requests.

Data broadcasting is a mature topic with a new life. Existing research works on data broadcast establish a solid foundation for further research in this area.

On the other hand, with the fast development of mobile communication techniques, new problems are keeping emerging waiting to be solved by future researchers. Hope this chapter can be a good guide for them to better understand the area and an inspiration for new questions and research topics.

Recommended Reading

1. S. Acharya, R. Alonso, M. Franklin, S. Zdonik, Broadcast disks: data management for asymmetric communication environments, in *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, San Jose, 22–25 May 1995, pp. 199–210 (1995)
2. S. Acharya, M.J. Franklin, S. Zdonik, Dissemination-based data delivery using broadcast disks. *IEEE Pers. Comm.* **2**(6), 50–60 (1995)
3. S. Acharya, S. Muthukrishnan, Scheduling on-demand broadcasts: new metrics and algorithms, in *MOBICOM'98*, Dallas, 25–30 Oct 1998, pp. 43–54
4. D. Aksoy, M. Franklin, Scheduling for large-scale on-demand data broadcasting, in *INFO-COM'98*, San Francisco, 29 Mar–2 Apr 1998, pp. 651–659
5. D. Aksoy, M. Franklin, $R \times W$: a scheduling approach for large-scale on-demand data broadcasting. *IEEE/ACM Trans. Netw.* **7**(6), 846–860 (1999)
6. D. Amarmend, M. Aritsugi, Y. Kanamori, An air index for data access over multiple wireless broadcast channels, in *International Conference on Data Engineering (ICDE'06)*, Atlanta, 3–8 Apr 2006, pp. 135
7. M.H. Ammar, J.W. Wong, The design of teletext broadcast cycles. *Perform. Eval.* **5**(4), 235–242 (1985)
8. M.H. Ammar, J.W. Wong, On the optimality of cyclic transmission in teletext systems. *IEEE Trans. Comm.* **35**(11), 1159–1170 (1987)
9. S. Anily, C.A. Glass, R. Hassin, The scheduling of maintenance service. *Discr. Appl. Math.* **82**, 27–42 (1998)
10. E. Ardizzone, A.A. Bertossi, M.C. Pinotti, S. Ramaprasad, R. Rizzi, M.V.S. Shashanka, Optimal skewed data allocation on multiple channels with flat broadcast per channel. *IEEE Trans. Comput.* **54**(5), 558–572 (2005)
11. A. Barnoy, R. Bhattacharyya, B. Schieber, Minimizing service and operation costs of periodic scheduling, in *The 1998 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, San Francisco, 25–27 Jan 1998, pp. 11–20
12. B. Belloul, S. Saunders, Cellular design services SPECTRUM PLANNING coverage accurate. *EBU Technical Review* (online), April 2004
13. W.T. Chan, T.W. Lam, H.F. Ting, W.H. Wong, New results on on-demand broadcasting with deadline via job scheduling with cancellation, in *Proceedings of the 10th Annual International Conference on Computing and Combinatorics*, Jeju Island, 17–20 Aug 2004, pp. 210–218
14. R.S. Chernock, R.J. Crinon, M.A. Dolan, J.R. Mick Jr., *Data Broadcasting: Understanding the ATSC Data Broadcast Standard* (McGraw-Hill Professional, New York, 2001)
15. C.-C. Chen, C. Lee, S.-C. Wang, On optimal scheduling for time-constrained services in multi-channel data dissemination systems. *Inform. Syst.* **34**(1), 164–177 (2009)
16. M.-S. Chen, K.-L. Wu, P.S. Yu, Optimizing index allocation for sequential data broadcasting in wireless mobile computing. *IEEE Trans. Knowledge Data Eng.* **15**(1), 161–173 (2003)
17. M.-S. Chen, P.S. Yu, K.-L. Wu, Indexed sequential data broadcasting in wireless mobile computing, in *ICDCS'97*, Baltimore, 27–30 May 1997, pp. 123–131
18. C.-H. Chu, H.-P. Hung, M.-S. Chen, A general framework of time-variant bandwidth allocation in the data broadcasting environment. *IEEE Trans. Knowledge Data Eng.* **22**(3), 318–333 (2010)
19. Y.D. Chung, M.H. Kim, Effective data placement for wireless broadcast. *Distrib. Parallel Databases* **9**(2), 133–150 (2001)

20. G. Ding, Z. Sahinoglu, P. Orlik, J. Zhang, B. Bhargava, Tree-based data broadcast in IEEE 802.15.4 and ZigBee networks. *IEEE Trans. Mobile Comput.* **5**(11), 1561–1574 (2006)
21. H.D. Dykeman, M. Ammar, J.W. Wong, Scheduling algorithms for videotex systems under broadcast delivery. *IEEE Int. Conf. Comm.* 1847–1851 (1986)
22. R. Gandhi, Y. Kim, S. Lee, J. Ryu, P.-J. Wan, Approximation algorithms for data broadcast in wireless networks, in *The 28th IEEE International Conference on Computer Communications (INFOCOM'09)*, Rio de Janeiro, 19–25 Apr 2009, pp. 2681–2685
23. M.R. Garey, D.S. Johnson, *Computers and Intractability; a Guide to the Theory of NP-Completeness* (W.H. Freeman & Co., New York, 1990)
24. C. Hsu, G. Lee, A. Chen, Index and data allocation on multiple broadcast channels considering data access frequencies, in *MDM'02*, Singapore, 8–11 Jan 2002, pp. 87–93
25. Q. Hu, W. Lee, D. Lee, A hybrid index technique for power efficient data broadcast. *Distrib. Parallel Databases* **9**(2), 151–177 (2001)
26. T. Hu, A. Tucker, Optimal computer search trees and variable-length alphabetic codes. *SIAM J. Appl. Math.* **21**(4), 514–532 (1971)
27. J.L. Huang, M.S. Chen, W.C. Peng, Broadcasting dependent data for ordered queries without replication in a multi-channel mobile environment, in *The 2003 International Conference on Data Engineering*, Bangalore, 5–8 Mar 2003, pp. 692–694
28. J.L. Huang, M.S. Chen, Dependent data broadcasting for unordered queries in a multiple channel mobile environment. *IEEE Trans. Knowledge Data Eng.* **16**(9), 1143–1156 (2004)
29. J.-L. Huang, J.-N. Lin, Data broadcast on a multi-system heterogeneous overlayed wireless network, in *International Conference on Parallel and Distributed Computing Applications and Technologies*, Taipei, 4–7 Dec 2006, pp. 358–363
30. J.-L. Huang, J.-N. Lin, Data broadcast on a multi-system heterogeneous overlayed wireless network. *J. Inform. Sci. Eng.* **24**, 819–840 (2008)
31. H.P. Hung, J.W. Huang, J.L. Huang, M.S. Chen, Scheduling dependent items in data broadcasting environments, in *The 2006 ACM Symposium on Applied Computing*, Dijon, 23–27 Apr 2006, pp. 23–27
32. A.R. Hurson, Y.C. Chehadeh, J. Hannan, Object organization on parallel broadcast channels in a global information sharing environment, in *The 2000 IEEE International Performance Computing and Communications Conference (IPCCC'00)*, Phoenix, 20–22 Feb 2000, pp. 347–353
33. A.R. Hurson, A.M. Muñoz-Avila, N. Orchowski, B. Shirazi, Y. Jiao, Power-aware data retrieval protocols for indexed broadcast parallel channels. *Pervasive Mobile Comput.* **2**(1), 85–107 (2006)
34. T. Imielinski, S. Viswanathan, B. Badrinath, Power efficient filtering of data on air, in *EDBT 1994*, Cambridge, 28–31 Mar 1994, pp. 245–258
35. T. Imielinski, S. Viswanathan, B. Badrinath, Data on air: organization and access. *IEEE TKDE* **9**(3), 353–372 (1997)
36. A. Itai, S. Rosenberg, A golden ratio control policy for a multiple-access channel. *IEEE Trans. Autom. Contr.* **29**(8), 712–718 (1984)
37. S. Jiang, N.H. Vaidya, Response time in data broadcast systems: mean, variance and tradeoff. *Mobile Netw. Appl.* **7**(1), 37–47 (2002)
38. S. Jung, B. Lee, S. Pramanik, A tree-structured index allocation method with replication over multiple broadcast channels in wireless environments. *IEEE Trans. Knowledge Data Eng.* **17**(3), 311–325 (2005)
39. B. Kalyanasundaram, M. Velauthapillai, On-demand broadcasting under deadline, in *Proceedings of the 11th Annual European Symposium on Algorithms*, Budapest, 15–20 Sep 2003, pp. 313–324
40. J.H. Kim, K.Y. Chwa, Scheduling broadcasts with deadlines. *Theor. Comput. Sci.* **325**(3), 479–488 (2004)
41. C. Kenyon, N. Schabanel, The data broadcast problem with non-uniform transmission time, in *The 1999 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Baltimore, 17–19 Jan 1999, pp. 547–556

42. M.V. Lawrence, L.S. Brakmo, W.R. Hamburgen, Energy management on handheld devices. *ACM Queue* **1**, 44–52 (2003)
43. H. Lee, G. Hong, H. Song, S. Han, Coding and presentation of multimedia for data broadcasting with broadcasting markup language. *Databases Netw. Inform. Syst. Lect. Note Comput. Sci.* **2544**, 147–160 (2002)
44. G. Lee, M.S. Yeh, S.C. Lo, A. Chen, A strategy for efficient access of multiple data items in mobile environments, in *The 2002 International Conference on Mobile Data Management (MDM'02)*, Singapore, 8–11 Jan 2002, pp. 71–78
45. W. Lee, D. Lee, Using signature techniques for information filtering in wireless and mobile environments. *Distrib. Parallel Databases* **4**(3), 205–227 (1996)
46. S.-C. Lo, A. Chen, Optimal index and data allocation in multiple broadcast channels, in *Proceedings of the 16th International Conference on Data Engineering*, San Diego, 28 Feb–3 Mar 2000, pp. 293–302
47. W. Mao, Competitive analysis of on-line algorithms for on-demand data broadcast scheduling, in *Proceedings of the 2000 International Symposium on Parallel Architectures, Algorithms and Networks*, Dallas, 7–10 Dec 2000, pp. 292–296
48. K. Prabhakara, K.A. Hua, J. Oh, Multi-level multi-channel air cache designs for broadcasting in a mobile environment, in *The 2000 International Conference on Data Engineering*, San Diego, 28 Feb–3 Mar 2000, pp. 167–176
49. N. Saxena, M.C. Pinotti, On-line balanced k-channel data allocation with hybrid schedule per channel, in *Proceedings of the 6th International Conference on Mobile Data Management (MDM)*, New York, 2005, pp. 239–246
50. Y. Shi, X. Gao, J. Zhong, W. Wu, Efficient parallel data retrieval protocols with MIMO antennae for data broadcast in 4G wireless communications, in *DEXA 2010*, Bilbao, 30 Aug–3 Sep 2010, pp. 80–95
51. N. Shivakumar, S. Venkatasubramanian, Efficient indexing for broadcast based wireless systems. *Mobile Netw. Appl.* **1**(4), 433–446 (1996)
52. N. Shivakumar, S. Venkatasubramanian, Energy-efficient indexing for information dissemination in wireless systems. *ACM Baltzer Mob. Netw. Appl.* (1995)
53. T. Sung, T. Wu, C. Yang, Y. Huang, Reliable data broadcast for Zigbee wireless sensor networks. *Int. J. Smart Sens. Intell. Syst.* **3**(5), 504–520 (2010)
54. K. Takenaga, Data broadcasting summary, Technical Subgroup of the Expert Group on International Economic and Social Classifications. New York, Feb 14–18, 2005.
55. H.-F. Ting, A near optimal scheduler for on-demand data broadcasts. *Theor. Comput. Sci.* **401**, 77–84 (2008)
56. N.H. Vaidya, S. Hameed, Scheduling data broadcast in asymmetric communication environments. *Wireless Netw.* **5**, 171–182 (1996)
57. N. Vaidya, S. Hameed, Log time algorithms for scheduling single and multiple channel data broadcast, in *The 1997 ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM)*, Budapest, 26–30 Sep 1997, pp. 90–99
58. N. Vaidya, S. Hameed, Efficient algorithms for scheduling single and multiple channel data broadcast. Technical Report 97-002, Texas A&M University, 1997
59. M. Vijayalakshmi, A. Kannan, A hashing scheme for multi-channel wireless broadcast. *J. Comput. Inform. Technol. CIT* **16**(3), 197–207 (2008)
60. A.B. Waluyo, B. Srinivasan, D. Taniar, Global indexing scheme for location-dependent queries in multi channels mobile broadcast environment. *Int. Conf. Adv. Inform. Netw. Appl.* **1**, 1011–1016 (2005)
61. S. Wang, H. Chen, TMBT: an efficient index allocation method for multi-channel data broadcast, in *AINAW'07*, Niagara Falls, 21–23 May 2007
62. J.W. Wong, Broadcast delivery. *Proc. IEEE* **76**, 1566–1577 (1988)
63. X. Wu, V. Lee, Wireless real-time on-demand data broadcast scheduling with dual deadlines. *J. Parallel Distrib. Comput.* **65**, 714–728 (2005)

64. H. Xu, J. Garcia-Luna-Aceves, Efficient broadcast in wireless ad hoc networks with a realistic physical layer, in *IEEE Global Telecommunications Conference (GlobeCom'08)*, New Orleans, 30 Nov–4 Dec 2008, pp. 683–687
65. J. Xu, W. Lee, X. Tang, Exponential index: a parameterized distributed indexing scheme for data on air, in *MobiSYS'04*, Boston, 2004
66. J. Xu, X. Tang, W.-C. Lee, Time-critical on-demand data broadcast: algorithms, analysis, and performance evaluation. *IEEE Trans. Parallel Distrib. Syst.* **17**(1), 3–14 (2006)
67. J. Xu, W. Lee, X. Tang, Q. Gao, S. Li, An error-resilient and tunable distributed indexing scheme for wireless data broadcast. *IEEE TKDE* **18**(3), 392–404 (2006)
68. Y. Yao, X. Tang, E. Lim, A. Sun, An energy-efficient and access latency optimized indexing scheme for wireless data broadcast. *IEEE TKDE* **18**(8), 1111–1124 (2006)
69. W.G. Yee, S.B. Navathe, E. Omiecinski, C. Jermaine, Efficient data allocation over multiple channels at broadcast servers. *IEEE Trans. Comput.* **51**(10), 1231–1236 (2002)
70. W.G. Yee, S.B. Navathe, Efficient data access to multi-channel broadcast programs, in *CIKM'03: Proceedings of the Twelfth International Conference on Information and Knowledge Management*, New York, 2003, pp. 153–160
71. P. Yu, W. Sun, B. Shi, A parameterized flexible indexing scheme for data broadcast in wireless mobile environment, in *Proceedings of The Sixth IEEE International Conference on Computer and Information Technology (CIT'06)*, Seoul, 20–22 Sep 2006, pp. 85
72. Z. Zhang, X. Gao, X. Zhang, W. Wu, H. Xiong, Three approximation algorithms for energy-efficient query dissemination in sensor database system, in *The 20th International Conference on Database and Expert Systems Applications (DEXA'09)*, Linz, 31 Aug–4 Sep 2009, pp. 807–821
73. B. Zheng, X. Wu, X. Jin, D.L. Lee, Tosa: a near-optimal scheduling algorithm for multi-channel data broadcast, in *The 2005 International Conference on Mobile Data Management*, Ayia Napa, May 9–13 2005, pp. 29–37
74. J. Zhong, W. Wu, Y. Shi, X. Gao, Energy-efficient tree-based indexing scheme for efficient retrieval under mobile wireless data broadcasting environment, in *Database Systems for Advanced Applications (DASFAA)*, Hong Kong, 22–25 Apr 2011, pp. 335–351 2011
75. Y. Zhu, X. Gao, J. Willson, C. Ma, J. Jue, W. Wu, Improving cell broadcasting scheme to support multi-lingual service in wireless networks. *IEEE Comm. Lett.* **13**(9), 634–636 (2009)

Optimization Problems in Online Social Networks

Jie Wang, You Li, Jun-Hong Cui, Benyuan Liu and Guanling Chen

Contents

1	Introduction	2456
2	Influence Spread	2459
2.1	Formulation	2459
2.2	NP-completeness	2459
2.3	#P-Hardness	2460
2.4	Approximations	2463
2.5	Other Influence Models	2466
3	Finding Communities	2468
3.1	Graph-Partitioning-Based Algorithms	2468
3.2	Hierarchical Clustering-Based Algorithms	2469
3.3	The Random Walker Algorithm	2471
4	Data Collection from OSNs	2473
4.1	Measuring and Analyzing OSNs	2473
4.2	Sampling Methods of OSNs	2474
4.3	MySpace Data Collection and Analysis	2477
4.4	Member Profile Patterns	2478
4.5	Temporal Usage Patterns	2482
4.6	Blog Content Patterns	2482

J. Wang (✉) • Y. Li • B. Liu • G. Chen

Department of Computer Science, University of Massachusetts, Lowell, MA, USA
e-mail: wang@cs.uml.edu; yli1@cs.uml.edu; bliu@cs.uml.edu; gchen@cs.uml.edu

J.-H. Cui

Department of Computer Science and Engineering, University of Connecticut, Storrs, CT, USA
e-mail: jcui@engr.uconn.edu

5 Multilayer Friendship Modeling in Location-Based OSNs.....	2484
5.1 A Multilayer Friendship Model.....	2485
5.2 Friendship Ranking.....	2495
5.3 Evaluation.....	2495
6 Conclusion.....	2499
Cross-References	2500
Recommended Reading.....	2500

Abstract

This chapter introduces and elaborates four major issues in online social networks. First, it describes the problem of maximizing the spread of influence in social networks and the problem of computing the spread. It shows that the problem of maximizing influence is NP-hard and the problem of computing the spread is #P-hard. It also presents a number of algorithms for finding approximations to the problem of influence maximization. Second, it describes the problem of detecting network communities and presents a few heuristic algorithms. In particular, it introduces and elaborates a new algorithm that uses random walkers to find communities effectively and efficiently. Third, it discusses how to collect data from online social networks and it presents the recent work on data collection. Forth, it discusses how to measure influence (friendship) in online social networks and elaborates the recent results on the location-based three-layer friendship modeling.

1 Introduction

Influence maximization (IM) is a combinatorial optimization problem in social sciences with important applications. Kempe et al. [20] formulated and studied IM as a discrete optimization problem and subsequently published additional results in [21]. Since then, a number of researchers have investigated the problem from different angles, including Kumura and Saito [23], Leskovec et al. [26], Narayanan and Narahari [32], and Chen et al. [6–8].

Domingos and Richardson [10] and Richardson and Domingos [38] were among the first to study the social influence problem from the algorithmic standpoint in connection to viral marketing. Suppose a company wants to promote a new product to a large group of people. To do so, the company plans to offer free samples to a small group of selected individuals who are more influential than the others, with the hope that these key individuals will help spread the information of the product to the largest number of individuals through their influence. How to find such key individuals motivated the study of the problem of influence maximization, which finds a small set of most influential individuals.

Social networking in the cyber space has become increasingly popular in recent years. Online social networks (OSNs), such as Facebook and MySpace, provide

a convenient platform for users to interact with each other through the Internet. Studying IM in the setting of OSNs may help reveal how social activities would take place in the electronic world. On the other hand, conducting research on OSNs has an added advantage: it is easier to collect data from OSNs than physical social networks. For example, one could either develop a crawler to collect information by crawling through the targeted OSN [11, 12] or simply use data dumps or the Application Programming Interface provided by the service provider. This chapter is focused on OSNs, but some of the results may also apply to general social networks.

To measure influence of one individual to another, an immediate solution is to determine how close the relationship would be between them. In OSNs, a user can declare other users as friends, even though they may not know each other personally. This chapter follows the notion of online friendships. Thus, one way to measure how influential an individual would be is to determine how many friends they have. The individual with the largest number of friends may be considered as the most influential individual. On the other hand, this individual may not have the most influence on a specific friend. For example, Alice is considered as the most influential individual because she has the largest number of friends, including Bob. But Bob is only a casual friend of Alice. Bob, on the other hand, has a close long-time friend Christie whom he trusts. Thus, Christie would be more influential to Bob than Alice. Simply going through the data collected from OSNs, however, it is difficult to differentiate such level of friendships, and so most studies still rely on the loose concept of online friendships adopted by OSNs.

Related to the problem of influence maximization is the problem of finding network communities. A network community is a collection of nodes within which connections are dense, while connections between communities are sparse. Finding communities in online social networks, biological networks, metabolic networks, and other large-scale networks has important applications, and it has been studied intensively and extensively in computer science, mathematics, biology, and other disciplines. Intuitively, network communities are formed as a subset of nodes with similar characteristics. In an OSN, a community may form from users with similar hobbies or interests. For example, users who like classical music may form a community; users who are interested in iPhone, iPad, and other Apple products may form another community. If a company wants to advertise a new product to as many people as possible effectively and efficiently, it would naturally want to choose communities with a keen interest in similar products. This chapter describes a number of heuristic algorithms for finding network communities, including a new algorithm that uses random walkers to search for communities. This algorithm was recently devised by Li et al. [28], which has only been reported in a technical report.

Studying how users are connected through OSNs would help to measure influence of OSN users. Intuitively, people sharing mutual interests, being geographically close to each other, or belonging to the same social communities are more likely to become friends. This chapter elaborates multilayer metrics recently devised by Li and Chen [27] to quantitatively model these intuitive friendships.

Social networks are often modeled as a weighted directed graph $G(V, E)$, where nodes represent individuals and edges represent relationships and interactions between individuals. To study social influence, each edge $u \rightarrow v$ is associated with a weight $p(u, v) \in [0, 1]$ that quantifies the level of influence on v by u . Such graphs are referred to as *influence graphs*. Node v is said to be node u 's neighbor if $u \rightarrow v \in E$.

Influence maximization seeks a small set of nodes that will influence the maximum number of nodes in the given influence graph. Kempe et al. [20] introduced two stochastic cascade models, also known as diffusion models, for studying the problem of influence maximization. They are *independent cascade* (IC) and *linear threshold* (LT), where each node in the graph is either active or inactive. Initially, both models specify an initial set of active nodes, referred to as *seed set*, while the rest of the nodes are inactive. Once a node is made active, meaning that it has been successfully influenced by its active neighbors, it will remain active throughout the entire process. The process of both models proceeds at discrete steps. Discrete steps are also referred to as rounds or iterations.

In the IC model, each active node u is only given one chance to activate each of its currently inactive neighbor v with the success probability $p(u, v)$, where $u \rightarrow v \in E$. This formulation is motivated by Goldenberg et al.'s work on viral marketing [16, 17]. If v has multiple active neighbors, their attempts to activate v are sequenced in an arbitrary order. In particular, if u is newly activated in round i , $u \rightarrow v \in E$, v is inactive, and u is given a chance to activate v in round $i + 1$, then v is activated by u with probability $p(u, v)$. If u succeeds, then v becomes active at time $i + 1$. If u fails to activate v , then u cannot activate v in subsequent rounds. The process continues until no more activations are possible.

In the LT model, each node $v \in V$ satisfies the following inequality:

$$\sum_{u \rightarrow v \in E} p(u, v) \leq 1.$$

At the beginning, each node v chooses a threshold value $\lambda_v \in [0, 1]$ uniformly and independently at random. An inactive node v becomes *active* in round i if

$$\sum_{\substack{u \rightarrow v \\ u \text{ is active}}} p(u, v) \geq \lambda_v.$$

The process continues until no more activations are possible. As noted by Kempe et al. [20], selecting thresholds at random is the only reasonable choice when the exact influence to v from its neighbors is unknown from the given data set.

This chapter is organized as follows. Section 2 defines the IM problem, shows that it is NP-hard for the independent cascade and linear threshold models, shows that the influence spread of an initial seed set is #P-hard for both models, and provides algorithms to approximate the IM problem. It also introduces other influence models. Section 3 introduces a number of heuristic algorithms for finding network communities. Section 4 describes how to collect data from online social networks. Section 5 discusses how to determine friendships. Section 6 concludes the chapter.

2 Influence Spread

Let S be an initial set of active nodes in G and $I_G(S)$ the expected number of active nodes at the end of the process in an influence model (IC or LT). Function I_G is referred to as the *influence function* of S and the value of $I_G(S)$ the *influence spread* of S , or simply *spread*.

2.1 Formulation

The *influence maximization* problem can be formulated as follows:

Influence maximization (IM)

Input: An influence graph $G(V, E)$ and a positive integer k .

Output: An initial seed set S of k nodes such that $I_G(S)$ is maximized for all k -node initial seed set.

The decision version of IM is defined below:

Decisional influence maximization (DIM)

Input: An influence graph $G(V, E)$ and positive integers k and N .

Question: Does there exist an initial seed set S of k nodes such that $I_G(S) \geq N$?

It is straightforward to see that if IM is polynomial-time computable, then so is DIM. But DIM is NP-complete for both the IC and LT models. Thus, IM is NP-hard on both models.

Counting how many k -node seed sets S that will maximize $I_G(S)$ is at least as hard as solving IM. Moreover, even computing the value of $I_G(S)$ for a fixed seed set S is #P-hard for both the IC and LT models.

Given a seed set S , computing the exact value of $I_G(S)$ is referred to as the *influence spread computation (ISC)* problem, formulated as follows:

Influence spread computation (ISC)

Input: An influence graph $G(V, E)$ and a seed set $S \subseteq V$.

Output: $I_G(S)$.

2.2 NP-completeness

The following results were obtained by Kempe et al. [20].

Theorem 1 ([20]) *DIM is NP-complete for both of the IC model and the LT model.*

Proof First consider the IC model. The goal is to construct a polynomial-time reduction from a known NP-complete problem to DIM for the IC model. The set cover (SC) problem is a suitable NP-complete problem for this purpose. Given a collection of subsets S_1, \dots, S_n of set $U = \{u_1, \dots, u_m\}$, SC decides whether there

exist k subsets such that the union of these sets is equal to U . Without loss of generality, assume that $k < m < n$.

Given an instance of SC, construct a directed bipartite graph $(V_1, V_2; E)$ of $m + n$ nodes such that each subset S_i corresponds to a unique node $i \in V_1$ and each element v_j corresponds to a unique node $j \in V_2$, where i and j are connected by a directed edge $i \rightarrow j \in E$ with weight $p_{ij} = 1$ if $v_j \in S_i$. Because $p_{ij} = 1$, the activation of a node is deterministic. It is straightforward to see that the instance of SC is positive if and only if there is a k -node seed set $S \subseteq U_1$ such that $I_G(S) \geq m + k$.

Thus, SC is polynomial-time reducible to DIM.

Now consider the LT model. The goal is to construct a polynomial-time reduction from a known NP-complete problem to DIM for the LT mode. The vertex cover (VC) problem is a suitable NP-complete problem for this purpose. Given an n -node undirected graph G and an integer k , VC decides whether there is a k -node set C such that every edge has at least one endpoint in C . Converting G to a directed graph by directing every edge in G to both directions and setting $p(u, v) = 1$ for every edge $u \rightarrow v$, it is straightforward to see that there exists a k -node set C to cover G if and only if $I_G(A) = n$. Thus, VC is polynomial-time reducible to DIM. This completes the proof. \square

2.3 #P-Hardness

The following result was obtained by Chen et al. [7].

Theorem 2 ([7]) *ISC for the IC model is #P-hard.*

Proof It suffices to show that if ISC is solvable in polynomial time for the IC model, then so is a known #P-complete problem. The two-terminal-network-reliability problem is an appropriate problem, which was shown to be #P-complete by Valiant [40]. This will establish that ISC is #P-hard.

The two-terminal-network-reliability problem is to count, on a given directed graph $G = (V, E)$ and two nodes $\{s, t\} \subseteq V$, the number of subgraphs in which there is a path from s to t . This is equivalent to computing the probability that there is a path from s to t under the condition that each edge in E has an independent probability of 1/2 to be selected and an independent probability of 1/2 to be discarded. Let $S = \{s\}$ and $p(e) = 1/2$ for all $e \in E$. Compute $I_1 = I_G(S)$. Next, add a new node $t' \notin G$ and a directed edge $t \rightarrow t'$ to obtain a new graph $G' = (V \cup \{t'\}, E \cup \{t \rightarrow t'\})$. Let $p(t, t') = 1$. Compute $I_2 = I_{G'}(S)$. By assumption, both I_1 and I_2 are polynomial-time computable and so is $I_2 - I_1$. Let $p(S, v, G)$ denote the probability that node v is influenced by the seed set S in G . Since t influences t' deterministically, it is straightforward to see that $I_2 = I_G(S) + p(S, t, G)$. Therefore, $p(S, t, G) = I_2 - I_1$ is the probability that s is connected to t , and it is polynomial-time computable. This completes the proof. \square

To show that ISC for the LT model is also #P-hard, it helps to consider the distribution over sets reachable from S via certain types of edges. Recall that in the LT model, each node v receives an influence weight $p(u, v) \geq 0$ from node u for $u \rightarrow v \in E$. Suppose that v selects at most one such incoming edge at random with probability $p(u, v)$ and selects no edges with probability $1 - \sum_u p(u, v)$. The selected edge $u \rightarrow v$ in this way is referred to as a *live* edge and the other edges as *blocked* edges. It can be shown that the distribution over active sets (i.e., sets of active nodes) by running the LT process starting from S is equivalent to the distribution over sets of nodes reachable from S via live-edge paths. A live-edge path is a path where all edges on the path are live edges.

To motivate the proof idea, consider a simple case that G is an acyclic graph. In an acyclic graph, nodes can be listed in a fixed topological ordering: v_1, v_2, \dots, v_n , such that all edges are from nodes of smaller indexes to nodes of larger indexes. It suffices to look at the distribution over active sets following this order. For each node v_i , suppose that the distribution over active subsets of its neighbors is already determined. Following the LT process, the probability that v_i will become active, given that a subset S_i of its neighbors is active, is equal to

$$\sum_{u \rightarrow v_i \in E} p(u, v_i).$$

This is the probability that the live incoming edge selected by v_i lies in S_i , and so it is straightforward to show that the two processes produce the same distribution over active sets by a simple induction on nodes in this order. For general graphs G , however, such an ordering of the nodes may not exist. Instead, an induction will be carried out over the number of rounds of the LT process. The following lemma was shown by Kempe et al. [20].

Lemma 1 ([20]) *For a given initial seed set S in an influence graph G , the distribution over active sets from S under the LT process is the same as that over sets reachable from S via live-edge paths.*

Proof Let S_i be the set of active nodes at the end of round i under LT process ($i = 0, 1, 2, \dots$), where $S_0 = S$. If node v has not become active by the end of round i , then the probability that it becomes active at round $i + 1$ equals the probability that the total influence from $S_i - S_{i-1}$ exceeds the threshold, given that its threshold was not exceeded already. This probability is

$$P_{u,v} = \frac{\sum_{u \in S_i - S_{i-1} \text{ & } u \rightarrow v \in E} p(u, v)}{1 - \sum_{u \in S_{i-1} \text{ & } u \rightarrow v \in E} p(u, v)}.$$

The following procedure can be used to identify reachable sets from the initial seed set S : Starting from the set S , for each node v with at least one edge $u \rightarrow v \in E$ and $u \in S$, if one of these edges becomes live, then v is reachable from S , resulting

in a new set S'_1 of reachable nodes from S . Repeat the same procedure recursively on edges from S'_1 to obtain a new reachable set S'_2 , from S'_2 obtain S'_3 . If node v is not yet reachable by the end of round i , then the probability that it will become reachable in round $i + 1$ is equal to the probability that its live edge comes from $S_i - S_{i-1}$, given that its live edge has not come from any of the earlier sets, which is equal to $\sum_{u \in S_i - S_{i-1} \text{ & } u \rightarrow v \in E} p(u, v) / (1 - \sum_{u \in S_{i-1} \text{ & } u \rightarrow v \in E} p(u, v))$. This is the same as $P_{u,v}$. Thus, a straightforward induction over LT rounds can establish that the live-edge process produces the same distribution over active sets by the LT process. This completes the proof. \square

The following result was obtained by Chen et al. [8].

Theorem 3 ([8]) *ISC for the LT model is #P-hard.*

Proof It suffices to show that if $I_G(S)$ is polynomial-time computable, then so is the problem of counting simple paths (CSP) in a directed graph. CSP is to count, given a directed graph $G = (V, E)$, the total number of simple paths in G , which was shown to be #P-complete by Valiant [40].

Given an influence graph G , construct a new influence graph $G' = (V', E')$ by adding a new node $s \notin G$ and connecting s to all the nodes in V . That is, the node s is a source node. Let d be the maximum in-degree of any node in G' . Assign a weight of $p(e) = w \leq 1/d$ to each edge $e \in E'$, where w is a constant, and it will be determined later. Clearly, for any node $v \in V'$,

$$\sum_{u \rightarrow v \in E'} p(u, v) \leq 1,$$

and so the weight requirement of the LT model is satisfied. Let the initial seed set be $S = \{s\}$. By assumption, $I_{G'}(S)$ is polynomial-time computable.

Let \mathcal{P} denote the set of all simple paths starting from $s \in G'$. It follows from Lemma 1 that

$$I_{G'}(S) = \sum_{\pi \in \mathcal{P}} \prod_{e \in \pi} p(e).$$

Let B_i be the set of simple paths of length i in \mathcal{P} , for $i = 0, 1, \dots, n$, and let $b_i = |B_i|$. Then

$$I_{G'}(S) = \sum_{i=0}^n \sum_{\pi \in B_i} \prod_{e \in \pi} p(e) = \sum_{i=0}^n \sum_{\pi \in B_i} w^i = \sum_{i=0}^n w^i b_i.$$

With the above equation, set w to $n + 1$ different values w_0, w_1, \dots, w_n , where $n = |V|$. For each value w_i , by assumption, $I_{G'}(S)$ is polynomial-time computable. Note that this process produces a set of $n + 1$ linear equations with b_0, \dots, b_n variables. The coefficient matrix M of these equations is a Vandermonde matrix

with $M_{ij} = w_i^j$ for $i, j = 0, \dots, n$. The system of these equations therefore has a unique solution and is easy to compute.

Finally, notice that for each $i = 1, \dots, n$, there is a one-to-one correspondence between paths in B_i and simple paths of length $i - 1$ in graph G . Therefore, $\sum_{i=1}^n b_i$ gives the answer to the total number of simple paths in graph G . This completes the proof. \square

2.4 Approximations

Approximation algorithms for the IM problem are based on the observation that the influence function I_G for both the IC and LT models is a *submodular* function.

2.4.1 Submodularity

A submodular function f is a function that maps a subset of a finite ground set U to a nonnegative real number satisfying the following inequality:

$$(\forall A \subseteq B \subseteq U)(\forall x \in U - B) : f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B).$$

A function f is *monotone* if for all $A \subseteq B : f(B) \geq f(A)$.
Submodular greedy algorithm (SGA) [9]:

1. Given a function $f : 2^U \rightarrow R^+ \cup \{0\}$.
2. Initially set $S \leftarrow \emptyset$.
3. Repeat the following for k times:
 - Choose $u \in U - S$ such that $f(S \cup \{u\})$ is the largest among all possible u 's.
 - Set $S \leftarrow S \cup \{u\}$.
4. Return S .

The following result was shown by Nemhauser et al. [33] and is well known. Its proof is omitted.

Lemma 2 ([33]) *Let f be a monotone submodular function. Let S be a k -element set obtained from SGA. Let S^* be a set that maximizes the value of f over all k -element sets. Then $f(S) \geq (1 - 1/e)f(S^*)$. That is, S is a $(1 - 1/e)$ -approximation to S^* .*

It is straightforward to see that I_G is monotone. It can be shown that I_G is submodular for the IC model and the LT model, which implies that the algorithm stated in Lemma 2 provides a $(1 - 1/e)$ -approximation ratio to IM.

Lemma 3 ([20]) *The influence function I_G is submodular for the IC model.*

Proof Consider a point in the IC process when node u , just becoming active, attempts to activate its neighbor v with a success probability $p(u, v)$.

The outcome of this random event may be thought of as the outcome of flipping an independent coin of bias probability $p(u, v)$. Because coin flips are independent, it does not matter whether the coin was flipped at the moment when v becomes active or it was flipped at the very beginning of the whole process and is being revealed now. Thus, assume that for each edge $u \rightarrow v$ in G , a coin of bias probability $p(u, v)$ is flipped at the very beginning of the process, independently of the coins for all other edges, and the result is stored for later use in the event that u is activated and v is inactive.

Flipping coins in advance, edges in G with coin flips indicate that an activation will be successful and declared live. The remaining edges are declared blocked. Fix the outcomes of the coin flips and then initially activate a seed set S . It is clear how to determine the full set of active nodes at the end of the cascade process [20]: A node x ends up active if and only if there is a live-edge path from some node in S to x . Consider the probability space in which each sample point specifies one possible set of outcomes for all the coin flips on the edges.

Let X denote one sample point in this space. Let $I_{G,X}(S)$ denote the total number of nodes activated by the process when S is the initial seed set and X the set of outcomes of all coin flips on edges. Because X is fixed, it is straightforward to see that $I_{G,X}(S)$ can be deterministically expressed as follows. Let $R(u, X)$ denote the set of all nodes reachable from u on a live-edge path. Thus, $I_{G,X}(S)$ is the number of nodes that can be reached on live-edge paths from any node in S , and so it is equal to the cardinality of the union $\cup_{u \in S} R(u, X)$.

For each fixed outcome X , the function $I_{G,X}(S)$ is submodular. To see this, let S and T be two sets of nodes such that $S \subseteq T$, and consider the quantity $I_{G,X}(S \cup \{u\}) - I_{G,X}(S)$. This is the number of elements in $R(u, X)$ that are not already in the union $\cup_{u \in S} R(u, X)$. It is at least as large as the number of elements in $R(u, X)$ that are not in the (bigger) union $\cup_{u \in T} R(u, X)$. It follows that

$$I_{G,X}(S \cup \{u\}) - I_{G,X}(S) \geq I_{G,X}(T \cup \{u\}) - I_{G,X}(T).$$

Since the expected number of nodes activated is just the weighted average over all outcomes, it implies that

$$I_G(S) = \sum_X \text{Prob}[X] \cdot I_{G,X}(S).$$

But a nonnegative linear combination of submodular functions is also submodular, and hence I_G is submodular, which completes the proof. \square

To show that I_G is submodular for the LT model, one may try to use a similar technique of the proof of [Lemma 3](#). For example, it may be reasonable to consider the behavior of the process after all node thresholds have been chosen. Unfortunately, for a fixed choice of thresholds, the number of activated nodes may not be a submodular function of the initial seed set. Thus, a more subtle technique is needed.

Lemma 4 ([20]) *The function I_G is submodular for the LT model.*

Proof Recall that each node v has an influence weight $p(u, v)$ from its neighbor u , where $\sum_{u \rightarrow v \in E} p(u, v) \leq 1$. Without loss of generality, extend this notation by letting $p(u, v) = 0$ if u is not v 's neighbor. Suppose that v picks at most one live edge $u \rightarrow v \in E$ with probability $p(u, v)$ and declares the rest of the edges blocked with probability $1 - \sum_{u \rightarrow v \in E} p(u, v)$. It follows from Lemma 1 that the LT model is equivalent to the reachability via live-edge paths. Similar to the proof of Lemma 3, it can be shown that I_G is submodular for the LT model. This completes the proof. \square

It follows from Lemmas 2–4 that if the influence function I_G is polynomial-time computable for the IC and LT models, then SGA is polynomial-time computable and provides a $(1 - 1/e)$ -approximation ratio to IM. However, computing $I_G(S)$ is #P-hard for both the IC and LT models, and so in general, I_G is unlikely to be polynomial-time computable. (There are some exceptions. For example, Chen et al. [8] showed that for the LT model, I_G can be computed in linear time for acyclic influence graph G .) Thus, one would naturally want to obtain a good estimate of I_G , which is justified by the following extension of Lemma 2, with a straightforward modification of the same proof as in [33].

Lemma 5 ([20]) *Let f be a monotone submodular function. Let S^* be a set that maximizes the value of f over all k -element sets. Suppose that for any $\varepsilon > 0$, there is a $\gamma > 0$ such that S is a k -element set obtained from SGA based on a g that is a $(1 + \gamma)$ -approximation to f . Then $g(S) \geq (1 - 1/e - \varepsilon)f(S^*)$. That is, S is a $(1 - 1/e - \varepsilon)$ -approximation to S^* .*

Kempe et al. [20] used Monte-Carlo simulations to obtain a close estimate of, on a given seed set S , the influence spread $I_G(S)$ with a large number of runs, resulting in high time complexity.

How to obtain a good estimate of $I_G(S)$ efficiently remains a challenge. Instead of finding close approximations to $I_G(S)$, researchers have devised new algorithms without needing to evaluate $I_G(S)$ (e.g., see [6–8, 26]). Research in this line typically follows two directions: (1) improving the time complexity of submodular greedy algorithm and (2) devising heuristics.

Presented below is an example of improving the time complexity of SGA. Note that each edge $u \rightarrow v$ in an influence graph $G = (V, E)$ is determined once whether it is activated with probability $p(u, v)$. Let G' be an induced subgraph of G as follows: Determine, for each $u \rightarrow v$ in the graph, whether it is selected and remove all edges that are not selected from G . Let $R_{G'}(S)$ denote the set of nodes reachable from S in G' . Performing a linear traverse of G' , depth first or breadth first, one can obtain $R_{G'}(S)$ and $|R_{G'}(\{u\})|$ for all $u \in V$.

The following greedy algorithm was devised by Chen et al. [6].

NewGreedyIC(G, k)

1. Set $S \leftarrow \emptyset$ and r a positive integer (e.g., $r = 20,000$)
2. Repeat the following steps for k times
3. Set $s_u \leftarrow 0$ for all $u \in V - S$
4. Repeat the following steps for r times
5. Construct G' by removing each edge $u \rightarrow v$ from G with probability $1 - p(u, v)$
6. Compute $R_{G'}(S)$
7. For each $u \in V - S$:
8. If $u \notin R_{G'}(S)$ then set $s_u \leftarrow s_u + |R_{G'}(S)|$
9. Set $s_u \leftarrow s_u/r$ for all $u \in V - S$
10. Set $s \leftarrow \arg \max_{u \in V - S} \{s_u\}$
11. Set $S \leftarrow S \cup \{s\}$
12. Output S

It is straightforward to see that *NewGreedyIC*(G, k) runs in $O(kr|E|)$ time. Simulation results [6] confirm that *NewGreedyIC*(G, k) produces an initial seed set S with an influence spread that is very close to what is produced by SGA, but is much more efficient.

For an example of heuristics, consider a simple degree discount for the IC model. According to the experimental results obtained by Kempe et al. [20], selecting nodes with maximum outgoing degrees as seeds could lead to larger influence spread than other previously known heuristics but still smaller than that of SGA. Chen et al. [6] presented a *degree discount heuristics* for the IC model based on this observation, but they discounted an edge from the degree. In particular, suppose that $u \rightarrow v \in E$ and u has been activated as a seed. To consider whether v should be included as a seed by looking at its degree, the edge $v \rightarrow u$ will not be counted in the degree. Simulations show the degree discount heuristics, when properly tuned, can achieve almost the same influence spread as SGA in significantly shorter time.

2.5 Other Influence Models

In addition to the IC and LT influence models, other influence models have also been proposed, including weighted cascade (WC) [20], maximum influence arborescence (MIA) [7], and local directed acyclic graph influence (LDAGI) [8]. The WC and MIA models are related to the IC model, while the LDAGI model is related to the LT model.

In the WC model, if u is activated in round i and $u \rightarrow v \in E$, then v is activated by u in round $i + 1$ with probability $1/d_v$, where d_v is the degree of node v . Similar to the IC model, each neighbor of v may activate v independently. Thus, if an inactive node v has m neighbors activated in or before the round i , the probability that v is activated in round $i + 1$ is $1 - (1 - 1/d_v)^m$.

In the MIA model, the calculation of the influence from one node to another node depends on the *maximum influence path* (MIP). In particular, given an influence graph $G = (V, E)$ and two nodes u and v , let $P_G(u, v)$ denote the set of all paths from u to v in G . Let $P = \langle u = u_1, u_2, \dots, u_m = v \rangle$ be a path from u to v . Define the propagation probability of P , denoted by $\text{pp}(P)$, as follows:

$$\text{pp}(P) = \prod_{i=1}^{m-1} p(u_i, u_{i+1}).$$

Let $\text{MIP}_G(u, v)$ denote the path from u to v with the largest propagation probability. That is,

$$\text{MIP}_G(u, v) = \arg \max_{P \in P_G(u, v)} \{\text{pp}(P)\}.$$

Given an influence threshold λ , define the *maximum influence in-arborescence* of node u , denoted by $\text{MIIA}(u, \lambda)$, as follows:

$$\text{MIIA}(u, \lambda) = \cup_{u \in V, \text{pp}(\text{MIP}_G(u, v)) \geq \lambda} \text{MIP}_G(u, v).$$

The *maximum influence out-arborescence* of u , denoted by $\text{MIOA}(u, \lambda)$, is as follows:

$$\text{MIOA}(u, \lambda) = \cup_{v \in V, \text{pp}(\text{MIP}_G(v, u)) \geq \lambda} \text{MIP}_G(v, u).$$

Given an initial seed set S , let the *activation probability* of any node $v \in \text{MIIA}(u, \lambda)$, denoted by $\text{aps}_S(v, \text{MIIA}(u, \lambda))$, be the probability that v is activated when S is the seed set and influence is propagated in $\text{MIIA}(u, \lambda)$.

In the MIA model, let I_G denote the influence spread of S , defined as

$$I_G(S) = \sum_{u \in V} \text{aps}_S(v, \text{MIIA}(u, \lambda)).$$

As in the IC model, it is straightforward to show the following result.

Theorem 4 *It is NP-hard to compute a k -node seed set S such that $I_G(S)$ is maximized for the MIA model. On the other hand, I_G is submodular and monotone, and so MIP can be approximated by SGA within a $(1 - 1/e)$ -approximation ratio.*

For more information on efficient greedy algorithms, the reader is referred to [7].

In the LDAGI model, each node v in G is associated with a local DAG $\text{LADG}(v)$, a subgraph of G containing v . Given a seed set S , it is assumed that the influence from S to v is only propagated within $\text{LADG}(v)$ according to the LT model. Let $\text{aps}_S(v)$ be the activation probability of v when influence is propagated within $\text{LADG}(v)$. Then the influence spread of S in the LDAG influence model, denoted by $I_G(S)$, is given by

$$I_G(S) = \sum_{v \in V} \text{aps}_S(v).$$

As in the LT model, it is straightforward to show the following result.

Theorem 5 *It is NP-hard to compute a k -node seed set S such that the influence spread $I_G(S)$ is maximized for the LDAGI model.*

For more information of efficient greedy algorithms, the reader is referred to [8].

3 Finding Communities

This section introduces and elaborates a number of heuristic algorithms for finding communities, including a new algorithm of low complexity, which uses random walkers to search for communities. This algorithm has only been reported in a technical report [28]. Simulation results show that the random walker method is both effective and efficient in finding network communities.

The existing algorithms for finding network communities can be roughly categorized into graph-partitioning-based algorithms and hierarchical clustering-based algorithms [35]. The Kernighan–Lin algorithm (K–L, in short) is one of the most popular graph-partitioning algorithms, and the agglomerative method and the divisive method are the common hierarchical clustering-based algorithms.

3.1 Graph-Partitioning-Based Algorithms

On a given (weighted) graph $G = (V, E)$ and a positive integer k , the graph partitioning problem tries to partition V into k subsets such that the size difference of any two subsets is at most 1 and the number of edges (or the summation of edge weights) between different components is minimized. This is an NP-hard problem. Graph partitioning has important applications in task scheduling under multiprocessor systems, sparse matrix reordering, parallelization of neural net simulations, particle calculation, and VLSI design, to name just a few. The K–L algorithm is a popular heuristic algorithm in VLSI design, which could be used to obtain rough network communities.

3.1.1 The Kernighan–Lin Algorithm

Let $G = (V, E)$ be a weighted graph with a weight function $w : E \rightarrow R$ that maps each edge to a real value. Initially, the algorithm partitions V at random into two subsets V_0 and V_1 such that their size difference is at most 1. Let

$$w(V_0, V_1) = \sum_{u \in V_0, v \in V_1} w(u, v)$$

denote the cost of the partition. For each node $u \in V_i$ ($i = 0, 1$), let E_u denote the external weight of u on edges from u to nodes in the other set, that is,

$$E_u = \sum_{v \in V_i \oplus 1} w(u, v),$$

where $0 \oplus 1 = 1$ and $1 \oplus 1 = 0$. Let I_u denote the internal weight of u within V_i , that is,

$$I_u = \sum_{v \in V_i} w(u, v).$$

Let $D_u = E_u - I_u$ be the difference between the external and internal weights of u . For each edge $(u, v) \in E$ with $u \in A$ and $v \in B$, let $I_{u,v} = D_u + D_v$. Swap u with v and obtain a new $I'_{u,v}$. Then

$$I_{u,v} - I'_{u,v} = D_u + D_v - 2w(u, v).$$

The heuristic of the K-L algorithm swaps nodes between $u \in V_0$ and $v \in V_1$ iteratively to maximize $I_{u,v} - I'_{u,v}$. The time complexity of the K-L algorithm is $O(|V|^3)$.

3.2 Hierarchical Clustering-Based Algorithms

The basic idea of the hierarchical clustering-based method is to find relations between nodes in the network and then extract communities from the network by adding or deleting edges.

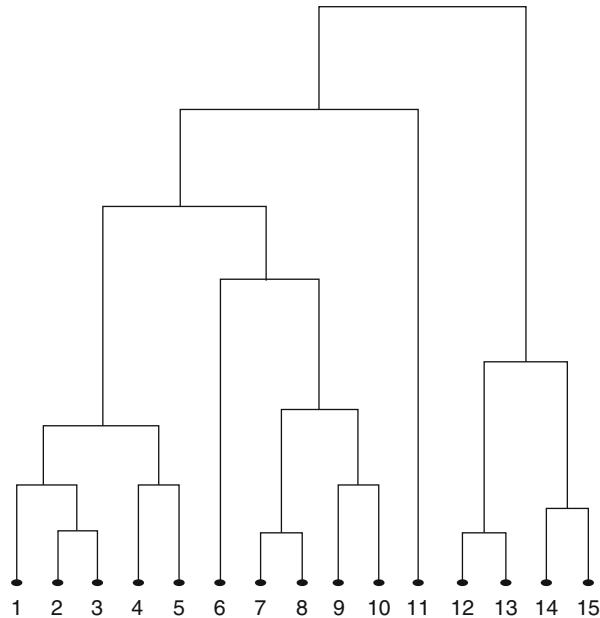
The hierarchical clustering method can be further divided into two subcategories: the agglomerative method and the divisive method. The agglomerative method follows a bottom-up approach. It starts with a completely disconnected network with no edges and proceeds in iterations. In each iteration, an edge is added based on a chosen measure. Pairs of clusters are merged as the algorithm moves up the hierarchy.

The divisive method follows a top-down approach. It starts from the original network and proceeds in iterations. In each iteration, it deletes an edge and splits a current subgraph recursively as the algorithm moves down the hierarchy.

The process of a hierarchical clustering method results in a tree structure known as “dendrogram” shown in Fig. 1.

3.2.1 The Agglomerative Method

This method uses a chosen measure of “structural similarity” of two nodes in the graph. The use of a different similarity measure may result in different communities. Such a similarity measure could either represent metrics of similarities or the strength of connection between nodes. For instance, one may use the distance between two points in a two-dimensional space as a similarity measure. In this case, using different notions of distance, such as the Euclidean distance, Manhattan distance, and maximum distance, gives rise to different variants of the similarity measure. A commonly used similarity measure is based on the comparison of

Fig. 1 The dendrogram

the neighboring nodes. More specifically, if two nodes have the same set of neighbors (or similar sets of neighbors), then they are considered “structurally equivalent” (or “structurally similar”) and should belong to the same community. For example, in an online social network, two persons in a friendship network are considered structurally equivalent if they have exactly the same set of friends [34]. But structural equivalence is rare in reality. Thus, structural similarities are often used instead. In particular, the following “Euclidean distance” [4] is often used to measure the similarity of two nodes u and v in a network. This Euclidean distance is not the same “Euclidean” commonly known:

$$x_{u,v} = \sqrt{\sum_{z \neq u,v} (a_{u,z} - a_{v,z})^2}, \quad (1)$$

where $a_{u,z} = 1$ if z is connected to u , and 0 otherwise. Clearly, if $x_{u,v} = 0$, then u and v have the exact same set of neighbors.

The agglomerative method adds edges from the network in iterations to increase the values of $x_{u,v}$. As more edges are added, connected subsets begin to form, which represent the forming of network communities. The components produced at each iteration are always subsets of other structures. The tree diagram, that is, the dendrogram, can be used to represent these subsets. Horizontal slices of the tree at a given level indicate the communities that exist above and below the value of the dissimilarity.

Running the agglomerative method requires a predetermination of how large each community should be. This is a drawback for finding natural communities.

3.2.2 The Divisive Method

Different from the agglomerative method that starts from a completely disconnected network and keeps adding edges to it in iteration, the divisive method starts from the original network and keeps deleting edges iteratively. The algorithm introduced by Girvan and Newman [13] is one of the most popular ones. The Girvan and Newman algorithm defines “edge betweenness” of an edge as the number of shortest paths between the two end nodes of the edge. If the network contains communities within which nodes are highly connected and between which nodes are sparsely connected, then all the shortest paths between different communities must go through one of these few edges between communities. Thus, at least one of the edges connecting communities will have a high edge betweenness. Removing such an edge, the groups of nodes are separated from one another, which may help reveal the underlying community structure of the network. Thus, the dendrogram is created from root to leaves. The Girvan and Newman algorithm involves intensive computation. For each iteration, the time complexity of calculating the betweenness of all existing edges in the network is $O(|V||E|)$. In the worst case, $O(|E|)$ edges may finally be removed, and so the worst-case running time is $O(|V||E|^2)$.

3.3 The Random Walker Algorithm

This section describes the random walker (RW) algorithm devised by Li et al. [28]. Different from the early algorithms, this algorithm uses random walkers to search for communities.

Given a graph $G = (V, E)$, the RW algorithm first generate, for each node, a number of walkers equal to its degree. The node for which a walker is generated is referred to as the *owner node* of the walker. Each walker is labeled with its owner node. The node where a walker is currently located is referred to as the *current node* of the walker. The RW algorithm runs in iterations. In each iteration, each walker chooses from two possible actions to proceed with certain probability. That is, with probability p_{move} , the walker chooses a node from the neighboring nodes of its current node and moves to it. With probability $p_{\text{jump}} = 1 - p_{\text{move}}$, it jumps back to its owner node.

Run the algorithm for a few iterations, where the number of iterations t is a parameter chosen by the user who runs the algorithm. When the algorithm stops, each node will have a certain number of walkers. If two nodes have the same set of walkers (or similar sets of walkers), then they are considered structurally equivalent (or structurally similar), and so they should belong to the same community. The algorithm uses the Pearson correlation to measure similarity of two nodes [34]. For any node u and v , if when the algorithm stops node u has a walker from node v , let $a_{u,v} = 1$; otherwise, let $a_{u,v} = 0$. The Pearson correlation of node u and v , denoted

by $x(u, v)$, is given below:

$$x_{u,v} = \frac{\sum_k (a_{u,k} - \mu_u)(a_{v,k} - \mu_v)/n}{\sigma_u \sigma_v}, \quad (2)$$

where

$$\begin{aligned} n &= |V|, \\ \mu_u &= \frac{1}{n} \sum_k a_{u,k}, \\ \sigma_u &= \frac{1}{n} \sum_k (a_{u,k} - \mu_u)^2 \end{aligned}$$

are the mean and variance of node u .

A larger value of $x_{u,v}$ would indicate that node u and v are more likely to be in the same community. The RW algorithm forms communities one at a time according to the similarities of nodes. In particular, it starts from an empty set S . It selects a pair of nodes with the highest similarity among all possible pairs that has not been selected and adds them to S . Next, expand S by repeatedly selecting new nodes in the remaining set of nodes, until a certain halting criterion is met. The halting criterion can be that either the size of the community constructed reaches a predefined threshold or the similarity between arbitrary node in the community and any node outside the community is smaller than a predefined threshold. Save S as a community and repeat the same process for the remaining nodes to find the next community, assuming that the nodes added into S will not belong to any other community. The RW algorithm stops after all nodes are considered. It is evident that the time complexity of the algorithm is $O(t(|V| + |E|))$.

Extensive simulation using both halting criteria was executed, which show that the RW algorithm is highly effective. The accuracy of the RW algorithm is computed as follows:

$$\text{Avg. accuracy} = \frac{1}{N_r} \sum_{i=1}^{N_r} \sum_{i=1}^{N_c} \frac{f^i}{s^i},$$

where N_r is the total number of simulation runs, N_c the number of communities, f^i the number of members that were correctly found for community i , and s^i the size of the community i .

For example, one set of simulations runs on two types of graphs of 1,000 nodes, with five communities of equal size. The edges in the first type are constructed with $p_{\text{low}} = 0.05$ and $p_{\text{high}} = 0.9$, while in the second type with $p_{\text{low}} = 0.1$ and $p_{\text{high}} = 0.75$. Each simulation runs for $t = 5$ iterations with $p_{\text{travel}} = p_{\text{jump}} = 0.5$. For the first type of graphs, the RW algorithm achieves on average near 100% accuracy using both halting criteria, where the size threshold is set to 200 and the similarity threshold is set to 2.0. In the second graph, the algorithm achieves on

average near 100 % accuracy if the first halting criterion is used, where the threshold of the community size is set to 200. If the second halting criterion is used, where the similarity threshold is set to 1.5, the RW algorithm achieves on average 93 % accuracy.

4 Data Collection from OSNs

Online social networks are popular platforms for people to connect, share information, and interact with each other. There have been intensive efforts to measure and analyze various characteristics of online social networks in recent years, including topological characteristics, user behaviors, and interactions between users.

Popular OSNs, such as Facebook and Twitter, are continuously growing to garner hundreds of millions of users. The sheer size of these OSNs and the dynamics of user activities make it impractical to measure the entire network. As a result, researchers often collect a small but representative sample of the network to study the properties of the network and test their algorithms.

This section reviews some of the recent studies on the measurement and analysis of OSNs, describes the graph sampling methods that have been used to collect data over OSNs, and presents an example of collecting data from MySpace with interesting statistical results.

4.1 Measuring and Analyzing OSNs

Mislove et al. [31] carried out a large-scale study on measuring and analyzing the structure of multiple online social networks, including Flickr, YouTube, LiveJournal, and Orkut. Their results confirmed that these OSNs are small world and scale free, and the degree distributions satisfy the power law. They also offered the following observations: (1) The in-degree of user nodes tends to match the out-degree. (2) Each OSN contains a densely connected core of high-degree nodes. (3) This core links small groups of strongly clustered, low-degree nodes at the fringes of the network.

Ahn et al. [1] measured and compared the structures of the following three online social networking services: Cyworld, MySpace, and Orkut, with respect to the degree distributions, clustering coefficients, degree correlations, and average path length. Krishnamurthy et al. [22] identified distinct classes of Twitter users and their behaviors, geographic growth patterns, and size of the network.

Gauvin et al. [11] measured and analyzed the gender-specific user publishing characteristics on MySpace using the technique of uniform sampling. They showed that there are recognizable patterns with respect to profile attributes, user interactions, temporal usages, and blog contents for users of different genders. In particular, gender neutral profiles (most of them are music bands, TV shows, and other commercial sites) differ significantly from normal male and female profiles for several publishing patterns. Caverlee and Webb [5] analyzed the frequencies of

words occurred in MySpace wall posts according to genders and ages. They looked at various statistics of male and female profiles in MySpace, including the degree distribution of friends, user demographics, language, and privacy preference.

Chun et al. examined the question of whether interactions between OSN users would follow the declaration of friends using the guestbook logs of Cyworld, the largest OSN in Korea. They constructed an activity network that reflected the online interactions between users and found that its structure is similar to the friend relationship network. However, when only reciprocated links are considered, the degree correlation distribution exhibited much more pronounced assortativity than the friends network. Wilson et al. [41] studied the relationship between friends and publishers and whether social links (social network friendships) were valid indicators of real user interaction in Facebook. They built a graph based on user interactions (i.e., messages exchanged between users). They observed that the interaction graph has much lower average node degree than the Facebook friends graph.

Benevenuto et al. [3] studied user workloads on four popular OSNs: Orkut, MySpace, Hi5, and LinkedIn. They analyzed key features of the social network workloads such as how frequently people were connected to social networks and for how long, as well as the types and sequences of activities that users conducted on these sites. They showed that browsing, which cannot be inferred from crawling publicly available data, accounted for 92 % of all user activities. Consequently, compared to using only crawled data, considering silent interactions like browsing friends' pages increases the measured level of interactions among users.

4.2 Sampling Methods of OSNs

A number of graph sampling methods have been adopted and developed for measuring large-scale OSNs. These methods can be classified into the categories of breadth-first search, random walk, and uniform sampling, among others.

4.2.1 Breadth-First Search (BFS)

The breadth-first-search traversal method was commonly used in recent measurement studies of OSNs (see, e.g., [1, 31, 41]). It is a classic graph search algorithm that begins from an initial seed node and explores all the neighboring nodes. The process is repeated, and at each iteration the earliest explored but not-yet-visited node is selected and its neighbors are explored. It is easy to see that the BFS method will discover all nodes within some distance from the seed node, but the sampled nodes are constrained within the connected component of the seed node. An incomplete BFS will cover and yield a full view (all nodes and edges) of some particular region in the graph. While the resulting sample graph readily enables the study of topological properties of the network (e.g., clustering coefficient, community structure), it is known that BFS introduces a bias toward nodes with high degrees [2, 25].

Kurant et al. [24] quantified the degree distribution bias of BFS sampling. For a random graph $\text{RG}(p_k)$ with a given (and arbitrary) degree distribution p_k , the node degree distribution expected to be observed by BFS as a function of the fraction of covered nodes (q_k) is given by

$$q_k = \frac{p_k(1 - (1 - t(f))^k)}{\sum_l p_l(1 - (1 - t(f))^l)}, \quad (3)$$

where $t(f)$ is the time that a fraction f of the nodes is covered, whose numerical value is straightforward to find.

The bias of BFS monotonically decreases with increasing fraction of sampled nodes f , and there is no bias when BFS is complete ($f = 1$). Kurant et al. also showed that, for $\text{RG}(p_k)$, all commonly used graph traversal techniques (breadth-first search, depth first search, forest fire, and snowball sampling) lead to the same bias. Given a biased sample; they also showed how to correct the bias by deriving an unbiased estimator of the original node degree distribution.

4.2.2 Random Walk

The random walk technique, successfully used for the World Wide Web (WWW) [19] and peer-to-peer network (P2P) [15, 36, 39] sampling, has been recently applied to OSN sampling [14]. In the classic random walk approach [29], a random walker starts from a node (u) and visits its neighboring nodes uniformly at random. The process continues and the given graph is sampled by the random walker. Denote the degree of node u by d_u , the probability that a node v is chosen as the next-hop node is given as follows:

$$P_{u,v} = \begin{cases} \frac{1}{d_u} & \text{if } v \text{ is a neighbor of } u, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

For a network with m edges, the steady-state distribution of the random walker being at a node v is

$$\pi(v) = \frac{d_v}{2m}. \quad (5)$$

A linear bias exists on high-degree nodes. Nodes with higher degrees will be visited more often than those with lower degrees in proportion to the node degree. To correct this bias, one can apply the Metropolis–Hastings algorithm [18, 30] to make the random walk converge to a desired probability distribution. In particular, it can be shown that the following transition probability achieves the uniform distribution among nodes [24]:

$$P_{u,w} = \begin{cases} \frac{1}{d_u} \min\left(1, \frac{d_u}{d_v}\right) & \text{if } v \text{ is a neighbor of } u, \\ 1 - \sum_{y \neq u} P_{u,y} & \text{if } v = u, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

It can be seen from the above equation that, to correct the degree bias in the classic random walk, the probability of visiting a node has been adjusted to reduce the transition to high-degree nodes.

In a random walk, the accuracy of estimating an underlying characteristic depends on the graph structure and the characteristic being estimated. The estimates may be inaccurate when the random walker gets “trapped” in a certain region of the graph. To mitigate this problem, one can start multiple independent random walks (MultipleRW) in different parts of the network [14]. This will reduce the chance of the sampling process being “trapped” in a region and has the advantage of being amenable to parallel implementation from different machines or different threads in the same machine. However, it was shown [37] that the MultipleRW approach may fail to reduce estimation errors. Starting vertices of MultipleRW in a component proportional to the number of edges of the component is a key factor to determine the accuracy of MultipleRW. While this approach can successfully mitigate estimation errors caused by disconnected components, it is not clear how it can be implemented in realistic OSNs such as MySpace and Facebook.

Ribeiro and Towsley [37] devised a new m -dimensional random walk sampling method called Frontier sampling. It performs m -dependent random walks in a graph. Starting from a collection of m randomly sampled vertices, the Frontier sampling method preserves all of the important statistical properties of a regular random walk (e.g., nodes are visited with a probability proportional to their degrees). For a given sampling budget of B steps, let c be the cost of randomly sampling a node. The sampling algorithm is given as follows:

Frontier Sampling

- 1 $n \leftarrow 0$ (n is the number of steps)
2. initialize $L = (v_1, \dots, v_m)$ with m randomly chosen nodes (uniformly)
3. **repeat**
4. select $u \in L$ with probability $d_u / \sum_{v \in L} d_v$
5. select an outgoing edge of u , (u, v) , uniformly at random
6. replace u by v in L and add (u, v) to the sequence of sampled edges
7. $n \leftarrow n + 1$
8. until $n \geq B - mc$

In Frontier sampling, while nodes are visited with a probability proportional to their degree, it can be shown that the joint steady-state distribution (the joint distribution of all m nodes) is closer to being uniform (the starting distribution) than k -independent random walkers, for any $k > 0$. This property has the potential to dramatically reduce the transient period of random walks. In simulations using real-world graphs, Frontier sampling can indeed effectively mitigate the large estimation

errors caused by disconnected or loosely connected components that can “trap” a random walker and distort the estimated graph characteristic in a variety of scenarios.

4.2.3 Uniform Node Sampling

As the name suggests, the uniform node sampling technique samples all the nodes in a graph uniformly and independently. There is no inherent measurement bias regardless of the distribution of the user IDs. When applicable, uniform node sampling can be used to establish the “ground truth” of the network properties and has the advantage of sampling disconnected graphs. Several recent studies have used the uniform node sampling technique to measure the network properties of MySpace and Facebook [11, 14].

For the uniform node sampling to work, however, an OSN must allow access to user profiles using unique numeric IDs. As such, it does not serve as a general solution for OSN sampling. Also, this approach may become highly inefficient when the space of user IDs is large and sparsely occupied, which was observed in both MySpace and Facebook [11, 14]. Since access to OSNs is often rate limited, the sparse distribution of the ID space will significantly increase the time to collect the data required for sampling.

4.3 MySpace Data Collection and Analysis

When sampling an online social network, the walker (a.k.a. crawler) collects the user profile information such as age, gender, education, work, location, friends/followers, and blogs, among others. The exact information that can be obtained from a user profile depends on the specific OSN design and the user’s privacy setting. The study carried out by Gauvin et al. on MySpace [11] provides an example of collecting and analyzing data of OSNs.

MySpace is a major online social networking site. Since launched in 2004, it has attracted a large number of users. Typical MySpace users are teenagers, music bands, artists, and commercial sites. MySpace is particularly popular among young people and music communities. Each MySpace user owns a profile containing basic personal information including name, age, gender, and location. A user profile whose owner’s gender is neither female nor male is treated as a gender “neutral” profile. A neutral profile in the data sets occurs when the gender is not specified. Many of the gender neutral profiles were commercial sites for music bands and TV shows. Despite the importance of these profiles, Gauvin et al. were the first to look at the publishing tendencies of gender neutral, commercial profiles.

In particular, Gauvin et al. [11] developed a crawling tool to sample MySpace, uniformly at random, to collect user profile attributes (name, gender, age, location, friends, among other things) and their corresponding blog information. Six million possible user profiles were randomly sampled between September 2008 and May 2009. When blogs for these random profiles were found, each blog was scanned to obtain the publisher profile attributes, the time stamp, and the content information.

A profile may be public (the default option) or private depending on its privacy setting. Private profiles only share the owner's name, gender, age, and location. No blog or friend information could be retrieved from these sites. As a result, the analysis was limited to the blogs posted on public profiles. The results of this data collection process produced 546,829 public and 68,141 custom profiles of which 92,653 sets of blogs were obtained and 1,867,299 blogs were collected. The number of blogs contained within a randomly scanned profile ranged from 0 to 85,643. To determine the publisher characteristics, 1,768,884 profiles were scanned.

4.4 Member Profile Patterns

4.4.1 Age, Gender, and Privacy

In the collected data set, female, male, and neutral profiles constitute, respectively, 52.8 %, 37.7 %, and 9.5 % of the total valid IDs. A larger percentage of female users (57.1 %) were privacy aware than male users (39.2 %). Almost all (99.999 %) gender neutral profiles were public as they were typical commercial sites trying to attract people to visit their pages.

[Figure 2](#) shows the age distribution of private account profiles. The majority of profile owners declared their ages in the range of 15–30. Beyond this range the distribution falls off rapidly, only to increase around age 69 and age 100 due to a

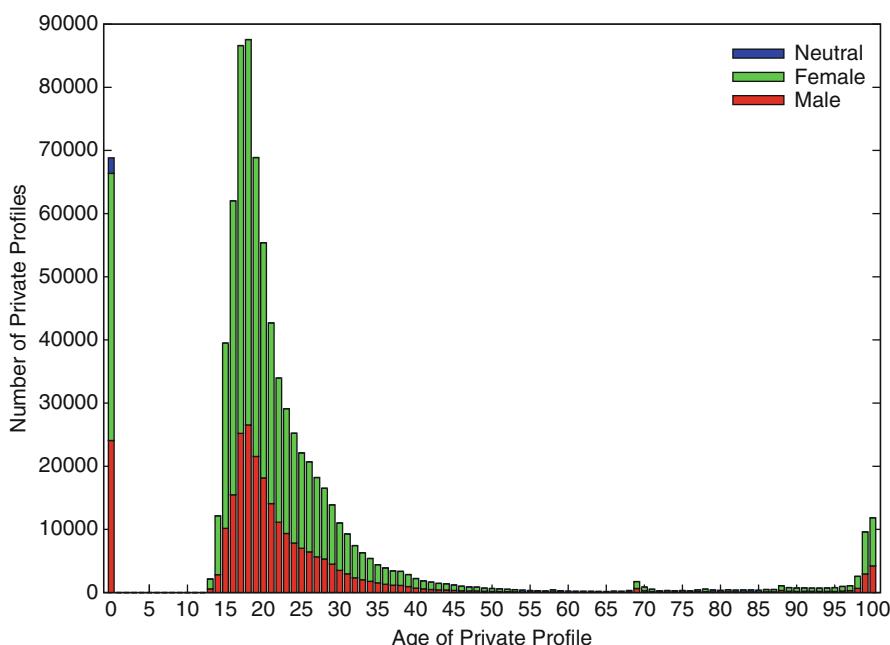


Fig. 2 MySpace private account and age distribution

common practice among underage users to elude the age restriction by selecting an exaggerated age.

Two age groups deserve a special mention. The first is the absence of ages from 1 to 12. MySpace protects users in this age group and does not create their profiles. The second is the age group under 16. MySpace automatically sets user profiles in this group to be private. This restricts the information that can be obtained for these sites. It can be noted that there was a large number of zero-age profiles. This is probably because accounts of underage owners were kept private. Also note that the data collected do contain some user profiles with ages under 16. This is contradictory to the current MySpace policy. This inconsistency may be the result of MySpace's policy change, where an old policy may actually allow users under age of 16 to publish their ages.

4.4.2 Publishers Versus Friends

In an online social network, friend relationships do not necessarily reflect real online interactions between users. The data sets depict an interesting relationship between the total number of friends of a user and the number of friends that actually publish on the user's blog wall.

[Figure 3](#) plots two sets of complementary cumulative distribution functions (CCDF), one set corresponding to the number of friends and the other corresponding to the number of distinct publishers among those friends. In each set, a CCDF is obtained for each profile gender: female, male, and neutral.

It can be seen that the friendship and publisher distributions for male and female profiles exhibit similar behaviors except at the tail, where the female distribution extends farther and the male distribution falls off sharply. The distributions for gender neutral profiles lie above the male and female distributions. This indicates that neutral profiles tend to have more friends and publishers than normal male and female profiles. This trend is consistent with the observation that neutral profiles were generally music bands or other commercial sites with a large number of friends publishing comments on the music, movie, or product being promoted.

[Figure 4](#) plots the number of distinct publishers versus the number of friends of MySpace users. It is observed that for most users, as the number of friends increases, only a small fraction of the users actually interact with each other through blogs. This is consistent with the observations made in [41] for Facebook users.

Interestingly, the publisher-versus-friend plots for male and female profiles nearly coincide with each other, while the plot for gender neutral profiles falls far below. This indicates that although gender neutral profiles tend to have more friends than normal male and female profiles, an even smaller fraction of these friends actually write on their blog walls. A possible explanation for this difference is that many users befriend commercial sites just to browse or receive information from the sites. The relationships between friends and gender neutral commercial sites are less "active" than those between normal users. It is worth noting that some profiles contain more distinct publishers than friends. This is a result of profile owners removing friends from their profiles and occurs rather rarely.

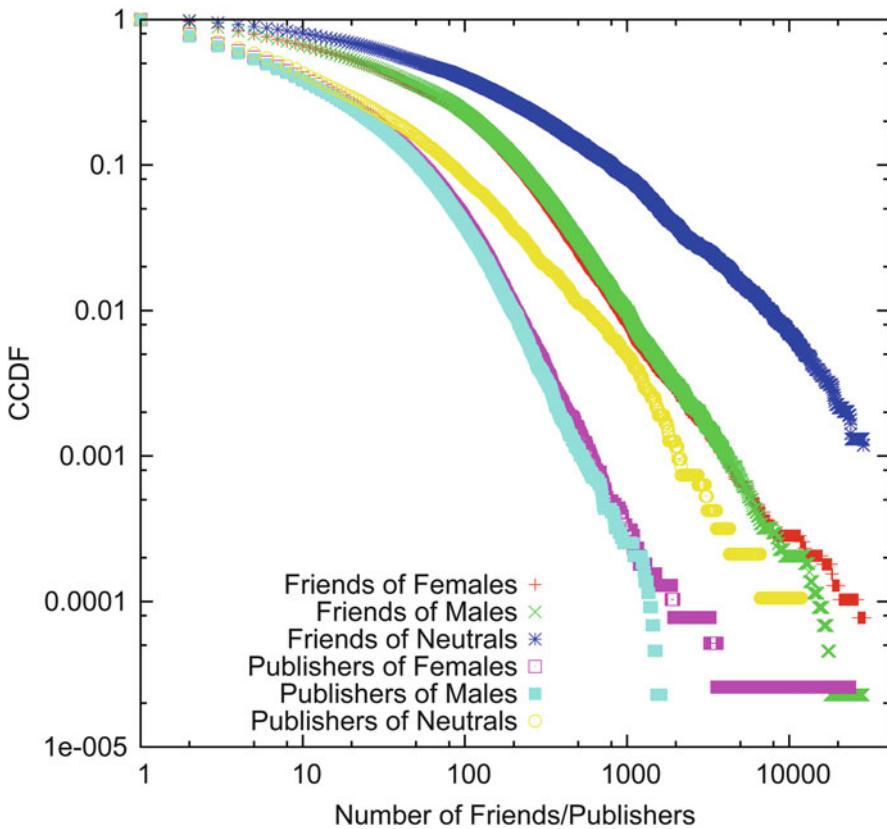


Fig. 3 Distribution of number of friends and publishers

4.4.3 Profile Owner and Publisher Age Difference

Figure 5 shows a density graph of the ages of publishers versus those of the profile owners. It can be observed that the density is greatest along the diagonal line (zero age difference between publishers and profile owners) and drops quickly as the age difference increases. This indicates that users tend to interact with friends of similar ages. A large fraction (75 %) of publishing activities occurs between people within a 5-year age difference. The shape of the age difference distribution can be described by an exponential function $f(x) = a \exp(b\|x\|)$, where $a = 0.334$ and $b = -0.470$, for age differences between 0 and 8 years. The shape of known distributions is unlikely to match the distribution of age differences larger than 8, likely because a fraction of the young users tend to lie about their age. Note that the other small islands in the figure are artifacts caused by the following two groups of users: underage users below 14 whose age information is not available and set to 0 and users with exaggerated ages around 69 and 100.

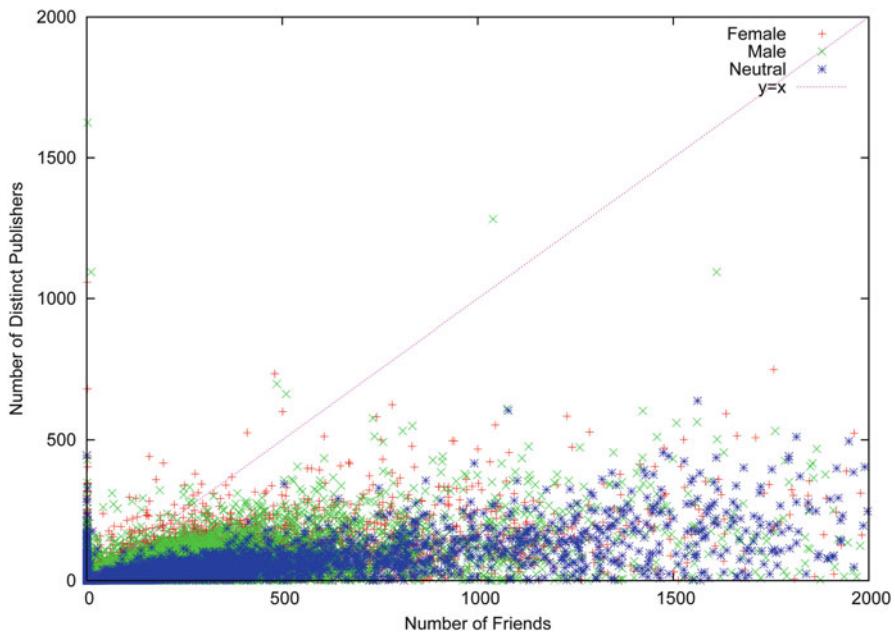


Fig. 4 Distribution of friends versus publishers

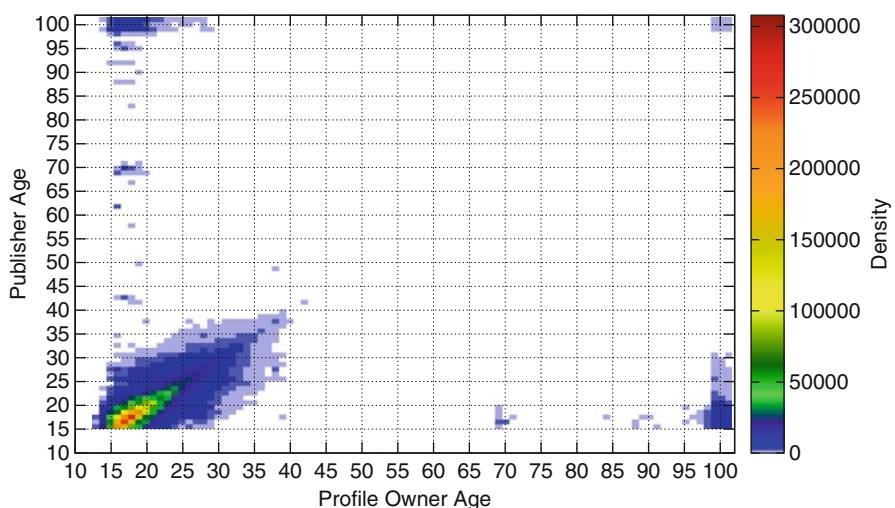


Fig. 5 Publisher age versus owners

4.5 Temporal Usage Patterns

Three specific elements of time are analyzed. They are time of day, day of week, and monthly usage patterns, as shown in Fig. 6.

All blog posting times are normalized to the local wall-clock time of the individual profiles. It can be observed from Fig. 6a that there is a significant increase in publishing activity from morning to afternoon, with the peak usage time at 10pm and a significant drop at midnight. The low activity between midnight and 12pm may reflect the times of rest and busiest time for labor in the day as far as school and work are concerned. As for the daily publishing pattern characterized in Fig. 6b, weekdays have a higher degree of activity than weekend days. Gender analysis for diurnal publishing patterns does not indicate any significant pattern difference between genders.

The total number of blogs and content per day for each month was also collected. The reason for gathering this information was to ascertain whether trends in publishing patterns can be used to predict the load of a social network server and determine peak usage times. This information may be used to determine false positives during spam detection and other security filtering policies. Many months displayed evenly distributed activities on each day where the daily blog counts did not deviate by more than 200 blogs for the collection set analyzed. There were some months (February, November, December), however, that stood out with respect to predicting publishing patterns, mainly due to the specific calendar events within the months. An example is given in Fig. 6c for February, which is a low publishing month in general, but there is a sharp contrast between the publishing activities on the 14th and those on other days. Similar high publishing activities are observed for other major calendar events including New Years Day, Mother's Day, 4th of July, Halloween, Thanksgiving, and Christmas.

4.6 Blog Content Patterns

A blog's entry contains four types of content, namely, words, hyper-references (HREFs), images, and objects. In most cases the objects were ColdFusion scripts or other animated objects. ColdFusion objects allow users to create animated objects used to make their profiles more attractive. Each blog is a composition of these four content types. The inclusion and makeup of these different content types is a reflection of each individual's writing style. The goal is to determine if specific content patterns can be identified.

Figure 7 depicts the complementary cumulative distribution functions (CCDF) of the counts of words, HREFs, images, and objects in a blog entry. It can be seen that the counts of these content types decay roughly according to power laws in certain regimes. In a blog entry, word count dominates the counts of other types. Object count is the smallest among all types. HREF and image counts are similar to each other and lie somewhere in-between. Of course, the counts of the different content types do not reflect their actual sizes (bytes). The size of an image is typically

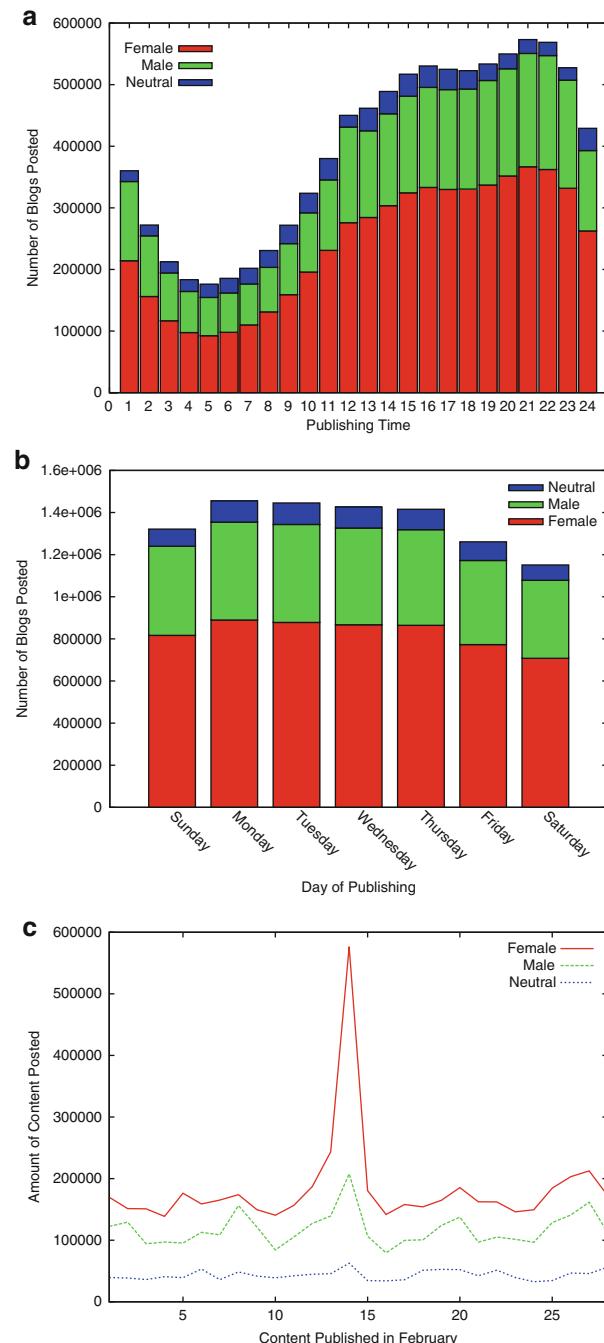


Fig. 6 Temporal publishing patterns. (a) Hourly pattern. (b) Daily pattern. (c) February daily pattern

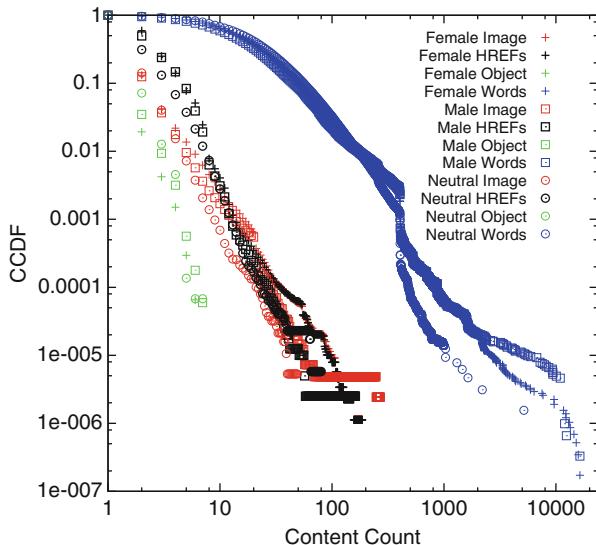


Fig. 7 Gender content publishing

significantly larger than that of a word. In general, users tend to post short messages. For example, only 10 % of the blog entries contain more than 50 words; 4 % of the blog entries contain 3 or more images. The use of objects is even less frequent.

There are a number of recognizable patterns with respect to the count of these content types across different genders. The word count distribution of gender neutral profiles drops off drastically around word count 410. This may be due to the need of a commercial site to have very specific and focused contents (e.g., advertisements) that fit people's attention spans. Another recognizable pattern is that females tend to publish more HREFs (hyperlink references) and images than male and neutral profiles.

5 Multilayer Friendship Modeling in Location-Based OSNs

How users are connected through an online social network is an interesting question. Intuitively, the formation of online friendship may reflect to certain degree the real-world counterpart. People who have mutual interests, are geographically close, or belong to the same social communities (e.g., churches, volunteer groups, and special interest groups) are more likely to become friends. The challenge is to devise adequate metrics to represent this intuitive understanding with a quantitative model. While there are a number of studies on structural properties of social networks, such as the distribution of node degrees, friendships (or connections) between users may be modeled with other user attributes.

In the context of influence maximization, a quantitative friendship model can provide an interesting perspective for studying influence propagation in OSNs, for the strength of the friendship often reflects the likelihood of how people influence each other. For example, for any two people who are OSN friends, it would be reasonable to combine their behaviors, such as whether they would buy the similar products or like the same brands, with the friendship model to come up with an empirical influence propagation model. For any two people who are not OSN friends, it would be useful to predict the possibility and the strength of their connection if they will ever become friends. Such information would be useful to find the most influential individuals in the network.

This section presents a friendship model for Brightkite, a location-based OSN. The primary service mode of Brightkite is to let users post and share updates with their friends or with the public. A user update could be a text note with a 140-character cap, a photo, or a check-in to announce a location. The user performs a check-in when they come to a new location. This is the typical usage scenario of Brightkite. All note and photo updates will be associated with the location of the user's latest check-in. Brightkite assumes that the user stays at the same location until they perform another check-in somewhere else.

To study how Brightkite users share location and how they become connected, Li and Chen [27] collected Brightkite user updates posted from March 21, 2008, to December 15, 2009. There were 4,460,161 updates posted by 70,337 users. They also collected each user's profile and friend list. They then built a three-layer friendship model to correlate the relationship between friend connections with attributes of their profiles, social graphs, and mobility patterns. For convenience, this model is referred to as the *three-layer friendship* (TLF) model. To validate the TLF model, they used new friend connections during the 168 days after the training data were collected. They showed that the TLF model can better predict user friendships than the classical naive Bayes and J48 tree algorithms.

5.1 A Multilayer Friendship Model

In social communities, friend connections have strong correlations to the social graph structures, profile attributes of individuals, and geographic locations. For social graph structures, the social distance between friends can be used to derive the empirical model of connections. For the profiles of individuals, the common interests between friends can be used to infer their friendships, which are reflected to some degree by the Brightkite tags in user profiles. For the geographic locations, it was observed that friends tend to send updates within a small geographic range with high probability. The TLF model exploits the correlations of friendship distance, tag attributes, and mobility patterns.

In a social network $G_s(V_s, E_s)$ where each node v_s denotes a user and each edge e_s denotes the friendship between two nodes. For any node (user) pair (a, b) , its attribute(s) can be calculated. Given the attribute values of two users, the empirical probability of them being friends can be computed as follows: Assuming that m_v is the attribute value, let $P_f(m_v)$ denote the probability of value m_v for friend pairs and

$P_{\text{nf}}(m_v)$ denotes the probability of value m_v for non-friend pairs. Using the *Bayes' rule* of conditional probability, the probability $P(ab|m_v)$ of existence of an edge between the two users (meaning they are friends) while observing attribute value m_v is

$$P(ab|m_v) = \frac{P(m_v|ab) \times P(ab)}{P(m_v)}, \quad (7)$$

where $P(m_v|ab)$ denotes the conditional probability that the attribute value m_v occurs given that (a, b) is a pair of friends, $P(ab)$ denotes the probability that node a and node b are friends, and $P(m_v)$ denotes the probability that m_v occurs in all possible pairs. Thus,

$$P(m_v|ab) = P_f(m_v), \quad (8)$$

$$P(ab) = \frac{|E_s|}{C_{|V_s|}^2}, \quad (9)$$

$$P(m_v) \cong P_{\text{nf}}(m_v), \quad (10)$$

where $C_{|V_s|}^2$ denotes all possible user pairs.

Equation 10 is satisfied because $C_{|V_s|}^2$ is much larger than $|E_s|$. In the social graph retrieved on April 14, 2009, $|E_s| = 175,309$ and

$$C_{|V_s|}^2 = 50,842 \times 50,842 / 2 \cong 1.29 \times 10^9.$$

The probability of m_v in non-friend pairs is approximately equal to that in all possible pairs. It follows from Eqs. 8–10 that Eq. 7 becomes

$$\begin{aligned} P(ab|m_v) &= \frac{P_f(m_v) \times |E_s|}{P_{\text{nf}}(m_v) \times C_{|V_s|}^2} \\ &= F(m_v) \times \frac{|E_s|}{C_{|V_s|}^2}. \end{aligned} \quad (11)$$

Let $F(m_v)$ denote the *ranking factor* defined as follows:

$$F(m_v) = \frac{P_f(m_v)}{P_{\text{nf}}(m_v)}. \quad (12)$$

Since $|E_s|/C_{|V_s|}^2$ is a constant for all user pairs, to determine which pair has a higher probability to become friends, it suffices to calculate $F(m_v)$ and compare its value over different user pairs.

Identifying the existing 175,309 friend pairs from the data set, it is straightforward to compute $P_f(m_v)$ using this empirical data. Since the number of non-friend pairs is huge, it would be hard to explore the entire sample space. Instead, Li and Chen [27] randomly selected 340,000 non-friend pairs and calculated $P_{nf}(m_v)$ from these samples.

In the real world, a single attribute is insufficient to recommend user's friendship. For example, potential friends could share the same tag(s) because of mutual interest(s), but may never send updates in nearby locations. Thus, if one only uses the location-based attribute, it would be difficult to accurately model the pair relationship. Instead, Li and Chen selected three attributes (details later) to calculate the probability of friends. Let P_1 , P_2 , and P_3 denote the probabilities that were calculated by three different attributes, respectively, and if they are independent, then

$$\begin{aligned} P(ab|m_{v1,v2,v3}) &= 1 - (1 - P_1) \cdot (1 - P_2) \cdot (1 - P_3) \\ &= 1 - (1 - P_1 - P_2 + P_1 \cdot P_2) \cdot (1 - P_3) \\ &= P_1 + P_2 + P_3 + O(P^2) - O(P^3) \\ &\cong P_1 + P_2 + P_3. \end{aligned} \quad (13)$$

The values of $O(P^2)$ and $O(P^3)$ are extremely small, and so they can be ignored. For Eq. 11,

$$|E_s|/\mathbf{C}_{|V_s|}^2 = 1.36 \times 10^{-4}$$

in the current social graph, and each rank factor $F(m_v)$ is usually no more than 300.

Ignoring the constant value, user friendship can be ranked by comparing the summation of the ranking factors of the three attributes:

$$F = F_1 + F_2 + F_3. \quad (14)$$

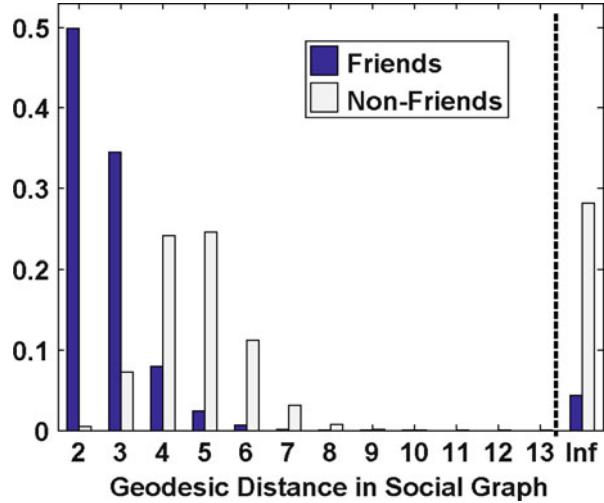
The three attributes used in this calculation will be discussed below. Note that in Eq. 14, if one or two attributes are missed, the model is still valid.

5.1.1 The Social Graph Layer

In the constructed social graph $G_s(V_s, E_s)$ obtained from Brightkite, where $|V_s| = 50,842$ and $|E_s| = 175,309$, each node v_s is a Brightkite user and each edge e_s represents a pair of friends. Since the "friendship" in Brightkite is mutual,¹ G_s is an undirected graph. Define the *friends distance* $DS_{i,j}$ of v_i and v_j in Eq. 15, where $e_{i,j}$ is the link between v_i and v_j :

¹At the time of retrieving the data trace, a friendship in Brightkite was mutual. Starting from December 2009, friendships were changed to directional. In other words, like Twitter, each user has a "friends list" and "fans list" corresponding to the "following" and "follower."

Fig. 8 Histogram of distance DS of friends/non-friends in the social graph



$$DS_{i,j} = \text{geodesic_distance}(v_i, v_j) \text{ in } G'_s(V_s, E_s - e_{i,j}). \quad (15)$$

In Eq. 15, geodesic distance is the shortest path of two nodes in the social graph. Equation 15 means if v_i and v_j are friends, then remove the existing edge between i and j and calculate their geodesic distance in the resulting social graph.

Figure 8 depicts the histogram of normalized DS including non-friend pairs (the bar at the left-hand side) and friend pairs (the bar at the right-hand side). It turns out that the distance of more than 49.9 % friend pairs was 2 and more than 34.5 % was 3. On the other hand, the distance of less than 7.8 % non-friend pairs was less than 4, and 28.2 % of non-friend pairs were disconnected nodes in the social graph, which suggested that the social graph in Brightkite was quite sparse.

The ranking factor of the social graph layer was calculated by dividing the friend pairs' proportion by non-friend pairs' proportion (Eq. 12). Consider the situations that the ranking factor is larger than 1 and ignore the other situations. Then

$$F_s(m_s) = \begin{cases} 94.81 & m_s = 2 \\ 4.75 & m_s = 3 \\ 0 & \text{others.} \end{cases} \quad (16)$$

5.1.2 Tag Graph Layer

To model the relationship between friendships and users' tags, Li and Chen [27] selected 1,000 popular tags to build a tag graph, denoted by $G_t(V_t, E_t)$. Since Brightkite allows users to use any words as their tags, it makes sense to only focus on the popular tags to obtain statistical results. The graph G_t is an undirected *complete* graph, and each node corresponds to a tag word. Connect each pair of tags with a nonnegative value w_t as weight to indicate the closeness of them, which is defined as follows:

$$\begin{aligned}
w_{t_1,t_2} &= \frac{P_f(t_1, t_2)}{P_a(t_1, t_2)} \\
&= \frac{P_f(t_1, t_2)}{P_a(t_1) \cdot P_a(t_2) + P_a(t_2) \cdot P_a(t_1)} \\
&= \frac{P_f(t_1, t_2)}{2 \cdot P_a(t_1) \cdot P_a(t_2)},
\end{aligned} \tag{17}$$

where $P_f(t_1, t_2)$ denotes the probability that tag t_1 and tag t_2 are shared by friends in all friend pairs. This probability can be calculated by first counting the number of pairs, where one user had tag t_1 and the other had tag t_2 , to get $CNT_f(t_1, t_2)$, then dividing it by the total number of friend pairs $|E_s|$. That is,

$$P_f(t_1, t_2) = \frac{CNT_f(t_1, t_2)}{|E_s|} \tag{18}$$

Note that t_1 and t_2 could be the same.

Let $P_a(t_1, t_2)$ denote the probability that for one user pair, one user has tag t_1 and the other has tag t_2 , in all possible user pairs, and $P_a(t_1)$ denote the probability that tag t_1 occurs in all users. The probability $P_a(t_1)$ can be calculated by first counting the number of users who used tag t_1 in all users to get $CNT_a(t_1)$ and then dividing it by the total number of users $|V_s|$. That is,

$$P_a(t_1) = \frac{CNT_a(t_1)}{|V_s|} \tag{19}$$

To rank how close tags are to each other, let

$$w_{t_1,t_2} = \frac{CNT_f(t_1, t_2)}{CNT_a(t_1) \cdot CNT_a(t_2)}, \tag{20}$$

which represents the weight of the tag pair of t_1 and t_2 . If neither t_1 nor t_2 is on the list of top 1,000 tags, then let $w_{t_1,t_2} = 0$. It turns out that the tag pairs with the highest weight are *music* versus *photography* and *photography* versus *travel*. The tag pairs with the lowest weight are *film* versus *php* and *linux* versus *snowboarding*.

Let user i 's tag list be $(t_{i1}, t_{i2}, \dots, t_{iM})$ and user j 's tag list be $(t_{j1}, t_{j2}, \dots, t_{jN})$. Define the tag metric $m_{t(i,j)}$ as follows:

$$m_{t(i,j)} = \sum_{n=1}^N \sum_{m=1}^M w_{t_{im}, t_{jn}}. \tag{21}$$

Figure 9 shows the histogram of tag metrics including both friend pairs (the bar on the right-hand side) and non-friend pairs (the bar on the left-hand side). It turns out that for most metric values, the tag metric distribution of friend pairs is

Fig. 9 Histogram of tag metric M_t of friends and non-friends

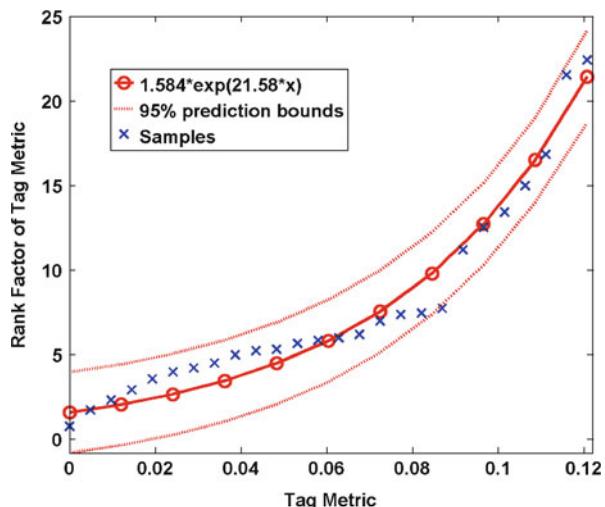
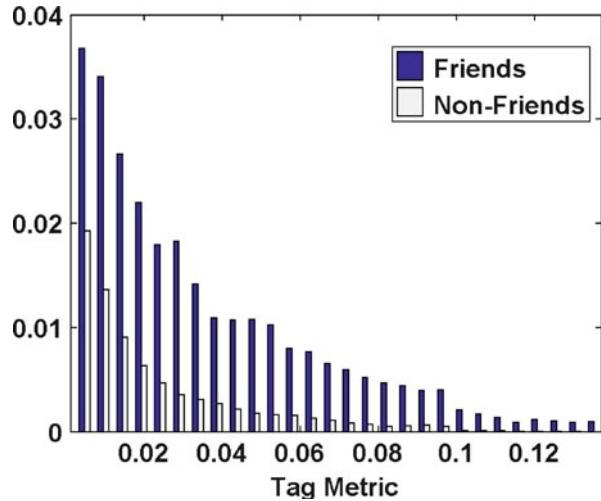


Fig. 10 Rank factor of tag metric

larger than that of non-friend pairs, especially in large value ranges, which means that friend pairs tend to share closer tags. The ranking factor of tag metric can be calculated as follows:

$$F_t = \frac{P_f(m_t)}{P_{nf}(m_t)}. \quad (22)$$

Figure 10 shows samples of F_m by cross markers. To guarantee the statistical significance of $P_{nf}(m_t)$, the tag count was required to be at least 50 for each sample in the figure. That is, values of $P_{nf}(m_t)$ for $m_t > 0.12$ were ignored. An exponential function was applied to fit the samples for producing a fitting function, shown as the

line of points. The two dotted lines depict the 95 % prediction bounds by the fitting function. When the tag metric is larger than 0.1, the values of the samples tend to be infinitely large, and so an arbitrary large number, 100, was assigned.

The function F_t can be denoted by

$$F_t(m_t) = \begin{cases} 0 & m_t = 0 \\ 1.584 \times e^{(21.58 \times m_t)} & 0 < m_t \leq 0.12 \\ 100 & 0.12 < m_t. \end{cases} \quad (23)$$

5.1.3 Location Graph Layer

To calculate location similarities among users, the conterminous 48 states in the USA were split into smaller regions based on population. Cities with large population were also divided into several regions. For a countryside with fewer people, a region may cover tens of square miles. [Figure 11](#) depicts a splitting example. The population data of the year 2000 from the US Census Bureau was used in the splitting,² which includes 36,269 cities in the USA. In particular, the conterminous 48 states were viewed as a rectangle, and the total population was counted. If the population was larger than the threshold of 50,000, split the rectangle both vertically and horizontally, resulting in four smaller rectangles evenly. Keep splitting each region until there is no more region with population larger than the threshold or there is only a single city in the region.

Count for each user how many updates fell into each region. The summation of two users' update counts in the same region indicates their location similarity, which is the metric for the location layer. Formally, let c_{in} denote the update count of user i in region n ; then the location metric of user i and user j , denoted by $m_l(i,j)$, can be calculated as follows:

$$m_l(i, j) = \sum_n \text{sng}(c_{in}) \times \text{sng}(c_{jn}) \times (c_{in} + c_{jn}), \quad (24)$$

where $\text{sng}(\cdot)$ is a sign function defined below:

$$\text{sng}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (25)$$

[Figure 12](#) shows the probability density function of the location metric for both friend pairs and non-friend pairs. The friend pairs tend to have smaller values of the location metric. To keep the statistical significance, samples with location metric larger than 500 were ignored, for there were few non-friend pairs in this range.

Similar to the tag metric, define the ranking factor of location metric as follows:

$$F_l = \frac{P_f(m_l)}{P_{nf}(m_l)}. \quad (26)$$

²<http://www.census.gov/>.

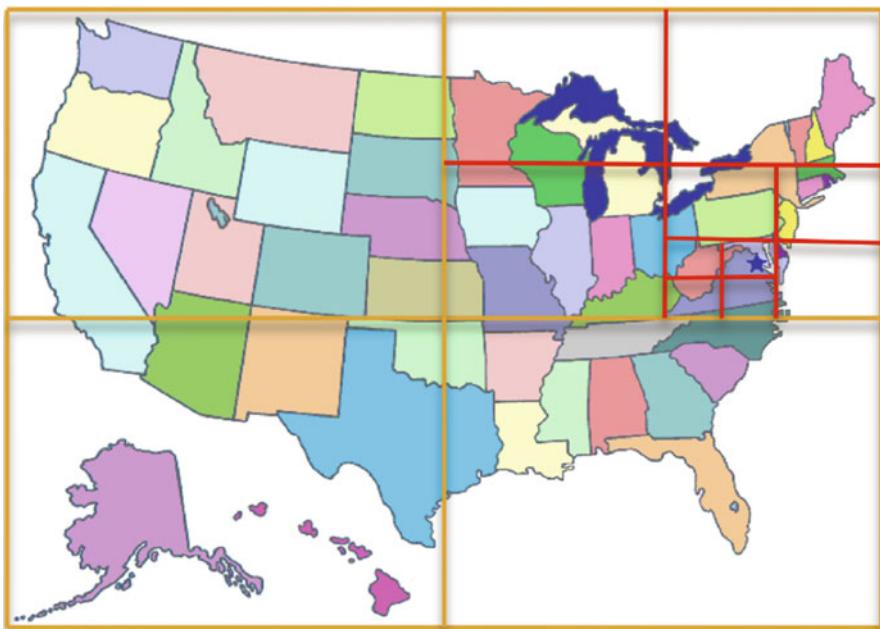


Fig. 11 An example of map splitting

Fig. 12 Probability distribution of location metric 1

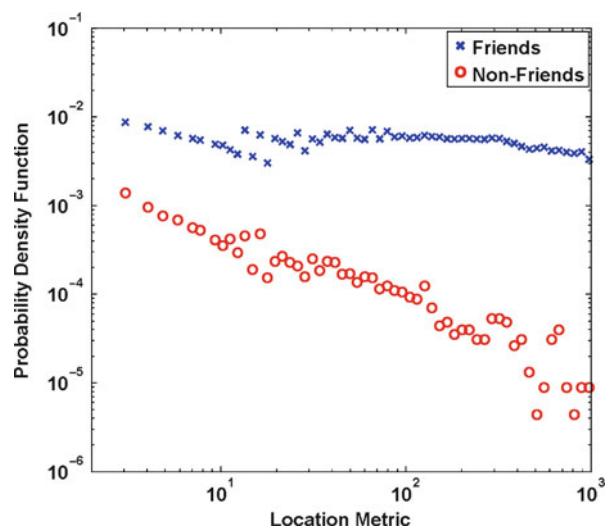


Fig. 13 Rank factor of location metric 1

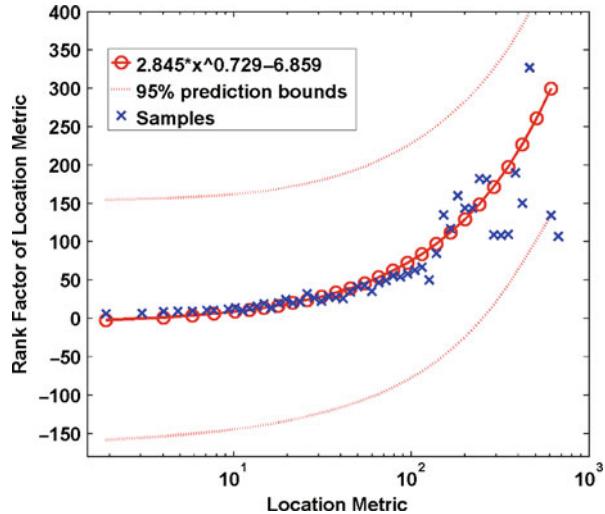


Figure 13 depicts the samples of F_l by cross markers. The power law function was used to fit the samples for producing a fitting function, shown as the line of points. The two dotted lines show the 95 % prediction bounds by the fitting function. The situations where $m_l > 500$ were ignored for statistical significance, and an arbitrary large number, 300, was assigned for $m_l > 500$.

$$F_l(m_l) = \begin{cases} 0 & m_l < 4 \\ 2.845 \times m_l^{0.729} - 6.859 & 4 \leq m_l \leq 500 \\ 300 & m_l > 500. \end{cases} \quad (27)$$

Lacking population data outside the USA, another location metric was used to calculate the user pairs who posted updates outside of the United States.

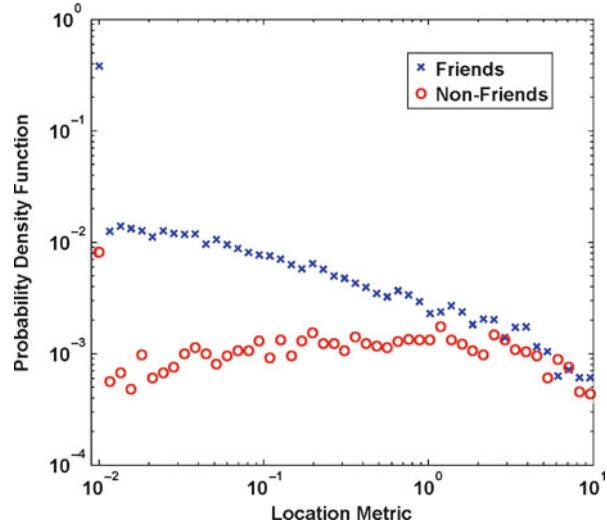
Assuming that user i sent updates in the locations $(l_{i1}, l_{i2}, \dots, l_{iN})$ and each location contained a number of updates $(c_{i1}, c_{i2}, \dots, c_{iN})$, respectively, let $\text{Dist}(l_1, l_2)$ denote the geographical distance of locations l_1 and l_2 .

Define the location metric $m'_{l(i,j)}$ between user i and j as follows:

$$m'_{l(i,j)} = \min_{\forall m \in (1, M), \forall n \in (1, N)} \left(\frac{\text{Dist}(l_{im}, l_{jn})}{c_{im} + c_{jn}} \right) \quad \text{when, } \text{Dist}(l_{im}, l_{jn}) < 30. \quad (28)$$

Equation 28 ignores the location pairs whose distance is larger than 30 miles. That is, the geographical correlation of two locations that are 30 miles away is ignored. The location metric is the minimum value of update distance divided by the summation of update times in the two locations. The user pair (A, B) is viewed closer than the user pair (C, D) if (A, B) has more updates than (C, D) , even if their

Fig. 14 Probability of location metric 2



update distances are the same. More updates for (A, B) mean that A and B could be neighbors or they worked in nearby locations. On the other hand, few updates for (C, D) mean that they happened to visit the same place and were not close as (A, B) .

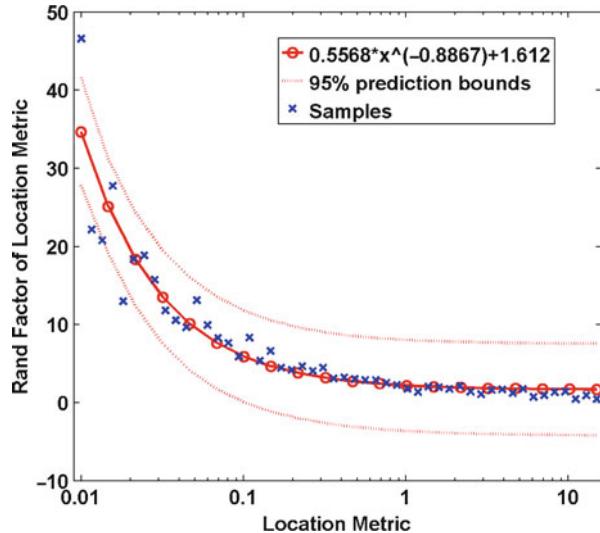
Figure 14 shows the probability density function of the location metric for both friend pairs and non-friend pairs. The friend pairs tend to have smaller values for the location metric.

Figure 15 shows the samples of F_l by cross markers. The power law function was used to fit the samples for producing a fitting function, shown as the point-line. The two dotted lines show the 95 % prediction bounds by the fitting function. The situations where $m'_l > 14$ were ignored because in these situations $F'_l < 1$. An arbitrary large number, 100, was assigned for $m'_l < 0.01$:

$$F'_l(m'_l) = \begin{cases} 100 & m'_l < 0.01 \\ 0.5568 \times m'^{-0.8867}_l + 1.612 & 0.01 \leq m'_l \leq 14 \\ 0 & m'_l > 14. \end{cases} \quad (29)$$

It can be seen that the second location metric is not as good as the first one, since it considers the two closest update locations and ignores the others. On the other hand, the first location metric reflects all update closeness between any two users. Finally, if the majority of updates of both users were in the conterminous 48 states of the USA, the first location metric was used; otherwise, the second location metric was used.

Fig. 15 Rank factor of location metric 2



5.2 Friendship Ranking

Given a user pair, its friendship ranking can be calculated as follows:

$$F(m_s, m_t, m_l) = F_s(m_s) + F_t(m_t) + F_l(m_l). \quad (30)$$

A larger value of $F(m_s, m_t, m_l)$ indicates that the two users would have a higher probability to be friends. For any given user, the system can calculate the friendship ranking value with other users and returns a list of high rank users as recommended friends to that user.

5.3 Evaluation

The proposed friendship model was validated using the changes of friends lists from April 14, 2009, to October 4, 2009. A total of 19,092 new friend pairs were found. Then randomly select another 200,000 non-friend pairs that have no overlaps with the 340,000 non-friend model training pairs. The two groups of user pairs were test sets for model evaluation.

Figure 16 shows the distributions of the ranking values of the two test sets. It is obvious that the distribution of ranking values of friend pairs and non-friend pairs is different. Given a threshold of 3, there were 93.01 % friend pairs whose ranking value is larger than 3. On the other hand, there were 86.32 % non-friend pairs whose ranking value is smaller than the threshold.

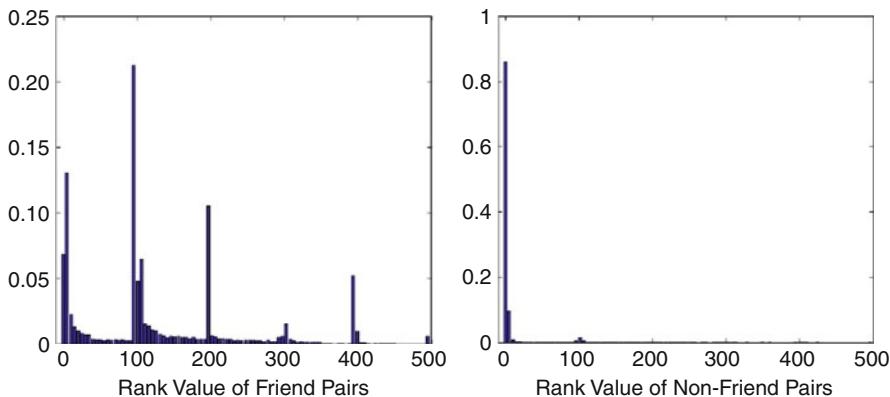


Fig. 16 Histogram of ranking value

5.3.1 Model Prediction Results

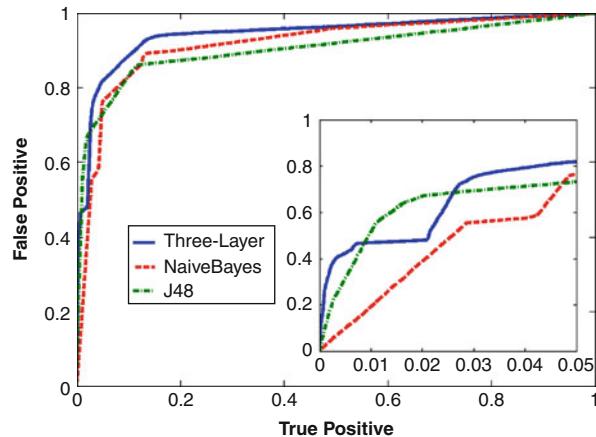
One of the applications of this friendship model is friend recommendation. Given a large user set (there were over 70,000 users in Brightkite, for example), one can use this model to rank all users against a specified user and present a recommended user list to the user as suggested friends based on the ranking values. On the other hand, this model can also be used to predict friendship in user pairs. To evaluate the model, if most of user pairs with high ranking value indeed became friends after a period of time, then the model is valid and effective.

The receiver operating characteristic (ROC) curve was used to compare the ranking performance of the proposed model with other classical data mining algorithms, as shown in Fig. 17. The ranking performance can also be considered as accuracy of prediction.

To plot the ROC curve, first calculate the ranking values of all test sets (19,092 new friend pairs + 200,000 non-friend pairs) and sort them in decreasing order of ranking values. If two pairs have the same ranking value, their order is random. Going through from top to bottom in the sorted list, draw points in the ROC figure from original point (0,0). If the current ranked user pair is indeed a friend, move the cursor along the y -axis with the normalized distance $\Delta y = 1/19,092$ and draw a point. If the current ranked user pair is not a friendship, move along the x -axis with distance $\Delta x = 1/200,000$ and draw a point. After drawing a point, check the next user pair and repeat the process until no more data is left. Viewing the ranking result as friendship prediction, the x -axis denotes the false positives and y -axis denotes the true positives.

For perfect friendship ranking, all the true friend pairs should aggregate at the top of the ranking values, the ROC curve thus yields the line from original point (0,0) to the upper left corner (0,1) and to the upper right corner (1,1). A completely random ranking, however, would give a diagonal line from the left bottom to the top right corner.

Fig. 17 ROC curves of friendship ranking



The non-diagonal solid line is the ROC curve for the proposed model. It can be seen that there were more than 80 % true friend pairs against about 10 % false positives. In other words, by using the friendship ranking, about 80 % friend pairs aggregate at the top 10 % of the data set.

The performance of the three-layer friendship model is superior to the classical data mining algorithms. In particular, the standard probabilistic naive Bayes classification algorithm and the C4.5 revision 8 decision tree learner algorithm (J48) were carried out on the same data sets used by the three-layer friendship model. Figure 17 also depicts the performance of these two algorithms, where the dashed line denotes the performance of naive Bayes algorithm and the dotted line denotes the performance of the J48 decision tree algorithm. The ROC area was calculated to quantify the overall performance, which is the area below the ROC curve. Of course, the ROC area of the perfect ranking is 1 and the random ranking is 0.5. Table 1 shows the comparison results of the three algorithms. Though the results are close, the three-layer model has a larger ROC area than the other two models, indicating better performance.

For a friend recommendation application, it is more interesting to compare user pairs for *high ranking values*. This is because when a recommended list of friends is provided to a user, they would probably only browse the top, say 100, suggestions and ignore the rest. Considering more than 70,000 users in Brightkite, the top 1 % or even top 0.1 % recommended users are the most important.

To show the performance of the three algorithms in the part of high ranking values, it suffices to show a sub-figure of magnifying ROC curves in Fig. 17. This figure includes the three ROC curves of the x -axis from 0 to 0.05. The three-layer friendship model shows a superior performance over the other two algorithms in most cases, where J48 is better than naive Bayes.

Table 2 shows the detailed numbers on the friend-pair prediction against different non-friend pairs (false positive rate). Each row indicates prediction results of the three algorithms in a given top user pairs. For example, the second row of Table 2

Table 1 The ROC area of three algorithms

	Multilayered	Naive Bayes	J48 tree
	0.906	0.882	0.869

Table 2 Predication performance comparisons

Top	Friend pairs number and percentage			J48 tree
	Multilayer	Naive Bayes		
10,000	8,672	86.7 %	6,498	65.0 %
5,000	4,804	96.1 %	3,277	65.5 %
1,000	982	98.2 %	639	63.9 %
500	487	97.4 %	324	64.8 %
100	98	98.0 %	68	68.0 %
50	49	98.0 %	37	74.0 %

indicates that in the top 100 user pairs with high ranking values, the three-layer model successfully predicted 98 friend pairs, which means that in those ranked list, there are 98 true friend pairs in the top 100 ranked pairs and the true friend pairs percentage is 98 %. At the same time, the J48 tree found 90.0 % true friend pairs and the naive Bayes found only 68.0 %. In the third row from bottom of [Table 2](#), the three-layer model successfully predicted 487 true friend pairs in the top 500 ranked pairs. Though the three algorithms have close overall prediction performance, [Table 1](#) shows that the three-layer friendship model achieved better prediction accuracy than the other two classical algorithms, especially for the part of high ranking values.

5.3.2 Location Metric

To explore the relationship between location-based metric and user friendship for location-based OSNs, the ROC area of the proposed model without the location layer was calculated. That is, the model has only two layers: the social graph and tag graph. The ROC area is 0.758, which incurs substantial performance reduction compared with the three-layer model. [Figure 18](#) shows the comparison of two ROC curves.

Additionally, one can use the *information gain* to quantify the impact of different attributes to friendship prediction. In addition to the three attributes of distance, tags, and locations, one may also calculate the information gain of gender difference and age difference in friendship prediction. [Table 3](#) shows the results.

It shows that the location-based metric provided comparable information gain as the social graph distance in friendship prediction. The tag metric has smaller information gain than the other two metrics, possibly for the reason that a large number of users used tags out of the top 1,000 popular ones, and so their tag metrics become zero. The gender and age attributes, however, provide few information gain

Fig. 18 Comparison of ROC curves with/without location metric

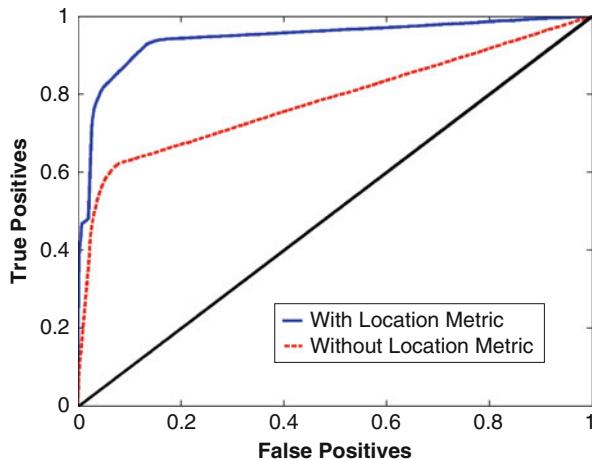


Table 3 Information gain of different attributes

Attributes	Info. gain
Social dist.	0.549
Loc. metric	0.405
Tag metric	0.034
Gender diff	0.020
Age diff	0.011

and so do not help predict friendship. One of a possible reason is that a large number of users hide such information for privacy consideration.

6 Conclusion

This chapter introduces and describes four major issues in the application research of online social networks. They are influence maximization, community detection, data collection, and multilayer friendship modeling. The first two are combinatorial optimization problems. The last two study how to collect data and determine friendships, which are needed for studying the first two issues from a holistic point of view. For each issue, this chapter provides theoretical results and approximation algorithms. Online social networks provide a new platform of rich applications for fruitful research of combinatorial optimizations.

Acknowledgements The author “Jie Wang” was supported in part by the NSF under grants CNS-0958477 and CNS-1018422. The author “You Li” was supported in part by the NSF under grant CNS-1018422. The author “Jun-Hong Cui” was supported in part by the US Department of Education under Grants P200A100141 and P200A090342. The author “Benyuan Liu” was supported in part by the NSF under grant CNS-0953620. The author “Guanling Chen” was supported in part by the NSF under grant IIS-0917112.

Cross-References

- ▶ Greedy Approximation Algorithms
- ▶ Network Optimization
- ▶ Social Structure Detection

Recommended Reading

1. Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, H. Jeong, Analysis of topological characteristics of huge online social networking services, in *Proceedings of the International World Wide Web Conference (WWW)*, Banff, 2007
2. L. Becchetti, C. Castillo, D. Donato, A. Fazzone, A comparison of sampling techniques for web graph characterization, in *Workshop on Link Analysis: Dynamics and Static of Large Networks (LinkKDD)*, Philadelphia, 2006
3. F. Benevenuto, T. Rodrigues, M. Cha, V. Almeida, Characterizing user behavior in online social networks, in *Proceedings of the ACM SIGCOMM Internet Measurement Conference (IMC)*, Chicago, 2009
4. R.S. Burt, Positions in networks. *Soc. Forces* **55**, 93–122 (1976)
5. J. Caverlee, S. Webb, A large-scale study of MySpace: observations and implications for online social networks, in *Proceeding of International Conference on Weblogs and Social Media*, Seattle, 2008
6. W. Chen, Y. Wang, S. Yang, Efficient influence maximization in social networks, in *KDD*, Paris, 2009
7. W. Chen, C. Wang, Y. Wang, Scalable influence maximization for prevalent viral marketing in large scale social networks, in *KDD*, Washington, DC, 2010
8. W. Chen, Y. Yuan, L. Zhang, Scalable influence maximization in social networks under the linear threshold model, in *ICDM*, Sydney, 2010
9. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms* (MIT, Cambridge, 2009)
10. P. Domingos, M. Richardson, Mining the network value of customers, in *KDD*, San Diego, 2001
11. W. Gauvin, B. Ribeiro, B. Liu, D. Towsley, J. Wang, Measurement and gender-specific analysis of user publishing characteristics on MySpace. *IEEE Netw.* **24**(5), 38–43 (2010)
12. W. Gauvin, Providing anonymity for online social network users. Ph.D. Dissertation, Department of Computer Science, University of Massachusetts Lowell, 2011
13. M. Girvan, M.E.J. Newman, Community structure in social and biological networks. *PNAS* **99**(12), 7821–7826 (2002)
14. M. Gjoka, M. Kurant, C.T. Butts, A. Markopoulou, Walking in Facebook: a case study of unbiased sampling of OSNs, in *Proceedings of the IEEE INFOCOM*, San Diego, 2010
15. C. Gkantsidis, M. Mihail, A. Saberi, Random walks in peer-to-peer networks, in *Proceedings of the IEEE INFOCOM*, Hong Kong, 2004
16. J. Goldenberg, B. Libai, E. Muller, Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Mark. Lett.* **12**(3), 211–223 (2001)
17. J. Goldenberg, B. Libai, E. Muller, Using complex systems analysis to advance marketing theory development. *Acad. Mark. Sci. Rev.* **9**, 1–19 (2001)
18. W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**(1), 97–109 (1970)
19. M.R. Henzinger, A. Heydon, M. Mitzenmacher, M. Najork, On near-uniform URL sampling, in *Proceedings of the International World Wide Web Conference (WWW)*, Amsterdam, 2000
20. D. Kempe, J.M. Kleinberg, E. Tardos, Maximizing the spread of influence through a social network, in *KDD*, Washington, DC, 2003

21. D. Kempe, J.M. Kleinberg, E. Tardos, Influential nodes in a diffusion model for social networks, in *ICALP*, Lisbon, 2005
22. B. Krishnamurthy, P. Gill, M. Arlitt, A few chirps about Twitter, in *Proceedings of the First Workshop on Online Social Networks (WOSN)*, Seattle, 2008
23. M. Kumura, K. Saito, Tractable models for information diffusion in social networks, in *ECML PKDD*, Berlin, 2006
24. M. Kurant, A. Markopoulou, P. Thiran, On the bias of BFS (Breadth-First-Search) sampling, in *Proceedings of the 22nd International Teletraffic Congress (ITC'22)*, Amsterdam, 2010
25. S.H. Lee, P.-J. Kim, H. Jeong, Statistical properties of sampled networks. *Phys. Rev. E* **73**, 102–109 (2006)
26. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N.S. Glance, Cost-effective outbreak detection in networks, in *KDD*, San Jose, 2007
27. N. Li, G. Chen, Multi-layer friendship modeling for location-based mobile social networks, in *MobiQuitous*, Toronto, 2009
28. Y. Li, Z. Fang, J. Wang, An efficient random-walk algorithm for finding network communities. Technical report, Department of Computer Science, University of Massachusetts, Lowell, 2012
29. L. Lovasz, Random walks on graphs, a survey. *Combinatorics* **2**, 1–46 (1993)
30. N. Metropolis, M. Rosenbluth, A. Rosenbluth, A. Teller, E. Teller, Equations of state Calculations by fast computing machines. *J. Chem. Phys.* **21**(6), 1087–1092 (1953)
31. A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, B. Bhattacharjee, Measurement and analysis of online social networks, in *Proceedings of the 5th ACM/USENIX Internet Measurement Conference (IMC'07)*, San Diego, 2007
32. R. Narayamm, Y. Narahari, A shapley value based approach to discover influential nodes in social networks. *IEEE Trans. Autom. Sci. Eng.* **8**,(1), 130–147 (2011)
33. G. Nemhauser, L. Wolsey, M. Fisher, An analysis of the approximations for maximizing submodular set functions. *Math. Program.* **14**, 265–294 (1978)
34. M.E.J. Newman, Detecting community structure in networks. *Eur. Phys. J. B* **38**, 321–330 (2004)
35. M.E.J. Newman, Modularity and community structure in networks. Technical report, physics/0602124, 2006
36. A. Rasti, M. Torkjazi, R. Rejaie, N. Duffield, W. Willinger, D. Stutzbach, Respondent-driven sampling for characterizing unstructured overlays, in *Proceedings of the IEEE Infocom Mini-Conference*, Rio de Janeiro, 2009
37. B. Ribeiro, D. Towsley, Estimating and sampling graphs with multidimensional random walks, in *Proceeding of ACM SIGCOMM Internet Measurement Conference (IMC)*, Melbourne 2010
38. M. Richardson, P. Domingos, Mining knowledge-sharing sites for viral marketing, in *Proceedings of the 8th International Conference on Knowledge Discovery and Data Mining*, Edmonton, 2002
39. D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, W. Willinger, On unbiased sampling for unstructured peer-to-peer networks, in *Proceeding of ACM SIGCOMM Internet Measurement Conference (IMC)*, Rio de Janeriro, 2006
40. L.G. Valiant, The complexity of enumeration and reliability problems. *SIAM J. Comput.* **8**(3), 410–421 (1979)
41. C. Wilson, B. Boe, A. Sala, K. Puttaswamy, B. Zhao, User interactions in social networks and their implications, in *Proceedings of the EuroSys*, Nuremberg, 2009

Optimizing Data Collection Capacity in Wireless Networks

Shouling Ji, Jing (Selena) He and Yingshu Li

Contents

1	Introduction	2505
2	Problems, Motivations, and Main Contributions	2506
3	Capacity Issues in Wireless Networks	2510
3.1	Capacity for Single-Radio Single-Channel Wireless Networks	2510
3.2	Capacity for Multichannel Wireless Networks	2512
3.3	Remarks	2513
4	Network Model and Preliminaries	2513
4.1	Interference Model	2514
4.2	Network Model	2515
4.3	Routing Tree	2516
4.4	Vertex Coloring Problem	2519
5	Capacity of Snapshot Data Collection	2519
5.1	Scheduling Algorithm for Snapshot Data Collection	2519
5.2	Capacity Analysis	2524
5.3	Discussion	2527
6	Capacity of Continuous Data Collection	2527
6.1	Compressive Data Gathering (CDG)	2527
6.2	Pipeline Scheduling	2528
6.3	Capacity Analysis	2530
6.4	Capacity of the Pipeline Scheduling Algorithm in Single-Radio Multichannel WSNs	2535
7	Simulations and Results Analysis	2538
7.1	Performance of MPS	2538
7.2	Performance of PS	2540
7.3	Impacts of N and M	2542

S. Ji (✉) • Y. Li

Department of Computer Science, Georgia State University, Atlanta, GA, USA
e-mail: sji@cs.gsu.edu; yli@cs.gsu.edu

J.S. He

Department of Computer Science, Kennesaw State University, Kennesaw, GA, USA
e-mail: jhe4@kennesaw.edu

8 Conclusion.....	2542
Cross-References.....	2544
Recommended Reading.....	2544

Abstract

Data collection is an important operation of wireless sensor networks (WSNs). The performance of data collection can be measured by its achievable *network capacity*. Most existing works focus on the capacity of *unicast*, *multicast*, or/and *snapshot data collection* (SDC) in single-radio single-channel wireless networks, and no works dedicatedly consider the *continuous data collection* (CDC) capacity for WSNs under the protocol interference model. In this chapter, firstly, a *multipath scheduling* algorithm for SDC in single-radio multichannel WSNs is proposed. Theoretical analysis of the multipath scheduling algorithm shows that its achievable network capacity is at least $\frac{W}{2\lceil(1.81\rho^2+c_1\rho+c_2)/H\rceil}$, where W is the channel bandwidth, H is the number of available orthogonal channels, ρ is the ratio of the interference radius over the transmission radius of a node, $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$, and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$, which is a tighter lower bound compared with the previously best result which is $\frac{W}{8\rho^2}$ (Chen et al. (2010) Capacity of data collection in arbitrary wireless sensor networks. In: IEEE INFOCOM, San Diego, USA). For CDC, an intuitive method is to pipeline existing SDC methods. However, such an idea cannot actually improve the network capacity. The reason for this failure is discussed and a novel *pipeline scheduling* algorithm for CDC in dual-radio multichannel WSNs is proposed. This pipeline scheduling algorithm significantly speeds up the data collection process and achieves a capacity of

$$\begin{cases} \frac{nW}{12M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M\Delta_e \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases},$$

where n is the number of the sensors, M is a constant value and usually $M \ll n$, Δ_e is the maximum number of the leaf nodes having a same parent in the routing tree (i.e., data collection tree), $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$, and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$. Furthermore, for completeness, the performance of the proposed pipeline scheduling algorithm in single-radio multichannel WSNs is also analyzed, which shows that for a long-run CDC, the lower bound of the achievable asymptotic network capacity is

$$\begin{cases} \frac{nW}{16M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M(\Delta_e + 4) \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases}.$$

Extensive simulation results indicate that the proposed algorithms improve the network capacity significantly compared with existing works.

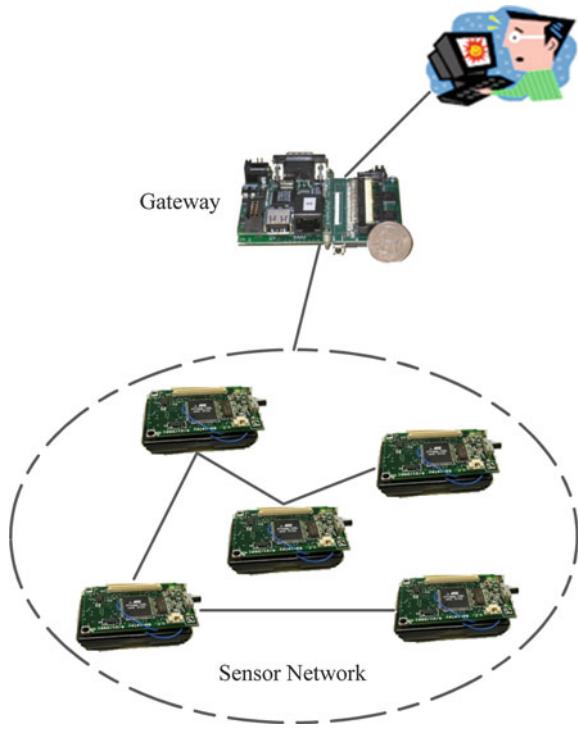
1 Introduction

Wireless sensor networks (WSNs) as shown in Fig. 1, consisting of small nodes with sensing, computation, and wireless communication capabilities, attract many interests from the research community recently.

Generally speaking, WSNs are deployed for monitoring and controlling systems where human intervention is not desirable or feasible. Therefore, WSNs are widely used in many applications [2, 15, 34]:

- *Disaster relief applications.* A typical scenario is wildfire detection, where a WSN consisting of sensor nodes that are equipped with thermometers and can determine their own locations (relative to each other or in absolute coordinates) is deployed. In such an application, sensor nodes should be cheap enough to be considered disposable since a large number is necessary.
- *Environment monitoring.* WSNs can be used to monitor environments. For instance, with respect to monitor chemical pollutants, a possible application is monitoring garbage dump sites. The main advantage of WSNs here is the long-term and unattended monitoring.
- *Intelligent buildings.* Buildings waste vast amounts of energy because of inefficient monitoring of humidity, ventilation, and/or air conditioning usage. A better, real-time, and high-resolution monitoring of temperature, airflow, humidity, and other physical parameters in a building by means of WSNs can considerably increase the comfort level of inhabitants and reduce the energy consumption.
- *Facility management.* In the management of facilities larger than a single building, WSNs also have a wide range of possible applications. Simple examples include keyless entry applications where people wear badges that allow a WSN to check which person is allowed to enter which areas of a larger company site. This example can be extended to the detection of intruders.
- *Machine surveillance and structural health monitoring (SHM).* One idea is to fix sensor nodes to difficult-to-reach areas of machinery where they can detect vibration patterns that indicate the need for maintenance.
- *Precision agriculture.* Applying WSNs to agriculture allows precise irrigation and fertilizing by placing humidity/soil composition sensors into the fields.
- *Medicine and health care.* The use of WSNs in healthcare applications is potentially very beneficial. Possibilities range from postoperative and intensive care, where sensors are directly attached to patients.
- *Logistics.* In several different logistics applications, it is conceivable to equip goods with simple sensors that allow a simple tracking of these objects during transportation or facilitate inventory tracking in stores or warehouses.
- *Traffic control.* Sensors embedded in the streets or road sides can gather information about traffic conditions at a much finer-grained resolution. Such deployed sensors could also interact with the cars to exchange danger warnings about road conditions or traffic jams ahead.

Fig. 1 A wireless sensor network (WSN)



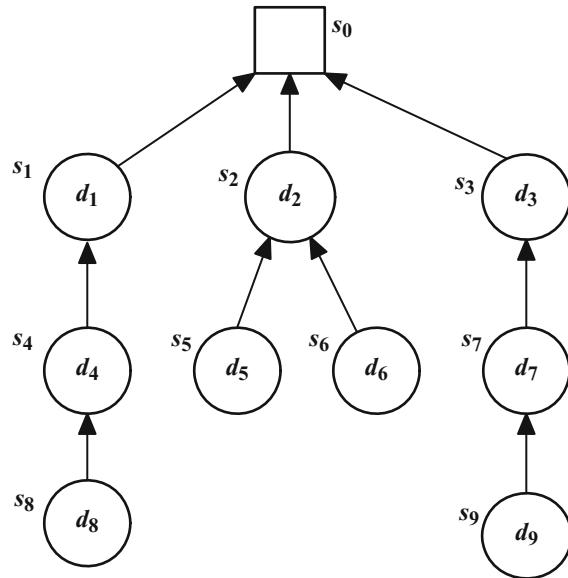
2 Problems, Motivations, and Main Contributions

As mentioned in the previous section, WSNs are mainly used for gathering data from the physical world. Data gathering can be categorized as *data aggregation* [1, 28, 33, 44, 57], which obtains aggregated values from WSNs, e.g., maximum, minimum, or/and average value of all the data, and *data collection* [11, 12, 31, 32, 46, 65, 66], which gathers all the data from a network without any data aggregation. For data collection, the union of all the sensing values from all the sensors at a particular time instance is called a *snapshot* [12, 31]. The problem of collecting all the data of one snapshot is called *snapshot data collection* (SDC). Similarly, the problem of collecting multiple continuous snapshots is called *continuous data collection* (CDC).

Taking the WSN shown in Fig. 2 as an example, this WSN consists of one sink node denoted by s_0 and nine sensors denoted by s_i ($1 \leq i \leq 9$). Assume the data value (packet) produced at sensor s_i ($1 \leq i \leq 9$) at a particular time instance is d_i ($1 \leq i \leq 9$) as shown in Fig. 2. Then, the set $\{d_i \mid 1 \leq i \leq 9\}$ is called a snapshot. In a data aggregation task, if the aggregation function is $\max(\cdot)$,¹ then

¹Under the $\max(\cdot)$ function, the maximize value of a snapshot will be collected by the sink, i.e., $\max(\{d_1, d_2, \dots, d_n\}) = d_{\max}$ such that $d_{\max} \in \{d_1, d_2, \dots, d_n\}$ and for $\forall d_i \in \{d_1, d_2, \dots, d_n\}$, $d_{\max} \geq d_i$.

Fig. 2 Explanation of *data aggregation and data collection*



only the maximize value of set $\{d_i \mid 1 \leq i \leq 9\}$ will be collected by the sink s_0 . On the other hand, in a data collection task, all the nine values of set $\{d_i \mid 1 \leq i \leq 9\}$ will be collected by the sink s_0 .

Different from wired networks, WSNs suffer from the interference problem, which degrades the network performance. Consequently, *network capacity*, which can reflect the achievable data transmission rate, is usually used as an important measurement to evaluate network performance. Particularly, for a data collection WSN, the average data receiving rate at the sink during the data collection process, referred to as *data collection capacity* [12, 31] (formally defined in Sect. 4.2), is used to measure its achievable network capacity, i.e., data collection capacity reflects how fast data have been collected by the sink. In this chapter, the snapshot and continuous data collection problems (formally defined in Sect. 4.2), as well as their achievable capacities in single/dual-radio multichannel WSNs, are studied.

After the first work [24], extensive works emerged to study the network capacity issue for variety of network scenarios, e.g., multicast capacity [40, 41, 48], unicast capacity [52, 53], broadcast capacity [42], and SDC capacity [11, 12, 46, 66]. Most of the previous studies on network capacity are for single-radio single-channel WSNs [4, 8–12, 20, 22, 24, 40, 41, 46, 48, 51–53, 55, 59, 60, 63, 66], where a network consists of a number of nodes which have only one radio and all the nodes communicate over a common single channel. Because of the inherent limitations of such networks, transmissions suffer from the *radio confliction* problem [13, 23, 37, 47, 56] and the *channel interference* problem [3, 6, 7, 14, 16–19, 21, 25, 29, 29, 35–37, 43, 45, 49, 54, 61, 64] seriously. This degrades network performance significantly. The radio confliction problem is caused by the fact that each node is equipped with only one radio, which means a node can only work on a *half-duplex* mode, i.e., this node cannot receive and transmit data simultaneously. The channel

Fig. 3 A dual-radio sensor node [30, 38]



interference problem is caused by all the nodes working over a common channel. When one node transmits data, all the other nodes within its interference radius cannot receive any other data and all the other transmissions that interfere with this transmission cannot be carried out simultaneously.

Fortunately, many current off-the-shelf sensor nodes are capable of working over multiple orthogonal channels, e.g., IEEE 802.11 b/g standard supports 3 orthogonal channels and IEEE 802.11a standard supports 13 orthogonal channels [5, 13], respectively, which can greatly mitigate the channel interference problem. Furthermore, with the development of hardware technologies and the decreasing of hardware cost, a sensor node can be equipped with multiple radios. As shown in Fig. 3, it is a dual-radio sensor node developed by HLJU and HIT [30, 38]. This helps with solving the radio confliction problem. Therefore, multi-radio multichannel WSNs currently begin to attract more and more interests from researchers [13, 23, 37, 47, 56].

Different from the previous works which investigate the capacity issues for single-radio single-channel WSNs, this chapter studies the network capacity problem for both CDC and SDC in dual-radio multichannel WSNs under the *protocol interference model*. Similarly as [12], in this chapter, the *data collection capacity* is defined as the average data rate at the sink to continuously receive data from sensor nodes. Two channel scheduling algorithms for both CDC and SDC are proposed, respectively. Meanwhile, the results obtained in this chapter can be extended to WSNs under the *physical interference model* [42]. The motivation of the work in this chapter lies on the fact that dual-radio multichannel WSNs can make nodes work in a *full-duplex* manner without incurring high hardware cost, while the channel interference problem can be mitigated significantly. Most of the previous works focus on addressing the SDC capacity problem, while the work in this chapter is the dedicated one investigating the CDC capacity problem in detail under the protocol interference model. Besides, the work in this chapter is suitable for dual-radio multichannel WSNs. The main contributions of this chapter are as follows:

- For the SDC problem in single-radio multichannel WSNs, a new *multipath scheduling* algorithm is proposed. Furthermore, theoretical analysis of the multipath scheduling algorithm shows that this algorithm can achieve the order-optimal network capacity $\Theta(W)$ and has a tighter lower bound $\frac{W}{2\lceil(1.81\rho^2+c_1\rho+c_2)/H\rceil}$ compared with the previously best result in [12], which is $\frac{W}{8\rho^2}$, where W is the channel bandwidth, H is the number of available orthogonal channels in a WSN, ρ is the ratio of the interference radius over the transmission radius of a node, $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$, and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$.

- Currently, there are no solutions dedicated for the CDC capacity problem. Therefore, a novel *pipeline scheduling* algorithm that combines *compressive data gathering* (CDG) [46] and *pipeline* together is proposed, which significantly improves the CDC capacity for dual-radio multichannel WSNs. Theoretical analysis shows that the achievable asymptotic network capacity of this pipeline scheduling algorithm in a long-run is

$$\begin{cases} \frac{nW}{12M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M\Delta_e \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases},$$

where n is the number of the sensors in a WSN, M is a constant value determined by the correlations of the data in a snapshot and usually $M \ll n$, Δ_e is the maximum number of the leaf nodes having a same parent in the routing tree (i.e., data collection tree), $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$, and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$. A straightforward upper bound of data collection of a dual-radio WSN is $2W$, since a dual-radio sink can simultaneously receive two data packets at most. Whereas, thanks to the benefit brought by the pipeline technique and CDG, analysis shows that the proposed pipeline scheduling algorithm can even achieve a capacity higher than $2W$.

- For completeness, the performance of the proposed pipeline scheduling algorithm in single-radio multi-channel WSNs is also examined, denoted by the *single-radio-based pipeline scheduling* algorithm. Theoretical analysis shows that for a long-run CDC, the lower bound of the achievable asymptotic network capacity of the single-radio-based pipeline scheduling algorithm for single-radio multi-channel WSNs is

$$\begin{cases} \frac{nW}{16M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M(\Delta_e + 4) \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases}.$$

- The simulation results indicate that the proposed algorithms have a better SDC capacity compared with the previously best works. Particularly, when $\rho = 2$ and $H = 3$, for SDC in a WSN with 4,000 nodes, the improvements of the capacity of the proposed multipath scheduling algorithm are 74.3 and 29 % compared with BFS [12] and SLR [6], respectively. For CDC in a WSN with 10,000 nodes, the proposed pipeline scheduling algorithm achieves a capacity 7.6 times of that of CDG [46], 22.8 times of that of BFS [12], and 19.4 times of that of SLR [6], respectively.

The rest of this chapter is organized as follows: Sect. 3 summarizes the advances in studying network capacity issues for wireless networks. Section 4 introduces the interference model, the network model, and the preliminaries. The multi-channel scheduling algorithm for SDC in single-radio multichannel WSNs is proposed and analyzed in Sect. 5. Section 6 presents a novel pipeline scheduling algorithm for CDC and its theoretical achievable asymptotic network capacity. For completeness, the performance of the pipeline scheduling algorithm in single-radio multichannel WSNs is also analyzed in Sect. 6. The simulations to validate the performance of the proposed algorithms are shown in Sect. 7. Section 8 concludes this chapter and points out possible future research directions.

3 Capacity Issues in Wireless Networks

In this section, the advances in studying network capacity issues for wireless networks are surveyed and summarized. At the end of this section, the differences between the work in this chapter and existing works are also compared.

3.1 Capacity for Single-Radio Single-Channel Wireless Networks

Following the seminal work [24] by Gupta and Kumar, extensive works emerged to study the network capacity issue. The works in [4, 8, 9, 22] focus more on the MAC layer to improve the network capacity. In [8], the network capacity with random-access scheduling is investigated. In this work, each link is assigned a channel access probability. Based on which some simple and distributed channel access strategies are proposed. Another similar work is [9], in which the authors studied the capacity of CSMA wireless networks. The authors formulated the models of a series of CSMA protocols and study the capacity of CSMA scheduling versus TDMA scheduling. They also proposed a CSMA scheme which combines a backbone-peripheral routing scheme and a dual carrier-sensing and dual-channel scheme. In [22], the authors considered the scheduling problem where all the communication requests are single-hop and all the nodes transmit at a fixed power level. They proposed an algorithm to maximize the number of links in one time slot. Unlike [22], the authors in [4] considered the power-control problem. A family of approximation algorithms were presented to maximize the capacity of arbitrary wireless networks.

The works in [40, 41, 48, 52, 53, 59] study the multicast and/or unicast capacity of wireless networks. The multicast capacity for wireless ad hoc networks under the protocol interference model and the Gaussian channel model is investigated in [40, 41], respectively. In [40], the authors showed that the network multicast capacity is $\Theta\left(\sqrt{\frac{n}{\log n}} \cdot \frac{W}{k}\right)$ when $k = O\left(\frac{n}{\log n}\right)$ and is $\Theta(W)$ when $k = \Omega\left(\frac{n}{\log n}\right)$, where W is the bandwidth of a wireless channel, n is the number of the nodes in a network, and k is the number of the nodes involved in one multicast session. In [41], the authors showed that when $k \leq \theta_1 \frac{n}{(\log n)^{2\alpha+6}}$ and $n_s \geq \theta_2 n^{1/2+\beta}$, the

capacity that each multicast session can achieve is at least $c_8 \frac{\sqrt{n}}{n_s \sqrt{k}}$, where k is the number of the receivers in one multicast session; n is the number of the nodes in the network; n_s is the number of the multicast sessions; θ_1 , θ_2 , and c_8 are constants; and β is any positive real number. Another similar work [48] studies the upper and lower bounds of multicast capacity for hybrid wireless networks consisting of ordinary wireless nodes and multiple base stations connected by a high-bandwidth wired network. Considering the problem of characterizing the unicast capacity scaling in arbitrary wireless networks, the authors proposed a general cooperative communication scheme in [52]. The authors also presented a family of schemes that address the issues between multi-hop and cooperative communication when the path loss exponent is greater than 3. In [53], the authors studied the balanced unicast and multicast capacity of a wireless network consisting of n randomly placed nodes and obtained the characterization of the scaling of the n^2 -dimensional balanced unicast and $n2^n$ -dimensional balanced multicast capacity regions under the Gaussian fading channel model. A more general (n, m, k) -casting capacity problem was investigated in [59], where n , m , and k denote the total number of the nodes in the network, the number of destinations for each communication group, and the actual number of communication group members that receive information, respectively. In [59], the upper and lower bounds for the (n, m, k) -cast capacity were obtained for random wireless networks.

In [29], the authors investigated the network capacity scaling in mobile wireless ad hoc networks under the protocol interference model with infrastructure support. In [64], the authors studied the network capacity of hybrid wireless networks with directional antenna and delay constraints. Unlike previous works, the authors in [36] studied the capacity of multi-unicast for wireless networks from the algorithmic aspects, and they designed provably good algorithms for arbitrary instances. The broadcast capacity of wireless networks under the protocol interference model is investigated in [35], where the authors derived the upper and lower bounds of the broadcast capacity in arbitrary connected networks. When the authors in [45] studied the data gathering capacity of wireless networks under the protocol interference model, they concerned the per source node throughput in a network where a subset of nodes send data to some designated destinations while other nodes serve as relays. To gather data from WSNs, a multi-query processing technology is proposed in [14]. In that work, the authors considered how to obtain data efficiently with data aggregation and query scheduling. Under different communication organizations, the authors in [17] derived the many-to-one capacity bound under the protocol interference model. Another work that studied the many-to-one capacity issue for WSNs is [49], where the authors considered to use data compression to improve the data gathering efficiency. They also studied the relation between a data compression scheme and the data gathering quality. In [19], the authors studied the scaling laws of WSNs based on an antenna-sharing idea. In that work, the authors derived the many-to-one capacity bounds under different power constraints. In [61], the authors studied the multicast capacity of MANETs under the physical interference model, called MotionCast. They considered the network capacity of MANETs in two particular situations, which are the LSRM (local-based speed-restricted) model and the GSRM (global-based speed-restricted) model.

The multi-unicast capacity of wireless networks is studied in [18] via percolation theory. By applying percolation theory, the authors obtained a tighter capacity bound for arbitrary wireless networks.

The SDC capacity of WSNs is studied in [10–12, 31, 46, 51, 66]. In [66], the authors considered the collision-free delay-efficient data-gathering problem. Furthermore, they proposed a family of path scheduling algorithms to collect all the data to the sink and obtained the network capacity through theoretical analysis. The authors of [12, 31] extended the work of [66]. They derived tighter upper and lower bounds of the capacity of data collection for arbitrary WSNs. Luo et al. [46] is a work studying how to distribute the data collection task to the entire network to achieve load balancing. In that work, all the sensors transmit the same number of data packets during the data collection process. In [10, 11, 32], the authors investigated the capacity of data collection for WSNs under the protocol interference model and the physical interference model, respectively. They proposed a grid partition method which divides the network into small grids to collect data and then derived the network capacity. The worst-case capacity of data collection of a WSN is studied in [51] under the physical and protocol interference models.

The capacity and energy efficiency of wireless ad hoc networks with multi-packet reception under the physical interference model are investigated in [60]. With the multi-packet reception scheme, a tight bound of the network capacity is obtained. Furthermore, the authors showed that a trade-off can be made between increasing the transport capacity and decreasing the energy efficiency. In [63], a scheduling partition method for large-scale wireless networks is proposed. This method decomposes a large network into many small zones, and then localized scheduling algorithms which can achieve the order optimal capacity as a global scheduling strategy are executed in each zone independently. A general framework to characterize the capacity of wireless ad hoc networks with arbitrary mobility patterns is studied in [20]. By relaxing the “homogeneous mixing” assumption in most existing works, the capacity of a heterogeneous network is analyzed. Another work [55] studies the relationship between the capacity and the delay of mobile wireless ad hoc networks, where the authors studied how much delay must be tolerated under a certain mobile pattern to achieve an improvement of the network capacity.

3.2 Capacity for Multichannel Wireless Networks

Since wireless nodes can be equipped with multiple radios, and each radio can work over multiple orthogonal channels, multi-radio multichannel wireless networks attract many research interests recently [3, 21, 25, 43]. In [21], the authors studied the data aggregation issue in multichannel WSNs under the protocol interference model. Particularly, they designed a constant factor approximation scheme for data aggregation in multi-channel WSNs modeled by unit disk graphs (UDGs). Unlike [21], this chapter studies the snapshot/continuous data collection capacity issue for WSNs. In [3, 25, 43], the authors investigated the joint channel assignment and routing problem for multi-radio wireless mesh networks, software-defined radio

networks, and multichannel ad hoc wireless networks, respectively. They focused on the channel assignment and routing issues, while in data collection, especially continuous data collection, one focuses on how to solve the data accumulation problem at the sensors near the sink to improve the achievable network capacity.

The issue of the capacity of multichannel wireless networks also attracts a lot of attention [6, 7, 16, 37, 54]. In [6, 7], the authors studied the connectivity and capacity problem of multichannel wireless networks. They considered a multi-channel wireless network under constraints on channel switching, proposed some routing and channel assignment strategies for multiple unicast communications and derived the per-flow capacity. The multicast capacity of multi-channel wireless networks is studied in [54]. In this work, the authors represented the upper bound capacity of per multicast as a function of the number of the sources, the number of the destinations per multicast, the number of the interfaces per node, and the number of the available channels. Subsequently, an order-optimal scheduling method is proposed under certain circumstances. In [16], the authors first proposed a multichannel network architecture, called MC-MDA, where each node is equipped with multiple directional antennas, and then obtained the capacity of multiple unicast communications, under arbitrary and random network models. The impact of the number of the channels, the number of the interfaces, and the interface switching delay on the capacity of multichannel wireless networks is investigated in [37]. In this work, the authors derived the network capacity under different situations for arbitrary and random networks.

3.3 Remarks

Unlike the above-mentioned works, this chapter has the following two main characteristics. First, most of the above-mentioned works are specifically for single-radio single-channel wireless networks, while the work in this chapter considers the network capacity for dual-radio multichannel WSNs. Second, the work in this chapter is the dedicated one that investigates the network capacity for CDC in detail under the protocol interference model, whereas most of the previous works study the network capacity for multicast or/and unicast and etc. For the works that study the data collection capacity of wireless networks, they focus on the SDC problem which is a special case of CDC. Compared with them, the methods proposed in this chapter are more universal.

4 Network Model and Preliminaries

In this section, the interference model, the network model, and the assumptions are described; the routing tree used for data collection is constructed; and some necessary preliminaries are made. For the frequently used notations in this chapter, they are listed in [Table 1](#).

Table 1 Notations used in this chapter

Notation	Description
$G(V, E)$	The network topology graph, V is the set of all the nodes, E is the set of all the possible links
n	The number of sensors in a WSN
ρ	The interference radius
$\lambda_1, \dots, \lambda_H$	The H available orthogonal channels
W	The bandwidth of a channel
b	The size of a data packet
t	A time slot
τ	The time consumption of snapshot/continuous data collection
N	The number of snapshots in a continuous data collection task
Υ	The snapshot/continuous data collection capacity
D/C	The set of dominators/connectors
G'	The graph constructed by the nodes in D
L'	The radius of G'
T	The data collection tree
$\mathfrak{R}(A)$	The conflicting graph of A
$\Delta(\cdot)/\delta(\cdot)$	The maximum/minimum degree of a graph
Δ_e	The maximum number of leaf nodes having a same parent in T
$\delta^*(\cdot)$	The inductivity of a graph
β_r	The number of the dominators within a half-disk with radius r
SDC	Snapshot data collection
CDC	Continuous data collection
CDG	Compressive data gathering
STS	Super time slot
CDS	Connected dominating set

4.1 Interference Model

When model the interference in wireless networks, three interference models are frequently used: the *protocol interference model* [12, 31], the *physical interference model* [11], and the *generalized physical interference model* [1, 41]. Assume u and v are two nodes in a wireless network and u is transmitting some data to v . Furthermore, assume \mathcal{T} is set of all the nodes that transmit data to some nodes concurrently with u . Then, the transmission of u and v under these three interference models will be introduced respectively in the following.

4.1.1 Protocol Interference Model

Under the protocol interference model, u can successfully transmit data to v only if

$$\forall w \in \mathcal{T}, w \neq u, \quad \|w - v\| \geq (1 + \theta) \cdot r, \quad (1)$$

where w is any node transmitting data concurrently with u , $\|\cdot\|$ is the Euclidian distance between two nodes, $\theta \geq 0$ is a constant value, and r is the communication radius of the nodes in a wireless network.

4.1.2 Physical Interference Model

Under the physical interference model, u can successfully transmit data to v only if the signal-to-interference-plus-noise ratio (SINR) value at v associated with u is no less than a threshold value as follows:

$$\Lambda(u, v) = SINR_{u,v} = \frac{P_u \cdot \|u - v\|^\eta}{N_0 + \sum_{w \in \mathcal{T}, w \neq u} P_w \cdot \|w - v\|^\eta} \geq \gamma, \quad (2)$$

where P_u (respectively, P_w) is the transmission power of u (respectively, w), $\eta > 2$ is the path loss exponent, N_0 represents the ambient noise, and γ is the threshold SINR value.

4.1.3 Generalized Physical Interference Model

Under the generalized physical interference model, the data receiving rate at v from u is given by

$$\mathcal{R}_{u,v} = W \cdot \log(1 + \Lambda(u, v)), \quad (3)$$

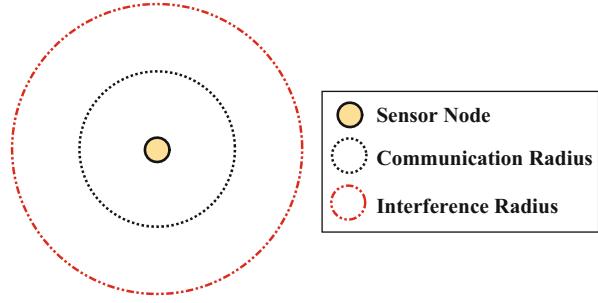
where W is the bandwidth of the channel on which u transmits data to v .

In this chapter, the snapshot and continuous data collection, as well as their achievable capacities, in single/dual-radio multichannel WSNs are investigated under the protocol interference model.

4.2 Network Model

In this chapter, a WSN consisting of n sensors and one sink is considered and represented by a connected undirected graph $G = (V, E)$, where V is the set of all the nodes in the network and E is the set of all the possible links among the nodes in V . Every sensor in the WSN produces one data packet in a snapshot. Each sensor has two radios and each radio has a fixed transmission radius normalized to one and a fixed interference radius, denoted by ρ , $\rho \geq 1$, as shown in Fig. 4. Since the protocol interference model is used, for any receiving node v , v can receive a data packet successfully from a transmitting node u if $\|u - v\| \leq 1$ and there is no other node s satisfying $\|s - v\| \leq \rho$ and trying to transmit a data packet simultaneously over the same channel with u . Furthermore, two links are *interfering links* if at least one transmission over them will fail if they transmit data simultaneously. Each radio can work over H orthogonal channels, denoted by $\lambda_1, \lambda_2, \dots, \lambda_H$, respectively. A fixed data-rate channel model [12] is adopted in this chapter, which means each sensor can transmit at a rate of W bits/second over a wireless channel. The size of every

Fig. 4 Communication radius vs. interference radius



data packet transmitted in the network is set to be b bits. All the transmissions are assumed to be synchronized and the size of a time slot is $t = b/W$ seconds.

The problems studied in this chapter are further formally defined as follows. For a WSN consisting of n sensors and one sink, every sensor produces a data packet with b bits at a particular time instant. The union of all the n data packets produced by the n sensors at a particular time instant is called a *snapshot*. The process to collect all the data of a snapshot to the sink is called *snapshot data collection* (SDC). The *SDC capacity* is defined as $\Upsilon = \frac{nb}{\tau}$, where τ is the time used to collect all the data of a snapshot to the sink, i.e., SDC capacity reflects the average data receiving rate at the sink during SDC. Similarly, the process to collect all the data of N continuous snapshots is called *continuous data collection* (CDC). The *CDC capacity* is defined as $\Upsilon = \frac{Nnb}{\tau}$, where τ now is the time consumption to collect all the data of these N snapshots to the sink, i.e., CDC capacity reflects the average data receiving rate at the sink during CDC. In this chapter, the SDC and CDC problems for WSNs, as well as their achievable network capacities, are studied.²

4.3 Routing Tree

G is a unitdisk graph representing a WSN. The sink s_0 is defined as the *center* of G . The *radius* of G with respect to s_0 is the maximum depth of the *breadth-first search* (BFS) tree rooted at s_0 . For a subset U of V , U is a *dominating set* (DS) of G if every node in V is either an element of U or adjacent³ to at least one node in U . If the subgraph of G induced by U is connected, then U is called a *connected dominating set* (CDS) of G . Since a CDS can serve as a virtual backbone of a WSN, it receives a lot of attention [26, 27, 39, 57, 58, 62] recently.

²In the following of this chapter, the snapshot/continuous data collection capacity and network capacity are used interchangeably without confusion.

³In this chapter, if two nodes u and v are said adjacent/connected, that means u and v are within the communication range of each other, i.e., $\|u - v\| \leq 1$.

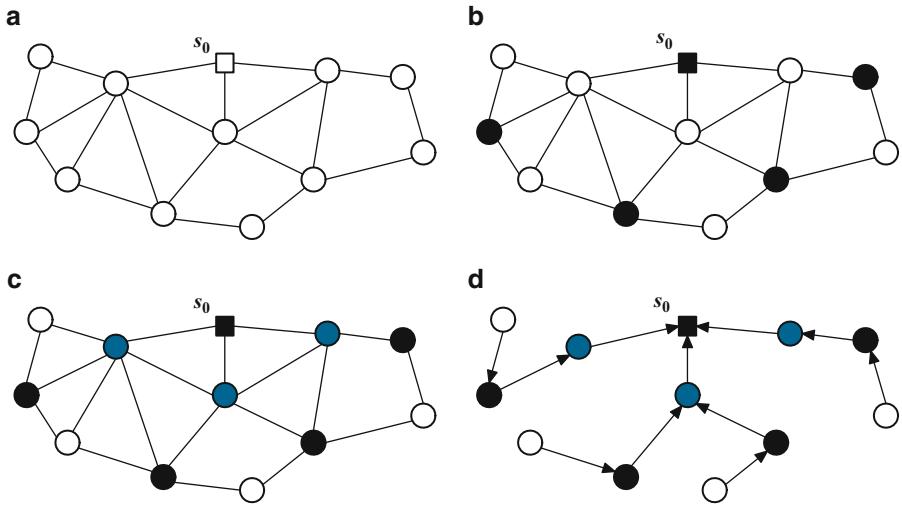
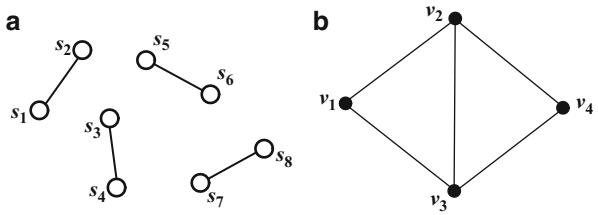


Fig. 5 The construction of a CDS based routing tree. s_0 is the sink. The *black nodes* in (b) are *dominators*, and the *blue nodes* in (c) are *connectors*

Taking the WSN shown in Fig. 5a as an example, a CDS-based routing tree T is built as shown in Fig. 5d using the method proposed in [57]. T is rooted at the sink and can be built according to the following steps. First, construct a BFS tree on G beginning at the sink and obtain a *maximal independent set* (MIS) D according to the search sequence. As shown in Fig. 5b, the set of all the black nodes is an MIS of the network shown in Fig. 5a. Note that D is also a DS of G and an element in D is called a *dominator*. Clearly, every dominator is out of the communication range of any other dominators. Let G' be a graph on D in which two nodes in D linked by an edge if and only if these two nodes have a common neighbor in G . Obviously, sink s_0 is in G' and also s_0 is defined as the center of G' . Suppose that the radius of G' with respect to s_0 is L' and the union of dominators at level l ($0 \leq l \leq L'$) is denoted as set D_l . Note that $D_0 = \{s_0\}$. Second, choose nodes, also called *connectors*, to connect all the nodes in D to form a CDS. Let S_l ($0 \leq l \leq L'$) be the set of the nodes adjacent to at least one node in D_l and at least one node in D_{l+1} and compute a minimal cover $C_l \subseteq S_l$ for D_{l+1} . Let $C = \cup_0^{L'-1} C_l$ and therefore $D \cup C$ is a CDS of G . As shown in Fig. 5c, the blue nodes are connectors chosen to connect the dominators in D . Meanwhile, the union of the dominators and connectors in Fig. 5c forms a CDS of the network shown in Fig. 5a. Finally, for any other node u , also called a *dominatee*, not belonging to $D \cup C$, choose the nearest dominator as u 's parent node. Thus, the routing tree T is obtained. For instance, the tree shown in Fig. 5d is the obtained routing tree for the network shown in Fig. 5a.

For each link in T , assign it a direction from the child node to the parent node along the data transmission flow to the sink as shown in Fig. 5d. Furthermore,

Fig. 6 (a) A link set and (b) its corresponding conflicting graph



the receiving node, i.e., the parent node, of a link is called a *head*. Similarly, the transmitting node, i.e., the child node, of a link is called a *tail*. Suppose that A is a set of links of T . The corresponding *conflicting graph* of A is denoted by $\mathfrak{R}(A) = (V_A, E_A)$, where each link in A is abstracted to a node in V_A and two nodes in V_A form an edge in E_A if the corresponding two links of these two nodes are interfering links. Taking the four links shown in Fig. 6a as an example, suppose $\iota_1 = (s_1, s_2)$, $\iota_2 = (s_3, s_4)$, $\iota_3 = (s_5, s_6)$, and $\iota_4 = (s_7, s_8)$. Let $\mathbb{I}(\iota_i)$ ($1 \leq i \leq 4$) denote the set of interfering links with ι_i and assume

- $\mathbb{I}(\iota_1) = \{\iota_2, \iota_3\}$;
- $\mathbb{I}(\iota_2) = \{\iota_1, \iota_3, \iota_4\}$;
- $\mathbb{I}(\iota_3) = \{\iota_1, \iota_2, \iota_4\}$;
- $\mathbb{I}(\iota_4) = \{\iota_2, \iota_3\}$.

Then, the corresponding conflicting graph of $\{\iota_1, \iota_2, \iota_3, \iota_4\}$ is shown in Fig. 6b, where ι_i is abstracted to v_i .

Lemma 1 in [57] can be used to derive some useful results of the routing tree T .

Lemma 1 ([57]) Suppose that O (respectively, O') is a disk (respectively, halfdisk) with radius r and U is a set of points with mutual distances of at least one. Then the number of the points α_r in a disk and the number of the points β_r in a half-disk are

$$\alpha_r = |U \cap O| \leq \frac{2\pi}{\sqrt{3}}r^2 + \pi r + 1 \quad (4)$$

and

$$\beta_r = |U \cap O'| \leq \frac{\pi}{\sqrt{3}}r^2 + \left(\frac{\pi}{2} + 1\right)r + 1. \quad (5)$$

From Lemma 1, Wan et al. in [57] derived the following properties of the routing tree T :

- For each $0 \leq l \leq L' - 1$, each connector in C_l is adjacent to at most four dominators in D_{l+1} .
- For each $1 \leq l \leq L' - 1$, each dominator in D_l is adjacent to at most 11 connectors in C_l .
- $|C_0| \leq 12$.

4.4 Vertex Coloring Problem

For a graph $G = (V, E)$, the *maximum degree* (respectively, *minimum degree*) of G is denoted by $\Delta(G)$ (respectively, $\delta(G)$). The *minimum degree* of G is denoted by $\delta(G)$. A subgraph of G on $U \subseteq V$ is denoted by $G(U)$. The *inductivity* of G is defined as $\delta^*(G) = \max_{U \subseteq V} \delta(G(U))$. A *vertex coloring* of G is a scheme of coloring all the vertices in G such that no two adjacent vertices share the same color. The *chromatic number* $\chi(G)$ of G is the least number of colors used to color G . Deciding the lower bound of $\chi(G)$ is a well-known NP-complete problem. However, the upper bound of $\chi(G)$ has been derived in *graph theory* [50, 57]. The following lemma was proven in [50, 57].

Lemma 2 $\chi(G) \leq 1 + \delta^*(G)$ and a vertex coloring scheme, called first-fit coloring, for G using at most $1 + \delta^*(G)$ colors can be found in polynomial time.

Given a link set A of T , the channel assignment problem for A can be abstracted to the vertex coloring problem for its corresponding conflicting graph $\mathfrak{R}(A)$. If the tail (respectively, head) of every link in A is a dominator, then Lemma 3 in [57] gives the upper bound of $\delta^*(A)$.

Lemma 3 [57] $\delta^*(A) \leq \beta_{\rho+1} - 1$.

Lemma 3 implies that in the worst case, at most $\beta_{\rho+1}$ channels may be assigned to all the links in A without channel interference by a first-fit coloring method.

5 Capacity of Snapshot Data Collection

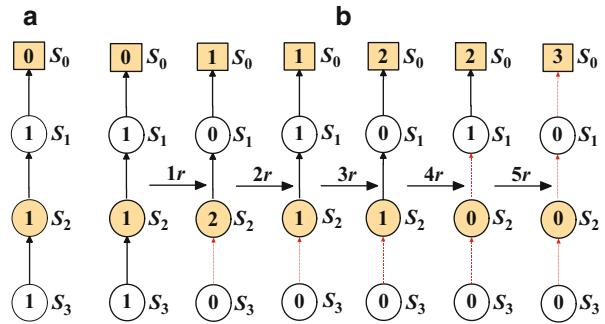
In this section, the traditional SDC problem is investigated first, a scheduling algorithm for this problem is proposed in single-radio multichannel WSNs, and the achievable capacity of the proposed algorithm is analyzed. Subsequently, the proposed algorithm is discussed further, and the reason that the proposed algorithm and most existing works cannot improve the CDC capacity of a network in order by the pipeline technology is also pointed out.

Since at any time slot, the sink can receive data from at most one neighboring sensor; therefore, the upper bound of the SDC capacity is W [12, 31]. Aiming at this upper bound, a scheduling algorithm for SDC which is order-optimal and has a tighter lower bound than the previously best result [12] is designed.

5.1 Scheduling Algorithm for Snapshot Data Collection

The idea of *single-path scheduling* has been used in [12, 66] to collect data for a WSN. However, their methods have a looser bound of the SDC capacity. In this

Fig. 7 (a) A single path and (b) its scheduling ($r = \text{round}$)



subsection, a new *multipath scheduling* algorithm based on the routing tree T built in Sect. 4 is designed, which is proven to have a better performance. Firstly, how to schedule a single path is studied. Subsequently, how to extend the single-path scheduling scheme to the scheduling of multipath in the routing tree T is investigated.

For simplicity of discussion, the concept of a *round* is defined. A *round* is a period of time which consists of multiple continuous time slots (the size of a round will be studied in Lemma 5 of the next subsection). Take the path, shown in Fig. 7a as an example to explain the idea of the single-path scheduling scheme. In Fig. 7a, the path, denoted by P consists of one sink s_0 and three sensors s_1, s_2 , and s_3 , where s_0 and s_2 are dominators, s_1 is a connector, and s_3 is a dominatee. The value marked in each node is the number of the data packets at this node to be transmitted during a time slot. At the initial time, every sensor on P has one data packet and there is no data packet at s_0 . P_o denotes the set of links on P whose heads are dominators and whose tails have at least one data packet to be transmitted. P_e denotes the set of links on P whose tails are dominators and whose tails have at least one data packet to be transmitted. For the path shown in Fig. 7a, $P_o = \{(s_3, s_2), (s_1, s_0)\}$ and $P_e = \{(s_2, s_1)\}$. P can be scheduled according to the following two steps and repeat them until all the data packets have been collected by s_0 .

Step 1: In an odd round, schedule every link in P_o once, i.e., assign a dedicated channel and one dedicated time slot to each link in P_o .

Step 2: In an even round, schedule every link in P_e once, i.e., assign a dedicated channel and one dedicated time slot to each link in P_e .

The detailed scheduling in Step 1 can be done in the following way: first, for any link $\ell_i \in P_o$, let $\mathbb{I}_{P_o}(\ell_i) = \{\ell_j | \ell_j \in P_o, \ell_i \text{ and } \ell_j \text{ are interfering links}\}$; second, sort the links in P_o according to $|\mathbb{I}_{P_o}(\ell_i)|$ ($1 \leq i \leq |P_o|$) in a nondecreasing order, where $|\cdot|$ denotes the cardinality of a set, and denote the resulting link sequence as $\{\ell'_1, \ell'_2, \dots, \ell'_{|P_o|}\}$; and finally, during the i -th ($1 \leq i \leq \lceil \frac{|P_o|}{H} \rceil$) time slot of a round, let the j -th ($(i-1)H < j \leq iH$) link in P_o work on channel $\lambda_{j \% H + 1}$. Here, sort the links in P_o first and subsequently assign channels based on the *first-fit coloring* scheme in Lemma 2. Furthermore, according to Lemmas 2 and 3,

the channel assignment plan for the links in P_o is interference/collision-free. The detailed scheduling in Step 2 is similar to that of Step 1.

The scheduling process of P in Fig. 7a is shown in Fig. 7b. During the first (odd) round, links (s_3, s_2) and (s_1, s_0) are scheduled and the data packets at s_3 and s_1 are transmitted to their parent nodes. After the first round, s_3 has no data packet to transmit. During the second (even) schedule, link (s_2, s_1) is scheduled and s_2 transmits one data packet to its parent node. This schedule process continues until all the data packets on path P has been transmitted to s_0 .

Now, consider the scheduling of the routing tree T built in Sect. 4. Suppose that there are m leaf nodes in T denoted by $s_1^l, s_2^l, \dots, s_m^l$, respectively. The path from leaf node s_i^l ($1 \leq i \leq m$) to the sink s_0 is denoted by P_i . Two paths P_i and P_j are said *intersecting* if they have at least one common node besides the sink node. Assume path P_i and path P_j are intersecting, and the lowest common ancestor of s_i^l (P_i) and s_j^l (P_j), i.e., the common node of P_i and P_j having the largest number of hops from the sink, is called an *intersecting point* of P_i and P_j . If path P_i intersects with other paths, the route from s_i^l to the nearest intersecting point of P_i is called a *sub-path*, denoted by F_i . Otherwise, F_i is actually P_i .

Taking the routing tree \widehat{T} shown in Fig. 8a as an example, \widehat{T} consists of one sink denoted by s_0 and ten sensor nodes denoted by s_i ($1 \leq i \leq 10$). \widehat{T} has three leaf nodes s_1, s_2 , and s_3 , which correspond to paths P_1, P_2 , and P_3 , respectively. In \widehat{T} , P_1 and P_2 are intersecting and their intersecting point is s_5 . Nevertheless, P_1 and P_3 , as well as P_2 and P_3 , are not intersecting since they have no common node beside the sink s_0 . For P_1 , the route from s_1 to s_5 is the sub-path of P_1 , denoted by F_1 . For P_3 , since it is not intersecting with any path, F_3 is P_3 itself.

To schedule multiple paths on the routing tree T , a *multipath scheduling* algorithm as shown in Algorithm 1 is proposed. In Algorithm 1, \mathcal{P} is the set of paths that has been scheduled simultaneously in a round, \mathcal{S} is the set of links from multiple paths that can be scheduled in a round, rd_i/rd_j indicates the number of available rounds that has been assigned to path P_i/P_j , and P_o^i/P_o^j (respectively, P_e^i/P_e^j) is the set of links on P_i/P_j whose heads (respectively, tails) are dominators and whose tails have at least one data packet to be transmitted. From Algorithm 1, one can see that lines 2–10 are used to schedule path P_i according to the single-path scheduling algorithm. Lines 11–20 are used to find other paths that can be scheduled simultaneously with P_i according to the single-path scheduling algorithm at the same round.

The proposed multipath scheduling algorithm is further explained by the routing tree \widehat{T} shown in Fig. 8a. Assume the interference radius $\rho = 1$, i.e., the interference radius is equal to the transmission radius, which implies that each *round* consists of two time slots. Furthermore, $\mathcal{I}(s_i)$ ($1 \leq i \leq n$) is used to denote the set of sensor nodes that cannot be transmitted data simultaneously with s_i . For the nodes in \widehat{T} , assume

- $\mathcal{I}(s_1) = \{s_4, s_5, s_6, s_7, s_8\}$;
- $\mathcal{I}(s_2) = \{s_6, s_7\}$;
- $\mathcal{I}(s_3) = \{s_9, s_{10}\}$;

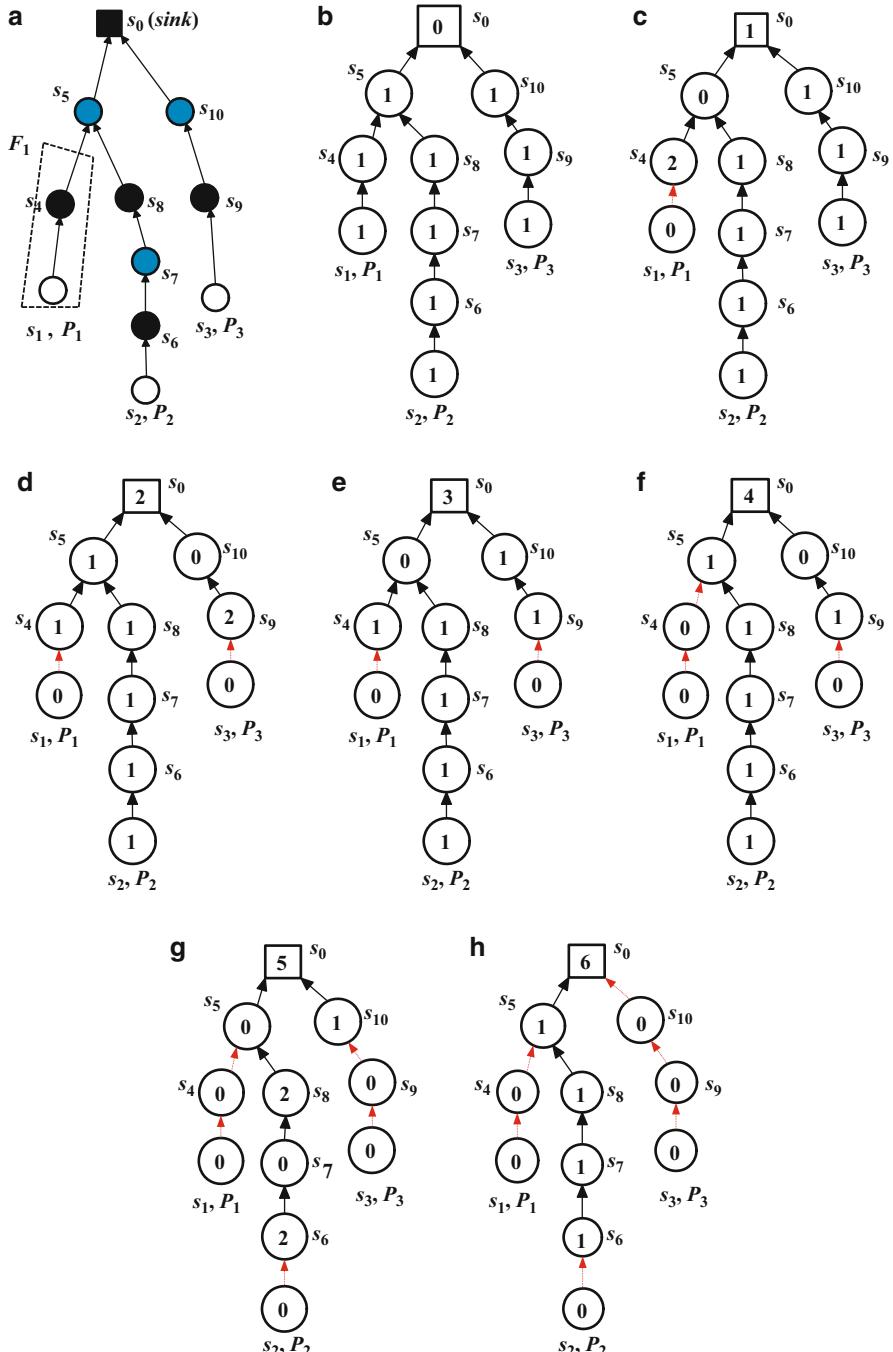


Fig. 8 A routing tree and its scheduling. In (a), the *black nodes* are dominators, the *blue nodes* are connectors, and the other nodes are dominatees. In (b)–(h), the number inside each node is the number of data packets at this node. The *red links* are the links removed from the data collection tree according to [Algorithm 1](#)

Algorithm 1 Multipath scheduling algorithm

```

input : a routing tree  $T$  with  $m$  leaf nodes
output: a schedule plan for the routing tree  $T$ 

1 for  $i = 1; i \leq m; i++$  do
2   while there is some data for transmission on  $F_i$  do
3      $\mathcal{P} \leftarrow \{P_i\}$ ;
4      $\mathcal{S} \leftarrow \emptyset$ ;
5     if  $rd_i \% 2 == 1$  then
6        $\mathcal{S} \leftarrow P_o^i$ ;
7        $rd_i++$ ;
8     else if  $rd_i \% 2 == 0$  then
9        $\mathcal{S} \leftarrow P_e^i$ ;
10       $rd_i++$ ;
11    for  $j = i + 1; j \leq m; j++$  do
12      if  $P_j$  is not intersecting with any path in  $\mathcal{P}$  && there is some data for
transmission on  $F_j$  then
13        if  $rd_j \% 2 == 1$  && all the transmissions in  $P_o^j$  and all the
transmissions in  $\mathcal{S}$  are interference/collision-free then
14           $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_j\}$ ;
15           $\mathcal{S} \leftarrow \mathcal{S} \cup P_o^j$ ;
16           $rd_j++$ ;
17        if  $rd_j \% 2 == 0$  && all the transmissions in  $P_e^j$  and all the
transmissions in  $\mathcal{S}$  are interference/collision-free then
18           $\mathcal{P} \leftarrow \mathcal{P} \cup \{P_j\}$ ;
19           $\mathcal{S} \leftarrow \mathcal{S} \cup P_e^j$ ;
20           $rd_j++$ ;
21    schedule the links in  $\mathcal{S}$  in a round as in the single-path scheduling algorithm;
22    if there is no data for transmission on  $F_i$  then
23      remove all the links on  $F_i$  from  $T$ .

```

- $\mathcal{I}(s_4) = \{s_1, s_5, s_7, s_8\};$
- $\mathcal{I}(s_5) = \{s_1, s_4, s_7, s_8, s_{10}\};$
- $\mathcal{I}(s_6) = \{s_1, s_2, s_7, s_8\};$
- $\mathcal{I}(s_7) = \{s_1, s_2, s_4, s_5, s_6, s_8\};$
- $\mathcal{I}(s_8) = \{s_1, s_4, s_5, s_6, s_7\};$
- $\mathcal{I}(s_9) = \{s_3, s_{10}\};$
- $\mathcal{I}(s_{10}) = \{s_3, s_5, s_9\}.$

Additionally, for path P_1 , P_2 , and P_3 ,

- $P_o^1 = \{(s_1, s_4), (s_5, s_0)\};$
- $P_e^1 = \{(s_4, s_5)\};$
- $P_o^2 = \{(s_2, s_6), (s_7, s_8), (s_5, s_0)\};$
- $P_e^2 = \{(s_6, s_7), (s_8, s_5)\};$
- $P_o^3 = \{(s_3, s_9), (s_{10}, s_0)\};$
- $P_e^3 = \{(s_9, s_{10})\}.$

At the beginning of [Algorithm 1](#), the network is shown in [Fig. 8b](#) with the number inside each node denoting the number of data packets at this node. According to the algorithm, during the first round, P_o^1 is scheduled, and path P_2 will not be scheduled since it is intersecting with P_1 . P_3 also will not be scheduled since the link (s_{10}, s_0) in P_o^3 and the link (s_5, s_0) in P_o^1 are not interference/confliction-free. Thus, after the first round, the network situation is shown in [Fig. 8c](#). During the second round, P_e^1 will be scheduled. Now, all the links in P_e^1 and all the links in P_o^3 are interference/confliction-free.⁴ Hence, P_1 and P_3 can be scheduled simultaneously at the second round. After the second round, the network situation is shown in [Fig. 8d](#). Similarly, according to [Algorithm 1](#), the network after the third, the fourth, the fifth, and the sixth round is shown in [Fig. 8e–h](#), respectively. Finally, for \widehat{T} shown in [Fig. 8a](#), it will take 13 rounds to collect all the data packets to the sink by the multipath scheduling algorithm. By contrast, it will take 18 rounds to collect all the data packets to the sink by the single-path scheduling algorithm.

5.2 Capacity Analysis

In this subsection, the achievable network capacity of the proposed multi-path scheduling algorithm is analyzed. The upper bound of the SDC capacity is W which has been explained. Consequently, the lower bound of the SDC capacity is focused on. In the worst case, all the paths in the routing tree T are intersecting, i.e., they have a common intersecting point, which means only one path can be scheduled at any time. In order to derive the lower bound of the capacity of the multipath scheduling algorithm, the number of the rounds needed to finish the scheduling of one single path is first investigated and then the number of the time slots in each round is studied. [Lemma 4](#) gives the maximum number of the rounds used for the scheduling of one single path.

Lemma 4 *For a single-path P of length L in T , it takes at most $2L - 1$ rounds to collect all the data packets on P at the sink node.*

Proof Suppose that the node sequence on P is $s_1, s_2, \dots, s_L, s_0$, where s_1 is the leaf node (dominatee) and s_0 is the sink node. Considering the building process of T , each link in P has either a dominator head or a dominator tail. According to the scheduling scheme of a single path, during the first (odd) round, the links in P_o are scheduled, which implies that each nondominator with at least one data packet transmits this data packet to its parent node. After the first round, the sink receives one data packet and all the other dominators of the links in P_o have two data packets to be transmitted. During the second (even) round, the links in P_e are scheduled, which implies that every dominator in P_e transmits one data packet to its

⁴Here, P_o^3 is considered instead of P_e^3 because for path P_3 , the current round is the first available round.

parent node. As a result, the sensor s_i ($2 \leq i \leq L$) has exactly one data packet to be transmitted and a new odd-even scheduling round begins. In summary, after every two rounds, the sink receives one data packet and the length of the data collection path decreases by 1. Since the length of P is L and s_0 is the destination of all the data packets which does not have to transmit any data, it takes at most $2L - 1$ rounds to collect all the data packets on P . \square

From [Lemma 4](#), it is straightforward to obtain the number of the rounds used to collect all the data on the sub-path F of P as shown in [Corollary 1](#).

Corollary 1 *For the sub-path F of length L_s in P , it takes at most $2L_s$ rounds to collect all the data packets on F .*

Proof The proof of [Corollary 1](#) is similar to that of [Lemma 4](#). Note that the intersecting point is not a sink node in this case and thus it needs one round to transmit its data packet. \square

By [Lemma 4](#) and [Corollary 1](#), the number of the rounds used to collect the data packets on a path is obtained. The maximum number of the time slots in a round is as follows.

Lemma 5 *In the single-path scheduling algorithm, a round has at most $\left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$ time slots, where $\beta_{\rho+1}$ is the number of the dominators in a halfdisk with radius $\rho + 1$ and H is the number of available orthogonal channels.*

Proof During every odd (respectively, even) round, the scheduled links are links in P_o (respectively, P_e). Since the heads (respectively, tails) of links in P_o (respectively, P_e) are dominators, all the links in P_o (respectively, P_e) can be scheduled in one time slot with at most $\beta_{\rho+1}$ channels in polynomial time by [Lemmas 3](#) and [2](#). Now, the considering WSN has H available channels, which means the scheduling can be finished within $\left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$ time slots. Therefore, the lemma holds. \square

Now, the lower bound of the achievable capacity of the multipath scheduling algorithm can be obtained as shown in [Theorem 1](#).

Theorem 1 *The capacity Υ at the sink of T of the multipath scheduling algorithm is at least $\frac{W}{2\lceil(1.81\rho^2+c_1\rho+c_2)/H\rceil}$, where $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$ and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$, which is orderoptimal.*

Proof Suppose that T has m paths and the length of each path is L_i ($1 \leq i \leq m$). In the worst case, all the m paths cannot be scheduled concurrently. Then by [Lemma 4](#), [Corollary 1](#), and [Lemma 5](#), the total time τ used to collect all the data packets of T at the sink is at most $t \cdot \sum_{i=1}^m 2L_i \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$. According to the multipath scheduling

algorithm, for any path P_i , the time used to collect data packets on P_i is equal to the time used to collect data packets on the corresponding sub-path F_i of P_i .⁵ Therefore,

$$\tau \leq t \cdot \sum_{i=1}^m 2L_i \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil \quad (6)$$

$$= t \cdot \sum_{i=1}^m 2|F_i| \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil \quad (7)$$

$$= 2t \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil \sum_{i=1}^m |F_i|. \quad (8)$$

Since the number of the links in T is equal to the number of the sensors in T , $\sum_{i=1}^m |F_i| = n$. Then, $\tau \leq 2nt \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil$. Therefore, the capacity

$$\Upsilon = \frac{nb}{\tau} \geq \frac{nb}{2nt \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil} \quad (9)$$

$$= \frac{b}{2t \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil} = \frac{W}{2 \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil}. \quad (10)$$

From Lemma 1,

$$\beta_{\rho+1} \leq \frac{\pi}{\sqrt{3}}(\rho+1)^2 + \left(\frac{\pi}{2} + 1\right)(\rho+1) + 1 \quad (11)$$

$$= \frac{\pi}{\sqrt{3}}\rho^2 + \left(\frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1\right)\rho + \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2 \quad (12)$$

$$\approx 1.81\rho^2 + c_1\rho + c_2, \quad (13)$$

where $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$ and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$. This implies

$$\Upsilon \geq \frac{W}{2 \left\lceil \frac{\beta_{\rho+1}}{H} \right\rceil} \geq \frac{W}{2 \left\lceil \frac{1.81\rho^2 + c_1\rho + c_2}{H} \right\rceil}. \quad (14)$$

⁵From lines 2–10 in Algorithm 1, the scheduling of path P_i is stopped when all the data packets on the sub-path F_i have been collected by the sink. As shown in Fig. 8f, g, after all the data packets on F_1 (the sub-path of P_1) have been collected by the sink, the scheduling of data transmission on P_2 is beginning. Additionally, based on the definition of a sub-path, F_3 in Fig. 8 is P_3 itself since P_3 does not intersect with any path. Therefore, the time used to collect data packets on P_i is equal to the time used to collect data packets on the corresponding sub-path F_i of P_i .

Since H is a constant and the upper bound of Υ is W [12, 31], Υ is orderoptimal. \square

From [Theorem 1](#), it is known that the achievable capacity of the multi-path scheduling algorithm is orderoptimal, and it also has a tighter lower bound compared with the previously best result in [12], which has a lower bound of $\frac{W}{8\rho^2}$.

5.3 Discussion

When address the CDC problem, an intuitive idea is to pipeline the existing SDC operations [12]. Nevertheless, such an idea cannot achieve a better performance. This is because the sink can receive at most one data packet at a time slot. By pipeline, data transmissions at the nodes far from the sink are really accelerated. However, the fact that a sink can receive at most one data packet at each time slot makes the data accumulated at the nodes near the sink. Finally, the network capacity cannot be improved even with pipeline. This motivates people to investigate new methods for CDC.

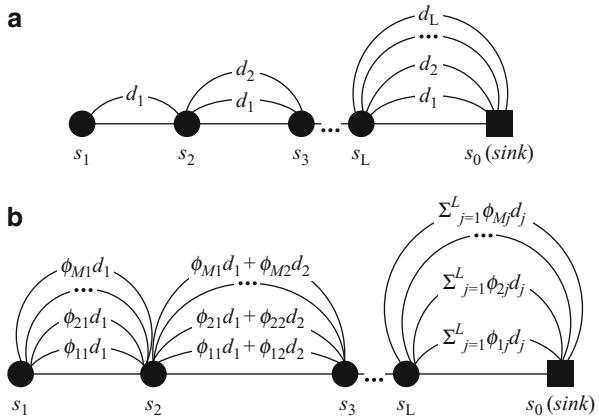
6 Capacity of Continuous Data Collection

Since multipath scheduling algorithm and existing works with pipeline cannot improve the capacity of CDC, in this section, a novel *pipeline scheduling* algorithm based on *compressive data gathering* (CDG) [46] is proposed in dual-radio multichannel WSNs, which augments the CDC capacity significantly. Here, dual-radio multichannel WSNs are considered because dual radios can make a *half-duplex* single-radio node work in a *full-duplex* mode, i.e., a dual-radio node can receive and transmit data simultaneously with the two radios over different channels. Furthermore, the full-duplex working mode is in favor of pipeline. For completeness, the achievable network capacity of the pipeline scheduling algorithm (a little modification is needed) in single-radio multichannel WSNs is also analyzed.

6.1 Compressive Data Gathering (CDG)

CDG is first proposed in [46] for SDC in single-radio single-channel WSNs. The basic idea of CDG is to distribute the data collection load uniformly to all the nodes in the entire network. Take the data collection on a path consisting of L sensors s_1, s_2, \dots, s_L and one sink s_0 as shown in [Fig. 9](#) [46] as an example to explain CDG. In [Fig. 9](#), the data packet produced at sensor s_j ($1 \leq j \leq L$) is d_j . In the basic data collection shown in [Fig. 9a](#), s_1 transmits one data packet d_1 to s_2 , s_2 transmits two data packets d_1 and d_2 to s_3 , and finally all the data packets on the path are transmitted to s_0 by s_L . Obviously, nodes near the sink have more transmission load compared with nodes far from the sink in the basic data collection. To balance the transmission load, the authors in [46] proposed the CDG method as shown in [Fig. 9b](#).

Fig. 9 Comparing of (a) basic data collection and (b) CDG [46]



Instead of transmitting the original data directly, \$s_1\$ multiplies its data with a random coefficient \$\phi_{i1}\$ (\$1 \leq i \leq M\$) and sends the \$M\$ results \$\phi_{i1}d_1\$ to \$s_2\$. Upon receiving \$\phi_{i1}d_1\$ (\$1 \leq i \leq M\$) from \$s_1\$, \$s_2\$ multiplies its data \$d_2\$ with a random coefficient \$\phi_{i2}\$ (\$1 \leq i \leq M\$), adds it to \$\phi_{i1}d_1\$, and then sends \$\phi_{i1}d_1 + \phi_{i2}d_2\$ as one data packet to \$s_3\$. Finally, \$s_L\$ does the similar multiplication and addition and sends the result \$\sum_{j=1}^L \phi_{ij}d_j\$ (\$1 \leq i \leq M\$) to \$s_0\$. After \$s_0\$ receives all the \$M\$ data packets, \$s_0\$ can restore the original data packets based on the compressive sampling theory [46]. By CDG, all the sensors send \$M\$ data packets to their parent nodes, which achieves the goal to uniformly distribute the data collection task to the entire network. The number of the transmitted data packets is \$O(n^2)\$ in Fig. 9a and is \$O(NM)\$ in Fig. 9b, and usually \$M \ll n\$ for large-scale WSNs. Therefore, CDG reduces the number of the transmitted data packets. Considering WSNs are usually large-scale networks, then \$M \ll n\$, the number of sensors in a WSN [46].

6.2 Pipeline Scheduling

Thanks to the benefit brought by CDG, the CDC problem can be addressed with the pipeline technique. From the building process of the routing tree \$T\$, it is known that the nodes in \$T\$ can be divided into sets by levels \$D_e, D_{L'}, C_{L'-1}, D_{L'-1}, C_{L'-2}, \dots, D_1, C_0, D_0 = \{s_0 | s_0 \text{ is the sink}\}\$ in a bottom-up way, where \$D_e\$ is the set of all the dominatees, i.e., leaf nodes, \$D_i\$ (\$0 \leq i \leq L'\$) is the set of the dominators at the \$i\$-th level, and \$C_i\$ (\$0 \leq i \leq L' - 1\$) is the set of the connectors at the \$i\$-th level. Since every node has two radios, one radio can be dedicated to receive data and the other dedicated to transmit data. Therefore, the nodes at every level can receive and transmit data simultaneously over different channels. Consequently, for a CDC task consisting of \$N\$ snapshots, a *pipeline scheduling* algorithm is proposed as follows:

Step 1: All the nodes in \$D_e\$ transmit the data packets of the \$j\$-th (\$1 \leq j \leq N - 1\$) snapshot to their parent nodes in the CDG way, i.e., for every node \$s \in D_e\$, \$s\$ multiplies its data with \$M\$ random coefficients, respectively, and sends the \$M\$ products to its parent node. After all the data packets of the \$j\$-th

snapshot have been transmitted successfully, the nodes in D_e immediately transmit the data packets of the $(j + 1)$ -th snapshot in the CDG way.

- Step 2: After all the nodes in D_l ($1 \leq l \leq L'$) receive all the data packets of the j -th snapshot from their childlevel, they send the data packets of the j -th snapshot to their parent nodes in the CDG way, i.e., every node $s \in D_l$ combines its data packet of the j -th snapshot with the received data packets of the j -th snapshot and sends the M new data packets to its parent node. After all the data packets of the j -th snapshot have been transmitted successfully, the nodes in D_l immediately transmit the data packets of the $(j + 1)$ -th snapshot to their parent nodes in the CDG way, if they have received all the data packets of the $(j + 1)$ -th snapshot from their childlevel.*
- Step 3: After all the nodes in C_l ($0 \leq l \leq L' - 1$) receive all the data packets of the j -th snapshot from their childlevel, they send the data packets of the j -th snapshot to their parent nodes in the CDG way, i.e., every node $s \in C_l$ combines its data packet of the j -th snapshot with the received data packets of the j -th snapshot and sends the M new data packets to its parent node. After all the data packets of the j -th snapshot have been transmitted successfully, the nodes in C_l immediately transmit the data packets of the $(j + 1)$ -th snapshot in the CDG way if they have received all the data packets of the $(j + 1)$ -th snapshot from their child-level.*

- Step 4: The sink restores the data of a snapshot in the CDG way after it receives all the data packets of this snapshot.*

Steps 1–4 provide the general frame of the proposed pipeline scheduling scheme. Now, how to prevent radio confliction and channel interference in Steps 1–3 is explained as follows. If two or more nodes have the same parent node, they are called *sibling* nodes. In Steps 1–3, radio confliction may arise if two or more sibling nodes send data to their parent node simultaneously even over different orthogonal channels. This is because every sensor only has one radio dedicated to receiving data. Suppose that there are at most Δ_e (respectively, Δ_d and Δ_c) nodes in D_e (respectively, D_l ($1 \leq l \leq L'$) and C_l ($1 \leq l \leq L' - 1$)) which have the same parent node. Usually, $\Delta_e < \Delta(T)$ except in one-hop WSNs, where any sensor is just one hop away from the sink, $\Delta_e = \Delta(T)$. Then, $\Delta_d \leq 4$ and $\Delta_c \leq 11$. (Note that $|C_0| \leq 12$.) (See Sect. 4.3.) To avoid confliction, the nodes in D_e (respectively, D_l ($1 \leq l \leq L'$) and C_l ($1 \leq l \leq L' - 1$)) can be partitioned into Δ_e (respectively, Δ_d and Δ_c) subsets to guarantee that each node belongs to one subset and no sibling nodes belong to the same subset. Then, when scheduling the nodes of each level, these subsets can be scheduled in a certain order. For the nodes in C_0 , they can be scheduled in a certain order, e.g., the nodes with small IDs are scheduled with high priority.

Different from the multipath scheduling algorithm, in which a sensor sends one data packet over a link in one time slot, the CDG way is employed in the pipeline scheduling algorithm, where a sensor sends M data packets for a snapshot. Thus, the concept of a *super time slot* (STS) which consists of M time slots is introduced. In an STS, a sensor can send M data packets over a channel for a snapshot. For the links working simultaneously, the channels and STSs can be assigned in the similar manner of the multipath scheduling algorithm.

6.3 Capacity Analysis

In this subsection, the achievable network capacity of the proposed *pipeline scheduling* algorithm is analyzed.

Lemma 6 indicates the inductivity (defined in Sect. 4.4) of the corresponding conflicting graph of the links scheduled simultaneously in the pipeline scheduling algorithm, which is used to obtain the upper bound of the number of the necessary channels to schedule these links.

Lemma 6 Suppose that A is the set of the links in T scheduled simultaneously in the pipeline scheduling algorithm and $\mathfrak{R}(A)$ is the corresponding conflicting graph of A , then, $\delta^*(\mathfrak{R}(A)) \leq 2\beta_{\rho+2} - 1$, where $\delta^*(\mathfrak{R}(A))$ is the inductivity of $\mathfrak{R}(A)$ and $\beta_{\rho+2}$ is the number of the dominators within a halfdisk of radius $\rho + 2$.

Proof Since the sibling nodes at every level have been divided into different subsets and different subsets are scheduled in a certain order, there is no radio confliction among the links in A . Furthermore, for any link in A , either the tail or the head of this link is a dominator according to the building process of the routing tree T . Suppose that A' is a subset of A and e is the link in A' whose tail, denoted by $t(e)$, or head, denoted by $h(e)$, is the bottommost dominator among all the dominators in A' . Then, it can be proven that the number of the links interfered with e , i.e., $\delta(\mathfrak{R}(A'))$, is at most $2\beta_{\rho+2} - 1$ case by case as follows:

Case 1: $t(e)$ is a dominator. In this case, assume that e' is another link in A' interfered with e and $t(e')$ is a dominator. Since $t(e)$ is the bottommost dominator, the necessary condition for e and e' to be interfering links is that $t(e')$ locates at the upper halfdisk centered at $t(e)$ with radius $\rho + 1$. On the other hand, if $h(e')$ is a dominator, then the necessary condition for e and e' to be interfering links is that $h(e')$ locates at the upper halfdisk centered at $t(e)$ with radius $\rho + 2$. By Lemma 1, the number of the dominators within a halfdisk of radius $\rho + 2$ is at most $\beta_{\rho+2}$. Since every dominator in A' is associated with at most two links, there are at most $2\beta_{\rho+2}$ links within the halfdisk of radius $\rho + 2$ centered at $t(e)$. Therefore, $\delta(\mathfrak{R}(A')) \leq 2\beta_{\rho+2} - 1$, where minus 1 means e is also in the halfdisk. As a result, $\delta^*(\mathfrak{R}(A)) = \max_{A' \subseteq A} \delta(\mathfrak{R}(A')) \leq 2\beta_{\rho+2} - 1$.

Case 2: $h(e)$ is a dominator. By the similar method as in case 1, it can be proven that the conclusion also holds in this case. \square

Based on the result of Lemma 6, the number of the STSSs used to schedule all the links in A can be determined as follows.

Lemma 7 For the links of set A in Lemma 6, one can use $\left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSSs to schedule them without channel interference.

Proof By Lemma 6, $\delta^*(\mathfrak{N}(A)) \leq 2\beta_{\rho+2} - 1$. By Lemma 2, one can use $1 + \delta^*(\mathfrak{N}(A)) \leq 2\beta_{\rho+2}$ channels to schedule all the links in A in one STS simultaneously. Now, a WSN has H channels, which implies all the links in A can be scheduled in $\left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs of H links in each STS. \square

From the pipeline scheduling algorithm, it is known that the transport of subsequent snapshots has sometime overlap with the transport of preceding snapshots. Therefore, in the following, the time used to collect the data packets of the first snapshot is firstly analyzed since it is the base of the pipeline, and then the achievable capacity of the entire pipeline is analyzed.

Theorem 2 *The number of the time slots used to collect the data packets of the first snapshot by the pipeline scheduling algorithm is at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1)$.*

Proof In Step 1 of the pipeline scheduling algorithm, the nodes in D_e are partitioned into Δ_e subsets and scheduled in a certain order. By Lemma 7, each scheduling uses at most $\left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs. Consequently, Step 1 needs at most $\Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs to finish the scheduling for the first snapshot. In Step 2 (respectively, Step 3), the nodes in D_l ($1 \leq l \leq L'$) (respectively, C_l ($1 \leq l \leq L' - 1$)) are partitioned into Δ_d (respectively, Δ_c) subsets and scheduled in a certain order. Since, $\Delta_d \leq 4$ (respectively, $\Delta_c \leq 11$), Step 2 (respectively, Step 3) needs at most $4L' \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ (respectively, $11(L' - 1) \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$) STSs to finish the scheduling for the first snapshot. Furthermore, it needs at most $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs for C_0 to transmit the data packets for the first snapshot to the sink. In summary, the total number of the STSs used for the first snapshot is at most

$$\Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil + 4L' \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil + 11(L' - 1) \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil + 12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil \quad (15)$$

$$= \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 4L' + 11L' - 11 + 12) \quad (16)$$

$$= \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1). \quad (17)$$

Since every STS has M time slots, then the number of the time slots used for the first snapshot is at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1)$. \square

On the basis of the result in [Theorem 2](#), one can obtain the time slots used to collect all the data packets of N continuous snapshots for the pipeline scheduling algorithm as shown in [Theorem 3](#).

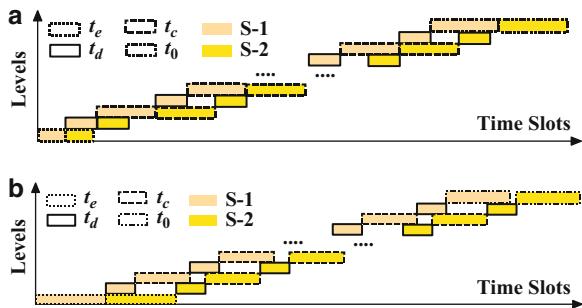
Theorem 3 *The time slots used for the pipeline scheduling algorithm to collect N continuous snapshots are at most*

$$\begin{cases} M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 12N - 11), & \text{if } \Delta_e \leq 12 \\ M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (N\Delta_e + 15L' + 1), & \text{if } \Delta_e > 12 \end{cases}.$$

Proof From the proof of [Theorem 2](#), it takes the nodes in D_e (respectively, D_l ($1 \leq l \leq L'$), C_l ($1 \leq l \leq L' - 1$), and C_0) at most $\Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ (respectively, $4 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, $11 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, and $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$) STSs to transmit data packets for a snapshot. In order to obtain the upper bound of the number of the time slots used, assume the STSs used by nodes in D_e (respectively, D_l ($1 \leq l \leq L'$), C_l ($1 \leq l \leq L' - 1$), and C_0) are $\Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ (respectively, $4 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, $11 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, and $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$) in the following proof. Then, [Theorem 3](#) can be proven by cases.

Case 1: $\Delta_e \leq 4$. For clearness, the transmission of two snapshots S-1 and S-2 shown in [Fig. 10a](#) is used as an example for explanation. In [Fig. 10a](#), the vertical axis denotes the levels in the routing tree T and the horizontal axis denotes time slots. $t_e = \Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, $t_d = 4 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, $t_c = 11 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, and $t_0 = 12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$, respectively. From [Fig. 10a](#), it is known that the nodes at the D_e -level begin to send data packets of S-2 immediately after they send out the data packets of S-1. Since $\Delta_e \leq 4$, after the nodes at the $D_{L'}$ -level receive all the data packets of S-2, they may still be busy with the transmission of the data packets of S-1. Nevertheless, from the $C_{L'-1}$ -level to the D_1 -level, the pipeline can be utilized in a maximum degree, which implies whatever the data packets of S-1 or the data packets of S-2, they can be sent immediately. After the data packets of S-2 are sent from the nodes at the D_0 -level to the nodes in C_0 , they may have to wait for a while at the nodes of the C_0 -level, since the transmission for the data packets of S-1 may last for as long as $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs. This implies the sink will receive all the data packets of S-2 in $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs after it receives all the data packets of S-1. According to the description of the pipeline scheduling algorithm, the subsequent snapshots will be transmitted in the same way, which implies the sink will receive all the data packets of a snapshot within at most every $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSs, after it receives the data packets of the first

Fig. 10 Data transport in (a) Case 1 and (b) Case 4



snapshot which takes at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1)$ time slots by **Theorem 2**. As a result, the number of time slots used to collect the data packets of N continuous snapshots by the pipeline scheduling algorithm is at most

$$M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1) + (N - 1) \cdot 12M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil \quad (18)$$

$$= M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1 + 12N - 12) \quad (19)$$

$$= M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 12N - 11). \quad (20)$$

Case 2: $4 < \Delta_e \leq 11$ and *Case 3:* $\Delta_e = 12$. These two cases can be proven by the similar method used in Case 1. The number of the time slots used to collect the data packets of N continuous snapshots is also at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 12N - 11)$.

Case 4: $\Delta_e > 12$. Take the data transmission of two snapshots shown in Fig. 10b as an example to show the proof in this case. The notations in Fig. 10b are the same as those in Fig. 10a. Since $\Delta_e > 12$, the pipeline can be utilized in a maximum degree at the $D_{L'}$ -level and continue to the C_0 -level. Then, the sink can receive all the data packets of a subsequent snapshot every $\Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSSs after it receives the data packets of the first snapshot. Therefore, the number of the time slots used to collect the data packets of N continuous snapshots is at most

$$M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1) + (N - 1) \cdot \Delta_e M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil \quad (21)$$

$$= M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1 + (N - 1)\Delta_e) \quad (22)$$

$$= M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (N\Delta_e + 15L' + 1). \quad (23)$$

As a conclusion, [Theorem 3](#) is true. \square

[Theorem 3](#) shows the number of the time slots used to collect N continuous snapshots. It is prepared to derive the achievable capacity of the pipeline scheduling algorithm. The lower bound of the achievable CDC capacity in a long-run is given in [Theorem 4](#).

Theorem 4 *For a long-run CDC, the lower bound of the achievable asymptotic network capacity of the pipeline scheduling algorithm is*

$$\begin{cases} \frac{nW}{12M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M\Delta_e \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases},$$

where $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$ and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$.

Proof [Theorem 4](#) can be proven by two cases.

Case 1: $\Delta_e \leq 12$. In this case the number of the time slots used to collect N continuous snapshots is at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 12N - 11)$ as proven in [Theorem 3](#). Therefore, the lower bound of the capacity of the pipeline scheduling algorithm is

$$\frac{N \cdot nb}{tM \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 12N - 11)} \quad (24)$$

$$= \frac{nb}{tM \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil \left(\frac{\Delta_e}{N} + \frac{15L'}{N} + 12 - \frac{11}{N} \right)} \quad (25)$$

$$= \frac{nW}{M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil \left(\frac{\Delta_e}{N} + \frac{15L'}{N} + 12 - \frac{11}{N} \right)}. \quad (26)$$

When $N \rightarrow \infty$, the above equation approaches to $\frac{nW}{12M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil}$. From [Lemma 1](#),

$$2\beta_{\rho+2} \leq 2 \left[\frac{\pi}{\sqrt{3}} (\rho + 2)^2 + \left(\frac{\pi}{2} + 1 \right) (\rho + 2) + 1 \right] \quad (27)$$

$$= \frac{2\pi}{\sqrt{3}}\rho^2 + \left(\frac{8\pi}{\sqrt{3}} + \pi + 2\right)\rho + \frac{8\pi}{\sqrt{3}} + 2\pi + 6 \quad (28)$$

$$\approx 3.63\rho^2 + c_3\rho + c_4, \quad (29)$$

where $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$ and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$. This implies the asymptotic network capacity in this case is $\frac{nW}{12M\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$.

Case 2: $\Delta_e > 12$. The lower bound of the asymptotic network capacity in this case is $\frac{nW}{M\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$, which can be proven by the similar method as in Case 1. \square

In a dual-radio multichannel WSN, since every node has two radios, the upper bound of the network capacity is $2W$. This is because the sink can receive at most two data packets in one time slot. From [Theorem 4](#), when $\Delta_e \leq 12$ and $M \leq \frac{n}{24\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$, or $\Delta_e > 12$ and $M \leq \frac{n}{2\Delta_e\lceil(3.63\rho^2+c_3\rho+c_4)/H\rceil}$, the achievable CDC capacity of the pipeline scheduling algorithm is greater than $2W$. By checking the reasons carefully, it can be found that the pipeline scheduling and CDG are responsible for this improvement. By forming a CDG-based pipeline, the time overlap of gathering multiple continuous snapshots conserves a lot of time, which accelerates the data collection process directly and significantly. These two reasons are also validated by the simulation results in [Sect. 7](#).

Furthermore, it can be found that the pipeline scheduling algorithm is more effective for large-scale WSNs, since large-scale WSNs incur large size routing trees, which are more suitable for pipeline. The pipeline scheduling algorithm is also more effective for a long-time CDC, which can also be seen from [Theorem 4](#).

6.4 Capacity of the Pipeline Scheduling Algorithm in Single-Radio Multichannel WSNs

For completeness, the achievable network capacity of the pipeline scheduling algorithm in single-radio multichannel WSNs is also analyzed in this subsection. Now, since each sensor node has one radio, some modifications of the pipeline scheduling algorithm are made as follows. For the nodes in D_e , D_l ($1 \leq l \leq L'$), and C_l ($1 \leq l \leq L' - 1$), instead of transmitting the data packets of the $(j+1)$ -th ($j \geq 1$) snapshot immediately after transmitting the data packets of the j -th snapshot, they wait until their parent nodes have transmitted all the data packets of the j -th snapshot successfully. (Note that the transmission of the data from the first snapshot does not have the waiting process.) For convenience, the modified pipeline scheduling algorithm is referred to as the *single-radio-based pipeline scheduling* algorithm. Then, by the similar proof technique shown in [Theorem 2](#), the following lemma can be proven.

Lemma 8 *The number of the time slots used to collect the data packets of the first snapshot by the single-radio-based pipeline scheduling algorithm for single-radio multichannel WSNs is at most $M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1)$.*

On the basis of [Lemma 8](#), the number of time slots used to collect all the data packets of N continuous snapshots by the single-radio-based pipeline scheduling algorithm is shown in [Theorem 5](#).

Theorem 5 *The time slots used by the single-radio-based pipeline scheduling algorithm to collect N continuous snapshots for single-radio multi-channel WSNs are at most*

$$\begin{cases} M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 16N - 15), & \text{if } \Delta_e \leq 12 \\ M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (N\Delta_e + 15L' + 4N - 3), & \text{if } \Delta_e > 12 \end{cases}.$$

Proof Similar as the analysis in the proof of [Theorem 3](#), when $\Delta_e \leq 12$, the sink will receive all the data packets of a snapshot within every $12 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil + 4 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil = 16 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSSs after it receives the data packets of the first snapshot, which implies that the number of time slots used to collect the data packets of N continuous snapshots by the single-radio-based pipeline scheduling algorithm is at most

$$M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1) + (N - 1) \cdot 16 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil \quad (30)$$

$$= M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 16N - 15). \quad (31)$$

Similarly, when $\Delta_e > 12$, the sink will receive all the data packets of a snapshot within every $\Delta_e \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil + 4 \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil = (\Delta_e + 4) \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil$ STSSs after it receives the data packets of the first snapshot, which implies the number of time slots used to collect the data packets of N continuous snapshots by the single-radio-based pipeline scheduling algorithm is at most

$$M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (\Delta_e + 15L' + 1) + (N - 1) \cdot (\Delta_e + 4) \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil \quad (32)$$

$$= M \left\lceil \frac{2\beta_{\rho+2}}{H} \right\rceil (N\Delta_e + 15L' + 4N - 3). \quad (33)$$

□

Table 2 Comparison of the multipath scheduling algorithm, the pipeline scheduling algorithm, and the best existing works

Algorithm name	SDC/CDC	IM	RWN/AWN	Υ
Zhu's algorithm [66]	SDC	PrIM	RWN	$\Theta(W)$
Chen's algorithm [12]	SDC	PrIM	RWN, AWN	$\Theta(W)$
Luo's algorithm (CDG) [46]	SDC	PrIM/PhIM	RWN	$\Theta(W)$
Chen's algorithm [11]	SDC/CDC	PhIM	RWN	$\Omega(W)$
Multipath scheduling	SDC	PrIM	RWN	$\Omega(W)$
Pipeline scheduling	CDC	PrIM	RWN	$\Omega\left(\frac{nW}{12M}\right)$ or $\Omega\left(\frac{nW}{M\Delta_e}\right)$

IM interference model, *PrIM* protocol interference model, *PyIM* physical interference model, *RWN* random wireless networks, *AWN* arbitrary wireless networks

Therefore, based on [Theorem 5](#), the lower bound of the achievable CDC capacity of the single-radio-based pipeline scheduling algorithm in a long-run is shown in [Theorem 6](#).

Theorem 6 *For a long-run CDC, the lower bound of the achievable asymptotic network capacity of the single-radio-based pipeline scheduling algorithm for single-radio multichannel WSNs is*

$$\begin{cases} \frac{nW}{16M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M(\Delta_e + 4) \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases},$$

where $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$ and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$.

Proof By the similar technique in the proof of [Theorem 4](#) and based on [Theorem 5](#), the results of this theorem hold. \square

From [Theorems 4](#) and [5](#), the capacity improvement ratio of the pipeline scheduling algorithm for multi-radio WSNs compared with the single-radio-based pipeline scheduling algorithm for single-radio WSNs is $\frac{4}{3}$ when $\Delta_e \leq 12$, or $\frac{\Delta_e + 4}{\Delta_e}$ when $\Delta_e > 12$.

In a sum, the achievable network capacities of the proposed algorithms are compared with the most recently published algorithms for data collection, and the result is shown in [Table 2](#).

7 Simulations and Results Analysis

In this section, simulations are conducted to verify the performances of the proposed algorithms through implementing them with the C language. For all the simulations, assume every WSN has one sink, and all the sensor nodes of each WSN are randomly distributed in a square area and the communication radius of each node is normalized to one. Suppose the network MAC layer works with TDMA, i.e., the network time can be slotted. Every node produces one data packet in a snapshot and the size of a data packet is normalized to one. Every available channel has the same bandwidth normalized to one. For any two different channels, suppose they are orthogonal, i.e., the communications initialized over any two channels have no wireless interference. Furthermore, assume a data packet can be transmitted over a channel within a time slot.

The compared algorithms are BFS [12], SLR [6], and CDG [46]. BFS is an SDC algorithm based on a *breadth-first search* tree and the scheduling is carried out path by path [12]. BFS is specifically proposed for single-radio single-channel WSNs. In the following simulations, BFS is extended to the dual-radio multichannel scenario for fairness. SLR is a *straight-line routing* method for multi-unicast communication in multichannel wireless networks with channel switching constraints [6]. For data collection, SLR works by setting every sensor having a unicast communication with the sink simultaneously. The channel switching constraints in SLR are also removed in the following simulations for fairness. Furthermore, the pipelined versions of BFS and SLR (i.e., add the pipeline technique to the data transmission in BFS and SLR), referred to as BFS-P and SLR-P, respectively, are also implemented, when evaluating the performance of the proposed pipeline scheduling algorithm. The basic idea of CDG is discussed in Sect. 6. The proposed *multipath scheduling* algorithm for SDC is referred to as MPS and the proposed *pipeline scheduling* algorithm for CDC is referred to as PS in the following discussions.

In the remainder of this section, the achievable capacities of MPS and PS are investigated through three groups of simulations, respectively. In the simulations, H is the number of the available channels, ρ is the interference radius, n is the number of the sensors in a WSN, AR refers to the square area where a WSN is deployed, and N is the number of the snapshots in a CDC task.

7.1 Performance of MPS

The SDC capacities of MPS, BFS, and SLR in different network scenarios are shown in Fig. 11. In Fig. 11a, the capacity of every algorithm increases when the number of the available channels increases. This is because more available channels enable more concurrent transmissions, which accelerates the data collection process resulting in a higher capacity. After the number of the available channels arrives at 4, the capacities of BFS and SLR almost maintain the same level. This is because four channels are enough to prevent channel interference. However, radio confliction becomes the main barrier of a higher capacity at this time. MPS achieves a

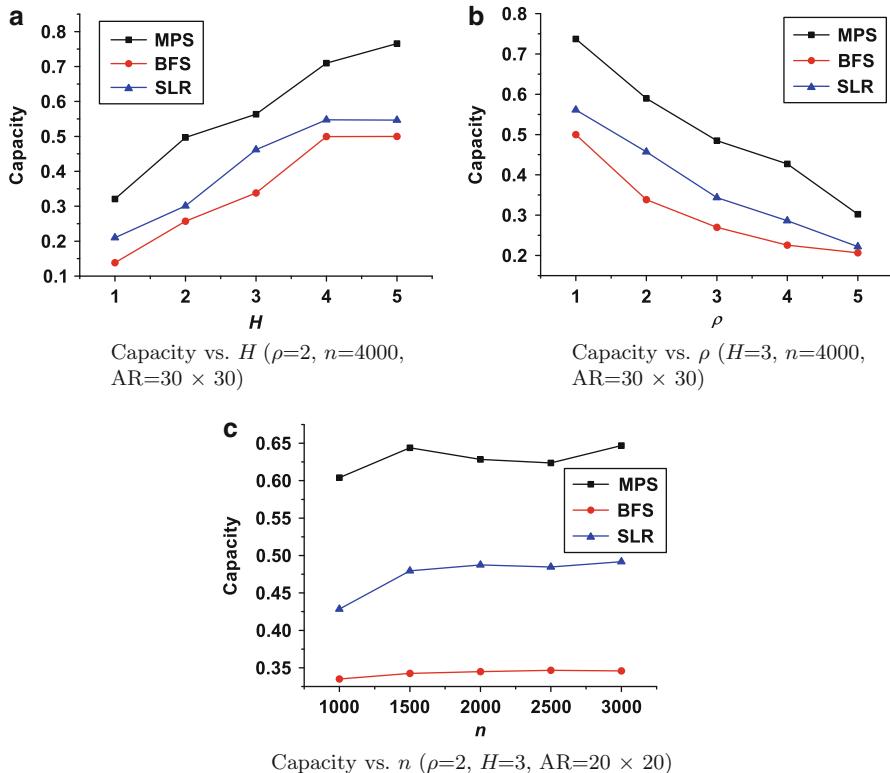


Fig. 11 SDC capacity

higher capacity compared with BFS and SLR. This is because MPS simultaneously schedules all the paths without radio confliction (except at the sink). Since radio confliction on a single path can be avoided easily, MPS can simultaneously schedule all the links without radio confliction on multiple paths, which implies that MPS can make use of channels in a maximum degree. Whereas, BFS just schedules links without radio confliction on one path every time and SLR schedules all the transmission links simultaneously, which leads to serious radio confliction. On average, MPS achieves 77.49 and 41.95 % more capacity than BFS and SLR, respectively.

The effect of the interference radius on the capacity is shown in Fig. 11b. With the increase of the interference radius, more transmission interference occurs, which leads to the decrease of the capacities of all the algorithms. Nevertheless, MPS still achieves the largest capacity since it simultaneously schedules multiple paths without radio confliction, which suggests a nice trade-off between BFS and SLR. On average, MPS achieves 67.45 and 37.37 % more capacity than BFS and SLR, respectively.

The effect of the number of the sensors on the capacity is shown in Fig. 11c. It can be seen that the number of the sensors in a network has a little impact on the capacities of MPS and SLR and almost no impact on the capacity of BFS. There are two reasons for this result. First, BFS is a single-path scheduling algorithm. Whatever the number of the sensors is, it schedules only one path every time. Second, the number of the channels is fixed to 2 in all of these three algorithms. This implies that whatever the number of the sensors is, they can simultaneously schedule at most two interfering links without radio confliction. On average, MPS achieves 83.51 and 32.87 % more capacity than BFS and SLR, respectively.

7.2 Performance of PS

The CDC capacities of PS, CDG, BFS-P, BFS, SLR-P, and SLR in different network scenarios are shown in Fig. 12. Figure 12a (respectively, Fig. 12b, c) and Fig. 12d (respectively, Fig. 12e, f) are the same except the achievable capacity of PS in Fig. 12d (respectively, Fig. 12e, f) is not shown. This is mainly for conveniently and clearly checking the achievable capacities of CDG, BFS-P, BFS, SLR-P, and SLR.

Figure 12a, d reflect the effect of the number of the available channels on the achievable CDC capacity. As explained before, the capacities of all the algorithms increase as more and more channels are available. This is because more channels can prevent channel interferences among concurrent transmission links. PS has a much higher capacity compared with the other five algorithms. This is because by forming a CDG-based pipeline, the time overlap of gathering multiple continuous snapshots conserves a lot of time, which accelerates the data collection process directly and significantly in PS. Furthermore, PS collects data in the CDG way, which can reduce the overall data transmission times. This also explains why CDG has a higher capacity compared with BFS and SLR. From Fig. 12d, it can also be seen that BFS-P and SLR-R have higher network capacities than BFS and SLR, respectively. This is because the use of the pipeline technique can accelerate the data collection process. On average, PS achieves a capacity of 8.22 times of that of CDG, 17.1 times of that of BFS-P, 21.94 times of that of BFS, 13.17 times of that of SLR-P, and 18.39 times of that of SLR, respectively.

The effect of the interference radius on the capacity is shown in Fig. 12b, e. With the increase of the interference radius, a transmission link will interfere with more and more other transmission links, which leads to the decrease of capacity whatever algorithm it is. PS has the highest capacity among all the algorithms since it works with pipeline and CDG. On average, PS achieves a capacity of 7.31 times of that of CDG, 15.9 times of that of BFS-P, 20.43 times of that of BFS, 12.53 times of that of SLR-P, and 16.08 times of that of SLR, respectively.

The effect of the number of the sensors on the capacity is shown in Fig. 12c, f. The increase of the number of the sensors has a little impact on BFS and SLR, and the reasons are similar to those explained in the previous subsection. Whereas, the capacities of PS and CDG have some improvement with more sensors in a WSN.

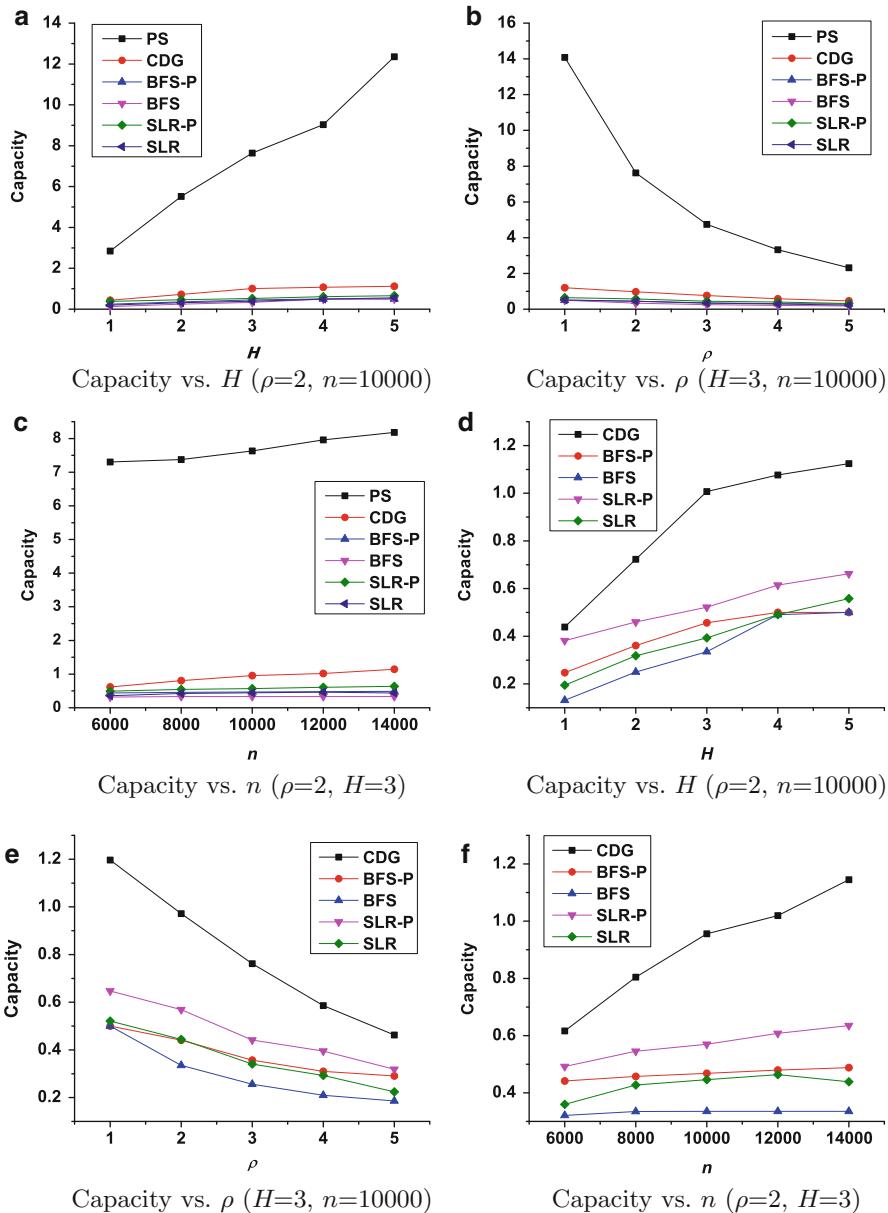


Fig. 12 CDC capacity in different scenarios (AR = 50 × 50, N = 1,000, M = 100)

This is because PS and CDG are more effective for large-scale WSNs. On average, PS achieves a capacity of 8.77 times of that of CDG, 15.48 times of that of BFS-P, 23.15 times of that of BFS, 12.49 times of that of SLR-P, and 18.06 times of that of SLR, respectively.

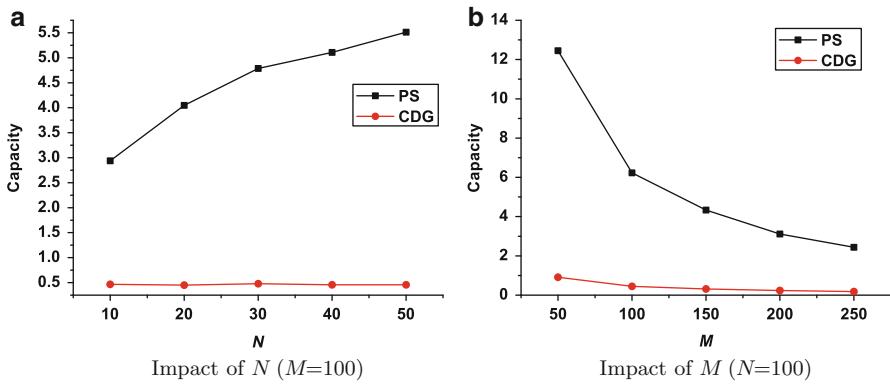


Fig. 13 The impacts of N and M to PS and CDG ($\rho = 2$, $H = 3$, $n = 5,000$, AR = 50×50)

7.3 Impacts of N and M

In this subsection, the impacts of the number of the snapshots and the value of M to the capacities of PS and CDG are investigated. As shown in Fig. 13a, with the increase of N in a CDC task, PS achieves about 87.54 % more capacity. This is straightforward from the analysis in Sect. 6. Since PS employs the pipeline technology, the transmissions of continuous snapshots are overlapped, which can significantly reduce the time used to collect all the snapshots data. With more snapshots in a CDC task, the capacity of PS approaches closer and closer to its theoretical asymptotic capacity. For CDG, the number of the snapshots has little impact on its capacity.

Since the performance of CDG depends on the value of M , the capacities of both PS and CDG decrease about 80 % with the increase of the value of M as shown in Fig. 13b. This is because a bigger M implies that more data packets have to be transmitted for every sensor, and longer transmission time for each snapshot is resulted. Nevertheless, considering that the value of M is usually much less than n , PS can still achieve a high capacity.

8 Conclusion

Motivated by the fact that there exist no works dedicated on studying the capacity of continuous data collection for WSNs under the protocol interference model, this problem in dual-radio multichannel WSNs is investigated in this chapter. A *multipath scheduling* algorithm for the snapshot data collection problem is first proposed. Theoretical analysis of the multi-path scheduling algorithm shows that its achievable network capacity is at least $\frac{W}{2\lceil(1.81\rho^2+c_1\rho+c_2)/H\rceil}$, where W is the channel bandwidth, H is the number of available orthogonal channels, ρ is the ratio of the

interference radius over the transmission radius of a node, $c_1 = \frac{2\pi}{\sqrt{3}} + \frac{\pi}{2} + 1$, and $c_2 = \frac{\pi}{\sqrt{3}} + \frac{\pi}{2} + 2$. For the continuous data collection problem, it is found that pipeline with the existing snapshot data collection methods cannot actually improve the network capacity. The reason of this is analyzed, and then a novel continuous data collection method, the *pipeline scheduling* algorithm, for dual-radio multichannel WSNs is proposed. This method speeds up the data collection process significantly. Theoretical analysis of the pipeline scheduling algorithm shows that the achievable asymptotic network capacity is

$$\begin{cases} \frac{nW}{12M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M\Delta_e \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases},$$

where n is the number of the sensors, M is a constant value and usually $M \ll n$, Δ_e is the maximum number of the leaf nodes having a same parent in the routing tree (i.e., data collection tree), $c_3 = \frac{8\pi}{\sqrt{3}} + \pi + 2$, and $c_4 = \frac{8\pi}{\sqrt{3}} + 2\pi + 6$. Furthermore, for completeness, the performance of the proposed pipeline scheduling algorithm in single-radio multichannel WSNs is also analyzed, which shows that for a long-run CDC, the lower bound of the achievable asymptotic network capacity is

$$\begin{cases} \frac{nW}{16M \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e \leq 12 \\ \frac{nW}{M(\Delta_e + 4) \left\lceil \frac{3.63\rho^2 + c_3\rho + c_4}{H} \right\rceil}, & \text{if } \Delta_e > 12 \end{cases}.$$

Simulation results indicate that the proposed algorithms improve the network capacity significantly compared with existing works.

The future work of this chapter can be conducted according to the following directions:

- First, in this chapter, the snapshot and continuous data collection problems are studied for single/dual-radio multichannel WSNs. It is expected to extend the work in this chapter to general multi-radio multichannel WSNs to study the achievable capacities of snapshot and continuous data collection.
- Second, the WSNs considered in this chapter are randomly deployed. It is expected to further investigate the snapshot/continuous capacity in arbitrary WSNs.
- Third, the parameter M is crucial for the performance of the pipeline scheduling algorithm. Although the authors in [46] indicated that $M = 3K \sim 4K$ is usually

sufficient for CDG, where K is a value determined by the correlations of the data of a snapshot and $K \ll n$, it is also expected to derive the function relation between M and n for random WSNs, as well as arbitrary WSNs, in the future work.

- Finally, to study the snapshot and continuous capacities for arbitrary WSNs under the *general physical interference model* will be another future research direction.

Acknowledgements The authors would like to thank the editors and reviewers for their time and valuable comments that help to improve the quality of this chapter. This chapter is also partly supported by the NSF under Grant No. CCF-0545667.

Cross-References

- [Advances in Scheduling Problems](#)
- [Connected Dominating Set in Wireless Networks](#)
- [Energy Efficiency in Wireless Networks](#)
- [Online and Semi-online Scheduling](#)
- [Online Frequency Allocation and Mechanism Design for Cognitive Radio Wireless Networks](#)
- [Optimization in Multi-Channel Wireless Networks](#)
- [Optimization Problems in Data Broadcasting](#)
- [Resource Allocation Problems](#)

Recommended Reading

1. A. Agarwal, P.R. Kumar, Capacity bounds for ad hoc and hybrid wireless networks. *ACM SIGCOMM Comput. Commun. Rev.* **34**(3), 71–81 (2004)
2. I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey. *Comput. Netw.* **38**, 393–422 (2002)
3. M. Alicherry, R. Bhatia, L.E. Li, Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks, in *ACM MobiCom*, Cologne, Germany, 28 Aug–2 Sept 2005
4. M. Andrews, M. Dinitz, Maximizing capacity in arbitrary wireless networks in the SINR model: complexity and game theory, in *IEEE INFOCOM*, Rio De Janeiro, Brazil, 19–25 Apr 2009
5. P. Bahl, R. Chandra, J. Dunagan, Ssch: slotted seeded channel hopping for capacity improvement in IEEE 802.11 Ad Hoc wireless networks, in *ACM MobiCom*, Philadelphia, PA, USA, 26 Sept–1 Oct 2004
6. V. Bhandari, N.H. Vaidya, Connectivity and capacity of multi-channel wireless networks with channel switching constraints, in *IEEE INFOCOM*, Anchorage, AK, USA, 6–12 May 2007
7. V. Bhandari, N.H. Vaidya, Capacity of multi-channel wireless networks with random (c, f) assignment, in *ACM MobiHoc*, Montreal, QC, Canada, 9–14 Sept 2007
8. D. Chafekar, D. Levin, V.S.A. Kumar, M.V. Marathe, S. Parthasarathy, A. Srinivasan, Capacity of asynchronous random-access scheduling in wireless networks, in *IEEE INFOCOM*, Phoenix, AZ, USA, 15–17 Apr 2008
9. C.-K. Chau, M. Chen, S.C. Liew, Capacity of large-scale CSMA wireless networks, in *ACM MobiCom*, Beijing, China, 20–25 Sept 2009

10. S. Chen, Y. Wang, X.-Y. Li, X. Shi, Order-optimal data collection in wireless sensor networks: delay and capacity, in *IEEE SECON*, Rome, Italy, 22–26 June 2009
11. S. Chen, Y. Wang, X.-Y. Li, X. Shi, Data collection capacity of random-deployed wireless sensor networks, in *IEEE Globecom*, Honolulu, HI, USA, 30 Nov–4 Dec 2009
12. S. Chen, S. Tang, M. Huang, Y. Wang, Capacity of data collection in arbitrary wireless sensor networks, in *IEEE INFOCOM*, San Diego, CA, USA, 15–19 Mar 2010
13. W. Cheng, X. Chen, T. Znati, X. Lu, Z. Lu, The complexity of channel scheduling in multi-radio multi-channel wireless networks, in *IEEE INFOCOM*, Rio De Janeiro, Brazil, 19–25 Apr 2009
14. O. Chipara, C. Lu, J. Stankovic, Dynamic conflict-free query scheduling for wireless sensor networks, in *IEEE ICNP*, Santa Barbara, CA, USA, 12–15 Nov 2006
15. D. Culler, M. Srivastava, D. Estrin, Guest editors' introduction: overview of sensor networks. *IEEE Comput.* **37**(8), 41–49 (2004)
16. H.-N. Dai, K.-W. Ng, R.C.-W. Wong, M.-Y. Wu, On the capacity of multi-channel wireless networks using directional antennas, in *IEEE INFOCOM*, Phoenix, AZ, USA, 15–17 Apr 2008
17. E.J. Duarte-Melo, M. Liu, Data-gathering wireless sensor networks: organization and capacity. *Comput. Netw.* **43**, 519–537 (2003)
18. M. Franceschetti, O. Dousse, D.N.C. Tse, P. Thiran, Closing the gap in the capacity of wireless networks via percolation theory. *IEEE Trans. Inf. Theory* **53**(3), 1009–1018 (2007)
19. H.E. Gamal, On the scaling laws of dense wireless sensor networks. *IEEE Trans. Inf. Theory* **51**(3), 1229–1234 (2005)
20. M. Garetto, P. Giaccone, E. Leonardi, On the capacity of Ad Hoc wireless networks under general node mobility, in *IEEE INFOCOM*, Anchorage, AK, USA, 6–12 May 2007
21. A. Ghosh, Ö.D. Incel, V.S. Anil Kumar, B. Krishnamachari, Multi-channel scheduling algorithms for fast aggregated convergecast in sensor networks, in *IEEE MASS*, Macau, China, 12–15 Oct 2009
22. O. Goussevskaia, R. Wattenhofer, M.M. Halldorsson, E. Welzl, Capacity of arbitrary wireless networks, in *IEEE INFOCOM*, Rio De Janeiro, Brazil, 19–25 Apr 2009
23. J. Gummesson, D. Ganesan, M.D. Corner, P. Shenoy, An adaptive link layer for range diversity in multi-radio mobile sensor networks, in *IEEE INFOCOM*, Rio De Janeiro, Brazil, 19–25 Apr 2009
24. P. Gupta, P.R. Kumar, The capacity of wireless networks. *IEEE Trans. Inf. Theory* **46**(2), 388–404 (2000)
25. B. Han, V.S. Anil Kumar, M.V. Marathe, S. Parthasarathy, A. Srinivasan, Distributed strategies for channel allocation and scheduling in software-defined radio networks, in *IEEE INFOCOM*, Rio De Janeiro, Brazil, 19–25 Apr 2009
26. J.(S.) He, Z. Cai, S. Ji, R. Beyah, Y. Pan, A genetic algorithm with immigrants schemes for constructing a reliable MCDS in probabilistic wireless sensor networks, in *WASA*, Chengdu, China, 11–13 Aug 2011
27. J.(S.) He, S. Ji, M. Yan, Y. Pan, Y. Li, Genetic-algorithm-based construction of load-balanced CDSs in wireless sensor networks, in *MILCOM*, Baltimore, MD, USA, 7–10 Nov 2011
28. S.C.-H. Huang, P.-J. Wan, C.T. Vu, Y. Li, F. Yao, Nearly constant approximation for data aggregation scheduling in wireless sensor networks, in *IEEE INFOCOM*, Anchorage, AK, USA, 6–12 May 2007
29. W. Huang, X. Wang, Q. Zhang, Capacity scaling in mobile wireless Ad Hoc network with infrastructure support, in *ICDCS*, Genoa, Italy, 21–25 June 2010
30. S. Ji, Study on routing protocols and query processing techniques in multi-radio wireless sensor networks (in Chinese). Master Thesis, Heilongjiang University, Harbin, Heilongjiang, China, 2010
31. S. Ji, Y. Li, X. Jia, Capacity of dual-radio multi-channel wireless sensor networks for continuous data collection, in *IEEE INFOCOM*, Shanghai, China, 10–15 Apr 2011
32. S. Ji, R. Beyah, Y. Li, Continuous data collection capacity of wireless sensor networks under physical interference model, in *IEEE MASS*, Valencia, Spain, 17–22 Oct 2011

33. C. Joo, J.-G. Choi, N.B. Shroff, Delay performance of scheduling with data aggregation in wireless sensor networks, in *IEEE INFOCOM*, San Diego, CA, USA, 15–19 Mar 2010
34. H. Karl, A. Willig, *Protocols and Architectures for Wireless Sensor Networks* (Wiley, Hoboken, 2005)
35. A. Keshavarz-Haddad, V. Ribeiro, R. Riedi, Broadcast capacity in multihop wireless networks, in *ACM MobiCom*, Los Angeles, CA, USA, 24–29 Sept 2006
36. V.S.A. Kumar, M.V. Marathe, S. Parthasarathy, A. Srinivasan, Algorithmic aspects of capacity in wireless networks, in *ACM SigMetrics*, Banff, AB, Canada, 6–10 June 2005
37. P. Kyasanur, N.H. Vaidya, Capacity of multi-channel wireless networks: impact of number of channels and interfaces, in *ACM MobiCom*, Cologne, Germany, 28 Aug–2 Sept 2005
38. J. Li, Research on key techniques of data management over sensor networks (in Chinese). Doctoral Dissertation, Harbin Institute of Technology, Harbin, Heilongjiang, China, 2007
39. Y. Li, M.T. Thai, F. Wang, D.-Z. Du, On the construction of a strongly connected broadcast arborescence with bounded transmission delay. *IEEE Trans. Mobile Comput.* **5**(10), 1460–1470 (2006)
40. X.-Y. Li, S. Tang, O. Frieder, Multicast capacity for large scale wireless Ad Hoc networks, in *ACM MobiCom*, Montreal, QC, Canada, 9–14 Sept 2007
41. S. Li, Y. Liu, X.-Y. Li, Capacity of large scale wireless networks under gaussian channel model, in *ACM MobiCom*, San Francisco, CA, USA, 14–19 Sept 2008
42. X.-Y. Li, J. Zhao, Y.W. Wu, S. Tang, X.H. Xu, X.F. Mao, Broadcast capacity for wireless Ad Hoc networks, in *IEEE MASS*, Atlanta, GA, USA, 29 Sept–2 Oct 2008
43. X. Lin, S.B. Rasool, Distributed and provably efficient algorithms for joint channel-assignment, scheduling, routing in multichannel Ad Hoc wireless networks. *IEEE/ACM Trans. Netw.* **17**(6), 1874–1886 (2009)
44. C. Liu, G. Cao, Distributed monitoring and aggregation in wireless sensor networks, in *IEEE INFOCOM*, San Diego, CA, USA, 15–19 Mar 2010
45. B. Liu, D. Towsley, A. Swami, Data gathering capacity of large scale multihop wireless networks, in *IEEE MASS*, Atlanta, GA, USA, 29 Sept–2 Oct 2008
46. C. Luo, F. Wu, J. Sun, C.W. Chen, Compressive data gathering for large-scale wireless sensor networks, in *ACM MobiCom*, Beijing, China, 20–25 Sept 2009
47. D. Lymberopoulos, N.B. Priyantha, M. Goraczko, F. Zhao, Towards energy efficient design of multi-radio platforms for wireless sensor networks, in *ACM/IEEE IPSN*, St. Louis, MO, USA, 22–24 Apr 2010
48. X. Mao, X.-Y. Li, S. Tang, Multicast capacity for hybrid wireless networks, in *ACM Mobicom*, Hong Kong SAR, China, 27–30 May 2008
49. D. Marco, E.J. Duarte-Melo, M. Liu, D.L. Neuhoff, On the many-to-one transport capacity of a dense wireless sensor network and the compressibility of its data, in *ACM/IEEE IPSN*, Palo Alto, CA, USA, 22–23 April 2003
50. D.W. Matula, L.L. Beck, Smallest-last ordering and clustering and graph coloring algorithms. *J. Assoc. Comput. Mach.* **30**(3), 417–427 (1983)
51. T. Moscibroda, The worst-case capacity of wireless sensor networks, in *ACM/IEEE IPSN*, Cambridge, MA, USA, 25–27 Apr 2007
52. U. Niesen, P. Gupta, D. Shah, On capacity scaling in arbitrary wireless networks. *IEEE Trans. Inf. Theory* **55**(9), 3959–3982 (2009)
53. U. Niesen, P. Gupta, D. Shah, The balanced unicast and multicast capacity regions of large wireless networks. *IEEE Trans. Inf. Theory* **56**(5), 2249–2271 (2010)
54. V. Ramamurthi, S.K.C. Vadrevu, A. Chaudhry, M.R. Bhatnagar, Multicast capacity of multi-channel multihop wireless networks, in *IEEE WCNC 2009*, Budapest, Hungary, 5–8 Apr 2009
55. G. Sharma, R. Mazumdar, N.B. Shroff, Delay and capacity trade-offs in mobile Ad Hoc networks: a global perspective. *IEEE/ACM Trans. Netw.* **15**(5), 981–992 (2007)

56. T. Stathopoulos, M. Lukac, D. McIntire, J. Heidemann, D. Estrin, W.J. Kaiser, End-to-end routing for dual-radio sensor networks, in *IEEE INFOCOM*, Anchorage, AK, USA, 6–12 May 2007
57. P.-J. Wan, S.C.-H. Huang, L. Wang, Z. Wan, X. Jia, Minimum-latency aggregation scheduling in multihop wireless networks, in *ACM MobiHoc*, New Orleans, LA, USA, 18–21 May 2009
58. P.-J. Wan, Z. Wang, Z. Wan, S.C.-H. Huang, H. Liu, Minimum-latency scheduling for group communications in multi-channel multihop wireless networks, in *WASA*, Boston, MA, USA, 16–18 Aug 2009
59. Z. Wang, H.R. Sadjadpour, J.J. Garcia-Luna-Aceves, A unifying perspective on the capacity of wireless Ad Hoc networks, in *IEEE INFOCOM*, Phoenix, AZ, USA, 15–17 Apr 2008
60. Z. Wang, H.R. Sadjadpour, J.J. Garcia-Luna-Aceves, The capacity and energy efficiency of wireless Ad Hoc networks with multi-packet reception, in *ACM MobiHoc*, Hong Kong SAR, China, 27–30 May 2008
61. X. Wang, Y. Bei, Q. Peng, L. Fu, Speed improves delay-capacity tradeoff in motionCast. *IEEE Trans. Parallel Distrib. Syst.* **22**(5), 729–742 (2011)
62. J. Wu, B. Wu, I. Stojmenovic, Power-aware broadcasting and activity scheduling in ad hoc wireless networks using connected dominating sets. *Wirel. Commun. Mobile Comput.* **3**(4), 425–438 (2003)
63. Y. Xu, W. Wang, Scheduling partition for order optimal capacity in large-scale wireless networks, in *ACM MobiCom*, Beijing, China, 20–25 Sept 2009
64. G. Zhang, Y. Xu, X. Wang, M. Guizani, Capacity of hybrid wireless networks with directional antenna and delay constraint. *IEEE Trans. Commun.* **58**(7), 2097–2106 (2010)
65. M. Zhao, Y. Yang, An optimization based distributed algorithm for mobile data gathering in wireless sensor networks, in *IEEE INFOCOM*, San Diego, CA, USA, 15–19 Mar 2010
66. X. Zhu, B. Tang, H. Gupta, Delay efficient data gathering in sensor networks, in *IEEE MSN*, Wuhan, Hubei, 13–15 Dec 2005

Packing Circles in Circles and Applications

Zhao Zhang, Weili Wu, Ling Ding, Qinghai Liu and Lidong Wu

Contents

1	Introduction	2550
2	An Inequality	2551
3	Packing Independent Points	2556
4	Sink Placement in Underwater Sensor Networks	2565
5	Data Aggregation Scheduling	2568
5.1	Pipelined Aggregation Scheduling (PAS)	2569
5.2	Enhanced Pipelined Aggregation Scheduling (EPAS)	2571
5.3	A Generic Method for $\rho > 1$	2572
6	Maximum Interference-Free Sets of Links	2573
7	Diameter Bounded Connected Dominating Sets	2576
8	Routing-Cost Bounded Connected Dominating Sets	2579
9	Conclusion	2581
	Cross-References	2581
	Recommended Reading	2581

Z. Zhang (✉) • Q. Liu

College of Mathematics and System Sciences, Xinjiang University, Urumqi, Xinjiang, People's Republic China

e-mail: hxhzz@sina.com; zhzhao@xju.edu.cn; liuqh506@163.com

W. Wu • L. Wu

Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA
e-mail: weiliwu@utdallas.edu; wxw020100@utdallas.edu; lidong.wu@utdallas.edu

L. Ding

Institute of Technology, University of Washington Tacoma, Tacoma, WA, USA
e-mail: lingding@u.washington.edu; ling.ding.sjtu@gmail.com

Abstract

Packing circles in a circle is a classic mathematical problem, which has been studied for a long time. Recently, this problem received a great deal of applications in the study of wireless networks. Those applications are related to many combinatorial optimization problems raised in wireless networks. In analyzing approximation algorithms for those combinatorial optimization problems, it is required to establish an upper bound for the number of untouched unit circles which can be packed into a circle of radius r , where a unit circle is a circle with diameter one. This bound can be easily derived from some classic bounds for the number of unit circles packed into a circle. Moreover, analysis of those approximation algorithms also promoted some new problems about packing circles in circles. In this chapter, a state of the art on the developments in this research direction is presented.

Keywords

Circle packing • Wireless networks

1 Introduction

A circle is *unit* if its diameter is one. What is the minimum radius $r(k)$ of a circle which can contain k nonoverlapping unit circles?

Kravitz [31] presented an upper bound of $r(k)$ for $k = 2, \dots, 16$ in 1967, and for $2 \leq k \leq 4$, the bound is trivially optimal. The optimality of Kravitz's bound is proved for $5 \leq k \leq 7$ by Graham [20] in 1968 and for $8 \leq k \leq 9$ by Piri [45] in 1969. In the same paper, Piri also determined the value of $r(10)$ which indicates that Kravitz's bound is not optimal for $k = 10$. Meanwhile, Piri [45] made a conjecture on optimal packings for $11 \leq k \leq 19$. His conjecture is proved for $k = 11$ by Melissen [40], for $k = 12$ by Fodor [13], for $k = 13$ by Fodor [14], and for $k = 19$ by Fodor [12]. Those values of $r(k)$ are as follows:

$$r(1) = 0.5,$$

$$r(2) = 1,$$

$$r(3) = 0.5 + \frac{1}{\sqrt{3}} = 1.077 \dots,$$

$$r(4) = (1 + \sqrt{2})/2 = 1.207 \dots,$$

$$r(5) = \left(1 + \sqrt{\frac{10 + 2\sqrt{5}}{5}}\right)/2 = 1.350 \dots,$$

$$r(6) = 1.5,$$

$$r(7) = 1.5,$$

$$r(8) = (1 + \csc(\pi/7))/2 = 1.652 \dots,$$

$$\begin{aligned}
r(9) &= \left(1 + \sqrt{4 + 2\sqrt{2}}\right)/2 = 1.806\dots, \\
r(10) &= 1.906\dots, \\
r(11) &= (1 + \text{scs}(\pi/9))/2 = 1.961\dots, \\
r(12) &= 2.014\dots, \\
r(13) &= (2 + \sqrt{5})/2 = 2.228\dots, \\
r(19) &= (1 + \sqrt{2} + \sqrt{6})/2 = 2.431\dots.
\end{aligned}$$

For $k = 14, 16, 17, 20$, Goldberg [18] improved the conjectured value of $r(k)$, and Reis [47] gave a further improvement for $k = 17$ and some dense packings for $21 \leq k \leq 25$. Graham et al. [23] found the best known packing for $26 \leq k \leq 65$. Graham and Lubachevsky [22] and Lubachevsky and Graham [36] gave dense packings for some families of infinitely many values of k . Lv and Huang [37] and Lubachevsky [35] designed computer programs to find dense packing for general k .

Tarnai and Miyazaki [51] found interesting connections between circle packing, sacred lotus, and Japanese Buddhist art. There also exist relationships between optimal packing and optimal codes [7, 21]. Recently, more and more applications of circle packing are found in the analysis of approximation algorithms for optimization problems in wireless sensor networks. In this chapter, a state of the art on the developments in this research direction is presented.

2 An Inequality

Let X be a compact convex region on the Euclidean plane. Suppose X contains a set V of centers of n nonoverlapping unit circles. Then

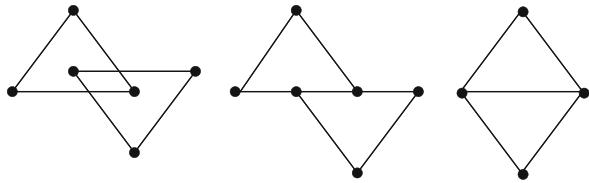
$$n \leq \frac{2}{\sqrt{3}}A(X) + \frac{1}{2}P(X) + 1,$$

where $A(X)$ and $P(X)$ denote the area and the perimeter of X , respectively. This inequality was conjectured by H. Zassenhaus in 1947 (see [59]) and independently proved by Groemer [24] in 1960 and Oler [44] in 1961. Folkman and Graham [15] extended this inequality to simplicial complex.

A point is a zero-dimensional simplex, a straight line segment is a one-dimensional simplex, and a triangle is a two-dimensional simplex. In general, a *simplex* is a polytope with the minimum number of vertices among all the polytopes of a certain dimension. For example, a tetrahedron is a three-dimensional polytope with the minimum number of vertices and hence a three-dimensional simplex.

Any simplex is the convex hull of its vertices. The convex hull of any subset of vertices in a simplex S is also a simplex, which is called a *face* of simplex S . A family Δ of simplexes is called a *simplicial complex* if it satisfies the following two conditions:

Fig. 1 Three families of simplexes



- (a) For $S \in \Delta$, every face of S is in Δ .
- (b) For $S, S' \in \Delta$, $S \cap S'$ is a face for both S and S' .
From (a) and (b), it is easy to see the following:
- (c) For $S, S' \in \Delta$, $S \cap S'$ is also a simplex in Δ .

In Fig. 1, there are three families of simplexes. While the first two are not simplicial complexes, the last one is.

For any simplex A , $|A|$ denotes the number of vertices in A . Thus, $|A| - 1$ is the dimension of A . The *Euler characteristic* of a simplicial complex Δ is defined by

$$\chi(\Delta) = \sum_{A \in \Delta, A \neq \emptyset} (-1)^{|A|-1}.$$

Let $m(A)$ denote the area of A for two-dimensional simplex A and the length of A for one-dimensional simplex A . For one-dimensional simplex A , let $\varepsilon(A, \Delta)$ denote the number of two-dimensional simplexes in Δ having A as a face. For simplicial complex Δ in the Euclidean plane, it is easy to see from (b) that $\varepsilon(A, \Delta) \leq 2$. When $\varepsilon(A, \Delta) = 1$, A is on the boundary of the union of the simplexes in Δ . When $\varepsilon(A, \Delta) = 2$, A is in the interior of the union of the simplexes in Δ . In terms of these terminologies, a definition can be given for the area $A(\Delta)$ and the perimeter $P(\Delta)$ of a simplicial complex Δ in the Euclidean plane:

$$A(\Delta) = \sum_{A \in \Delta, |A|=3} m(A)$$

and

$$P(\Delta) = \sum_{A \in \Delta, |A|=2} (2 - \varepsilon(A, \Delta))m(A).$$

The inequality of Folkman and Graham [15] is as follows:

Theorem 1 (Folkman-Graham [15]) *Let Δ be a simplicial complex in the Euclidean plane. Suppose for any two distinct vertices x and y in Δ , $\|xy\| \geq 1$. Then*

$$|\Delta| \leq \frac{2}{\sqrt{3}}A(\Delta) + \frac{1}{2}P(\Delta) + \chi(\Delta)$$

where $|\Delta|$ is the number of vertices in Δ and $\|xy\|$ is the Euclidean distance between x and y .

To prove this inequality, the following two lemmas were proved first.

Lemma 1 Let a, b, c be the lengths of the three edges of a triangle Δ . Suppose $a \geq b \geq c \geq 1$. Then

$$\frac{4}{\sqrt{3}}A(\Delta) + a \geq b + c.$$

Proof By Hero's formula,

$$\begin{aligned} A(\Delta) &= \frac{1}{4}\sqrt{(a+b+c)(a+b-c)(a-b+c)(-a+b+c)} \\ &\geq \frac{1}{4}\sqrt{(a+b+c)(-a+b+c)} \\ &\geq \frac{1}{4}\sqrt{3c(-a+b+c)} \\ &\geq \frac{1}{4}\sqrt{3(-a+b+c)^2} \\ &= \frac{\sqrt{3}}{4}(-a+b+c). \end{aligned}$$

The inequality of the lemma follows. \square

Lemma 2 Let $BCDE$ be a quadrilateral in the Euclidean plane with area A and perimeter P . Suppose the length of every diagonal of $BCDE$ is not less than the length of every edge of $BCDE$ and the length of every edge is at least one. Then

$$\frac{4}{\sqrt{3}}A - P + 2 \geq 0.$$

Proof Without loss of generality, assume $\angle B + \angle D \leq \pi$. Note that in this case, one must have $A = A(\triangle BCE) + A(\triangle CDE)$. Since diagonal CE is the longest edge in $\triangle BCE$ and in $\triangle DEC$, one has $\angle B \geq \pi/3$ and $\angle D \geq \pi/3$. Hence, $\pi/3 \leq \angle B \leq 2\pi/3$ and $\pi/3 \leq \angle D \leq 2\pi/3$. Therefore,

$$\begin{aligned} A &= \frac{1}{2}(|BC| \cdot |BE| \cdot \sin \angle B + |DC| \cdot |DE| \cdot \sin \angle D) \\ &\geq \frac{\sqrt{3}}{4}(|BC| \cdot |BE| + |DC| \cdot |DE|). \end{aligned}$$

Thus,

$$\begin{aligned} \frac{4}{\sqrt{3}}A - P + 2 \\ \geq |BC| \cdot |BE| + |DC| \cdot |DE| - (|BC| + |BE| + |DC| + |DE|) + 2 \end{aligned}$$

$$\begin{aligned}
&= (|BC| - 1)(|BE| - 1) + (|DC| - 1)(|DE| - 1) \\
&\geq 0.
\end{aligned}$$

The lemma is proved. \square

Now, it is ready to prove [Theorem 1](#).

Proof of Theorem 1. The proof uses an induction on the number of one-dimensional simplexes contained in Δ . First, suppose Δ contains no one-dimensional simplex. Then $A(\Delta) = P(\Delta) = 0$ and $\chi(\Delta)$ is equal to the number of vertices. Therefore, the Folkman-Graham inequality is true.

Next, suppose Δ contains k one-dimensional simplexes and $k \geq 1$. Assume that for every simplicial complex with less than k one-dimensional simplexes, the Folkman-Graham inequality holds. Let $\text{union}(\Delta)$ denote the union of simplexes in Δ . Then it is easy to see that for two simplicial complexes Δ and Γ , if $\text{union}(\Delta) = \text{union}(\Gamma)$, then $A(\Delta) = A(\Gamma)$ and $P(\Delta) = P(\Gamma)$. Furthermore, one also has $\chi(\Delta) = \chi(\Gamma)$, since the Euler characteristic $\chi(\Delta)$ depends only on $\text{union}(\Delta)$ (see, e.g., [46]). Therefore, it suffices to show the validity of Folkman-Graham inequality for a specific simplicial complex among the simplicial complexes with the same union. Without loss of generality, suppose that Δ is the one having the minimum total length of one-dimensional simplexes among those simplicial complexes with the same union and the same number of one-dimensional simplexes as Δ has.

Consider a one-dimensional simplex σ in Δ with the longest length. There are three cases in the following.

Case 1. $\varepsilon(\sigma, \Delta) = 0$. In this case, $\Delta - \{\sigma\}$ is a simplicial complex and $P(\Delta - \{\sigma\}) = P(\Delta) - 2m(\sigma)$. Therefore, by induction hypothesis,

$$\begin{aligned}
|\Delta| &= |\Delta - \{\sigma\}| \\
&\leq \frac{2}{\sqrt{3}}A(\Delta - \{\sigma\}) + \frac{1}{2}P(\Delta - \{\sigma\}) + \chi(\Delta - \{\sigma\}) \\
&= \frac{2}{\sqrt{3}}A(\Delta) + \frac{1}{2}P(\Delta) - m(\sigma) + \chi(\Delta) + 1 \\
&\leq \frac{2}{\sqrt{3}}A(\Delta) + \frac{1}{2}P(\Delta) + \chi(\Delta).
\end{aligned}$$

Case 2. $\varepsilon(\sigma, \Delta) = 1$. Let τ be the two-dimensional simplex in Δ having σ as a face. Let σ' and σ'' be the other two one-dimensional faces of τ . Without loss of generality, assume $m(\sigma') \geq m(\sigma'')$. From the choice of σ , it can be seen that $m(\sigma) \geq m(\sigma') \geq m(\sigma'') \geq 1$. By [Lemma 1](#),

$$\frac{4}{\sqrt{3}}m(\tau) + m(\sigma) \geq m(\sigma') + m(\sigma'').$$

Note that $\Gamma = \Delta - \{\tau, \sigma\}$ is a simplicial complex. By induction hypothesis,

$$\begin{aligned}
& \frac{2}{\sqrt{3}}A(\Delta) + \frac{1}{2}P(\Delta) + \chi(\Delta) \\
&= \frac{2}{\sqrt{3}}A(\Gamma) + \frac{1}{2}P(\Gamma) + \chi(\Gamma) + \frac{2}{\sqrt{3}}(A(\tau) + \frac{1}{2}(m(\sigma) - m(\sigma') - m(\sigma''))) \\
&\geq \frac{2}{\sqrt{3}}A(\Gamma) + \frac{1}{2}P(\Gamma) + \chi(\Gamma) \\
&\geq |\Gamma| = |\Delta|.
\end{aligned}$$

Case 3. $\varepsilon(\sigma, \Delta) = 2$. Let τ and τ' be the two-dimensional simplexes in Δ having σ as a face. Let B and D be the two vertices of σ . Suppose C is the third vertex of τ and E is the third vertex of τ' . By (b) of the definition of simplicial complex, it can be seen that $BCDE$ is a convex quadrilateral. It follows that replacing τ and τ' by $\triangle BCE$ and $\triangle DEC$, a simplicial complex can be obtained from Δ which has the same union and the same number of one-dimensional simplexes as Δ has. By the choice of Δ , one has $m(\sigma) = |BD| \leq |CE|$. Since σ has the longest length, the condition of Lemma 2 is satisfied, and thus

$$\frac{4}{\sqrt{3}}(A(\tau) + A(\tau')) - P(\tau \cup \tau') + 2 \geq 0.$$

Note that $\Gamma = \Delta - \{\sigma, \tau, \tau'\}$ is a simplicial complex. By induction hypothesis,

$$\begin{aligned}
& \frac{2}{\sqrt{3}}A(\Delta) + \frac{1}{2}P(\Delta) + \chi(\Delta) \\
&= \frac{2}{\sqrt{3}}A(\Gamma) + \frac{1}{2}P(\Gamma) + \chi(\Gamma) + \frac{2}{\sqrt{3}}(A(\tau) + A(\tau')) - \frac{1}{2}P(\tau \cup \tau') + 1 \\
&\geq \frac{2}{\sqrt{3}}A(\Gamma) + \frac{1}{2}P(\Gamma) + \chi(\Gamma) \\
&\geq |\Gamma| = |\Delta|.
\end{aligned}$$

Summing the above three cases, Theorem 1 is proved. \square

Zassenhaus-Groemer-Oler inequality can be derived from Folkman-Graham inequality.

Corollary 1 (Zassenhaus-Groemer-Oler) *Let X be a compact convex region on the Euclidean plane. Suppose X contains a set V of centers of n nonoverlapping unit circles. Then*

$$n \leq \frac{2}{\sqrt{3}}A(X) + \frac{1}{2}P(X) + 1$$

where $A(X)$ and $P(X)$ denote the area and the perimeter of X , respectively.

Proof Let H be the convex hull of V . By the convexity of X , one has $A(X) \geq A(H)$ and $P(X) \geq P(H)$. Let Δ be a simplicial complex with vertex set V , whose two-dimensional faces form a triangulation of H . Then the union of Δ equals H and $\chi(\Delta) = 1$. By Folkman-Graham inequality,

$$\frac{2}{\sqrt{3}}A(X) + \frac{1}{2}P(X) + 1 \geq \frac{2}{\sqrt{3}}A(\Delta) + \frac{1}{2}P(\Delta) + 1 \geq |\Delta| = |V|.$$

Zassenhaus-Groemer-Oler Inequality follows. \square

The next corollary follows immediately from Zassenhaus-Groemer-Oler inequality.

Corollary 2 *If a circle with radius r contains n nonoverlapping unit circles, then*

$$n \leq \frac{2}{\sqrt{3}}\pi(r - 0.5)^2 + \pi(r - 0.5) + 1.$$

3 Packing Independent Points

In this section, $\|uv\|$ denotes the Euclidean distance between two points u and v .

A set of points in the Euclidean plane is *independent* if the distance between every pair of points in the set is (strictly) larger than one. For packing independent points, one has similar inequalities with the sign “ \leq ” replaced by “ $<$ ”.

Theorem 2 *Let Δ be a simplicial complex in the Euclidean plane with at least one one-dimensional face. Suppose for any two distinct vertices x and y in Δ , $\|xy\| > 1$. Then*

$$|\Delta| < \frac{2}{\sqrt{3}}A(\Delta) + \frac{1}{2}P(\Delta) + \chi(\Delta)$$

where $|\Delta|$ is the number of vertices in Δ .

Proof Notice that under the condition that every pair of vertices has distance strictly larger than one, the inequality sign “ \geq ” can be replaced by sign “ $>$ ” in Lemmas 1 and 2. With these modified lemmas, the proof of Folkman-Graham inequality can be modified to a proof of Theorem 2. \square

Corollary 3 *Let X be a compact convex region on the Euclidean plane. Suppose X contains n independent points. Then*

$$n < \frac{2}{\sqrt{3}}A(X) + \frac{1}{2}P(X) + 1$$

where $A(X)$ and $P(X)$ denote the area and the perimeter of X , respectively.

Corollary 4 If a circle with radius r contains n independent points, then

$$n < \frac{2}{\sqrt{3}}\pi r^2 + \pi r + 1.$$

As one can see from the following sections, Corollary 4 has a lot of applications. But in some other applications, Corollary 4 is far from tight. For example, in the study of wireless sensor networks, the following problem is often considered: Let T be a tree lying in the Euclidean plane with n vertices, satisfying the condition that each edge has length at most one. Let $Q(T)$ be the union of the n disks with radius one and centers at the vertices of T . How many independent points could $Q(T)$ contain? Let $\text{mip}(n)$ denote the maximum number of independent points lying in $Q(T)$ for T over all the trees with n vertices whose maximum edge lengths are at most one. For $n = 1$, Corollary 4 results in $\text{mip}(1) \leq \lfloor (2/\sqrt{3} + 1)\pi + 1 \rfloor = 7$. However, the exact value of $\text{mip}(1)$ is five, determined by Wan et al. in [53].

Lemma 3 (Wan et al. [53]) $\text{mip}(1) = 5$.

Proof Let D_v denote the disk with center v and radius one. Let a_1, a_2, \dots, a_k be the independent points lying in D_v such that va_1, va_2, \dots, va_k are in counterclockwise ordering. Since $\|va_i\| \leq 1$ for $i = 1, 2, \dots, k$ and $\|a_i a_j\| > 1$ for $i \neq j$, one has $\angle a_1 v a_2 > \pi/3, \angle a_2 v a_3 > \pi/3, \dots, \angle a_k v a_1 > \pi/3$. Thus, $2\pi = \angle a_1 v a_2 + \angle a_2 v a_3 + \dots + \angle a_k v a_1 > k\pi/3$. Hence, $k < 6$. This means $\text{mip}(1) \leq 5$.

On the other hand, it is easy to arrange five independent points on the boundary of D_v , which means $\text{mip}(1) \geq 5$. \square

Since the upper bound of $\text{mip}(n)$ plays an important role in analyzing approximation algorithms for the virtual backbone of wireless networks, a lot of works focus on the improvement of $\text{mip}(n)$.

Unlike wired network, there is no fixed infrastructure in a wireless network. In such a circumstance, if all the sensors participate in the transmission, it is not only a waste of energy, but also can cause too much interference such that nodes cannot decode the transmitted information correctly (which is known as *broadcast storm*). To solve this problem, virtual backbone was proposed [3, 8, 49], which corresponds to a connected dominating set in a graph. A *dominating set* (DS) of a graph $G = (V, E)$ is a vertex set D such that every vertex in $V \setminus D$ has at least one neighbor in D . It is a *connected dominating set* (CDS) if $G[D]$, the subgraph of G induced by D , is connected. Using a CDS as the virtual backbone, computations and transmissions are mainly carried out in the CDS, and thus energy is saved; interference is alleviated. Clearly, one expects the size of the CDS to be as small as possible, resulting in the *minimum connected dominating set* problem (MCDS).

A widely used model of homogeneous wireless sensor network is the *unit disk graph* (UDG), for which every vertex corresponds to a sensor on the plane; two vertices are adjacent in the graph if and only if the Euclidean distance between their corresponding sensors is at most a constant which is scaled to one.

Consider the MCDS problem in a unit disk graph (UDG) G . A lot of approximation algorithms follow a two-phase strategy [3, 5, 33, 50, 53]. The first phase constructs a *maximal independent set* (MIS) I (i.e., a set of independent points such that adding any point into it breaks the independency, notice that any MIS is also a DS). The second phase selects a set C of *connectors* such that $D = I \cup C$ is a CDS. The differences of these works lie in how I and C are selected.

To analyze the approximation ratio, consider an optimal solution D^* . If $|I| \leq \beta_1|D^*| + \beta_2$ and $|C| \leq \gamma_1|D^*| + \gamma_2$, then $|D| \leq (\beta_1 + \gamma_1)|D^*| + (\beta_2 + \gamma_2)$, and thus D is a $(\beta_1 + \gamma_1)$ -approximation solution.

Implementation of Lemma 3 for mip(1) already shows that $|I| \leq 5|D^*|$. A lot of improvements on the upper bound were made continually by Funke et al. [16], Gao et al. [17], Wan et al. [53, 54], Wu et al. [58], and Li et al. [33], which will be introduced in the following.

For the ease of statement, some notations should be fixed first. For a positive real number r , use D_w^r to denote the disk centered at node w with radius r . For a node set W , denote by $D_W^r = \bigcup_{w \in W} D_w^r$. Call them as the *r -neighbor area* of w and W , respectively. For an MIS I , denote by $I_w^r = I \cap D_w^r$ and $I_W^r = I \cap D_W^r$ the set of MIS nodes in the r -neighbor area of w and W , respectively. When $r = 1$, the superscript r is omitted and the notations D_W , I_W , and terminology *neighbor area* are used. Thus, $Q(T)$ in the above is exactly $D_{V(T)}$, and $\text{mip}(n)$ is exactly $\max\{|I_{V(T)}| : I \text{ is an MIS and } T \text{ is a tree on } n \text{ nodes in a UDG}\}$. What one is interested in is the spanning tree T of $G[D^*]$ where D^* is an optimal CDS.

Theorem 3 $|I| \leq 4|D^*| + 1$.

Proof The proof makes use of the following observation: Suppose W is a connected node set and w is another node which is connected to W , then

$$|I_{W \cup \{w\}} \setminus I_W| \leq 4 \quad (1)$$

(refer to Fig. 2 for an illustration), that is, connecting a node to an already connected node set increases the number of MIS nodes in the neighbor area by at most four. For an optimal solution D^* , order the nodes as $v_1, v_2, \dots, v_{|D^*|}$ such that for any $i = 1, 2, \dots, |D^*|$, the subgraph of $G[D^*]$ induced by $W_i = \{v_1, v_2, \dots, v_i\}$ is connected. Then

$$|I| = |I_{v_1}| + \sum_{i=2}^{|D^*|} |I_{v_i} \setminus I_{W_i}| \leq 5 + \sum_{i=2}^{|D^*|} 4 = 5 + 4(|D^*| - 1) = 4|D^*| + 1.$$

The upper bound is proved. \square

For $n = 2$, the exact value of $\text{mip}(2)$ has been determined in [58].

Lemma 4 (Wu et al. [58]) $\text{mip}(2) = 8$.

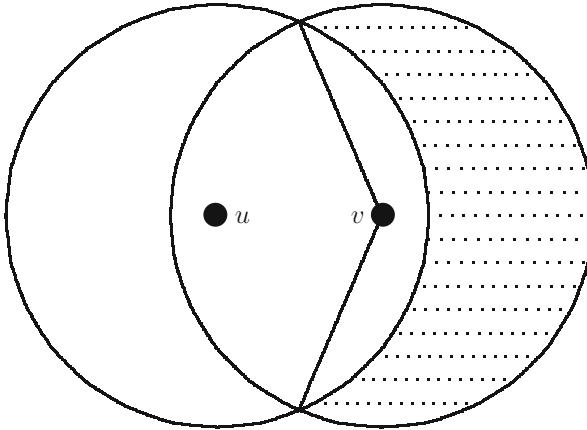


Fig. 2 Adding node v increases the region D_u by the *shaded area* which contains at most four MIS nodes

Proof Let u and v be two points with distance at most one.

First, it was shown that $\text{mip}(2) \leq 8$. Suppose that there exists an independent set I of more than eight points lying in $D_u \cup D_v$. A contradiction will be found through a proof consisting of two parts, each part showing a fact.

Fact 1. The intersection area $A = D_u \cap D_v$ contains exactly one vertex in I .

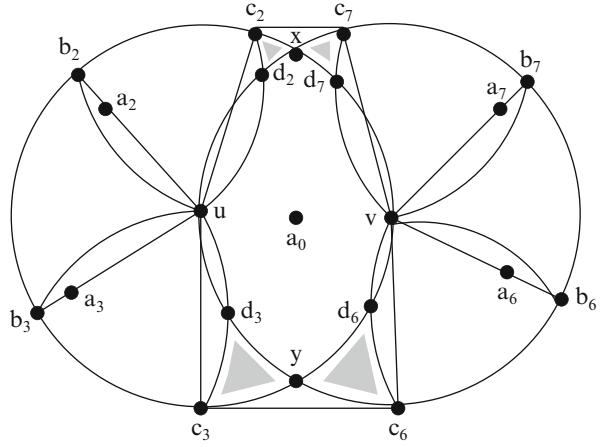
Indeed, suppose A contains k vertices in I . By Lemma 3, $D_u - A$ contains at most $5 - k$ vertices in I and $D_v - A$ contains at most $5 - k$ vertices in I . Thus, $D_u \cup D_v$ contains at most $10 - k$ vertices in I . Hence, by the assumption $|I| \geq 9$, one has $k \leq 1$. To show $k \geq 1$, let x and y be the two intersection points of the boundaries of D_u and D_v . Since $d(u, v) \leq 1$, one has $\angle xvy = \angle yux \geq 2\pi/3$. Thus, $D_u - A$ contains at most four vertices in I , so does $D_v - A$. This means that I contains at most $8 + k$ vertices. Since $|I| \geq 9$, one has $k \geq 1$.

Let a_0 denote the unique point of I lying in $D_u \cap D_v$. As a consequence, each of $D_u - A$ and $D_v - A$ contains exactly four vertices in I and $|I| = 9$. Assume $I = \{a_0, a_1, \dots, a_8\}$, where a_0, a_1, \dots, a_4 lie counterclockwise in D_u and a_0, a_5, \dots, a_8 lie counterclockwise in D_v . Denote by ub_i the radius through a_i for $i = 2, \dots, 4$ and by vb_i the radius through a_i for $i = 5, \dots, 8$. Using radius one, draw four arc-triangles ub_2c_2 , ub_3c_3 , vb_6c_6 , and vb_7c_7 as shown in Fig. 3. Their boundaries intersect the boundary of $D_u \cap D_v$ at d_2, d_3, d_6, d_7 , respectively. Note that none of a_1, a_4, a_5, a_8 can lie in the four arc-triangles ub_2c_2 , ub_3c_3 , vb_6c_6 , and vb_7c_7 and $D_u \cap D_v$. Therefore, a_1, a_4, a_5, a_8 must lie in the four small dark areas xc_2d_2 , yc_3d_3 , yc_6d_6 , and xc_7d_7 , respectively, as shown in Fig. 3.

Fact 2. There exist two small dark areas which are too close to contain two independent vertices. (Therefore, this fact results in a contradiction.)

To show this fact, notice that $\angle b_2ub_3 > \pi/3$ and $\angle c_2ub_2 = \angle b_3uc_3 = \pi/3$. Hence, $\angle c_2uc_3 > \pi$ and $\angle c_3uc_2 < \pi$ (note that $\angle c_3uc_2$ is the one obtained by sweeping c_3u counterclockwise to c_2u). Similarly, $\angle c_7vc_6 < \pi$. Therefore,

Fig. 3 An illustration of the four small dark areas



$\angle uc_2c_7 + \angle c_2c_7v + \angle vc_6c_3 + \angle c_6c_3u > 2\pi$. This means that either $\angle uc_2c_7 + \angle c_2c_7v > \pi$ or $\angle vc_6c_3 + \angle c_6c_3u > \pi$. Without loss of generality, assume the former occurs. Then, it can be shown that the dark area $xc_2d_2 \cup xc_7d_7$ cannot contain two independent points.

To do so, enlarge the area xc_7d_7 by turning the arc-triangle vb_7c_7 around v until vc_7 is parallel to uc_2 (Fig. 4). At this position, quadrilateral c_2uvc_7 becomes a parallelogram so that $\|c_2c_7\| = \|uv\| \leq 1$. It follows that the distance between two points in areas xc_2d_2 and xc_7d_7 cannot exceed $\max(\|c_2c_7\|, \|c_2d_7\|, \|d_2c_7\|, \|d_2d_7\|)$. Moreover, it can be proved that $\|d_2d_7\| \leq \max(\|c_2d_7\|, \|d_2c_7\|)$. In fact, by noticing that $\angle c_7d_7d_2 + \angle d_7d_2c_2 > \pi$, one has either $\angle c_7d_7d_2 > \pi/2$ or $\angle d_7d_2c_2 > \pi/2$. Therefore, either $\|d_2c_7\| > \|d_2d_7\|$ or $\|c_2d_7\| > \|d_2d_7\|$.

Now, to complete the proof of Fact 2, it suffices to prove that $\|c_2d_7\| \leq 1$ and $\|d_2c_7\| \leq 1$. To see $\|c_2d_7\| \leq 1$, make $\|c_2d_7\|$ longer by moving v away from u until $\|uv\| = 1$ (Fig. 5). At this limit position, $\|uv\| = \|vb_7\| = \|b_7d_7\| = \|d_7u\| = 1$. Therefore, uvb_7d_7 is a parallelogram. It follows that $\|d_7b_7\| = \|c_2c_7\| = 1$ and d_7b_7 is parallel to uv and hence parallel to c_2c_7 . Thus, $c_2d_7b_7c_7$ is a parallelogram. Therefore, $\|c_2d_7\| = \|c_7b_7\| = 1$. Similarly, one can show $\|d_2c_7\| \leq 1$.

Fact 2 induces a contradiction, which implies that $\text{mip}(2) \leq 8$.

On the other hand, eight independent points can be easily placed in $D_u \cup D_v$ when $\|uv\| = 1$. This means that $\text{mip}(2) \geq 8$. \square

Making use of Lemma 4, Wu et al. [58] improved the upper bound of $|I|$ as follows:

Theorem 4 $|I| \leq 3.8|D^*| + 1.2$.

Proof In [58], the authors first showed that any unit disk graph has a minimum spanning tree in which every vertex has degree at most five. Let T be such a spanning tree of $G[D^*]$. The upper bound is established by induction on $|T|$, the number

Fig. 4 Turn unit arc-triangle vb_7c_7 until vc_7 parallels uc_2

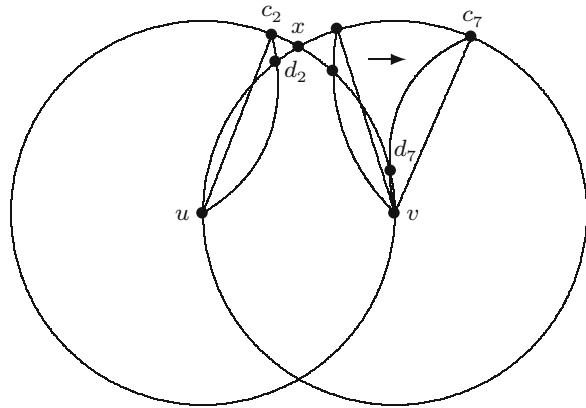
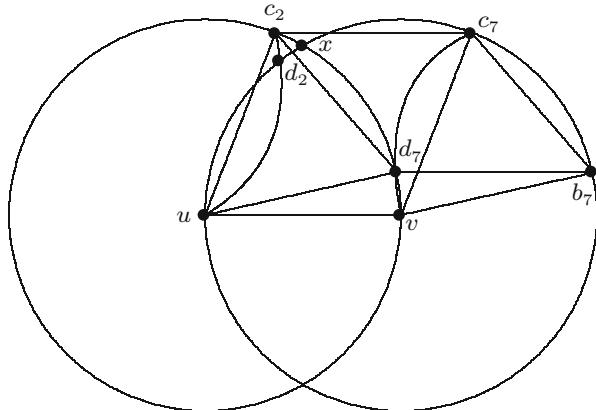


Fig. 5 Move u until $\|uv\| = 1$



of nodes in T . The validity for $|T| = 1$ or 2 follows from [Lemmas 3](#) and [4](#). For $|T| \geq 3$, it is easy to see that T has a non-leaf node v which is adjacent with at most one non-leaf node u (if all the neighbors of v in T are leaves, let u be an arbitrary neighbor of v). Let the remaining neighbors of v be x_1, x_2, \dots, x_k . Then $k \leq 4$. By the induction hypothesis, the neighbor area of $T' = T - \{v, x_1, \dots, x_k\}$ contains at most $3.8(|T| - k - 1) + 1.2$ MIS nodes. Denote $T_0 = T' \cup \{vx_k\}$ and $T_i = T_{i-1} \cup \{vx_i\}$ for $i = 1, 2, \dots, k - 1$. By [Lemma 4](#), it can be seen that $|I_{\{v, x_k\}} \setminus I_u| \leq 7$. By [Lemma 3](#), $|I_{x_i} \setminus I_v| \leq 4$. Thus $|I_{T_0} \setminus I_{T'}| \leq 7$ and $|I_{T_i} \setminus I_{T_{i-1}}| \leq 4$ ($i = 1, 2, \dots, k - 1$). Then, $|I| = |I_{T'}| + |I_{T_0} \setminus I_{T'}| + \sum_{i=1}^{k-1} |I_{T_i} \setminus I_{T_{i-1}}|$ is upper bounded by

$$\begin{aligned} |I| &\leq 3.8(|T| - k - 1) + 1.2 + 7 + 4(k - 1) = 3.8|D^*| + 1.2 + 0.2(k - 4) \\ &\leq 3.8|D^*| + 1.2. \end{aligned}$$

□

From the above analysis, one obtains the feeling that if some nodes in D^* are more compact, then the number of MIS nodes in their neighbor area will be smaller. This intuition is further explored in [54].

Theorem 5 (Wan et al. [54]) $\text{mip}(n) \leq \frac{11}{3}n + \frac{4}{3}$.

Proof A set W of nodes is called a *star* if there exists a node $w \in W$ such that $W \subseteq D_w$. A star containing k nodes is called a k -star. Thus a star is a compact set of nodes, and the larger k is, the more compact the nodes are. It was proved in [54] that any k -star W satisfies

$$|I_W| \leq \frac{11k}{3} + 1. \quad (2)$$

This inequality is obtained by showing that

$$|I_W| \leq \phi_k = \begin{cases} 3k + 2, & \text{for } k \leq 2, \\ \min\{3k + 3, 21\}, & \text{for } k \geq 3, \end{cases}$$

and $\phi_k \leq \frac{11k}{3} + 1$. The intuition for the expression of ϕ_k is that adding a new node into a star increases the number of MIS nodes in the neighbor area by at most 3, and when k is large enough (in fact, when $k > 6$), adding new nodes into the star will no longer increase the number of MIS nodes in the neighbor area.

A *nontrivial star* is a k -star with $k \geq 2$. Decompose D^* into nontrivial stars S_1, S_2, \dots, S_p such that for any $i = 1, 2, \dots, p$, the subgraph of $G[D^*]$ induced by $\bigcup_{j=1}^i S_j$ is connected (this can be done as proved in [54]). By (2),

$$|I_{S_1}| \leq \frac{11|S_1|}{3} + 1.$$

For $i = 2, 3, \dots, p$, let u_i be a node in $\bigcup_{j=1}^{i-1} S_j$ which is adjacent with S_i . Then $(I_{S_i} \setminus \bigcup_{j=1}^{i-1} S_j) \cup \{u_i\}$ is independent. Again by (2),

$$|I_{S_i} \setminus \bigcup_{j=1}^{i-1} I_{S_j}| \leq \frac{11}{3}|S_i|.$$

It follows that

$$|I| = |I_{S_1}| + \sum_{i=2}^p |I_{S_i} \setminus \bigcup_{j=1}^{i-1} I_{S_j}| \leq \frac{11}{3}|D^*| + 1. \quad \square$$

Funke et al. [16] presented a new approach and outlined a proof for

$$\text{mip}(n) \leq 3.453n + 8.291.$$

However, Wan et al. [54] thought that the proof for a key geometric extreme property underlying this claim is missing, and thus the proof is incomplete. Gao et al. [17] proved the key geometric extreme property in detail and found that the proof of Funke et al. actually leads to a better bound.

Lemma 5 (Gao et al. [17]) $\text{mip}(n) \leq 3.453n + 4.839$.

To understand the idea in deducing this bound, it is useful to first look at the deduction of a looser bound:

$$|I| \leq 3.748|D^*| + 5.252. \quad (3)$$

To prove (3), notice that the disks $\{D_u^{0.5} \mid u \in I\}$ are mutually disjoint and they are all contained in $D_{D^*}^{1.5}$. Hence

$$|I| \leq \frac{\text{Area}(D_{D^*}^{1.5})}{\pi \cdot 0.5^2}. \quad (4)$$

To obtain an upper bound for $\text{Area}(D_{D^*}^{1.5})$, notice that adding a disk $D_v^{1.5}$ to a disk $D_u^{1.5}$ increases the 1.5-neighbor area by at most $\pi \cdot 1.5^2 - 2 \left(1.5^2 \arccos \frac{1}{3} - \frac{\sqrt{2}}{2} \right) \approx 2.9435$ (this can be illustrated also by Fig. 2; in this setting, the disks have radius 1.5, and the upper bound is achieved when the distance between nodes u and v is exactly one). Then by an analysis similar to Theorem 3, it can be seen that

$$\text{Area}(D_{D^*}^{1.5}) \leq \pi \cdot 1.5^2 + 2.9435(|D^*| - 1) \approx 2.9435|D^*| + 4.1251. \quad (5)$$

Then inequality (3) follows from (4).

The above deduction is not tight, because the set of disks $\{D_u^{0.5} \mid u \in I\}$ cannot overspread the whole area of $D_{D^*}^{1.5}$; there are a lot of “wasted” fragments the areas of which are counted in $\text{Area}(D_{D^*}^{1.5})$ but have no contribution to $\text{Area}(\bigcup_{u \in I} D_u^{0.5})$.

To overcome this shortcoming, Funke et al. [16] proposed using Voronoi cells instead of disks of radius 0.5 to aid the analysis. For a node $u \in I$, the *Voronoi cell* of u , denoted by $V(u)$, is the set of points on the plane which are closer to u than to the other nodes of I . Call $TV(u) = V(u) \cap D_{D^*}^{1.5}$ as the *truncated Voronoi cell* of u . Clearly, $D_{D^*}^{1.5} = \bigcup_{u \in I} TV(u)$. The advantage is that there is no waste in using truncated Voronoi cell. The key step in the paper [17] is to establish a lower bound:

$$\text{Area}(TV(u)) \geq 0.8525 (\forall u \in I). \quad (6)$$

Replacing the denominator of (4) by 0.8525 and using (5), the upper bound in Lemma 5 is obtained.

In the case that the region $D_{D^*}^{1.5}$ has no holes, Gao et al. [17] proved an upper bound:

$$|I| \leq 3.399|D^*| + 4.874.$$

It was obtained by combining the lower bounds for the truncated Voronoi cells with the Euler's formula. In the case when $D_{D^*}^{1.5}$ has holes, they proved

$$|I| \leq 3.473|D^*| + 4.874.$$

Li et al. [33] improved the approach of Funke et al. [16] and proved the following:

Lemma 6 (Li et al. [33]) $\text{mip}(n) \leq 3.4306n + 4.81854$.

Proof This bound is obtained by further exploring the lower bounds for the areas of the truncated Voronoi cells. The authors partition I into two subsets: $I_1 = \{u \in I \mid D_u^{1/\sqrt{3}} \subseteq D_{D^*}^{1.5}\}$ and $I_2 = I \setminus I_1$. It was proved that

$$\text{Area}(TV(u)) \geq \begin{cases} \sqrt{3}/2, & \text{for } u \in I_1, \\ \sigma, & \text{for } u \in I_2, \end{cases}$$

where $\sigma \approx 0.855$. Hence

$$\text{Area}(D_{D^*}^{1.5}) \geq \frac{\sqrt{3}}{2}|I_1| + \sigma|I_2| = \frac{\sqrt{3}}{2}|I| - \left(\frac{\sqrt{3}}{2} - \sigma\right)|I_2|. \quad (7)$$

Then the authors proved that

$$|I_2| \leq \frac{\text{Peri}\left(D_{D^*}^{1.5-1/\sqrt{3}}\right)}{2(1 - 1/\sqrt{3})}, \quad (8)$$

where Peri represents the perimeter of the region. By induction on $|D^*|$, it can be proved that

$$\text{Peri}\left(D_{D^*}^{1.5-1/\sqrt{3}}\right) \leq 2\left(3 - \frac{2}{\sqrt{3}}\right)\left((|D^*| - 1) \arcsin \frac{1}{3 - 2/\sqrt{3}} + \frac{\pi}{2}\right). \quad (9)$$

Then, the desired upper bound follows from (5), (7), (8), and (9). □

Vahdatpour et al. [52] claimed that they proved

$$\text{mip}(n) \leq 3n + 3. \quad (10)$$

If their proof is correct, then this is best possible because Wan et al. [54] have constructed an example showing that

$$\text{mip}(n) \geq 3n + 3.$$

Unfortunately, an important part of the proof was missing from [52]. Therefore, their result cannot be accepted by many researchers in the literature. What is the important part which was missing from the proof? To explain, the terminology of semi-exclusive neighboring set is needed. Let T be an arbitrary spanning tree of $G[D^*]$ and assume $v_1, v_2, \dots, v_{|T|}$ is an arbitrary traversal of T and v_1 is not a leaf vertex. For any i , $2 \leq i \leq |T|$, the set $U_i = I_{v_i} \setminus I_{\{v_1, \dots, v_{i-1}\}}$ (the subset of MIS nodes which are adjacent with v_i but not adjacent to any of the previous nodes v_1, v_2, \dots, v_{i-1}) is called the *semi-exclusive neighboring set of node v_i* . By (1), one has $|U_i| \leq 4$ for $i = 2, \dots, |T|$.

Vahdatpour et al. tried to prove (10) by using induction on $|T|$. If there exists a leaf vertex v_j with $|U_j| \leq 3$, then the induction hypothesis can be applied on $T - v_j$ to yield $|I_{D^*}| \leq (3(|T|-1) + 3) + 3 = 3|T| + 3$. Hence one may assume that every leaf vertex v_j has $|U_j| = 4$. For each leaf vertex v_j , denote by $f(v_j)$ the first fork vertex (i.e., a vertex of degree at least three in T) that is met when one starts from v_j and goes along the path on T toward the root v_1 . If there exists a vertex v_i between v_j and $f(v_j)$ such that $|U_i| \leq 2$, denote the path on T between v_j and v_i as P . Notice that $T - V(P)$ is still a tree. Hence one can apply the induction hypothesis to $T - V(P)$ to obtain $|I_{D^*}| \leq (3(|T|-t) + 3) + 2 + 3(t-2) + 4 = 3|T| + 3$, where t is the number of nodes on P . What remains is the case that for every leaf vertex v_j , every vertex v_i between v_j and $f(v_j)$ has $|U_i| \geq 3$, and the focus is on the distribution of MIS nodes near a fork vertex. This is the most complicated case, but Vahdatpour et al. did not provide a sufficient argument to deal with it.

Due to the above reason, one can conclude that up to now, the best known upper bound for $\text{mip}(n)$ for general n is the one in Lemma 6 given by Li et al. [33].

In all the above works, geometry is explored deeper and deeper to obtain a tighter bound for the packing of disks with radius 0.5 in the neighbor area of a minimum CDS. The interested readers are referred to the mentioned literatures for the detailed implementation of geometry.

4 Sink Placement in Underwater Sensor Networks

Underwater Sensor Network has a lot of differences from the Terrestrial Sensor Network. It uses acoustic channels instead of radio-frequency signals, which cannot travel far in water. Thus a message which is forwarded through more intermediate nodes will experience a significantly longer data latency. To solve this problem, multiple UnderWater-Sink (UW-Sink) architecture was proposed [1]. Since UW-Sinks are very expensive, it is desirable to use as less UW-Sinks as possible while the number of relays for each sensor can be controlled within a given constant d . This consideration leads to the model of *minimum d -hop sink placement* (MdHSP) problem, the goal of which is to place a minimum number of sinks such that each sensor is at most d -hops away from at least one sink.

Even in UDG, MdHSP problem is NP-hard [29]. However, this problem admits a PTAS in UDG [57]. In the following, the ideas in [29] which gave a constant approximation algorithm for MdHSP are introduced.

A node set I_d of a graph $G = (V, E)$ is a *d-hop independent set* (*dIS*) if any pair of nodes in I_d have hop distance greater than d , it is a *maximal d-hop independent set* (*dMIS*) if any super-set of I_d is no longer a *d-IS*, and it is a *d-hop dominating set* (*dDS*) if every node $v \in V \setminus D$ is at most d -hops away from some node $u \in D$ (v is said to be *d-hop dominated by u*). Clearly, a *dMIS* is also a *dDS*.

Let G be the UDG for the sensors. The algorithm just finds a *dMIS* I_d in G and place a sink at the location of each node in I_d . Clearly, I_d is a solution to *MdHSP*. The authors in [29] proved that I_d approximates the optimal solution within a factor of $O(d)$.

First, they proved that

$$opt_{dDS} \leq 5opt_{MdHSP}, \quad (11)$$

where opt_{dDS} and opt_{MdHSP} are the sizes of a minimum *dDS* and an optimal *MdHSP*, respectively. To show (11), suppose OPT_{MdHSP} is an optimal solution to *MdHSP*. For each $c \in \text{OPT}_{MdHSP}$, find an MIS I_c in $N(c)$, where $N(c)$ is the set of sensors at Euclidean distance at most one from c . Let $I' = \bigcup_{c \in \text{OPT}_{MdHSP}} I_c$. For each node $v \in V$, it is at most d -hops away from some $c \in \text{OPT}_{MdHSP}$. Let (v, w_1, \dots, w_k, c) be a shortest path from v to c , where $k \leq d - 1$. Then w_k is either in I_c or adjacent with a node in I_c . In any case, v is at most d -hops away from some node in I_c . Hence I' is a *dDS* of G , and thus $opt_{dDS} \leq |I'|$. By Lemma 3, one has $|I'| \leq 5opt_{MdHSP}$.

In view of inequality (11), what remains to show is that

$$|I_d| \leq O(d)opt_{dDS} \quad (12)$$

holds for any *dMIS* I_d .

Since a *dMIS* must be an MIS, an upper bound of factor $O(d^2)$ follows easily from Corollary 4: Each node in I_d is *d-hop dominated* by some node in OPT_{dDS} ; nodes which are *d-hop dominated* by $u \in \text{OPT}_{dDS}$ are in the disk centered at u with radius d ; since the nodes in I_d have mutual distances greater than one, Corollary 4 gives

$$|I_d| \leq \left(\frac{2}{\sqrt{3}}\pi d^2 + \pi d + 1 \right) opt_{dDS} = O(d^2)opt_{dDS}.$$

To reduce the order of the factor, an intuitive idea might be that two nodes which are *d-hop independent* are far from each other. Unfortunately, this intuition is not true, as can be seen from Fig. 6. Thus a more elaborated analysis is in need.

For a node u , denote by $I_d(u)$ the set of nodes of I_d which are *d-hop dominated* by u . Divide $I_d(u)$ into two parts: $I_d^{(1)}(u) = \{v \in I_d(u) \mid v \text{ is } \lceil d/2 \rceil\text{-hop dominated by } u\}$ and $I_d^{(2)}(u) = I_d(u) \setminus I_d^{(1)}(u)$. It was proved in [29] that for any node u ,

$$|I_d^{(1)}(u)| \leq 5. \quad (13)$$

$$|I_d^{(2)}(u)| \leq O(d). \quad (14)$$

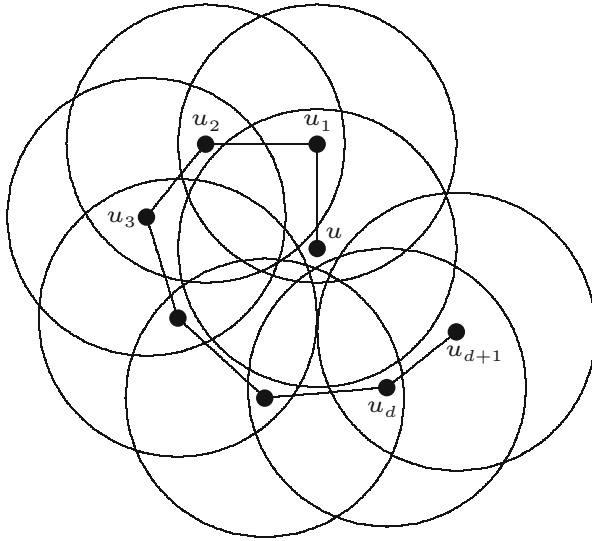


Fig. 6 The node u_{d+1} is d -hop independent with node u ; their Euclidean distance is only a little larger than one

To show (13), for each node $v \in I_d^{(1)}(u)$, let P_v be a shortest path from v to u , and let w_v be the last node on P_v . Then $W(u) = \{w_v \mid v \in I_d^{(1)}(u)\}$ is an independent set. In fact, if this is not true, suppose w_{v_1} is adjacent with w_{v_2} , then there is a path from v_1 to v_2 through edge $w_{v_1}w_{v_2}$ with hop distance at most $2(\lceil d/2 \rceil - 1) + 1 \leq d$, contradicting that v_1 and v_2 are d -hop independent. Thus it follows from Lemma 3 that $|I_d^{(1)}(u)| \leq |W(u)| \leq 5$.

To show (14), for each node $v \in I_d^{(2)}(u)$, let P_v be a shortest path from v to u , say, $P_v = w_0w_1w_2 \dots w_p u$, where $w_0 = v$. Notice that $p \geq \lceil d/2 \rceil$. For each $i = 0, 1, \dots, \lceil d/2 \rceil - 1$, draw a disk $D_{w_i}^{0.5}$ with center w_i and radius 0.5. Denote $A(v) = \bigcup_{i=1}^{\lceil d/2 \rceil - 1} D_{w_i}^{0.5}$. First, notice that $A(v_1) \cap A(v_2) = \emptyset$ for any pair of nodes $v_1, v_2 \in I_d^{(2)}(u)$; otherwise, similarly to the above paragraph, one might obtain a contradiction that there is a path from v_1 to v_2 with hop distance at most d . Second, observe that the disks $D_{w_0}^{0.5}, D_{w_2}^{0.5}, D_{w_4}^{0.5} \dots$ are mutually disjoint since P_v is a shortest path. Thus

$$\text{Area}(A(v)) \geq \left\lfloor \frac{1}{2} \left\lceil \frac{d}{2} \right\rceil \right\rfloor \cdot \pi \cdot (0.5)^2 \geq \frac{(d-3)\pi}{16}.$$

Since $\bigcup_{v \in I_d^{(2)}(u)} A(v)$ is contained in the disk centered at u with radius $d + 0.5$, one has

$$|I_d^{(2)}(u)| \leq \frac{\pi(d+0.5)^2}{(d-3)\pi/16} = O(d).$$

Then, (12) follows from (13) and (14).

Theorem 6 *The MdHSP problem in unit disk graph admits an approximation of factor $O(d)$.*

The above analysis also shows that the *minimum d-Hop dominating set* problem can be approximated within factor $O(d)$, which provides an approximation guarantee for the previous works such as [2], and can also be a step stone for some other more constraint problems such as *minimum connected d-hop dominating set* [43], *minimum d-hop connected d-hop dominating set* [30, 48], and *minimum connected k-fold d-hop dominating set* [32, 60].

5 Data Aggregation Scheduling

In data aggregation problem, a sink node s collects an information from the other nodes. During the aggregation, each node other than the sink combines all the information it receives with its own and sends the combined information only once. For example, to detect a burst of fire, the base station is interested in the highest temperature in a sensed field. Each node in the field detects its local temperature. When a node receives a data sent from another node, it compares the received temperature with the highest temperature stored in it, chooses the higher one, and puts it into the current data as well as the corresponding location. When it has aggregated all the data that are planned to be sent to it, it forwards the aggregated data. It can be seen that the topology of an aggregation can be modeled as an inward arborescence rooted at the sink node, call it as an *aggregation arborescence*.

Because of interference, the aggregation might be delayed. In the *protocol interference model*, node u can receive data from node v only when u is in the transmission range of v and no other nodes are in the interference range of u . Suppose each node has a fixed transmission radius which is scaled to one and an interference radius $\rho > 1$. Then two links u_1v_1 and u_2v_2 *interfere* with each other if either u_1v_2 or u_2v_1 has length at most ρ . A set of links are *interference-free* if they are mutually interference-free.

In an *aggregation schedule*, one has to find an aggregation arborescence and assign time-slots to its links such that (i) a node sends the combined data to its parent only after it has collected all the data from its children and (ii) all the links assigned to a common time-slot are interference-free. The *latency* of the aggregation schedule is the number of time-slots used. The goal of the *minimum-latency aggregation schedule problem* (MLAS) is to find an aggregation schedule with the minimum latency.

Even for the case that $\rho = 1$, the MLAS problem is NP-hard [6]. For a set of nodes V in the plane, let G be the unit disk graph on V . The maximum hop distance from the sink node s to the other nodes is the *radius of G with respect to s* , denoted by R . The approximation algorithms given in [6] and [27] achieve latencies at most $(\Delta - 1)R$ and $23R + \Delta - 18$, respectively, where Δ is the maximum degree of G . In [55], Wan et al. presented three approximation algorithms called SAS, PAS, and EPAS, which produce latencies at most $15R + \Delta - 4$, $2R + \Delta + O(\log R)$, and

$\left(1 + \Theta\left(\log R / \sqrt[3]{R}\right)\right)R + \Delta$. In Sects. 5.1 and 5.2, the main ideas of PAS and EPAS are introduced, respectively.

An aggregation schedule for interference radius $\rho = 1$ is said to be *well separated* if at each time-slot of the schedule, either all the transmitting nodes have mutual distances greater than one or all the receiving nodes have mutual distances greater than one. Using well-separated schedule, Wan et al. [55] introduced a generic method for the aggregation scheduling problem with $\rho > 1$. This method is presented in Sect. 5.3.

Parts of the above works were generalized to many-to-one data aggregation scheduling problem in multichannel multi-hop wireless networks [34].

5.1 Pipelined Aggregation Scheduling (PAS)

A basic step for PAS is the establishment a *single-hop aggregation schedule* from X to Y , where X, Y are two disjoint node sets. The schedule is described in Algorithm 1 called *iterative minimal covering* (IMC), in which data are sent through the links specified by S . In each iteration, a minimal cover C of the remaining node set X' is formed. By the minimality of C , every node $y \in C$ has a *private neighbor* x in X' , that is, y is the only neighbor of x in C . Then xy is the link through which x is set to report its data. The integer $\ell(xy)$ indicates the time-slot assigned to the link xy . It can be seen that

$$\ell(xy) \text{ is no more than the number of neighbors of } y \text{ in } X. \quad (15)$$

By choosing x to be a private neighbor of y , the links with the same label are interference-free.

Algorithm PAS first constructs a connected dominating set (CDS) D of the unit disk graph G . For each node u in $V \setminus D$, assign an arbitrary dominator v as its parent, indicating that u reports data through link uv . In the first phase of the aggregation, data are sent from $V \setminus D$ to D using the single-hop aggregation schedule IMC. By the property in (15), this phase takes at most $\Delta - 1$ time-slots. In the second phase of the aggregation, transmissions are scheduled within the CDS D . In the following, the focus is put on the construction of D and the scheduling within D .

Though the problem of computing a minimum CDS has constant approximations, the CDS D in [55] is constructed in a specific way so that it has stronger properties to be used in analyzing the latency. It starts from constructing a breadth-first-search tree T of G and finds a maximal independent set (MIS) U induced by T in a first-fit manner: order the nodes of T layer by layer as v_1, v_2, \dots, v_n , where $v_1 = s$; set $U = \{v_1\}$ initially; for $i = 2$ to n , add v_i to U if v_i is not adjacent with any node in U . Notice that any MIS of G is also a dominating set of G . After that, more nodes W called *connectors* are added in the following way such that $D = U \cap W$ is a CDS of G : Let H be the graph with node set U ; two nodes in U are adjacent in H if they have a common neighbor in G . Suppose the radius of H

Algorithm 1 IMC

```

1:  $S = \emptyset$ ,  $\ell = 0$ ,  $X' = X$ ,  $Y' = Y$ .
2: while  $X' \neq \emptyset$  do
3:    $\ell = \ell + 1$ .
4:   Let  $C \subseteq Y'$  be a minimal cover of  $X'$ .
5:   for each  $y \in C$  do
6:     Let  $x \in X'$  be a private neighbor of  $y$ .
7:      $S = S \cup \{xy\}$ .
8:      $\ell(xy) = \ell$ .
9:      $X' = X' \setminus \{x\}$ .
10:     $Y' = C$ .
11:   end for
12: end while
13: Output  $S$  and  $\ell$ .

```

with respect to s is R' . It can be seen that $R' \leq R - 1$. For $i = 0, 1, \dots, R'$, let U_i be the set of nodes at distance i from s in H , and let P_i be the set of nodes which are adjacent with some node in U_i and some node in U_{i+1} . Compute a minimal cover $W_i \subseteq P_i$ for U_{i+1} . The connector set $W = \bigcup_{i=1}^{R'-1} W_i$.

Since all the nodes in U are contained in the disk centered at s with radius R , it follows from Corollary 4 that

$$|U| \leq \frac{2\pi}{\sqrt{3}}R^2 + \pi R + 1. \quad (16)$$

Furthermore, since W_i is a *minimal* cover of U_{i+1} , one has $|W_i| \leq |U_{i+1}|$, and thus $|W| \leq |U| - 1$. Hence

$$|U \cup W| \leq \frac{4\pi}{\sqrt{3}}R^2 + 2\pi R + 1. \quad (17)$$

Let $D = U \cap W$. Denote the subgraph of G induced by D as $G[D]$. Clearly, $G[D]$ is connected. Let R_D be the radius of $G[D]$ with respect to s . For $i = 0, 1, \dots, R_D$, let V_i be the set of nodes at distance i from s in $G[D]$. The aggregation arborescence T_D within D is constructed together with a link labeling and a node ranking rank as in algorithm CBFS.

For $0 \leq i \leq R_D$ and $0 \leq j \leq \text{rank}(s)$, let $t_{ij} = (R_D - i) + 44j$. Each link uv of T_D is scheduled to the time-slot $t_{ij} + 4(\ell - 1)$, where i is the depth of u in T_D (i.e., $u \in V_i$), $j = \text{rank}(u)$, and ℓ is the label of the link uv .

It was proved in [55] that the above method gives an aggregation schedule with latency at most $t_{0,\text{rank}(s)}$. Notice that the node ranking produced by CBFS has the property that $\text{rank}(s) \leq \log |D|$. Combining this with (17), $\text{rank}(s) \leq O(\log R)$. Hence, by observing $R_D \leq 2(R - 1)$, one has

$$r_{0,\text{rank}(s)} = R_D + 44 \cdot \text{rank}(s) \leq 2R + O(\log R).$$

Algorithm 2 CBFS

```

1:  $T_D = (D, \emptyset)$ .
2: For each  $u \in V_{R_D}$ ,  $\text{rank}(u) = 0$ .
3: for  $i = R_D$  down to 1 do
4:   Let  $J = \{\text{rank}(v) : v \in V_i\}$ .
5:   for each  $j \in J$  do
6:     Let  $V_{ij} = \{v \in V_i : \text{rank}(v) = j\}$ .
7:     Augment  $T_D$  by applying Algorithm IMC to the sets  $V_{ij}$  and  $V_{i-1}$ .
8:   end for
9:   for each  $u \in V_{i-1}$  do
10:    if  $u$  has no child then  $\text{rank}(u) = 0$ 
11:    else Let  $r$  be the maximum rank of the children of  $u$ .
12:      if  $u$  has only one child with rank  $r$  then  $\text{rank}(u) = r$ 
13:      else  $\text{rank}(u) = r + 1$ .
14:      end if
15:    end if
16:   end for
17: end for
18: Output  $T$  and rank.

```

Taking the first phase of aggregation into account, the latency of the scheduling produced by the algorithm PAS is at most $2R + O(\log R) + \Delta$.

5.2 Enhanced Pipelined Aggregation Scheduling (EPAS)

EPAS uses the same outline as PAS but different set of connectors and different function of time-slot assignment.

The set W of connectors is constructed based on the same MIS U as in PAS, which is induced by the breadth-first-search tree T . For each node v , denote by $p^i(v)$ the i th ancestor of v , that is, $p^i(v)$ is i hops away from v on the path of T from v to s . Fix a positive integer k . Set $W' = \emptyset$ initially. For each node $u \in U \setminus \{s\}$, suppose i is the smallest positive integer such that $p^i(u)$ is in U , add $\{p^j(u) : 1 \leq j \leq \min\{i-1, k\}\}$ into W' . Notice that $\min\{i-1, k\} \geq 1$, and thus by the construction of U , the subgraph $G[U \cup W']$ is connected. In $G[U \cup W']$, construct a shortest path tree T' from s to all the nodes in $U \setminus \{s\}$. Let W be the subset of W' which is contained in $V(T')$.

Since for each node $u \in U \setminus \{s\}$, at most k nodes are added into W' , one has

$$|W| \leq |W'| \leq k(|U| - 1).$$

Combining this with (16),

$$|D| = |U \cup W| \leq (k+1) \left(\frac{2\pi}{\sqrt{3}} R^2 + \pi R \right) + 1. \quad (18)$$

Furthermore, it was proved in [55] that

$$R_D \leq \left(1 + \frac{1}{k}\right) R, \quad (19)$$

where R_D is the radius of the subgraph $G[D]$ with respect to s .

The EPAS uses parameter $k = \Theta(\sqrt[3]{R} / \log R)$ to find a connector set W . Let T_D be the aggregation arborescence of $G[U \cup W]$ constructed by CBFS. Suppose ℓ_0 is the maximum label on the links of T_D . For $0 \leq i \leq R_D$ and $0 \leq j \leq \text{rank}(s)$, let $t_{ij} = (R_D - i) + 3\ell_0 j$. Each link uv of T_D is scheduled to the time-slot $t_{ij} + 3(\ell - 1)$, where i is the depth of u in T_D , $j = \text{rank}(u)$, and ℓ is the label of the link uv .

Similar to PAS, the number of time-slots used by EPAS is at most $t_{0,\text{rank}(s)}$. By (18), (19), and the observation $\text{rank}(s) \leq O(\log R)$,

$$t_{0,\text{rank}(s)} = R_D + 3\ell_0 \text{rank}(s) \leq \left(1 + \Theta\left(\frac{\log R}{\sqrt[3]{R}}\right)\right) R.$$

Taking the first phase of aggregation into account, EPAS produces a schedule with latency at most $\left(1 + \Theta\left(\log R / \sqrt[3]{R}\right)\right) R + \Delta$.

5.3 A Generic Method for $\rho > 1$

Let $\mathcal{S} = \{\mathcal{S}_i : 1 \leq i \leq \ell\}$ be the output of a well-separated schedule using interference radius $\rho = 1$, where ℓ is the latency of the schedule, and \mathcal{S}_i is the set of links scheduled at time-slot i . For each $1 \leq i \leq \ell$, links in \mathcal{S}_i might have interference with respect to $\rho > 1$ and thus need to be further scheduled.

For a link set L , the *conflict graph* of L is the undirected graph $CG(L)$ with vertex set L ; two vertices are adjacent in $CG(L)$ if the corresponding two links interfere with each other. It can be seen that scheduling \mathcal{S}_i into interference-free subsets is equivalent to properly coloring the vertices of $CG(\mathcal{S}_i)$, with each color class corresponding to a set of interference-free links.

Suppose H is a graph to be colored, and $V(H)$ is ordered as v_1, v_2, \dots, v_n . The *first-fit coloring* assigns colors to the vertices sequentially. When vertex v_i is to be colored, it is assigned the first available color, namely, the color with the smallest index which has not been used by any of its previous neighbors. Such a coloring strategy uses at most $\delta^* + 1$ colors, where $\delta^* = \max_{1 \leq i \leq n} \{d_{\text{prev}}(v_i)\}$ and $d_{\text{prev}}(v_i)$ is the number of previous neighbors of v_i in the ordering (also see [39]).

By symmetry, it suffices to consider the case that

the transmitting nodes of \mathcal{S}_i have mutual distances greater than one. (20)

Order the vertices of $CG[\mathcal{S}_i]$ as $u_1 v_1, u_2 v_2, \dots, u_p v_p$ such that the tails u_1, u_2, \dots, u_p of the corresponding links are located from left to right. When

vertex $u_i v_i$ is to be colored in $CG[\mathcal{S}_i]$, all those links corresponding to the previous neighbors of vertex $u_i v_i$ which interfere with link $u_i v_i$ have their tails in the left half of the disk centered at u_i with radius $\rho + 1$. Combining this with (20), Zassenhaus-Groemer-Oler inequality ([Corollary 1](#)) tells us that $u_i v_i$ has at most $\beta_\rho - 1$ previous neighbors, where

$$\beta_\rho = \frac{1}{\sqrt{3}}\pi(\rho + 1)^2 + \left(\frac{\pi}{2} + 1\right)(\rho + 1) + 1.$$

Thus $CG[\mathcal{S}_i]$ can be properly colored with at most β_ρ colors, and thus \mathcal{S}_i can be further scheduled into at most β_ρ interference-free subsets with respect to interference radius $\rho > 1$. The total number of time-slots for the whole scheduling is at most $\beta_\rho \ell$.

Notice that PAS is a well-separated aggregation schedule for $\rho = 1$. Hence the EPAS can produce a schedule with latency at most $\beta_\rho(2R + \Delta + O(\log R))$.

6 Maximum Interference-Free Sets of Links

In a wireless sensor network, interferences are inevitable, which causes delays of transmissions. In a *physical interference model*, when a node u is transmitting at a power p_u , the signal power caught by another node v is

$$cp_v(u) = \frac{p_u}{\|uv\|^\alpha},$$

where $\|uv\|$ is the Euclidean distance between u and v and α is the *path-loss exponent* which is a constant usually between two and six. A node can correctly receive a signal if and only if the *signal-to-interference-plus-noise ratio* (SINR) is above a threshold. To speak it concretely, suppose \mathcal{L} is a set of transmission requests specified by a set of source-destination links. For a link subset $L \subseteq \mathcal{L}$, denote by L^s and L^r the set of source nodes and the set of destination nodes of the links in L , respectively. For a link uv and a vertex set W not containing u , the SINR of uv with respect to W is defined as

$$\text{SINR}(W; uv) = \frac{cp_v(u)}{N + \sum_{w \in W} cp_v(w)}, \quad (21)$$

where N is the ambient noise. Node v correctly receives the message from u if and only if

$$\text{SINR}(L^s; uv) \geq \beta, \quad (22)$$

where L is the set of the other links which are scheduled to transmit at the same time-slot as uv . A subset L of links is said to be *interference-free* if $\text{SINR}(L^s \setminus \{u\}; uv) \geq \beta$ holds for each link $uv \in L$. In other words, links in L can be scheduled to transmit

at the same time-slot such that all the destination nodes of L can correctly receive the messages from their corresponding sources.

Starting with the pioneering work of Gupta and Kumar [26], the capacity of networks with physical interference has been extensively studied in recent years. A lot of works claim the superiority of the physical interference model over the binary interference model [25, 38, 42]. However, because of the nonconvexity and the nonlocality property of the physical interference model, theoretical studies are notoriously hard.

Consider the problem of *maximum interference-free set of links* (MISL), also known as *one-shot scheduling* problem. Given a set \mathcal{L} of transmission requests, the goal is to find a maximum subset L of \mathcal{L} which is interference-free. With uniform power assignment (i.e., every node transmits at a fixed power P), Goussevskaia et al. [19] gave the first constant approximation algorithm for MISL, but their analysis is only valid when the ambient noise $N = 0$. For $N > 0$, Wan et al. [56] gave a constant approximation algorithm in which [Corollary 4](#) is implemented. The following part introduces the major ideas of the algorithm in [56] and how [Corollary 4](#) is integrated in the analysis of the algorithm.

Let $R = \left(\frac{P}{\beta N}\right)^{1/\alpha}$. By (21) and (22), R is the maximum length that a transmission can be done without considering the interferences from the other nodes. A simple calculation shows that $\text{SINR}(W; uv) \geq \beta$ if and only if

$$\beta \frac{\sum_{w \in W} (\|uv\| / \|wv\|)^\alpha}{1 - (\|uv\| / R)^\alpha} \leq 1.$$

In the light of this, Wan et al. defined the *relative interference of a node w ($\neq u$) to the link uv* as

$$RI(w; uv) = \beta \frac{(\|uv\| / \|wv\|)^\alpha}{1 - (\|uv\| / R)^\alpha},$$

and the *relative interference of a set W ($u \notin W$) to uv* as

$$RI(W; uv) = \sum_{w \in W} RI(w; uv).$$

Thus a link set L is interference-free if and only if $RI(L^s \setminus \{u\}; uv) \leq 1$ holds for every link $uv \in L$.

The algorithm uses a parameter $\phi \in (0, 1)$ which is to be chosen to minimize the final approximation ratio. It also makes use of the Riemann zeta function $\zeta(x) = \sum_{j=1}^{\infty} j^{-x}$. For a link uv , denote

$$\rho_{uv, \phi} = 1 + \left(\frac{\beta(16\zeta(\alpha - 1) + 8\zeta(\alpha) - 6)}{\phi(1 - (\|uv\| / R)^\alpha)} \right)^{1/\alpha}.$$

The algorithm is presented in [Algorithm 3](#).

Algorithm 3 Approximation algorithm for MISL

```

1: Let  $\mathcal{S}$  be the set of transmission requests sorted by the increasing order of lengths.
2: Let  $I = \emptyset$ ,  $U = \emptyset$ .
3: while  $\mathcal{S} \neq \emptyset$  do
4:   Let  $uv$  be the first link in  $\mathcal{S}$ .
5:    $\mathcal{S} = \mathcal{S} \setminus \{uv\}$ ,  $I = I \cup \{uv\}$ ,  $U = U \cup \{u\}$ .
6:   Let  $\mathcal{S}' = \{u'v' \in \mathcal{S} : \|u'u'\| < \rho_{\|uv\|,\phi} \cdot \|uv\|\}$ .
7:    $\mathcal{S} = \mathcal{S} \setminus \mathcal{S}'$ .
8:   Let  $\mathcal{S}'' = \{u''v'' \in \mathcal{S} : RI(U; u''v'') > 1 - \phi\}$ .
9:    $\mathcal{S} = \mathcal{S} \setminus \mathcal{S}''$ .
10: end while
11: Output  $I$ .

```

In the algorithm, the transmission requests are sorted by the increasing order of lengths and scanned sequentially. That is, preference is given to those links with shorter lengths. When a link uv is selected into the interference-free set I , the remaining links whose sources are near to u to some extend are discarded (see \mathcal{S}' in line 6), then the remaining links whose relative interferences to uv is greater than $1 - \phi$ are discarded (see \mathcal{S}'' in line 8).

The final output I of [Algorithm 3](#) is indeed an interference-free set of links. In fact, for each link $uv \in I$, denote by I_{uv}^- and I_{uv}^+ the sets of links of I which are chosen before uv and after uv , respectively. Since uv is not discarded as a member of \mathcal{S}'' , one has $RI((I_{uv}^-)^s; uv) \leq 1 - \phi$. The use of \mathcal{S}' ensures that the mutual distances between the nodes in $(I_{uv}^+)^s \cup \{u\}$ are at least $\rho_{\|uv\|,\phi} \cdot \|uv\|$. Wan et al. proved that such a property implies $RI((I_{uv}^+)^s; uv) < \phi$. Then, by the additivity of relative interference, $RI(I \setminus \{uv\}; uv) \leq 1$.

Let

$$\begin{aligned}\rho_0 &= 1 + \left(\frac{\beta(16\zeta(\alpha - 1) + 8\zeta(\alpha) - 6)}{\phi} \right)^{1/\alpha}, \\ \mu_1 &= \left\lfloor \frac{2\pi}{\sqrt{3}} \left(\frac{\rho_0}{\beta^{1/\alpha} - 1} \right)^2 + \pi \frac{\rho_0}{\beta^{1/\alpha} - 1} \right\rfloor + 1, \\ \mu_2 &= 5 \left\lceil \left((1 - \phi)^{-1/\alpha} + 2\beta^{-1/\alpha} \right)^\alpha \right\rceil.\end{aligned}$$

It was proved in [56] that [Algorithm 3](#) has approximation ratio $(\mu_1 + \mu_2 + 1)$. To prove this, suppose $|I| = k$. For $i = 1, 2, \dots, k$, let $u_i v_i$ be the link chosen into I and $\mathcal{S}'_i, \mathcal{S}''_i$ be the sets discarded from \mathcal{S} in the i th iteration. Denote $L' = \bigcup_{i=1}^k \mathcal{S}'_i$ and $L'' = \bigcup_{i=1}^k \mathcal{S}''_i$. Consider an optimal solution OPT. It can be proved that $|\text{OPT} \cap L''| \leq \mu_2 k$ and

$$|\text{OPT} \cap \mathcal{S}'_i| \leq \mu_1 \text{ holds for each } 1 \leq i \leq k. \quad (23)$$

Then the approximation ratio follows from

$$|\text{OPT}| = |\text{OPT} \cap I| + |\text{OPT} \cap L'| + |\text{OPT} \cap I''| \leq (1 + \mu_1 + \mu_2)k.$$

It is in proving (23) that [Corollary 4](#) is used. In fact, Wan et al. proved that the sources of all the links in $\text{OPT} \cap \mathcal{S}'_i$ have mutual distances at least

$$\left(\frac{\beta^{1/\alpha}}{(1 - (\|u_i v_i\|/R)^\alpha)^{1/\alpha}} - 1 \right) \|u_i v_i\|.$$

Combining this with the observation that the sources of all the links in $\text{OPT} \cap \mathcal{S}'$ lie in the disk of radius $\rho_{\|u_i v_i\|, \phi} \cdot \|u_i v_i\|$, [Corollary 4](#) tells us that

$$|\text{OPT} \cap \mathcal{S}'_i| \leq \frac{2\pi}{\sqrt{3}} \left(\frac{\rho_{\|u_i v_i\|, \phi}}{\frac{\beta^{1/\alpha}}{(1 - (\|u_i v_i\|/R)^\alpha)^{1/\alpha}} - 1} \right)^2 + \pi \cdot \frac{\rho_{\|u_i v_i\|, \phi}}{\frac{\beta^{1/\alpha}}{(1 - (\|u_i v_i\|/R)^\alpha)^{1/\alpha}} - 1} + 1.$$

Then inequality (23) follows from an analytic calculation.

7 Diameter Bounded Connected Dominating Sets

Using connected dominating set (CDS) as virtual backbone has a lot of advantages, such as saving energy and reducing interferences. However, there are some other concerns. For example, the probability of message transmission failure increases if a message is sent through a longer path [\[41\]](#). Hence one expects the diameter of the virtual backbone to be as small as possible. In [\[28\]](#), Kim et al. initiated the study on minimum CDS with bounded diameter. Two approximation algorithms were presented in [\[28\]](#) for the unit disk graph (UDG), which were named by the authors as CDS-BD-C1 and CDS-BD-C2, respectively. The former one has a smaller diameter but a larger size of CDS, and the latter one has a smaller size of CDS but a larger diameter.

Both algorithms are based on a breadth-first-search tree T rooted at an arbitrary node s . Suppose T has depth R . It can be seen that

$$\text{Diam}^* \geq R - 2, \quad (24)$$

where Diam^* is the diameter of a CDS D^* with the smallest diameter. In fact, consider two nodes $u, v \in V(G)$ with $\text{dist}_G(u, v) = \text{Diam}(G)$. Let u', v' be the nodes in D^* dominating u, v , respectively (u' may coincide with u if u is in D , the same coincidence may occur for v and v'). Then

$$\text{Diam}^* \geq \text{dist}_{D^*}(u', v') \geq \text{dist}_G(u', v') \geq \text{dist}_G(u, v) - 2 = \text{Diam}(G) - 2 \geq R - 2.$$

For $i = 0, 1, \dots, R$, let L_i be the set of nodes at hop distance i from node s . Find a maximal independent set (MIS) I induced by T by, for example, the

Algorithm 4 Connector 1

```

1:  $C = \emptyset$ .
2: for  $i = 2$  to  $R$  do
3:   for each node  $u \in I \cap L_i$  do
4:     if  $u$  has no neighbor in  $C \cap L_{i-1}$  then
5:       Let  $v$  be a neighbor of  $u$  in  $L_{i-1}$  and add  $v$  into  $C$ .
6:     end if
7:   end for
8: end for

```

first-fit strategy used in Sect. 5 or a coloring strategy as in [28]. Such an MIS has the property that

$$\text{each node } u \in L_i \setminus I \text{ is adjacent with some node in } I \cap (L_{i-1} \cup L_i). \quad (25)$$

Then a set C of connectors is found layer by layer.

First consider CDS-BD-C2. The construction of C is described in Algorithm 4. Notice that since s is an MIS node, there is no MIS nodes in L_1 , and thus the for-loop starts from $i = 2$. By property (25), it is easy to see that $D = I \cup C$ is a CDS.

Theorem 7 *CDS-BD-C2 produces a connected dominating set D with*

$$\begin{aligned} |D| &\leq 6.862\text{opt}^* + 8.638, \\ \text{Diam}(G[D]) &\leq 4(\text{Diam}^* + 1), \end{aligned}$$

where opt^* and Diam^* are the size and the diameter of the optimal solution D^* .

Proof Since in each iteration, at most one node is added into C , one has $|C| \leq |I| - 1$. Thus $|D| \leq 2|I| - 1$ and the first inequality follows from the inequality in Lemma 6.

Denote by r_i the maximum distance of a node in $I \cap L_i$ from s in $G[D]$. By induction on i , it can be proved that

$$r_1 = 1 \text{ and } r_i \leq 2(i - 1) \text{ for } i \geq 2. \quad (26)$$

In fact, it is trivial to see that $r_1 = 1$ and $r_2 = 2$. Suppose $i \geq 3$ and (26) is true for r_1, r_2, \dots, r_{i-1} . Let u be a node in $I \cap L_i$ such that $\text{dist}(s, u) = r_i$. Suppose v is the neighbor of u in $C \cap L_{i-1}$. By property (25), v is adjacent to a node $u' \in I \cap (L_{i-2} \cup L_{i-1})$. Then, it follows from the induction hypothesis that

$$r_i = \text{dist}(s, u) \leq d(s, u') + 2 \leq 2(i - 1) - 2 + 2 = 2(i - 1).$$

In particular, $r_R \leq 2(R - 1)$. Notice that a node which is farthest from s in $G[D]$ must be an MIS node. Hence

$$\text{Diam}(G[D]) \leq 2r_R \leq 4(R - 1). \quad (27)$$

Algorithm 5 Connector 2

```

1:  $C = \emptyset$ .
2: for  $i = 1$  to  $\lfloor R/2 \rfloor$  do
3:   for each node  $u \in I \cap L_{2i}$  do
4:     if  $u$  has no neighbor in  $C \cap L_{2i-1}$  then
5:       Let  $v$  be a neighbor of  $u$  in  $L_{2i-1}$  and add  $v$  into  $C$ .
6:       if  $v$  has no neighbor in  $(I \cup C) \cap L_{2i-2}$  then
7:         Let  $w$  be a neighbor of  $v$  in  $L_{2i-2}$  and add  $w$  into  $C$ .
8:       end if
9:     end if
10:   end for
11:   for each node  $u \in I \cap L_{2i+1}$  do
12:     if  $u$  has no neighbor in  $C \cap L_{2i}$  then
13:       Let  $v$  be a neighbor of  $u$  in  $L_{2i}$  and add  $v$  into  $C$ .
14:     end if
15:   end for
16: end for

```

Then the second inequality follows from (27) and (24). \square

CDS-BD-C1 differs from CDS-BD-C2 in the construction of C which is described in [Algorithm 5](#). The authors in [28] gave a weight function on the node set to specify the priority in selecting the ancestors, which is a heuristic to raise the performance in the simulation. Since this does not affect the theoretical analysis, the algorithm is presented in a simpler form as [Algorithm 5](#).

Theorem 8 *CDS-BD-C1 produces a connected dominating set D with*

$$|D| \leq 10.293\text{opt}^* + 12.457,$$

$$\text{Diam}(G[D]) \leq 3\text{Diam}^* + 5.$$

Proof Again by property (25), the output $D = I \cup C$ is a CDS. Notice that for each MIS node, at most two ancestors are added into C . Hence $|C| \leq 2(|I| - 1)$, and thus $|D| \leq 3|I| - 2$. The first inequality follows from [Lemma 6](#).

Using the same notation r_i as in [Theorem 7](#), it can be proved that for CDS-BD-C1,

$$r_i \leq \left\lfloor \frac{3i - 1}{2} \right\rfloor \text{ holds for } i \geq 1. \quad (28)$$

Notice that $r_1 = 1$ and $r_2 = 2$ satisfy (28). Suppose (28) is true for $r_1, r_2, \dots, r_{2j-1}, r_{2j}$. Let u be a node in $I \cap L_{2j+1}$ with $\text{dist}(s, u) = r_{2j+1}$. Suppose

v is the parent of u in C . Since v is adjacent with some MIS node u' in layer L_{2j} or L_{2j-1} , one has

$$r_{2j+1} = \text{dist}(s, u) = \text{dist}(s, u') + 2 \leq r_{2j} + 2 \leq 3j + 1 = \left\lfloor \frac{3(2j + 1) - 1}{2} \right\rfloor.$$

Next consider a node $u \in I \cap L_{2j+2}$ with $\text{dist}(s, u) = r_{2j+2}$. By line 4–9 of the algorithm, u has an ancestor $w \in (I \cup C) \cap L_{2j}$. If $w \in I$, let $u' = w$; otherwise, let u' be an MIS neighbor of w in layer L_{2j} or L_{2j-1} . Then

$$r_{2j+2} = \text{dist}(s, u) \leq \text{dist}(s, u') + 3 \leq r_{2j} + 3 \leq 3j + 2 = \left\lfloor \frac{3(2j + 2) - 1}{2} \right\rfloor.$$

Thus (28) is true by induction.

It follows that $\text{Diam}(G[D]) \leq 2r_R \leq 3R - 1$. Combining this with (24), the second inequality is proved. \square

8 Routing-Cost Bounded Connected Dominating Sets

Using a connected dominating set (CDS) as the virtual backbone of a network G , energy is saved; interference is alleviated. However, the routing cost might be increased. For a node set D , denote by $d_D(u, v)$ the length of a shortest path between two nodes u and v such that all the internal nodes on this path belong to D (length here means *hop distance*). Consider the example in Fig. 7a, the set D of black nodes is a CDS. One can see that $d_D(A, C) = 4 > 2 = d_G(A, C)$.

Taking the routing cost into consideration, Ding et al. [9] proposed the problem of *minimum routing cost CDS* (MOC-CDS), the goal of which is to find a minimum CDS D with the property that $d_D(u, v) = d_G(u, v)$ holds for any pair of nodes $u, v \in V(G)$. The set of black nodes in Fig. 7b is an MOC-CDS. It was proved in [9] that MOC-CDS cannot be polynomially approximated within factor $(1 - \varepsilon) \ln \Delta$ unless $NP \subseteq DTIME(n^{O(\log \log n)})$, where Δ is the maximum degree of the graph. The authors also gave a $(1 - \ln 2 + 2 \ln \Delta)$ -approximation algorithm.

However, the size of an MOC-CDS might be conspicuously larger than the size of a CDS. To balance the competing considerations of energy saving and routing cost, Du et al. [11] proposed the problem of *guaranteed routing cost CDS*: Given a constant $\beta \geq 1$, the goal is to find a minimum CDS D such that $d_D(u, v) \leq \beta d_G(u, v)$ holds for any pair of nodes $u, v \in V(G)$. Such a CDS is called a β -GOC-CDS. For $\beta = 5$, a constant approximation algorithm was given in [11].

The authors first proved that for any pair of nodes $u, v \in V(G)$, $d_D(u, v) - 1 \leq \beta(d_G(u, v) - 1)$ if and only if for any pair of nodes $u, v \in V(G)$ with $d_G(u, v) = 2$, $d_D(u, v) - 1 \leq \beta$. The necessity is obvious. The sufficiency is established by considering a shortest (u, v) -path in G and sequentially replacing sub-paths of length two by paths of length at most $\beta + 1$ whose internal nodes all belong to D (see the

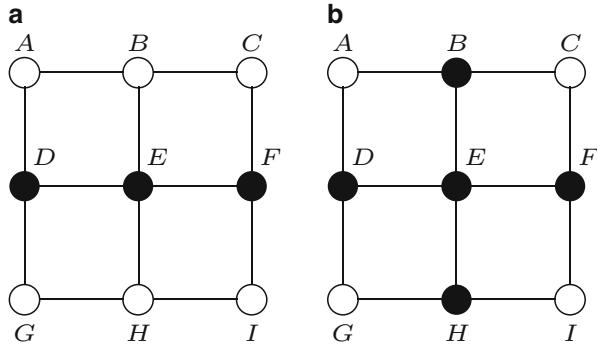


Fig. 7 (a) a CDS, (b) an MOC-CDS

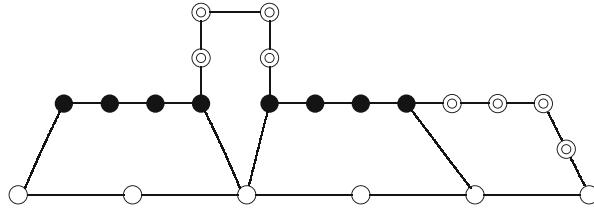


Fig. 8 The *white nodes* are on the shortest path. The *blackened nodes* and the *double-circle nodes* belong to D . First replace sub-paths of length two by the paths whose internal nodes are the blackened ones. Then further replace sub-paths of length two at the intersection points by the paths whose internal nodes are the *double-circle* ones

illustration in Fig. 8). As a consequence, if $d_D(u, v) \leq \beta + 1$ holds for any pair of nodes $u, v \in V(G)$ with $d_G(u, v) = 2$, then D is a β -GOC-CDS.

The algorithm first computes a maximal independent set (MIS) I . Then for every pair of nodes $u, v \in I$ with $d_G(u, v) \leq 4$, the internal nodes of a shortest (u, v) -path are added. Denote the set of the added nodes as C . The output $D = I \cup C$ has the property that for any pair of nodes $u, v \in I$ with $d_G(u, v) \leq 4$, the distance $d_D(u, v) \leq 4$. Consider two nodes $u', v' \in V(G)$ with $d_G(u', v') = 2$. Since I is a dominating set, there exist $u, v \in I$ such that u' is dominated by u and v' is dominated by v . Then $d_G(u, v) \leq 2 + d_G(u', v') = 4$, and thus $d_D(u, v) \leq 4$. Hence $d_D(u', v') \leq 2 + d_D(u, v) \leq 6$. By the arbitrariness of u', v' and the claim in the previous paragraph, D is a 5-GOC-CDS.

To analyze the approximation ratio of the algorithm, the authors first proved that for any node $u \in I$, the number of nodes $v \in I$ with $0 < d_G(u, v) \leq 4$ is at most 80. In fact, using Corollary 4, the upper bound can be reduced to 70: Since any such node v lies in the disk centered at u with radius 4, the number of such nodes is upper bounded by

$$\left\lfloor \frac{2}{\sqrt{3}} \cdot 16\pi + \frac{1}{2} \cdot 8\pi + 1 \right\rfloor - 1 = 70.$$

To bound $|C|$, construct an auxiliary graph H with vertex set I ; two vertices are adjacent in H if their distance is at most four in G . By the above analysis, the maximum degree of H is upper bounded by 70, and thus there are at most $35|I|$ edges in H . Corresponding to each edge in H , at most three nodes are added into C . Hence $|C| \leq 105|I|$, and thus $|D| \leq 106|I|$. By using the inequality in [Lemma 6](#), one has $|I| \leq 3.431 \cdot \text{opt}_{MCDS} + 4.819$, where opt_{MCDS} is the number of nodes in a minimum CDS, which is clearly a lower bound for the size of a minimum GOC-CDS. Then $|D| \leq 363.686 \cdot \text{opt}_{GOC-CDS} + 510.814$, which is a constant approximation solution to the GOC-CDS problem.

Based on the above constant approximation solution, a PTAS was given in [\[10\]](#).

9 Conclusion

In this chapter, an overview of some typical applications of packing circle theory to the wireless network is presented. These applications also propose new challenges to the study on packing circle theory. This chapter aims to serve as a spur to induce more researchers to come forward with valuable contributions.

Acknowledgements The work is supported in part by National Natural Science Foundation of China under grant 61222201 and Specialized Research Fund for the Doctoral Program of Higher Education (20126501110001) and National Science Foundation of USA under grants CNS0831579 and CCF0728851.

Cross-References

- ▶ [A Unified Approach for Domination Problems on Different Network Topologies](#)
- ▶ [Algorithmic Aspects of Domination in Graphs](#)
- ▶ [Connected Dominating Set in Wireless Networks](#)
- ▶ [Coverage Problems in Sensor Networks](#)
- ▶ [Partition in High Dimensional Spaces](#)

Recommended Reading

1. I.F. Akyildiz, D. Pompili, T. Melodia, Underwater acoustic sensor networks: research challenges. *J. Ad Hoc Netw.* **3**(3), 257–279 (2005)
2. A.D. Amis, R. Prakash, T.H.P. Vuong, D.T. Huynh, Max-min D-cluster formation in wireless Ad Hoc networks, in *INFOCOM*, Tel Aviv, 2000
3. V. Bhargavan, B. Das, Routing in Ad Hoc networks using minimum connected dominating sets, in *International Conference on Communications'97*, Montreal, Canada, June 1997
4. J.A. Bondy, U.S.R. Murty, *Graph Theory with Applications* (North-Holland, New York, 1976)
5. M. Cadei, X. Cheng, D.-Z. Du, Connected domination in Ad Hoc wireless networks, in *Proceedings of the 6th International Conference on Computer Science and Informatics*, Durham, 2002
6. X.J. Chen, X.D. Hu, J.M. Zhu, Minimum data aggregation time problem in wireless sensor networks. *Lect. Notes Comput. Sci.* **3794**, 133–142 (2005)

7. J.H. Conway, N.J.A. Sloane, *Sphere Packing, Lattices and Groups*, 2nd edn. (Springer, New York, 1993)
8. B. Das, R. Sivakumar, V. Bhargavan, Routing in Ad-Hoc networks using a spine, in *International Conference on Computers and Communications Networks'97*, Las Vegas, Sept 1997
9. L. Ding, X.F. Gao, W.L. Wu, W.J. Lee, X. Zhu, D.-Z. Du, Distributed construction of connected dominating sets with minimum routing cost in wireless network, in *ICDCS*, Genova 2010
10. H.W. Du, Q. Ye, J.F. Zhong, Y.X. Wang, W.J. Lee, H. Park, PTAS for minimum connected dominating set with routing cost constraint in wireless sensor network, in *COCOA 2010*, Kailua-Kona. Lecture Notes in Computer Science, vol. 6508, 2010, pp. 252–259
11. H.W. Du, Q. Ye, W.L. Wu, W.J. Lee, D.Y. Li, D.-D. Du, S. Howard, Constant approximation for virtual backbone construction with guaranteed routing cost in wireless sensor networks, in *INFOCOM*, Shanghai, 2011
12. F. Fodor, The densest packing of 19 congruent circles in a circle. *Geom. Dedic.* **74**, 139–145 (1999)
13. F. Fodor, The densest packing of 12 congruent circles in a circle. *Beitr. zur Algebra und Geom. Contrib. Algebra Geom.* **41**, 401–409 (2000)
14. F. Fodor, The densest packing of 13 congruent circles in a circle. *Beitr. zur Algebra und Geom. Contrib. Algebra Geom.* **44**, 431–440 (2003)
15. J.H. Folkman, R.L. Graham, A packing inequality for compact convex subsets of the plane. *Can. Math. Bull.* **12**, 745–752 (1969)
16. S. Funke, A. Kesselman, U. Meyer, M. Segal, A simple improved distributed algorithm for minimum CDS in unit disk graphs. *ACM Trans. Sens. Net.* **2**, 444–453 (2006)
17. X. Gao, Y. Wang, X. Li, W. Wu, Analysis on theoretical bonds for approximating dominating set problems. *Discret. Math. Algorithms Appl.* **1**(1), 71–84 (2009)
18. M. Goldberg, Packing of 14, 16, 17 and 20 circles in a circle. *Math. Mag.* **44**(3), 134–139 (1971)
19. O. Goussevskaia, M.M. Halldóorsson, R. Wattenhofer, E. Welzl, Capacity of arbitrary wireless networks, in *IEEE INFOCOM*, Rio de Janeiro, 2009, pp. 1872–1880
20. R.L. Graham, Sets of points with given minimum separation (Solution to Problem El921). *Am. Math. Mon.* **75**, 192–193 (1968)
21. R.L. Graham, N.J.A. Sloane, Pemmy-packing and two-dimensional codes. *Discret. Comput. Geom.* **5**, 1–11 (1990)
22. R.L. Graham, B.D. Lubachevsky, Dense packings of $3k(k + 1) + 1$ equal disks in a circle for $k = 1, 2, 3, 4$ and 5, in *Proceedings of the First International Conference on Computing and Combinatorics (COCOON'95)*, Xi'an. Springer Lecture Notes in Computer Science, vol. 959, 1996, pp. 303–312
23. R.L. Graham, B.D. Lubachevsky, K.J. Nurmela, P.R.J. Östergrd, Dense packings of congruent circles in a circle. *Discret. Math.* **181**, 139–154 (1998)
24. H. Groemer, Über die Einlagerung von Kreisen in einen konvexen Bereich. *Math. Z.* **73**, 285–294 (1960)
25. J. Grönkvist, A. Hansson, Comparison between graph-based and interference-based STDMA scheduling, in *Mobicom*, Long Beach, 2001, pp. 255–258
26. P. Gupta, P.R. Kumar, The capacity of wireless networks. *IEEE Trans. Inf. Theory* **46**, 388–404 (2000)
27. S.C.-H. Huang, P.-J. Wan, C.T. Vu, Y. Li, F. Yao, Nearly constant approximation for data aggregation scheduling in wireless sensor networks, in *IEEE INFOCOM* (IEEE Operations Center, Piscataway, 2007)
28. D.H. Kim, Y.W. Wu, Y.S. Li, F. Zou, D.-Z. Du, Constructing minimum connected dominating sets with bounded diameters in wireless networks. *IEEE Trans. Parallel Distrib. Syst.* **20**(2), 147–157 (2009)
29. D. Kim, W. Wang, N. Sohaee, C. Ma, W. Wu, W. Lee, D.-Z. Du, Minimum Data Latency Bound k-Sinks Placement Problem in Wireless Sensor Networks, *IEEE/ACM Transactions on Networking (TON)*, **19**(5), 1344–1353 (2011)

30. R.S. Ko, C.H. Huang, Using density-based d-Hop connected d-dominating sets routing scheme to achieve load balancing for wireless sensor networks, in *Wireless Communications and Networking Conference (WCNC)* (IEEE, Piscataway, 2007)
31. S. Kravitz, Packing cylinders into cylindrical containers. *Math. Mag.* **40**(2), 65–71 (1967)
32. D.Y. Li, L. Liu, H.Q. Yang, Minimum connected r-hop k-dominating set in wireless network. *Discret. Math. Algorithms Appl.* **1**(1), 45–57 (2009)
33. M. Li, P.-J. Wan, F.F. Yao, Tighter approximation bounds for minimum CDS in wireless ad hoc networks, in *ISAAC'2009*, Honolulu. Lecture Notes in Computer Science, vol. 5878, 2009, pp. 699–709
34. D. Li, Q. Zhu, H. Du, W. Wu, H. Chen, W. Chen, Conflict-free many-to-one data aggregation scheduling in multi-channel multi-hop wireless sensor networks, in *ICC2011*, Kyoto, Japan, 2011, pp.1–5
35. B.D. Lubachevsky, How to simulate billiards and similar systems. *J. Comput. Phys.* **94**, 255–283 (1991)
36. B.D. Lubachevsky, R.L. Graham, Curved hexagonal packings of equal disks in a circle. *Discret. Comput. Geom.* **18**, 179–194 (1997)
37. Z. Lv, W. Huang, PERM for solving circle packing problem. *Comput. Oper. Res.* **35**, 1742–1755 (2008)
38. R. Maheshwari, S. Jain, S.R. Das, A measurement study of interference modeling and scheduling in low-power wireless networks, in *SenSys*, Raleigh, 2008, pp. 141–154
39. D.W. Matula, L.L. Beck, Smallest-last ordering and clustering and graph coloring algorithms. *J. Assoc. Comput. Mach.* **30**, 417–427 (1983)
40. H. Melissen, Densest packing of eleven congruent circles in a circle. *Geom. Dedic.* **50**, 15–25 (1994)
41. K. Mohammed, L. Gewali, V. Muthukumar, Generating quality dominating sets for sensor network, in *ICCIMA*, Las Vegas, 2005, pp. 204–211
42. T. Moscibroda, R. Wattenhofer, Y. Weber, Protocol design beyond graph-based models, in *Hotnets*, Irvine, 2006, pp. 25–30
43. T.N. Nguyen, D.T. Huynh, Connected D-hop dominating sets in mobile Ad Hoc networks modeling and optimization in mobile, in *4th International Symposium on Ad Hoc and Wireless Networks* (IEEE, Piscataway, 2006)
44. N. Oler, An inequality in the geometry of numbers. *Acta Math.* **105**, 19–48 (1961)
45. U. Piri, Der Mindestabstand von n in der Einheitakreisscheibe gelegenen Punkten. *Math. Nachr.* **40**, 111–124 (1969)
46. L.S. Pontryagin, *Combinatorial Topology* (Graylock, New York, 1952)
47. G.E. Reis, Dense packing of equal circles within a circle. *Math. Mag.* **48**(1), 33–37 (1975)
48. M.Q. Rieck, S. Pai, S. Dhar, Distributed routing algorithms for multi-hop ad hoc networks using d -hop connected d -dominating sets. *Comput. Netw.* **47**, 785–799 (2005)
49. R. Sivakumar, B. Das, V. Bharghavan, An improved spine-based infrastructure for routing in Ad Hoc networks, in *IEEE Symposium on Computers and Communications'98*, Athens, Greece, June 1998
50. I. Stojmenovic, M. Seddigh, J. Zunic, Dominating sets and neighbor elimination based broadcasting algorithms in wireless networks, in *Proceedings of the IEEE Hawaii International Conference on System Sciences*, Maui, Jan 2001
51. T. Tarnai, K. Miyazaki, Circle packing and the sacred lotus. *Leonardo* **36**(2), 145–150 (2003)
52. A. Vahdatpour, F. Dabiri, M. Moaveni, M. Sarrafzadeh, Theoretical bound and practical analysis of connected dominating set in ad hoc and sensor networks, in *Proceedings of 22nd International Symposium on Distributed Computing (DISC)*, Arcachon, France, 2008, pp. 481–495
53. P.-J. Wan, K.M. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks. *ACM/Springer Mob. Netw. Appl.* **9**(2), 141–149 (2004). A preliminary version of this paper appeared in *IEEE INFOCOM*, 2002
54. P.-J. Wan, L.X. Wang, F.F. Yao, Two-phased approximation algorithms for minimum CDS in wireless ad hoc networks, in *IEEE ICDCS*, Beijing, 2008, pp. 337–344

55. P.-J. Wan, S.C.-H. Huang, L.X. Wang, Z.Y. Wan, X.H. Jia, Minimum-latency aggregation scheduling in multihop wireless networks, in *ACM MOBIHOC* (ACM, New York, 2009)
56. P.-J. Wan, X. Jia, F. Yao, Maximum independent set of links under physical interference model, in *WASA*, Boston, 2009, pp. 169–178
57. W. Wang, D.J. Kim, N. Sohaee, C.C. Ma, W.L. Wu, A PTAS for minimum d -hop underwater sink placement problem in 2-D underwater sensor networks. *Discret. Math. Algorithms Appl.* **1**(2), 283–289 (2009)
58. W. Wu, H. Du, X. Jia, Y. Li, S.C.-H. Huang, Minimum connected dominating sets and maximal independent sets in unit disk graphs. *Theor. Comput. Sci.* **352**(1–3), 1–7 (2006)
59. H. Zassenhaus, Modern development in the geometry of numbers. *Bull. Am. Math. Soc.* **67**, 427–439 (1961)
60. Z. Zhang, Q.H. Liu, D.Y. Li, Two algorithms for connected r -hop k -dominating set. *Discret. Math. Algorithms Appl.* **1**(4), 485–498 (2009)

Partition in High Dimensional Spaces

Zhao Zhang and Weili Wu

Contents

1	Introduction	2586
2	History of partition method	2587
3	The Basic Ideas	2590
3.1	The First Observation: Local Computability	2590
3.2	The Second Observation: Divisibility	2591
3.3	The Third Observation: A Probabilistic Property	2592
4	Some Basic Techniques	2593
4.1	The First Technique: Vacancy	2594
4.2	The Second Technique: Expansion	2595
5	Connection	2597
5.1	The Algorithm	2597
5.2	Correctness	2598
5.3	Time Complexity	2599
5.4	The Performance Ratio	2600
6	Multilayer Partition	2604
6.1	Maximum Independent Set in Ball Graph	2605
6.2	Minimum Sum Radii Cover	2609
7	Growing Partition	2613
7.1	Maximum Independent Set	2614
7.2	Minimum Dominating Set	2616
7.3	Minimum Connected Dominating Set	2617
8	Conclusion	2620
	Cross-References	2621
	Recommended Reading	2621

Z. Zhang (✉)

College of Mathematics and System Sciences, Xinjiang University, Urumqi, Xinjiang, People's Republic of China
e-mail: hxhzz@sina.com

W. Wu

Department of Computer Science, The University of Texas at Dallas, Richardson, Dallas, TX,
USA
e-mail: weiliwu@utdallas.edu

Abstract

Partition combined with shifting strategy is a powerful method in producing polynomial time approximation scheme (PTAS) for many NP-hard geometric problems. The focus of this chapter is on the ideas and the techniques of partition method which can be applied to high dimensional space, including the basic ideas, the small compensation technique which is used to deal with the requirement of connection, the multilayer partition technique which is used to deal with the situation when the geometric objects are not uniform, and the growing partition technique which does not require the geometric representation of the graph.

1 Introduction

The performance of an approximation algorithm is measured by its time complexity and performance ratio. For an NP-hard problem \mathcal{P} , suppose \mathcal{A} is an approximation algorithm for \mathcal{P} , the performance ratio of \mathcal{A} is

$$PR(\mathcal{A}) = \begin{cases} \sup_{\mathcal{I} \in \mathcal{P}} \frac{apx(\mathcal{I})}{opt(\mathcal{I})}, & \text{if } \mathcal{P} \text{ is a minimization problem;} \\ \sup_{\mathcal{I} \in \mathcal{P}} \frac{opt(\mathcal{I})}{apx(\mathcal{I})}, & \text{if } \mathcal{P} \text{ is a maximization problem,} \end{cases}$$

where the supremum is taken over all instances \mathcal{I} of \mathcal{P} , $apx(\mathcal{I})$ is the value of the solution output by the algorithm when it is applied to instance \mathcal{I} , and $opt(\mathcal{I})$ is the optimal value of instance \mathcal{I} .

A polynomial time approximation scheme (PTAS) is an approximation algorithm such that for any fixed positive real number $\varepsilon > 0$, the algorithm can produce an approximation solution in polynomial time (depending on the size of the input and ε) with performance ration $1 + \varepsilon$. Furthermore, if the running time is polynomial in the size of the input and $1/\varepsilon$, then the algorithm is a fully polynomial time scheme (FPTAS).

In this chapter, partition method is explored to device PTASs for some geometric NP-hard problems in high dimensional space. These problems are all strong NP-hard [4, 8, 18, 20, 25, 28, 31, 33, 39, 45]. Therefore, there are no FPTAS for these problems unless P=NP. Hence, the best approximation algorithm one can expect is a PTAS.

Partition has long been proved to be a powerful method in producing PTAS for many geometric NP-hard problems. It partitions the region into small subregions, finds exact solutions to the small subregions, and combines them together to yield a feasible solution to the original large region. The key observation for this method lies in two aspects. The first is that the exact solution for the problem can be found in time depending on the size of the region, and thus if the sizes of the subregions are bounded by a constant (which is the meaning of “small”), then the problem on the small subregions can be polynomially solvable. The second is that

the solutions for the subregions can be combined into a “feasible” solution to the original large region. The key analysis for the performance ratio of the method lies in the probabilistic observation that the probability for an optimal solution to “cross the boundary region” is very small.

Instead of a comprehensive survey of the large amount of literatures in this rich realm, this chapter focuses on the ideas of the partition method which can produce PTASs for the problems in high dimensional space. For the sake of simplicity, only three-dimensional problems are used as illustrations. However, all the methods introduced in this chapter can be easily adapted to fit higher dimensional problems.

The following sections are organized as follows. In Sect. 2, the history of partition method is briefly introduced. Section 3 works on the minimum unit ball cover problem to illustrate the basic ideas of the partition and shifting strategy. Section 4 introduces some techniques to deal with different problems, including the vacancy technique for the maximum independent set problem and the expansion technique for the minimum dominating set problem. Section 5 introduces the technique used in conquering the minimum connected dominating set problem. Section 6 introduces the idea of multilayer partition which is used to deal with the problems in which the geometric objects have different sizes. In Sect. 7, a partition method of a different taste, which is called growing partition, is introduced. This method does not require the geometric representation of the problems.

2 History of partition method

As early as 1977, Karp [38] has used the probabilistic method to analyze the performance of the partition algorithm for the traveling-salesman problem (TSP). The algorithm partitions the region into small subregions, each of which contains at most a constant number of cities. An optimal tour is constructed within each subregion. And then these subtours are combined to yield a tour through all the cities. Karp showed that with probability 1, the algorithm produces a PTAS for the TSP.

This method was applied to the rectilinear minimum Steiner tree problem by Komlos and Shing [43].

As a derandomization of Karp’s probabilistic analysis, the partition and shifting method was proposed by Baker [5] and Hochbaum and Maass [32].

In solving covering and packing problems on planar graphs, Baker [5] devised a technique to produce PTASs. The class of problems for which this technique provides PTASs includes maximum independent set, maximum tile salvage, partition into triangles, maximum H-matching, minimum vertex cover, minimum dominating set, and minimum edge dominating set on planar graphs. Taking the maximum independent set problem, for example, nodes are assigned to different levels according to their depths from the exterior face. A planar graph is said to be *k*-outerplanar if there is no node of level $> k$ in the graph. The key observation is that for a fixed k , the maximum independent set problem is solvable in linear time on *k*-outerplanar graphs by dynamic programming. In the case that the graph G is not *k*-outerplanar, for each $0 \leq r \leq k$, let G_r be the graph obtained from

G by deleting all those nodes whose levels are congruent to $r \pmod{k+1}$. Then G_r is composed of components of k -outerplanar subgraphs. Solving the problem on each of the components, then the union of the solutions forms an approximation solution A_r to the problem on G_r . Let r_0 be the index such that $|A_{r_0}| = \max\{|A_r| : 0 \leq r \leq k\}$, and let A^* be an optimal solution. Notice that there exists an index $0 \leq r^* \leq k$ such that at most $1/(k+1)$ nodes of A^* are of level r^* . Thus, $|A_{r_0}| \geq |A_{r^*}| \geq (1 - \frac{1}{k+1})|A^*|$. Taking $k = \lceil 1/\varepsilon \rceil$, one obtains a PTAS.

Similar idea was used by Hochbaum and Maass [32] to solve some other covering and packing problems (not only on the plane). It divides the region into small subregions, solves the problem optimally on each subregion, and combines the solutions for the subregions to yield a feasible solution to the original region. Then it shifts the partition one unit at a time to obtain a series of partitions, solves the problem as the above on each of the partitions, and chooses the best one. Hochbaum and Maass illustrated the strategy by solving the problem of ball covering in a d -dimensional space, where the diameters of all the balls are the same. They also remarked that this method can be easily generalized for problems where one covers with objects other than balls. However, the running time will depend exponentially on the ratio D/\tilde{D} , where D is the maximum diameter of the object and \tilde{D} is the maximum side length of a d -dimensional cube which can be contained in a covering object of the considered shape. In some special cases, such as the covering with rectilinear blocks, the ratio D/\tilde{D} can be eliminated from the exponent of the running time.

An application of the above method to the cellular networks can be found in [30], in which Glaßer et al. considered the problem of selecting a base station configuration from a set of possible configurations, such that as much as possible of the traffic is served while interference is kept to a minimum. PTASs were given under the assumption that there is an upper bound for the transmission ranges and a lower bound for the distance between different base stations.

Hunt et al. [34] were the first to implement the partition and shifting strategy to obtain PTASs for many NP-hard problems on unit disk graphs. Unit disk graphs arise naturally in the field of wireless networks. A graph G is a unit disk graph, if every vertex corresponds to a node on the plane; two vertices are adjacent in G if and only if the Euclidean distance between the corresponding nodes is not greater than one. Notice that a unit disk graph is not necessarily a planar graph. Hence, the method used for planar graphs is invalid in this circumstance. The NP-hard problems which were studied in [34] include the maximum independent set problem, the minimum vertex cover problem, and the minimum dominating set problem. Their method is also valid for graphs which can be drawn in an (r, s) -civilized manner, that is, the length of each edge is $\leq r$ and the distance between any two nodes is $\geq s$. Roughly speaking, a graph can be drawn in a civilized manner if the distances between vertices are not too large and the vertices are distributed not too densely.

Requiring connection significantly adds to the difficulty. The first PTAS for the minimum connected dominating set problem on unit disk graph was given by Cheng et al. in [17], in which a new idea was proposed to harmonize the solutions in the subregions. In their algorithm, a constant factor approximation solution D_0 is computed; the shifting strategy is used to select a partition $P(a)$ such that the

number of nodes of D_0 which fall into the boundary region of $P(a)$ is the minimum; then these boundary nodes are combined into the union of the optimal solutions for the subregions. The key idea is to use the boundary nodes to connect otherwise independent connected components of the subregions. However, the analysis of Cheng's algorithm depends on a property which is only valid on the plane. Hence, their method cannot be generalized to higher dimensional space.

Later, Zhang et al. [63] gave a PTAS for the minimum connected dominating set problem on unit ball graphs. Their algorithm is essentially the same as that in [17]. The difference lies in the theoretical analysis, where a new idea of *small compensation* was used. This technique is also valid for some other connection problems, such as the minimum connected vertex cover problem [64].

Some other applications of the partition and shifting strategy can be found in [48, 56, 58], etc.

In an intersection graph G , each vertex corresponds to a geometric object, and two vertices are adjacent in G if and only if the two objects corresponding to them have a non-empty intersection. In particular, if the geometric objects are disks of the same radii on the plane, then the intersection graph is equivalent to a unit disk graph. But if the shapes and the sizes of the geometric objects differ a lot, the difficulty is increased significantly.

In [22, 23], Erlebach et al. developed a new technique called *multilayer partition* to give PTASs for the maximum independent set problem and the minimum vertex cover problem in the intersection graphs in which the geometric objects are disks on the plane with varying radii. Before their work, for the intersection disk graphs with varying radii, the best known approximation algorithms have performance ratio 5 for the maximum independent set problem [47] and $3/2$ for the minimum weight vertex cover problem [46]. The idea in [22, 23] is to divide both the disks and the region into layers, consider an approximate problem in which the disks intersecting the lines of their own level are ignored, show that the approximate problem can be solved polynomially using dynamic programming, and the solution to the approximate problem differs from the solution to the original one by only a small fraction.

A PTAS for the fractional chromatic number problem on intersection disk graphs, using the above PTAS for the minimum weight independent set problem as a subroutine, was obtained by Jansen and Porkolab in [35].

The method is also applicable to the intersection graphs consisting of *fat* objects. Using instead shifted quadtree, Chan [15] described a PTAS for the packing problem which takes $n^{O(1/\varepsilon^{d-1})}$ time and space. Chan also gave a PTAS for the packing problem which takes $n^{O(1/\varepsilon^d)}$ time and linear space. This algorithm uses a completely different idea, by extending Smith and Wormald's separator theorem [55] for highly overlapping objects. Using the separator approach, Chan obtained the first PTAS for the piercing problem for fat objects with varying sizes.

An interesting relation between the intersection graphs and the planar graphs is as follows: every planar graph is a *coin graph*, that is the intersection graph of a set of interior-disjoint disks [42]. Therefore, every planar graph is a special intersection graph, and thus the above works generalized the work for the planar graphs.

The multilayer partition technique is also used by Lev-Tov and Peleg [44] in solving the minimum sum radii cover problem.

For all the above methods, the geometric representation is required, that is the coordinates of the nodes are known. This requirement is both reasonable and realistic, since Breu and Kirkpatrick [12] have proved that the recognition of unit disk graphs is NP-hard, and in the real-world applications, the location information is always given.

A method without requiring the geometric representation was proposed by Nieberg et al. in [51, 52], who gave a PTAS for the maximum independent set problem and a PTAS for the minimum dominating set problem in unit disk graphs. Their idea is to grow a subregion from a node until the boundary of the region is fairly *thin*; remove the nodes which have been searched and repeat this procedure until the region is partitioned into disjoint subregions; then the algorithm uses the union of the optimal solutions to the subregions to approach the optimal solution to the original problem. Something might be lost at the boundary regions. However, since the boundary is very thin, the loss can be controlled within a small fraction of the optimal solution. It was proved that the size of the subregions can be bounded by a constant depending on ε but not on the size of the input graph. As a consequence, the running time of their algorithm is $O(n^{\frac{1}{\varepsilon}} \log^{\frac{1}{\varepsilon}})$ to achieve a $(1 + \varepsilon)$ -approximation.

The growing partition method was also used on the problems with connection requirement. Gfeller and Vicari [29] used it to give a PTAS for the minimum connected dominating set problem in growth-bounded graphs. Roughly speaking, in a growth-bounded graph, the size of any maximum independent set can be controlled. The implementation of the method on the minimum d -Hop connected dominating set problem in growth-bounded graphs was done by Gao et al. in [27]. PTASs for the minimum dominating set and the minimum connected dominating set problems in unit disk graphs were also given by Wiese and Kranakis [61] using growing partition, focusing on the locality of the algorithm.

The partition method has many variations and is extensively used in many other problems. For example, in proving a lower bound for the k -steiner ratio, Du et al. [21] proved a useful lemma for the shifting technique in tree partition. With this lemma, Jiang et al. [36] and Wang et al. [59] designed a PTAS for the tree alignment problem. Wang et al. [60] introduced a new partition of balanced binary trees which results in a more efficient PTAS.

3 The Basic Ideas

The basic idea of the partition and shifting method lies in the following three observations.

3.1 The First Observation: Local Computability

For many geometric problems in a small region with fixed size, it is possible to obtain an optimal solution in polynomial time by exhaust search.

For example, consider the minimum unit ball cover problem (MUBC): a unit ball is a ball with diameter one; given a set of nodes in the space, the goal is to find a minimum number of unit balls to cover all the nodes. Suppose the nodes are enclosed in a cube Q_ℓ with side length ℓ , where ℓ is a fixed integer. Let $p = \lceil \sqrt{3}\ell \rceil$. Divide Q_ℓ into $p \times p \times p$ small cubes such that each small cube has side length at most $1/\sqrt{3}$. Notice that every unit ball whose center lies at the center of a small cube e covers all the nodes in e . Hence, the minimum number of unit balls needed is at most $p^3 = \lceil \sqrt{3}\ell \rceil^3$.

Notice that there are infinitely many choices for the unit balls. However, one can make the number to be finite by considering only *standard positions*. In fact, if there is a node u with a distance more than one from all the other nodes, then one unit ball has to be spent to cover u . If there are two nodes u_1, u_2 within distance one from each other such that they have a distance more than one from all the remaining nodes, then one unit ball has to be spent to cover u_1 and u_2 . In these two cases, such unit balls can be regarded as unique. For a unit ball which encloses at least three nodes, one can shift it to a position such that at least three nodes are on the surface of the unit ball. Such a position is called standard. For simplicity of statement, it is assumed that no three nodes are colinear. Then any three nodes determine at most two standard positions. In fact, suppose u_1, u_2, u_3 are three nodes which can be enclosed in a unit ball. Draw a triangle with u_1, u_2, u_3 as its extremes. On each side of the plane determined by the triangle, one can find a unique position which has distance $1/2$ from each of the three nodes (the positions on the two sides may collapse into one if the circumcenter of the triangle has distance $1/2$ from the three nodes). Then the two unit balls whose centers are located at these two positions are the only standard unit balls determined by u_1, u_2, u_3 . It follows that the number of standard positions is at most $O(n^3)$, where n is the number of nodes in the space.

It is easy to see that there exists an optimal ball cover such that every ball which encloses at least three nodes is in the standard position. Hence, to find an optimal ball cover, one only needs to select some balls from $N = O(n^3)$ candidate balls. By trying out every subset of candidate balls with cardinality at most $\lceil \sqrt{3}\ell \rceil^3$, one can find the desired ball cover in time at most $\binom{N}{1} + \binom{N}{2} + \binom{N}{\lceil \sqrt{3}\ell \rceil^3} = O(N^{\lceil \sqrt{3}\ell \rceil^3})$, which is polynomial in n when ℓ is a constant.

The keys to the success of the above method lie in two properties:

- (i) The number of elements in an optimal solution is bounded above by a polynomial in the size of the region.
- (ii) The locating problem can be transformed to a selecting problem by considering some standard candidates.

3.2 The Second Observation: Divisibility

When a large region is divided into small regions, then for many geometric problems, the union of the optimal solutions in every small region is a feasible solution for the large region. Consider the MUBC problem again. Suppose the n

nodes are enclosed in a cube Q with side length L , where L may be polynomial in n . Use planes to divide Q into smaller cubes Q_ℓ with side length ℓ , where ℓ is a constant. Such smaller cubes are called as *cells*. Solve the MUBC problem on each Q_ℓ as described in Sect. 3.1. Let U be the union of all these local optimal solutions. Clearly U is a ball cover of the n nodes.

3.3 The Third Observation: A Probabilistic Property

When one throws a ball to a square randomly, the probability that the ball hits the boundary lines is zero. If the boundary line is enlarged to a boundary region while the region is still very thin compared to the side length, then the probability that the ball falls into the boundary region is very small. This observation, combining with shifting strategy, leads to the theoretical analysis of the performance ratio for the above divide-and-conquer method.

Let U^* be an optimal solution to the MUBC problem, and consider its restriction in a cell Q_ℓ , that is let $U_{Q_\ell}^*$ be the set of the balls of U^* which have nonempty intersections with Q_ℓ . Clearly, $U_{Q_\ell}^*$ is a ball cover of the nodes in Q_ℓ . In other words, $U_{Q_\ell}^*$ is a feasible solution to the local problem on Q_ℓ . Hence, $|U_{Q_\ell}^*| \geq |U^*(Q_\ell)|$, where $U^*(Q_\ell)$ is the local optimal solution on Q_ℓ . It follows that

$$|U| = \sum |U^*(Q_\ell)| \leq \sum |U_{Q_\ell}^*|, \quad (1)$$

where $U = \bigcup U^*(Q_\ell)$ is the output of the approximation algorithm, and the sum is taken over every cell which contains at least one node to be covered.

The problem is that one ball in U^* might intersect more than one cell, and thus is counted more than once in the right-hand side of (1). Such balls are called as *boundary balls*. Suppose U^* has m boundary balls. Notice that if the side length ℓ is larger than one, then one unit ball intersects at most eight cells. It follows that

$$|U| \leq |U^*| + 7m. \quad (2)$$

Hence, to make the performance ratio small, one expects m to be small. Intuitively, the larger the side length ℓ is, the fewer the number of boundary balls might be. However, this is not necessarily true if the partition is badly chosen such that some planes which are used to divide the region go through too many balls. Nevertheless, by shifting the partition, it is possible to find one such that the number of boundary balls is small.

To state the idea more strictly with mathematical language, suppose the region enclosing all the nodes is $Q = \{(x_1, x_2, x_3) \mid 0 \leq x_i < L, i = 1, 2, 3\}$ (notice that Q is half closed and half open). Let $q = \lceil L/\ell \rceil$. Enlarge Q to $\tilde{Q} = \{(x_1, x_2, x_3) \mid 0 \leq x_i < (q+1)\ell, i = 1, 2, 3\}$. Divide \tilde{Q} into $(q+1) \times (q+1) \times (q+1)$ smaller cubes (called *cells*) with side length ℓ such that for $j_1, j_2, j_3 \in \{0, 1, \dots, q\}$, the (j_1, j_2, j_3) -th cell is $Q_\ell^{j_1 j_2 j_3} = \{(x_1, x_2, x_3) \mid j_i \ell \leq x_i < (j_i + 1)\ell, i = 1, 2, 3\}$ (notice that each cell is also half closed and half open). Denote this partition as $P(0)$.

Algorithm 1 PTAS for MUBC

```

1: for each partition  $P(a)$  do
2:   for each cell  $Q_\ell(a)$  in  $P(a)$  which contains at least one node do
3:     Compute a minimum ball cover  $U_\ell(a)$  of the nodes in  $Q_\ell(a)$ .
4:   end for
5:   Set  $U(a) = \bigcup_{Q_\ell^a} U_\ell(a)$ .
6: end for
7: Choose  $a^*$  such that  $|U(a^*)| = \min\{|U(a)| : a = 0, 1, \dots, \ell - 1\}$ .
8: Output  $U(a^*)$ .

```

For $a = 0, 1, \dots, \ell - 1$, let $P(a)$ be the partition obtained from $P(0)$ by shifting it such that the left-bottom-back corner of $P(a)$ lies at the coordinate $(-a, -a, -a)$. It should be noted that the enlargement of Q guarantees that after shifting, all the nodes to be covered are still enclosed by every partition $P(a)$.

The algorithm is executed as in [Algorithm 1](#).

Since every unit ball can serve as a boundary ball at most once (the assumption that every Q_ℓ is half closed and half open is used here), there must exist a partition $P(\hat{a})$ for which U^* has at most $|U^*|/\ell$ boundary balls. Then by (2) and Step 7 of [Algorithm 1](#), one has $|U(a^*)| \leq |U(\hat{a})| \leq (1 + 7/\ell)|U^*|$. Taking $\ell = \lceil 7/\varepsilon \rceil$, one has $|U(a^*)| \leq (1 + \varepsilon)|U^*|$, which implies that there is a PTAS for the MUBC problem.

4 Some Basic Techniques

To deal with various different conditions requires more techniques. They are illustrated by considering the maximum independent set problem (MIS), the minimum vertex cover problem (MVC), and the minimum dominating set problem (MDS) in a unit ball graph.

In a *unit ball graph* (UBG), each vertex corresponds to a node in the space. Two vertices are adjacent in the graph if and only if the Euclidean distance between the corresponding two nodes is not greater than one. UBGs are widely used in the field of wireless sensor networks. A wireless sensor network is an ad hoc wireless network which consists of a huge amount of static or mobile sensors. The sensors collaborate to sense, collect, and process the raw information of the phenomenon in the sensing area and transmit the processed information to the observers. Suppose the sensors of the network are uniform and have omnidirectional antennas. Then the transmission range of each sensor is a ball, and the transmission radii of all the sensors are the same. For simplicity of statement, the transmission radius can be scaled to one. It is widely adopted that two sensors can communicate if and only if they fall into the transmission ranges of each other, in other words, if and only if the Euclidean distance between them is not greater than one. Thus, in modeling a wireless sensor network, one arrives at an UBG. When all the sensors are distributed on the plane, then a unit ball graph degenerates to a *unit disk graph* (UDG).

A set I of vertices is an *independent set* of graph G if no vertices in I are adjacent in G . A set U of vertices is a *vertex cover* of G if every edge of G has at least one end in U . A set D of vertices is a *dominating set* of G if every vertex in $V(G) \setminus D$ has at least one neighbor in D . For an independent set, one wants its size to be as large as possible, which results in the maximum independent set problem (MIS). For a vertex cover or a dominating set, one wants its size to be as small as possible, which results in the minimum vertex cover problem (MVC) and the minimum dominating set problem (MDS).

There are an abundance of studies on the above problems. For example, see [3, 6, 7, 24, 26, 40, 50, 54] for the MVC problem, and [11, 31, 37, 53] for the MDS problem. In particular, with the rapid progress in wireless networks, the studies are flourishing on the unit disk graphs [2, 10, 14, 16, 17, 19, 47, 49, 57, 62].

Notice that different from the MUBC problem, the above three problems are already selecting problems. Thus, the step of making a locating problem to a selecting problem by using standard positions (as property (ii) of the first observation) can be omitted.

For an UBG, since a cell Q_ℓ of side length $\sqrt{3}/3$ contains at most one vertex from an independent set, and one vertex in Q_ℓ is sufficient to dominate all the other vertices in Q_ℓ , it can be seen that property (i) of the first observation holds for the MIS and the MDS problems. It is well known that a vertex set U is a vertex cover of G if and only if $V(G) \setminus U$ is an independent set of G . Thus, the MVC problem also satisfies property (i). Hence, these three problems can also be solved in polynomial time if the size of the region is a constant.

In terms of the second observation, it is also valid for the MDS problem in UDG. However, it is no longer true for the MIS problem. The trouble occurs at the boundary region: two vertices which belong to the MISs of their own cells might not be independent if they fall into the boundary region of two adjacent cells. The following technique is very useful in dealing with such cases.

4.1 The First Technique: Vacancy

The technique creates a vacant region at the boundary: for each cell Q_ℓ , let $Q_\ell^{(i)}$ contain all those points of Q_ℓ which are at least $1/2$ distance away from the boundary of Q_ℓ (to be more strict, it is assumed that $Q_\ell^{(i)}$ is half closed and half open as for the case of Q_ℓ). Call $Q_\ell^{(i)}$ the *inner region* of Q_ℓ . Let $Q_\ell^{(b)} = Q_\ell \setminus Q_\ell^{(i)}$ and call it the *boundary region* of Q_ℓ . The algorithm for computing an MIS in UDG is executed as in [Algorithm 2](#).

The essential difference of [Algorithm 2](#) from [Algorithm 1](#) lies in Step 3, which computes a local optimal solution for each *inner* region instead of the whole region in a cell. Since nodes in two different inner regions are at least one distance away from each other, it can be seen that the set $I(a)$ as in Step 5 of [Algorithm 2](#) is an independent set.

Algorithm 2 PTAS for MIS

```

1: for each partition  $P(a)$  do
2:   for each cell  $Q_\ell(a)$  in  $P(a)$  whose inner region  $Q_\ell^{(i)}(a)$  contains at least one node do
3:     Compute a maximum independent set  $I_\ell(a)$  of the nodes in  $Q_\ell^{(i)}(a)$ .
4:   end for
5:   Set  $I(a) = \bigcup_{Q_\ell^{(i)}(a)} I_\ell(a)$ .
6: end for
7: Choose  $a^*$  such that  $|I(a^*)| = \max\{|I(a)| : a = 0, 1, \dots, \ell - 1\}$ .
8: Output  $I(a^*)$ .

```

To analyze the performance ratio of [Algorithm 2](#), compare the output $I(a^*)$ with an optimal solution I^* . For a partition $P(a)$, let $I_\ell^*(a)$ be the restriction of I^* on $Q_\ell^{(i)}(a)$. Clearly, $I_\ell^*(a)$ is an independent set of the nodes in $Q_\ell^{(i)}(a)$. Hence,

$$|I_\ell^*(a)| \leq |I_\ell(a)|,$$

where $I_\ell(a)$ is the *maximum* independent set computed in Step 3 of [Algorithm 2](#). Denote by $B(a)$ the union of the boundary regions of $P(a)$, and set $I_B^*(a) = I^* \cap B(a)$. Since each node can fall into at most one of the $B(a)$'s, there exists a partition $P(\hat{a})$ such that

$$|I_B^*(\hat{a})| \leq \frac{|I^*|}{\ell}.$$

It follows that

$$|I^*| = \sum_{Q_\ell^{(i)}(\hat{a})} |I_\ell^*(\hat{a})| + |I_B^*(\hat{a})| \leq \sum_{Q_\ell^{(i)}(\hat{a})} |I_\ell(\hat{a})| + \frac{|I^*|}{\ell} \leq |I(a^*)| + \frac{|I^*|}{\ell},$$

and thus

$$|I^*| \leq \left(1 + \frac{1}{\ell - 1}\right) |I(a^*)|.$$

Taking $\ell = \lceil 1/\varepsilon \rceil + 1$, it can be seen that $|I^*| \leq (1 + \varepsilon)|I(a^*)|$. Thus, [Algorithm 2](#) is a PTAS for the MIS problem.

4.2 The Second Technique: Expansion

Though the MDS problem on UBG satisfies the second observation, a transplantation of the analysis in [Sect. 3.3](#) to this problem does not produce the desired performance ratio. The difficulty here lies in the observation that the restriction of an optimal solution D^* to a cell Q_ℓ might not be a feasible solution to the local

Algorithm 3 PTAS for MDS

```

1: for each partition  $P(a)$  do
2:   for each cell  $Q_\ell(a)$  in  $P(a)$  whose expansion  $Q_\ell^{(e)}(a)$  contains at least one vertex do
3:     Compute a minimum dominating set  $D_\ell(a)$  of the vertices in  $Q_\ell^{(e)}(a)$ .
4:   end for
5:   Set  $D(a) = \bigcup_{Q_\ell^{(e)}(a)} D_\ell(a)$ .
6: end for
7: Choose  $a^*$  such that  $|D(a^*)| = \min\{|D(a)| : a = 0, 1, \dots, \ell - 1\}$ .
8: Output  $D(a^*)$ .

```

problem, since it is possible that some vertex in Q_ℓ has all its dominators outside of Q_ℓ . Hence, a modification has to be made to the algorithm.

For each cube Q_ℓ , suppose $Q_\ell = \{(x_1, x_2, x_3) \mid j_i\ell \leq x_i < (j_i + 1)\ell, i = 1, 2, 3\}$, let $Q_\ell^{(e)} = \{(x_1, x_2, x_3) \mid j_i\ell - 1 \leq x_i < (j_i + 1)\ell + 1, i = 1, 2, 3\}$ be the expansion of Q_ℓ , which expands the boundary of Q_ℓ by one unit on each of the six sides. The essential difference of [Algorithm 3](#) from [Algorithm 1](#) lies in Step 3, in which a minimum dominating set is computed for the *expanded* cell instead of the original cell. The output of [Algorithm 3](#) is clearly a dominating set of the unit ball graph.

Let D^* be a minimum dominating set of the unit ball graph. For a partition $P(a)$, let $D_\ell^*(a)$ be the restriction of D^* on $Q_\ell^{(e)}(a)$. Since every node in $Q_\ell(a)$ must have its dominators in $Q_\ell^{(e)}(a)$, it can be seen that $D_\ell^*(a)$ is a dominating set of the nodes in $Q_\ell(a)$. It follows that

$$|D_\ell(a)| \leq |D_\ell^*(a)|,$$

where $D_\ell(a)$ is the minimum dominating set computed in Step 3 of [Algorithm 3](#). Thus,

$$|D(a)| \leq \sum_{Q_\ell^{(e)}(a)} |D_\ell(a)| \leq \sum_{Q_\ell^{(e)}(a)} |D_\ell^*(a)|. \quad (3)$$

Notice that nodes in D^* might be repeatedly added in the right-hand side of inequality (3). Let $B(a) = \bigcup Q_\ell^{(e)}(a) \setminus Q_\ell(a)$, where the union is taken over all the expanded cubes which contain at least one node. Clearly, only those nodes which fall into $B(a)$ can be added more than once. Furthermore, a node in $B(a)$ can be repeatedly added at most eight times. Denote by $D_B^*(a)$ the set of nodes of D^* which fall in $B(a)$. By shifting the partitions, it can be seen that a node can fall into at most two of the $B(a)$'s. Thus, there exists a partition $P(\hat{a})$ such that

$$|D_B^*(\hat{a})| \leq \frac{2|D^*|}{\ell}.$$

Then by Step 7 and inequality (3), one has

$$|D(a^*)| \leq |D(\hat{a})| \leq |D^*| + 7|D_B^*(\hat{a})| \leq \left(1 + \frac{14}{\ell}\right) |D^*|.$$

Taking $\ell = \lceil 14/\varepsilon \rceil$, one has the desired $|D(a^*)| \leq (1 + \varepsilon)|D^*|$.

5 Connection

In many real-world applications, connection plays an important role. For example, in a wireless sensor network, establishing a virtual backbone can be modeled as finding a minimum connected dominating set of the UBG [19]. A vertex subset D of a graph G is said to be a *connected dominating set* (CDS) if D is a dominating set of G , and the subgraph of G induced by D is connected. Thus, in a wireless sensor network, all the sensors can communicate directly or indirectly with each other through nodes in D . For example, if a node u has a message to be transmitted to another node v . Suppose neither u nor v are in D . Then the message can be sent to the dominator of u in D , relayed to the dominator of v in D , and finally reaches node v . By restricting the major processing and transmission within the virtual backbone, a lot of energy is saved, and it is also saved a lot from suffering the broadcasting storm which is caused by too many nodes speaking at the same time. The MCDS problem asks for a connected dominating set with the minimum cardinality.

To devise a PTAS for the MCDS problem in a UBG needs a completely new idea for the connection part. Roughly speaking, in addition to computing local optimal solutions for all the inner regions, it computes a connected dominating set D^0 which is a constant approximation solution and uses the nodes of D^0 which fall into the boundary region to play the role of connection.

Two questions have to be answered. The first is how to compute a constant approximation solution. The second is how to harmonize D^0 with the local optimal solutions such that the output is indeed a connected dominating set with the desired performance ratio.

For the MCDS problem in UBG, Butenko and Ursulenko [13] gave a distributed 22-approximation. In [66], Zou et al. improved the performance ratio to $13 + \ln 10 \approx 15.303$. In [41], Kim et al. further improved the performance ratio to 14.937, using the solution for the Gregory–Newton problem concerning with the kissing number [65] and the Euler’s formula relating the number of edges, vertices, and faces of a surface in the geometric analysis.

The following introduces the PTAS given in [63] for the computation of an MCDS in UBG.

5.1 The Algorithm

For each cell Q_ℓ , the *boundary region* B_ℓ of Q_ℓ is the region contained in Q_ℓ such that each point in this region is at most distance 3 from the boundary of Q_ℓ .

Algorithm 4 PTAS for MCDS

Input: The geometric representation of a connected unit ball graph G and a positive real number $\varepsilon < 1$.

Output: A connected dominating set D of G .

- 1: Let $\ell = \lceil 300\rho/\varepsilon \rceil$.
 - 2: Use the ρ -approximation algorithm to compute a connected dominating set D_0 of G .
 - 3: For each $a \in \{0, 1, \dots, \ell - 1\}$, denote by $D_0(a)$ the set of vertices of D_0 lying in the boundary region of $P(a)$. Choose a^* with the minimum $|D_0(a)|$.
 - 4: For each cell Q_ℓ of $P(a^*)$, compute a local optimal solution D_ℓ satisfying the requirement in (4), using exhaust search.
 - 5: Let $D = D_0(a^*) \cup \bigcup_{Q_\ell \in P(a^*)} D_\ell$.
-

The *inner region* C_ℓ of Q_ℓ is the region of Q_ℓ such that each point is at least distance 2 away from the boundary of Q_ℓ . Note that B_ℓ and C_ℓ have an overlap.

The algorithm is described in [Algorithm 4](#). It first computes a connected dominating set D_0 which is a ρ -approximation solution. For example, $\rho = 22$ by the algorithm given by Butenko and Ursulenko [13], $\rho = 14.937$ by Kim et al. [41]. Then, for the partition $P(a^*)$ whose boundary region contains the minimum number of nodes in D_0 , and for the inner region of each cell, the algorithm computes a local optimal solution in the following sense. For each cell Q_ℓ of $P(a^*)$, denote by G_ℓ the subgraph of G induced by the vertices in the inner region C_ℓ . A vertex subset D_ℓ of vertices in Q_ℓ is said to be a *local optimal solution* if

$$\begin{aligned} \text{each connected component } H \text{ of } G_\ell \text{ is dominated by} \\ \text{only one connected component of } G[D_\ell], \end{aligned} \tag{4}$$

where $G[D_\ell]$ is the subgraph of G induced by the vertex set D_ℓ . Finally, the algorithm outputs the union of the local optimal solutions together with the nodes of D_0 which fall into the boundary region of $P(a^*)$.

5.2 Correctness

The following lemma shows the correctness of the algorithm.

Lemma 1 *The output D of [Algorithm 4](#) is a CDS of G .*

Proof First, it is shown that D is a dominating set. Let x be a vertex in $V(G)$. Suppose x is in cell Q_ℓ . If $x \in C_\ell$, then x is dominated by D_ℓ . If $x \in Q_\ell \setminus C_\ell$, then x is at distance less than two from the boundary of Q_ℓ . If $x \in D_0$, then $x \in D_0(a^*)$. If $x \notin D_0$, then the vertex $y \in D_0$ which dominates x is in $D_0(a^*)$ (notice that the overlapping of B_ℓ and C_ℓ plays an important role here). By the arbitrariness of x , D is a dominating set of G .

To show that $G[D]$ is connected, it is first shown that any components of $G[D_0(a^*)]$ are connected in $G[D]$. Let F_1, F_2 be two components of $G[D_0(a^*)]$. Since D_0 is a CDS of G , it can be seen that F_1 and F_2 can be connected by D_0 through the inner region of some cell Q_ℓ . Then there exist two vertices $x_1 \in V(F_1) \cap B_\ell \cap C_\ell$ and $x_2 \in V(F_2) \cap B_\ell \cap C_\ell$ such that x_1, x_2 are in a same component H of G_ℓ . By requirement (4), x_1 and x_2 are connected through D_ℓ , and thus F_1 and F_2 are also connected through $D_\ell \subseteq D$.

Let \tilde{G} be the component of $G[D]$ containing all the vertices of $D_0(a^*)$. It is to be shown that $\tilde{G} = G[D]$. Suppose this is not true, then there exists a cell Q_ℓ and a component R of $G[D_\ell]$ such that $V(R) \cap D_0(a^*) = \emptyset$ and R is not adjacent with any vertex in $D_0(a^*)$. Since D_0 is a CDS of G , there exists a vertex $x \in D_0$ which dominates some vertex $y \in V(R)$ (y may coincide with x). Since $x \notin D_0(a^*)$, one has $x \in Q_\ell \setminus B_\ell$. Hence, $y \in C_\ell$. Let H be the connected component of G_ℓ containing y . By requirement (4), it can be seen that R dominates H . Since $G[D_0]$ is connected, there is a path in $G[D_0]$ connecting x to the other parts of G outside of cell Q_ℓ . Such a path must contain a vertex $z \in D_0 \cap B_\ell \cap C_\ell \subseteq D_0(a^*)$. Note that z is also in H . Hence, there is a vertex w in $V(R)$ dominating z , contradicting that R is not adjacent with any vertex in $D_0(a^*)$. Hence, $\tilde{G} = G[D]$, and thus $G[D]$ is connected. \square

5.3 Time Complexity

It is easy to see that the most time-consuming step of [Algorithm 4](#) is the fourth step. In this section, it will be shown that exhaust search can be done in polynomial time. For this purpose, one has to show that the number of vertices chosen in Step 4 is upper bounded by a constant (depending on n and ε , where n is the number of vertices in G).

First study the relationship between the size of a dominating set and the size of a connected dominating set. For two subgraphs G_1, G_2 of G , let $dist_{\text{hop}}(G_1, G_2)$ be the hop distance between G_1 and G_2 , that is the number of edges of the shortest path connecting G_1 and G_2 in G . The next lemma is well known (see, e.g, [31]).

Lemma 2 *Let G be a connected graph and D be a dominating set of G . Then $G[D]$ has two connected components G_1 and G_2 such that $dist_{\text{hop}}(G_1, G_2) \leq 3$.*

Proof Let G_1, G_2 be two connected components of $G[D]$ such that they are the nearest to each other. Let $P = x_1x_2\dots x_k$ be a shortest path of G between G_1 and G_2 , where $x_1 \in V(G_1)$ and $x_k \in V(G_2)$. Suppose, to the contrary, that $dist_{\text{hop}}(G_1, G_2) > 3$, then $k \geq 5$. Since D is a dominating set of G , there is a vertex $y \in D$ which dominates x_3 . Since P is a shortest path between G_1 and G_2 , it can be seen that y is in a component G_3 of $G[D]$ which is different from G_1 and G_2 . But then G_3 and G_2 are nearer than G_1 and G_2 , a contradiction. \square

The following lemma is an easy consequence of [Lemma 2](#).

Lemma 3 *Let G be a connected graph and D be a dominating set of G . There exists a connected dominating set of G with at most $3|D| - 2$ vertices.*

Proof Suppose $G[D]$ has k components. By [Lemma 2](#), adding at most two vertices can reduce the number of components by one. Repeatedly merging the components like this, at most $2(k - 1)$ vertices are added to obtain a connected dominating set of G . Then the lemma follows from the observation that $k \leq |D|$. \square

Now, it is ready to show the time complexity of the algorithm.

Theorem 1 *Algorithm 4 runs in time $n^{O(1/\varepsilon^3)}$.*

Proof Clearly, the most time-consuming part is the fourth step. Since any vertex in a small cube with side length $\sqrt{3}/3$ dominates any other vertices in the same cube, it can be seen that a minimum dominating set of Q_ℓ uses at most $\lceil \sqrt{3}\ell \rceil^3$ vertices. By [Lemma 3](#), the local optimal solution $|D_\ell| \leq 3\lceil \sqrt{3}\ell \rceil^3$. Hence, the exhaust search takes time at most $\sum_{k=0}^{3\lceil \sqrt{3}\ell \rceil^3} \binom{n_\ell}{k} = n_\ell^{O(\ell^3)}$ to compute D_ℓ , where n_ℓ is the number of vertices in Q_ℓ . It follows that the total time complexity is bounded by $\sum_{Q_\ell \in P(a^*)} n_\ell^{O(\ell^3)} = n^{O(\ell^3)} = n^{O(1/\varepsilon^3)}$. \square

5.4 The Performance Ratio

To show that [Algorithm 4](#) is a PTAS for CDS in UBG, one has to compare the size of an optimal solution D^* with the size of the output D of the algorithm.

5.4.1 Estimates for the Boundary Nodes

Similarly to the previous problems, shifting strategy yields

$$|D_0(a^*)| \leq \frac{\varepsilon}{25} |D^*|. \quad (5)$$

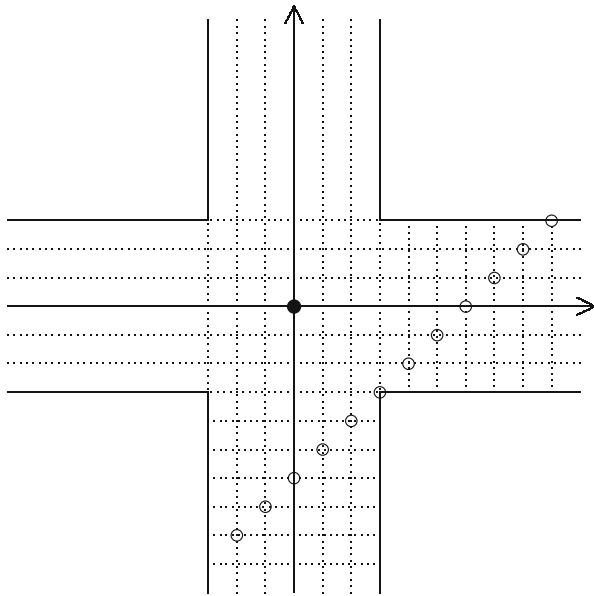
In fact, when a runs over $0, 1, \dots, \ell - 1$, each vertex belongs to at most 12 boundary regions of the $P(a)$'s (see [Fig. 1](#)). Hence,

$$|D_0(0)| + |D_0(1)| + \dots + |D_0(\ell - 1)| \leq 12|D_0|.$$

Since D_0 is a ρ -approximation for D^* , one has

$$|D_0(a^*)| \leq \frac{12}{\ell} |D_0| \leq \frac{12\rho}{\ell} |D^*| \leq \frac{\varepsilon}{25} |D^*|.$$

Fig. 1 When the partition shifts, each vertex falls into at most 12 boundary regions



5.4.2 Modification of Local Solution

Notice that the restriction of D^* to a cell Q_ℓ , denoted by D_ℓ^* , is a dominating set of G_ℓ . In fact, each vertex $x \in V(G_\ell) \setminus D^*$ is dominated by a vertex $z \in D^*$. Since x is in the inner region of Q_ℓ , it can be seen that vertex z is also in Q_ℓ and thus in D_ℓ^* .

However, D_ℓ^* is not necessarily a local feasible solution: a connected component of G_ℓ might be jointly dominated by several connected components of $G[D_\ell^*]$. Thus, to compare $|D_\ell^*|$ with $|D_\ell|$, where D_ℓ is the local optimal solution computed in Step 4 of the algorithm, one has to modify D_ℓ^* to another set \tilde{D}_ℓ such that \tilde{D}_ℓ is a local feasible solution (and thus $|D_\ell| \leq |\tilde{D}_\ell|$). To ensure the performance ratio $1 + \varepsilon$, the modification must be so slight that $|\tilde{D}_\ell|$ is larger than $|D_\ell^*|$ by only a very small amount.

If a connected component H of G_ℓ is jointly dominated by more than one connected components of $G[D_\ell^*]$, a natural way for the modification is to add some nodes to merge the components into one. The following lemma provides us an upper bound for the number of nodes that need to be added.

Let H be a subgraph of G . For two subgraphs R_1 and R_2 of G , the distance between R_1 and R_2 in H , denoted by $dist_H(R_1, R_2)$, is the length of the shortest path connecting R_1 and R_2 in H . In other words, if $dist_H(R_1, R_2) = d$, then R_1 and R_2 can be connected through at most $d - 1$ vertices of H .

Lemma 4 *Let H be a connected subgraph of G , and D be a subset of $V(G)$ dominating H . If H is not dominated by only one connected component of $G[D]$, then there exist two components R_1 and R_2 of $G[D]$ such that $dist_H(R_1, R_2) \leq 3$.*

Proof Let R_1, R_2, \dots, R_k ($k \geq 2$) be a minimum set of components of $G[D]$, the union of which dominates H . The *minimality* ensures that every R_i is adjacent with H . Since H is connected, it can be seen that $\bigcup_{i=1}^k R_i$ can be connected through vertices in H . Choose i and j such that $\text{dist}_H(R_i, R_j)$ is the minimum. Let $P = x_1 x_2 \dots x_t$ be the shortest path connecting R_i and R_j such that the inner vertices $x_2, \dots, x_{t-1} \in V(H) \setminus \bigcup_{i=1}^k V(R_i)$ and the end vertices $x_1 \in V(R_i), x_t \in V(R_j)$. Suppose $t \geq 5$. Let R_ℓ be a component dominating x_3 . Then, similarly to the proof of [Lemma 2](#), it can be seen that $\ell \neq i, j$ and $\text{dist}_H(R_\ell, R_j) < \text{dist}_H(R_i, R_j)$, contradicting the choice of i and j . \square

Suppose a component H of G_ℓ is jointly dominated by k components of $G[D_\ell^*]$. By [Lemma 4](#), it can be seen that at most $2k$ nodes are needed in order that requirement (4) is satisfied for H .

5.4.3 Compensation for The Added Nodes

Next, an upper bound is given for the number of added nodes in terms of ε . The idea is to use the nodes in $D_0(a^*) \cap Q_\ell$ to *compensate* for the nodes added for merging. By (5), the compensation can be expected to be small.

The compensation is *not one-to-one*, that is one node in $D_0(a^*) \cap Q_\ell$ might be used to compensate for more than one added nodes. However, the repetition is bounded by a constant which can be guaranteed by the result of the famous Gregory-Newton problem concerning about kissing number. The *kissing number* is the maximum number of unit balls that can simultaneously touch the surface of a unit ball (“touch” means two balls have exactly one point in common). It is known that the kissing number is 12 (see [[65](#)]).

Lemma 5 *For any vertex u in a unit ball graph G , the neighborhood $N_G(u)$ contains at most 12 independent vertices.*

Proof Let $S(u)$ be the unit ball with center u and $\{x_1, \dots, x_t\}$ be the maximum set of independent vertices in $S(u)$. For each $i = 1, \dots, t$, draw a radial r_i with origin u which goes through x_i . Suppose r_i intersects the surface of $S(u)$ at point \tilde{x}_i . Let S_i be a unit ball touching $S(u)$ at \tilde{x}_i . Since x_1, \dots, x_t are independent, the angle between any two radials is at least $\pi/3$. Hence, S_1, \dots, S_t are nonintersecting. It follows that t is at most the kissing number, which is 12. \square

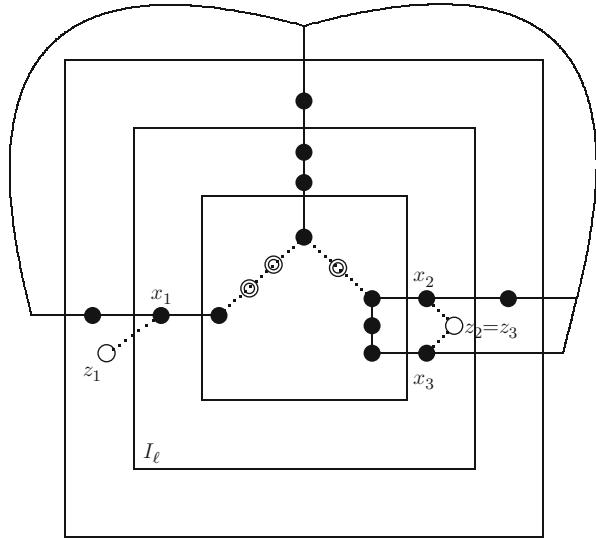
As a consequence of [Lemma 5](#), one has

Corollary 1 *Let G be a unit ball graph, D be a dominating set of G , and I be an independent set of G . Then $|I| \leq 12|D|$.*

Suppose a component H of G_ℓ is jointly dominated by k components of $G[D_\ell^*]$, say R_1, R_2, \dots, R_k . Then at most $2k$ nodes are added to merge them into one big component.

The following analysis is illustrated in [Fig. 2](#).

Fig. 2 An illustration for the compensation. The *solid nodes* are in D_ℓ^* , they are connected through D^* from the outside of the cell Q_ℓ . The *double-cycle nodes* are the ones added for merging. The *small circles* are the nodes used for compensation. Notice that x_2 and x_3 are compensated by a same node



Suppose that the components are merged in the order that R_1 is merged with R_2 , R_3 is merged with R_4, \dots, R_{2k-1} is merged with R_{2k} . To simplify the presentation of the idea, it is first assumed that the R_i 's are all distinct components of the original $G[D_\ell^*]$.

Denote by I_ℓ the region of Q_ℓ between distance 1 and 2 from the boundary of Q_ℓ . For each $i = 1, 2, \dots, k$, let x_i be a vertex in $V(R_{2i-1}) \cap I_\ell$. Notice that such x_i exists since R_{2i-1} dominates some vertex in H which is a component in the inner region of Q_ℓ (hence, R_{2i-1} is within distance 1 from the inner region), and $G[D^*]$ is connected (hence, R_{2i-1} is accessible from the outer side of Q_ℓ). Because D_0 is a dominating set of G , there is a vertex $z_i \in D_0$ dominating x_i . Since $x_i \in I_\ell$, one has $z_i \in B_\ell$, and thus $z_i \in D_0(a^*) \cap Q_\ell$. Note that for $i \neq j$, it is possible that $z_i = z_j$. However, in this case,

$$\begin{aligned} x_i \text{ and } x_j \text{ are independent} \\ \text{since they are in different components of } G[D_\ell^*]. \end{aligned} \tag{6}$$

Hence, by Lemma 5, a vertex serves at most 12 times as z_i 's. Thus, it has been shown that

$$k \leq 12|D_0(a^*) \cap Q_\ell|, \tag{7}$$

and thus the number of nodes added to merge R_1, R_2, \dots, R_k is at most $24|D_0(a^*) \cap Q_\ell|$.

Next, consider the case that there are some repetitions among the R_i 's. For example, suppose R_3 is the component of the new $G[\tilde{D}_\ell^*]$ obtained by merging R_1 and R_2 . Since x_1 is chosen to be in $V(R_1) \cap I_\ell$, one can choose $x_3 \in V(R_2) \cap I_\ell$.

In general, one is always able to choose x_i 's such that they are in different components of the original $G[D_\ell^*]$. Hence, (7) holds in any case.

Let H_1, H_2, \dots, H_t be the connected components of G_ℓ . Execute the above procedure for each H_j ($j = 1, 2, \dots, t$). Notice that a vertex z in $D_0(a^*) \cap Q_\ell$ might be used for compensation for different H_j 's. However, a similar observation as in (6) shows that the total number of times that z is used is still upper bounded by 12. Hence, one has the following:

Lemma 6 *One can modify D_ℓ^* into a local feasible solution \tilde{D}_ℓ satisfying (4) such that $|\tilde{D}_\ell| \leq |D_\ell^*| + 24|D_0(a^*) \cap Q_\ell|$.*

5.4.4 Analysis of the Performance Ratio

Now, it is ready to show that [Algorithm 4](#) is a PTAS.

Theorem 2 *[Algorithm 4](#) is a $(1 + \varepsilon)$ -approximation for CDS in UBG.*

Proof Let D^* be an optimal CDS of G , and let $\tilde{D} = \bigcup_{Q_\ell} \tilde{D}_\ell$, where \tilde{D}_ℓ is the modification of D_ℓ^* as in [Lemma 6](#). Since \tilde{D}_ℓ is a local feasible solution for Q_ℓ , one has

$$|D_\ell| \leq |\tilde{D}_\ell|,$$

where D_ℓ is the local optimal solution found in the fourth step. Then it follows from [Lemma 6](#) and inequality (5) that

$$\begin{aligned} \left| \bigcup_{Q_\ell \in P(a^*)} D_\ell \right| &= \sum_{Q_\ell \in P(a^*)} |D_\ell| \leq \sum_{Q_\ell \in P(a^*)} |\tilde{D}_\ell| \\ &\leq \sum_{Q_\ell \in P(a^*)} (|D_\ell^*| + 24|D_0(a^*) \cap Q_\ell|) \\ &= |D^*| + 24|D_0(a^*)| \leq \left(1 + \frac{24\varepsilon}{25}\right) |D^*. \end{aligned} \quad (8)$$

Combining inequalities (5) and (8), one has

$$|D| \leq \left| \bigcup_{Q_\ell \in P(a^*)} D_\ell \right| + |D_0(a^*)| \leq (1 + \varepsilon) |D^*|,$$

where D is the output of the algorithm. This proves the theorem. □

6 Multilayer Partition

The MIS problem in intersection graphs has interesting applications in map labeling and data mining [1, 9]. When the sizes of the objects do not differ too much, a slight modification of the above partition and shifting strategy will do, the only

difference is that the running time depends on D_{\max}/D_{\min} , where D_{\max} and D_{\min} are the sizes of the largest object and the smallest object, respectively. However, if D_{\max}/D_{\min} can be arbitrarily large, new ideas have to be implemented.

This section first illustrates the idea of multilayer partition by the MIS problem for the intersection ball graph, then introduces the idea used by Lev-Tov and Peleg in solving the MSRC problem.

6.1 Maximum Independent Set in Ball Graph

In an *intersection graph* G , every vertex corresponds to a geometric object in the space, and two vertices are adjacent in G if and only if the two objects corresponding to them intersect. If for an intersection graph, all the objects are balls, then it is an *intersection ball graph*, or one simply calls it a *ball graph* in this section.

6.1.1 The Partition

In the multilayer partition, both the region and the balls are partitioned into layers.

First, the balls are scaled such that the maximum one has diameter *a little less* than 1. Let d_{\min} be the minimum diameter after the scaling. For a fixed constant ℓ , set $j_0 = \lceil \log_{\ell+1}^{1/r_{\min}} \rceil - 1$. For each $0 \leq j \leq j_0$, the *layer j balls* (or simply, j -balls) are those with diameter $d \in \left[\frac{1}{(\ell+1)^{j+1}}, \frac{1}{(\ell+1)^j} \right)$. Then each ball belongs to exactly one layer.

Next, the region containing all the centers of the balls is partitioned as follows (refer to Fig. 3a for an illustration): similarly to Sect. 3.3, let Q be the region enclosing all the centers with side length L , and let \tilde{Q} be the enlarged region of

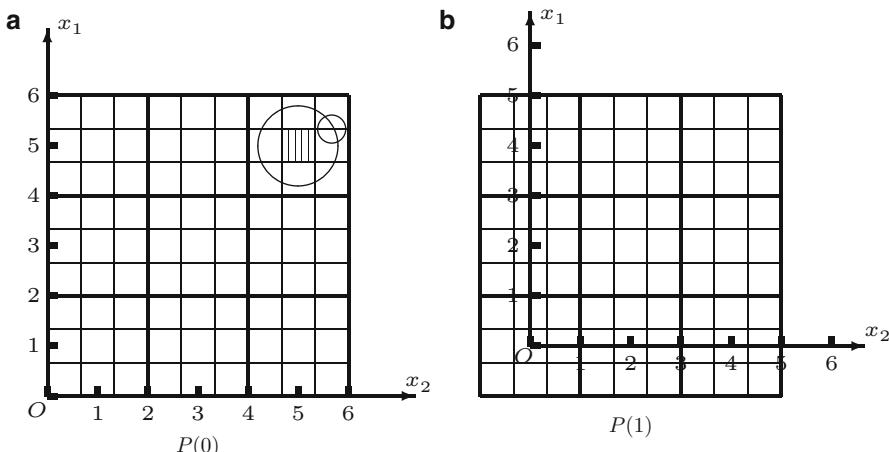


Fig. 3 Multilayer partition with $k = 2$ and $\ell = 1$. The *heavy lines* indicate the 0-planes and the *light lines* indicate the 1-planes

Q with side length $(q+1)\ell$, where $q = \lceil L/\ell \rceil$. For each $0 \leq j \leq j_0$, divide \tilde{Q} into $(q+1)(\ell+1)^j \times (q+1)(\ell+1)^j \times (q+1)(\ell+1)^j$ smaller cubes with side length $\ell/(\ell+1)^j$. Each smaller cube is called a j -cell, and the planes dividing j -cells are called j -planes. An important observation is that a j -plane is also a $(j+1)$ -plane. In fact, suppose a $(j+1)$ -plane has x_1 -coordinate $k \cdot \ell/(\ell+1)^{j+1}$, since $k \cdot \ell/(\ell+1)^{j+1} = k(\ell+1) \cdot \ell/(\ell+1)^j$, it is also a j -plane. This multilayered partition is denoted as $P(0)$. For $a = 0, 1, \dots, \ell-1$, the multilayer partition $P(a)$ is obtained from $P(0)$ by shifting it to the position such that the left-bottom-back corner is located at $(-a, -a, -a)$.

6.1.2 A Reduced Problem

A j -ball is called a *hitting ball* if it hits some j -planes (by hitting, the ball either intersects or touches the j -plane). It should be emphasized that one only considers those balls hitting the planes of their own levels. A j -ball which hits a j' -plane for $j' \neq j$ is not regarded as hitting the plane. It should also be noted that whether a ball is a hitting ball depends on the partition $P(a)$. When a shifts, since the maximum ball has diameter a little less than 1, it can be seen that every ball serves as a hitting ball at most three times (see Fig. 4).

Let $\mathcal{B}(a)$ be the set of balls obtained from \mathcal{B} by removing all the hitting balls with respect to the partition $P(a)$. The *reduced problem* is to find a maximum independent set for $\mathcal{B}(a)$. The idea is that the removal of hitting balls reduces the interlacing of balls at the boundaries, and thus the reduced problem can be solved polynomially by piecing up optimal local solutions. Simply piling them up does not do. Dynamic programming has to be used.

Lemma 7 *A maximum independent set of $\mathcal{B}(a)$ can be found in time $n^{O(\ell^6)}$ using dynamic programming.*

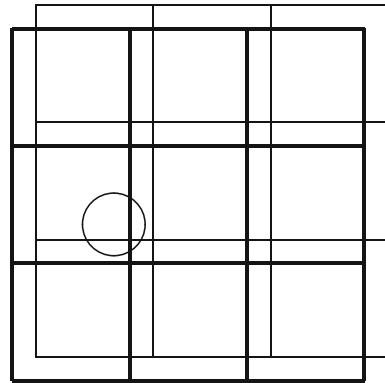
Proof A j -cell is said to be *relevant* if it contains a j -ball. For example, in Fig. 3a, the right-top 1-cell is a relevant 1-cell, but the shaded one is not. Though the shaded 1-cell contains a 0-ball, it does not contain any 1-ball.

Suppose $j' > j$ and a relevant j' -cell e' is contained in a relevant j -cell e . If for any j'' with $j < j'' < j'$, there is no relevant j'' -cell containing e' which is contained in e , then e' is called a *child* of e and e as a *parent* of e' . A *maximal relevant cell* is a relevant cell which has no parent.

For any index $j \in \{0, 1, \dots, j_0\}$ and any relevant j -cell e , let I be a set of independent balls hitting e whose levels are $< j$. Define $T(e, I)$ to be a *maximum* set of independent balls with levels $\geq j$ which are contained in e and independent with I . It should be noticed that since all the hitting balls are removed, every ball with level $\geq j$ which has nonempty intersection with e is completely contained in e . Clearly, the optimal solution to the reduced problem is

$$OPT = \bigcup_e T(e, \emptyset),$$

Fig. 4 In a d -dimensional space, every ball serves as a hitting ball at most d times. This figure is an illustration for $d = 2$



where the maximum is taken over all the maximal relevant cells e .

The element $T(e, I)$ can be calculated using the following recurrence relation:

$$T(e, I) = \arg \max_{I'} \left\{ \left| I' \cup \bigcup_{e'} T(e', (I \cup I')|_{e'}) \right|_{e'} \right\}, \quad (9)$$

where the set I' is taken over all the sets of independent balls with level j , the cell e' is taken over all the relevant children of e , and $(I \cup I')|_{e'}$ is the restriction of $I \cup I'$ to the cell e' , that is the set of independent balls in $I \cup I'$ which hit e' . The recurrence relation (11) is used bottom-up, that is, for $j = j_0, j_0 - 1, \dots, 0$. For a j_0 -cell, its children are empty.

Next it will be shown that the table $\{T(e, I)\}$ can be calculated in polynomial time depending on ℓ .

For each relevant j -cell e , using the assumption that a j -ball has radius at least $1/2(\ell + 1)^{j+1}$, one has

$$|I'| \leq \frac{\left(\frac{\ell}{(\ell+1)^j}\right)^3}{\frac{4}{3}\pi \left(\frac{1}{2(\ell+1)^{j+1}}\right)^3} = O(\ell^6).$$

Hence, for each fixed e , the number of choices for I' is at most $n^{O(\ell^6)}$.

To show that $|I|$ is upper bounded for each j -cell e , enlarge e on each of its six sides by $1/(\ell + 1)^j$. Denote the enlarged cube by e_e . Notice that for any $j' < j$, any j' -ball hitting e occupies at least $\frac{4}{3}\pi \left(\frac{1}{2(\ell+1)^{j'}}\right)^3$ volume of e_e . Since I is independent, one has

$$|I| \leq \frac{\left(\frac{\ell+2}{(\ell+1)^j}\right)^3}{\frac{4}{3}\pi \left(\frac{1}{2(\ell+1)^j}\right)^3} = O(\ell^3).$$

Hence, for each fixed e , there are at most $n^{O(\ell^3)}$ choices for I .

Then it follows that the size of the table $\{T(e, I)\}$ is bounded above by $n \cdot n^{O(\ell^3)} = n^{O(\ell^3)}$, and the total time for the dynamic programming is upper bounded by $n^{O(\ell^6)} \cdot n^{O(\ell^3)} = n^{O(\ell^6)}$. \square

6.1.3 The Algorithm and the Performance

Algorithm 5 uses the optimal solutions for the reduced problems to compute an independent set for the original ball graph G .

Clearly, the running time of **Algorithm 5** is $\ell n^{O(\ell^6)}$.

To show that the performance ratio is $(1 + \varepsilon)$, let I^* be a maximum independent set of G . For each $a = 0, 1, \dots, \ell - 1$, let $I_h^*(a)$ be the set of balls of I^* which are hitting balls with respect to $P(a)$. Since each ball serves as a hitting ball at most three times, one has

$$|I_h^*(0)| + |I_h^*(1)| + \dots + |I_h^*(\ell - 1)| \leq 3|I^*|,$$

and thus there is an index \tilde{a} such that

$$|I_h^*(\tilde{a})| \leq \frac{3}{\ell}|I^*|.$$

Since $I^* \setminus I_h^*(\tilde{a})$ is an independent set for $\mathcal{B}(\tilde{a})$, one has

$$|I| = |I(a^*)| \geq |I(\tilde{a})| \geq |I^*| - |I_h^*(\tilde{a})| \geq \left(1 - \frac{3}{\ell}\right)|I^*|,$$

and thus

$$|I^*| \leq \frac{\ell}{\ell - 3}|I| \leq \varepsilon|I|.$$

Algorithm 5 PTAS for MIS in Ball Graph

Input: A ball graph G and a positive real number $\varepsilon < 1$.

Output: An independent set I of G .

- 1: Set $\ell = \lceil \frac{3}{\varepsilon} + 3 \rceil$.
 - 2: **for** $a = 0, 1, \dots, \ell - 1$ **do**
 - 3: Find a maximum independent set $I(a)$ for $\mathcal{B}(a)$
 - 4: **end for**
 - 5: Let a^* be the index with $|I(a^*)| = \max\{|I(a)| : a = 0, 1, \dots, \ell - 1\}$.
 - 6: Output $I = I(a^*)$.
-

6.2 Minimum Sum Radii Cover

In an MSRC problem, one is given a set \mathcal{S} of servers and a set \mathcal{C} of clients. Each server $s \in \mathcal{S}$ is assigned a transmission range $r(s)$. A client $c \in \mathcal{C}$ can be served by $s \in \mathcal{S}$ if c falls into the transmission range of s . The goal is to find a subset $\mathcal{S}' \subseteq \mathcal{S}$ and an assignment of transmission ranges to the servers in \mathcal{S}' , such that every client is served and the total transmission ranges (which corresponds to the total energy consumed) are minimized. Although the transmission range can be chosen for a given server, in any optimal solution, for each chosen radius $r(s)$, there must be a client on the boundary of the ball with center s and radius $r(s)$. Hence, the problem is equivalent to the following: for each $s \in \mathcal{S}$ and each $c \in \mathcal{C}$, let B_s^c be the ball with center s and radius $dist(s, c)$, where $dist(s, c)$ is the Euclidean distance between s and c . A weight $w(B_s^c) = dist(s, c)$ is assigned to the ball B_s^c . Let $\mathcal{B} = \{B_s^c : s \in \mathcal{S}, c \in \mathcal{C}\}$. The goal is to select a subset $\mathcal{B}^* \subseteq \mathcal{B}$ such that \mathcal{B}^* covers all the clients in \mathcal{C} and the total weight $w(\mathcal{B}^*) = \sum_{B \in \mathcal{B}^*} w(B)$ is minimized.

The key idea of Lev-Tov and Peleg's method [44] is to cover the clients by fragments instead of whole balls. The key observation to the running time is that there exists an optimal solution owning some disjoint property.

6.2.1 Semi-disjointness

Two balls $B_{s_1}^{c_1}$ and $B_{s_2}^{c_2}$ are said to be *semi-disjoint* if neither $s_2 \in B_{s_1}^{c_1}$ nor $s_1 \in B_{s_2}^{c_2}$. A set of balls \mathcal{B}' is semi-disjoint if any two balls in \mathcal{B}' are semi-disjoint.

Lemma 8 *There exists an optimal solution to the MSRC problem which is semi-disjoint.*

Proof Let \mathcal{B}^* be an optimal solution to the MSRC problem and $B_{s_1}^{c_1}$ and $B_{s_2}^{c_2}$ be two balls in \mathcal{B}^* with $dist(s_1, c_1) \geq dist(s_2, c_2)$. Suppose they are not semi-disjoint, then $s_2 \in B_{s_1}^{c_1}$ (see Fig. 5). If $B_{s_2}^{c_2} \setminus B_{s_1}^{c_1}$ has no clients, then simply removing $B_{s_2}^{c_2}$ from \mathcal{B}^* will result in a cover with less weight, which is a contradiction. Hence, $B_{s_2}^{c_2} \setminus B_{s_1}^{c_1}$ contains a client. Choose c_0 to be such a client with $dist(s_1, c_0) = \max\{dist(s_1, c) : c \in B_{s_2}^{c_2}\}$. Then the ball $B_{s_1}^{c_0}$ contains all the clients in $B_{s_1}^{c_1} \cup B_{s_2}^{c_2}$. Notice that

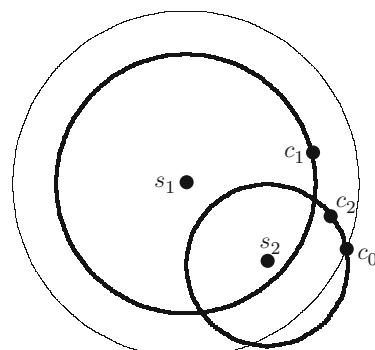


Fig. 5 An illustration for the semi-disjointness of the optimal solutions for the MSRC problem

$dist(s_1, c_0) \leq dist(s_1, c_1) + dist(s_2, c_2)$; thus, replacing $B_{s_1}^{c_1}$ and $B_{s_2}^{c_2}$ by $B_{s_1}^{c_0}$ will result in a cover whose weight is not larger than $w(\mathcal{B}^*)$. \square

The semi-disjointness is crucial in controlling the size of the table of the dynamic programming.

6.2.2 Fragment Covering

Unlike the MIS problem, to solve the MSRC problem, hitting balls cannot be removed. For example, the removal of a large hitting ball might greatly increase the number of balls in the solution. The key new idea is to use fragments instead of whole balls to cover all the clients.

The partition is the same as that in the above section. When a j -ball intersects some j -planes (by intersecting, the ball has more than one common points with the plane, touching is not regarded as intersecting), it is divided into *fragments*. It should be emphasized that one only considers those balls divided by the planes of their own levels. A j -ball which intersects a j' -plane for $j' \neq j$ is not regarded as being divided by the plane. Since the diameter of a j -ball is less than the spacing between j -planes, it can be seen that

$$\text{every ball corresponds to at most eight fragments.} \quad (10)$$

The idea is to cover the clients with these fragments together with those balls which fall completely in the cells of their own levels, such that the total weight is minimum. Such a problem is called a *fragment covering problem*. For simplicity of statement, a ball which falls completely into a cell of its own level is also referred to as a fragment. Clearly, those fragments which do not contain any client is useless in the covering; hence, in the following, when the term fragment is used, it is always referring to a fragment containing some clients. The advantage is that such a covering can be found in polynomial time using dynamic programming.

For a fragment F , denote by B_F the ball corresponding to F . For a set \mathcal{F} of fragments, let $\mathcal{B}_{\mathcal{F}} = \{B_F : F \in \mathcal{F}\}$ be the set of balls corresponding to the fragments in \mathcal{F} . A fragment F is said to be of level j if B_F is of level j , and a set \mathcal{F} of fragments is said to be semi-disjoint if $\mathcal{B}_{\mathcal{F}}$ is semi-disjoint.

Lemma 9 *The fragment covering problem can be solved in time $n^{O(\ell^6)}$ using dynamic programming.*

Proof For any index $j \in \{0, 1, \dots, j_0\}$ and any relevant j -cell e , denote by S_e the set of clients in e whose levels are $\geq j$. Let \mathcal{F} be a semi-disjoint set of fragments with levels $< j$ such that each fragment in \mathcal{F} contains some client in e . Denote by $S_{e,\mathcal{F}}$ the subset of clients of S_e which are not covered by the fragments in \mathcal{F} . Define $T(e, \mathcal{F})$ to be a set of semi-disjoint fragments with levels $\geq j$ which covers $S_{e,\mathcal{F}}$ such that the total weight is minimum. If such a covering does not exist, define $T(e, \mathcal{F})$ to be null and its value to be infinity. Clearly, every element of $T(e, \mathcal{F})$ is completely contained in e , $T(e, \mathcal{F}) \cup \mathcal{F}$ is a fragment covering of clients in e , and

the optimal fragment covering $OPT = \bigcup_e T(e, \emptyset)$, where the union is taken over all the maximal relevant cells e . The element $T(e, \mathcal{F})$ can be calculated using the following recurrence relation:

$$T(e, \mathcal{F}) = \arg \min_{\mathcal{F}'} \left\{ w \left(\mathcal{F}' \cup \bigcup_{e'} T(e', (\mathcal{F} \cup \mathcal{F}')|_{e'}) \right) \right\}, \quad (11)$$

where \mathcal{F}' is taken over all the set of semi-disjoint fragments with level j which is also semi-disjoint from \mathcal{F} , e' is taken over all the relevant children of e , and $(\mathcal{F} \cup \mathcal{F}')|_{e'}$ is the restriction of $\mathcal{F} \cup \mathcal{F}'$ to the cell e' , that is the fragments in $\mathcal{F} \cup \mathcal{F}'$ which contain some clients in e' .

Next, it will be shown that the table $T = \{T(e, \mathcal{F})\}$ can be calculated in polynomial time depending on ℓ . For this purpose, the following two claims are proved first. Let e be a relevant j -cell and $\mathcal{F}, \mathcal{F}'$ be the sets of semi-disjoint fragments as in the recurrence relation (11). Notice that no two fragments of \mathcal{F} and \mathcal{F}' can be induced by a same ball. Hence, to show the upper bounds for $|\mathcal{F}|$ and $|\mathcal{F}'|$, it suffices to show the upper bounds for $|\mathcal{B}_{\mathcal{F}}|$ and $|\mathcal{B}_{\mathcal{F}'}|$.

Claim 1 $|\mathcal{F}'| = |\mathcal{B}_{\mathcal{F}'}| \leq O(\ell^6)$.

Enlarge e on each of its six sides by $1/2(\ell+1)^j$. Denote the enlarged cube by e_e . Then e_e has side length $1/(\ell+1)^{j-1}$.

Recall that every ball $B \in \mathcal{B}_{\mathcal{F}'}$ has its diameter in $[\frac{1}{(\ell+1)^{j+1}}, \frac{1}{(\ell+1)^j}]$. Since B contains some client in e , its center is in e_e . Furthermore, for any two balls $B_1, B_2 \in \mathcal{B}_{\mathcal{F}'}$, by the semi-disjointness of \mathcal{F}' , the distance between the centers of B_1 and B_2 is larger than $1/2(\ell+1)^{j+1}$. Divide e_e into smaller cubes of side length $\frac{1}{2\sqrt{3}(\ell+1)^{j+1}}$ (with the exception of side strips whose side length might be smaller if the side length of e_e is not an integral multiplicity of $\frac{1}{2\sqrt{3}(\ell+1)^{j+1}}$). Then any point in a same small cube has distance at most $1/2(\ell+1)^{j+1}$, and thus each small cube contains at most one center of a ball in $\mathcal{B}_{\mathcal{F}'}$. It follows that

$$|\mathcal{B}_{\mathcal{F}'}| \leq \left(\frac{\frac{1}{(\ell+1)^{j-1}}}{\frac{1}{2\sqrt{3}(\ell+1)^{j+1}}} \right)^3 = O(\ell^6). \quad (12)$$

Claim 2 $|\mathcal{F}| = |\mathcal{B}_{\mathcal{F}}| \leq O(\ell^3)$.

Notice that for every ball $B \in \mathcal{B}_{\mathcal{F}}$, its diameter is at least $1/(\ell+1)^j$. Similarly to the above, it can be seen that the number of balls in $\mathcal{B}_{\mathcal{F}}$ whose centers belong to e_e is at most $O(\ell^3)$ (the difference is that the denominator of (12) is changed to $\frac{1}{2\sqrt{3}(\ell+1)^j}$).

Next, consider the balls of $\mathcal{B}_{\mathcal{F}}$ whose centers lie outside of e_e . In order that the idea can be understood easier, the analysis will first be illustrated on

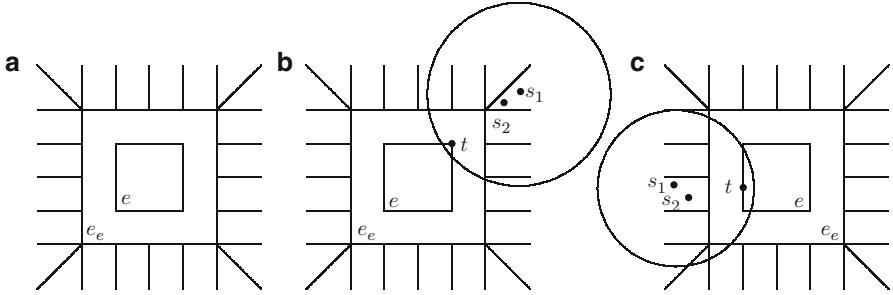


Fig. 6 An illustration for the division of the space outside of e_e

two-dimensional case. Divide the square outside of e_e into subregions as shown in Fig. 6a, where any two consecutive parallel lines have distance $1/2(\ell+1)^j$. It will be shown that each subregion R contains at most one center of a disk in \mathcal{B}_F . Suppose this is not true, let B_1, B_2 be two disks in \mathcal{B}_F whose centers s_1, s_2 are both in R . Suppose $dist(s_1, e) \geq dist(s_2, e)$, where $dist(s, e) = \min\{dist(s, s'): s' \in e\}$ is the distance between a point s and the square e . Since the ball B_{s_1} which corresponds to s_1 contains a client in e , its radius is at least as large as the length of s_1t in Fig. 6b, c. Then it can be seen that B_{s_1} contains s_2 , contradicting the assumption that B_{s_1} and B_{s_2} are semi-disjoint (the observation that B_{s_1} has radius $\geq 1/2(\ell+1)^j \geq$ the distance between any two consecutive parallel lines is used in considering the case of Fig. 6c).

For the three-dimensional case, the argument is almost the same except that the distance between any two consecutive parallel planes should be $1/2\sqrt{2}(\ell+1)^j$. Then, it can be seen that

$$|\mathcal{B}_F| \leq 6 \left(\frac{\frac{1}{(\ell+1)^{j-1}}}{\frac{1}{2\sqrt{2}(\ell+1)^j}} \right)^2 + 12 \left(\frac{\frac{1}{(\ell+1)^{j-1}}}{\frac{1}{2\sqrt{2}(\ell+1)^j}} \right) = O(\ell^2).$$

Then, Claim 2 follows by summing up the above analysis.

By Claim 1 and Claim 2, it can be seen that the running time for the dynamic programming is at most $n \cdot n^{O(\ell^3)} \cdot n^{O(\ell^6)} = n^{O(\ell^6)}$. \square

6.2.3 The Algorithm

Algorithm 6 uses the optimal solutions for the fragment coverings to obtain a solution to the MSRC problem.

6.2.4 The Performance Ratio

When a ball is used more than once, that is, when t fragments of a ball are used for covering, the ball is counted t times, which incurs some repetition. However, it can

Algorithm 6 PTAS for MSRC

Input: A set of servers \mathcal{S} , a set of clients \mathcal{C} , and a positive real number $\varepsilon < 1$.

Output: A solution \mathcal{B}' to the MSRC problem.

- 1: Determine the set $\mathcal{B} = \{B_s^c : s \in \mathcal{S}, c \in \mathcal{C}\}$ of possible balls.
 - 2: Set $\ell = \lceil 7/\varepsilon \rceil$.
 - 3: **for** $a = 0, 1, \dots, \ell - 1$ **do**
 - 4: Find a minimum weight fragment cover $\mathcal{F}(a)$ for the partition $P(a)$.
 - 5: **end for**
 - 6: Let a^* be the index with $w(\mathcal{F}(a^*)) = \min\{w(\mathcal{F}(a)) : a = 0, 1, \dots, \ell - 1\}$.
 - 7: Output $\mathcal{B}' = \mathcal{B}_{\mathcal{F}(a^*)}$.
-

be shown that by using the shifting strategy, the repetition can be controlled within a small portion of the optimal solution.

Let \mathcal{B}^* be a minimum ball cover. By the observation (10), the number of fragments resulted from \mathcal{B}^* is at most $8|\mathcal{B}^*|$ for all the partitions. Since there are ℓ partitions, there exists a partition $P(\tilde{a})$ such that the number of fragments resulted from \mathcal{B}^* for $P(\tilde{a})$ is at most $8|\mathcal{B}^*|/\ell$. To speak it more concretely, let \mathcal{B}_I^* be the set of the intersecting balls of \mathcal{B}^* with respect to $P(\tilde{a})$, and let $\mathcal{B}_B^* = \mathcal{B}^* \setminus \mathcal{B}_I^*$. Let \mathcal{F}^* be the set of fragments resulted from \mathcal{B}_I^* . Then $|\mathcal{F}^*| \leq 8|\mathcal{B}^*|/\ell$. It is clear that $\mathcal{F}^* \cup \mathcal{B}_B^*$ is a fragment covering for the partition $P(\tilde{a})$. Thus,

$$w(\mathcal{B}') \leq w(\mathcal{F}(a^*)) \leq w(\mathcal{F}(\tilde{a})) \leq w(\mathcal{F}^* \cup \mathcal{B}_B^*) \leq \left(1 + \frac{7}{\ell}\right) w(\mathcal{B}^*) \leq (1 + \varepsilon) w(\mathcal{B}^*).$$

7 Growing Partition

This section introduces the idea of growing partition which was first proposed by Nieberg et al. [51] to give a PTAS for the MIS problem in unit disk graphs. The method has the advantage that it does not require the geometric representation of the graph (recall that the recognition of a unit disk graph is NP-hard [12]). In fact, their algorithm is *robust* in the sense that it either gives a $(1 + \varepsilon)$ -approximation if the graph is a unit disk graph or gives a certificate that the given graph does not belong to the class of unit disk graphs.

Nieberg and Hurink [52] also implemented the method on the MDS problem in unit disk graphs. Gfeller and Vicari [29] generalized it to the MCDS problem for growth-bounded graphs. A graph G is *growth bounded* if there is a polynomial bounding function $f(r)$ such that for any integer $r \geq 0$, the size of any independent set in any r -closed neighborhood is upper bounded by $f(r)$. The implementation of the method on the d -hop CDS problem for growth-bounded graphs was made by Gao et al. [27].

This section illustrates the ideas of growing partition by MIS, MDS, and MCDS problems on unit ball graphs.

7.1 Maximum Independent Set

For a vertex $u \in V(G)$ and a nonnegative integer r , the closed r -neighborhood of u is $N^r[u] = \{v \in V(G) \mid \text{dist}_G(u, v) \leq r\}$. For a vertex subset $U \subseteq V(G)$, $N^r[U] = \bigcup_{u \in U} N^r[u]$ is the closed r -neighborhood of U . The subgraph of G induced by the vertex subset U is denoted by $G[U]$. The maximum independent set of $G[U]$ is denote by $\text{MIS}(U)$.

7.1.1 The Algorithm

The algorithm starts with an arbitrary vertex $u \in V(G)$, computes the minimum integer r such that $|\text{MIS}(N^{r+1}(u))| \leq (1 + \varepsilon)|\text{MIS}(N^r(u))|$. Remove $N^{r+1}[u]$ from the vertex set, and repeat this procedure in the remaining graph until there is no vertex left. The output is the union of $\text{MIS}(N^r[u])$ found in the iterations. The details are described in [Algorithm 7](#).

Suppose the algorithm iterates K times, and the vertex u and the integer r found in the i th iteration are u_i and r_i , respectively. Then $V(G)$ is partitioned into $N^{r_1+1}[u_1] \cup \dots \cup N^{r_K+1}[u_K]$. It should be noted that $N^{r_i+1}[u_i]$ refers to the $(r_i + 1)$ -closed neighborhood of u_i in the graph $G - \bigcup_{j=1}^{i-1} N^{r_j+1}[u_j]$ instead of G . Notice that the sets $\text{MIS}(N^{r_1}[u_1]), \dots, \text{MIS}(N^{r_K}[u_K])$ are mutually disjoint, since $N(N^{r_i}[u_i]) \subseteq N^{r_i+1}[u_i]$, which is disjoint from $N^{r_j+1}[u_j]$ for $i \neq j$. Hence, the output I is indeed an independent set.

The essence of the algorithm is still partitioning. The difference is that the partition is not predetermined as in the previous sections, it is obtained dynamically while the algorithm is being executed. The subgraphs resulted from the partition are $\{G[N^{r_i+1}[u_i]]\}_{i=1}^K$. Then each subgraph is shrink to $G[N^{r_i}[u_i]]$, and a local optimal solution is found for the shrink subgraph. This is in fact an application of vacancy technique introduced in [Sect. 4.1](#), the goal of which is to ensure that the union of the local optimal solutions is still independent.

7.1.2 The Performance Ratio

Let I^* be a maximum independent set of G . For $i = 1, 2, \dots, K$, denote $I_i^* = I^* \cap N^{r_i+1}[u_i]$. Then I_i^* is an independent set of $G[N^{r_i+1}[u_i]]$ and thus $|I_i^*| \leq |\text{MIS}(N^{r_i+1}[u_i])| \leq (1 + \varepsilon)|\text{MIS}(N^{r_i}[u_i])|$. It is clear that $I_i^* \cap I_j^* = \emptyset$ for $i \neq j$.

Algorithm 7 Growing PTAS for MIS

Input: A unit ball graph G and a positive real number $\varepsilon < 1$.

Output: An independent set I of G .

- 1: Set $U = V(G)$, $I = \emptyset$.
 - 2: **while** $U \neq \emptyset$ **do**
 - 3: Choose a vertex $u \in U$. Compute the minimum integer r such that $|\text{MIS}(N^{r+1}[u])| \leq (1 + \varepsilon)|\text{MIS}(N^r[u])|$.
 - 4: Set $I = I \cup \text{MIS}(N^r[u])$, $U = U \setminus N^{r+1}[u]$.
 - 5: **end while**
 - 6: Output I .
-

Thus, $|I^*| = \sum_{i=1}^K |I_i^*| \leq (1 + \varepsilon) \sum_{i=1}^K |MIS(N^{r_i}[u_i])| = (1 + \varepsilon)|I|$. Hence, [Algorithm 7](#) is a $(1 + \varepsilon)$ -approximation algorithm.

7.1.3 The Running Time

Notice that the most time-consuming step is to compute $MIS(N^r[u])$ and determine the minimum integer r satisfying the condition

$$|MIS(N^{r+1}[u])| \leq (1 + \varepsilon)|MIS(N^r[u])| \quad (13)$$

in Line 3 of [Algorithm 7](#).

Lemma 10 *In any unit ball graph G , the number of independent vertices in an r -closed neighborhood is upper bounded by $(2r + 1)^3$.*

Proof Let I^r be a maximum independent set in $N^r[u]$. Use $B_a(v)$ to denote the open ball with radius a and center v . For each vertex $v \in I^r$, draw an open ball $B_{1/2}(v)$. Notice that $N^r[u]$ is contained in the ball $B_r(u)$. Hence, all the balls $\{B_{1/2}(v)\}_{v \in I^r}$ are contained in the ball $B_{r+1/2}(u)$. Since I^r is independent, $B_{1/2}(v) \cap B_{1/2}(w) = \emptyset$ for $v, w \in I^r$ and $v \neq w$. It follows that $|I^r| \leq \frac{\frac{4}{3}\pi(r+\frac{1}{2})^3}{\frac{4}{3}\pi(\frac{1}{2})^3} = (2r + 1)^3$. \square

Lemma 11 *There is a constant $c_0 = O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ such that $r_i \leq c_0$ for all $i = 1, 2, \dots, K$.*

Proof By the choice of r_i and the observation $|MIS(N^0[u_i])| = 1$, one has $|MIS(N^{r_i}[u_i])| > (1 + \varepsilon)|MIS(N^{r_i-1}[u_i])| > (1 + \varepsilon)^2|MIS(N^{r_i-2}[u_i])| > \dots > (1 + \varepsilon)^{r_i}|MIS(N^0[u_i])| = (1 + \varepsilon)^{r_i}$. Combining this with [Lemma 10](#), one has

$$(2r_i + 1)^3 > (1 + \varepsilon)^{r_i}. \quad (14)$$

With the increase of r_i , since the increasing speed of $(1 + \varepsilon)^{r_i}$ is much higher than $(2r_i + 1)^3$, there must be a constant c_0 such that inequality (14) holds only for $r_i \leq c_0$.

The order of c_0 can be calculated from (14). \square

For $i = 1, 2, \dots, K$, denote by $n_i = |N^{r_i+1}[u_i]|$. As a consequence of [Lemma 10](#), the exhaust search for $MIS(N^r[u])$ runs in time at most $|N^r[u]|^{O(r^3)}$. It follows that the running time of [Algorithm 7](#) is upper bounded by $\sum_{i=1}^K r_i n_i^{O(r_i^3)} \leq c_0 \sum_{i=1}^K n_i^{O(c_0^3)} = c_0 n^{O(c_0^3)} = O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon} \cdot n^{O((\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})^3)})$.

7.2 Minimum Dominating Set

For a vertex subset $U \subseteq V(G)$, denote by $DS(U)$ the minimum vertex subset of $V(G)$ which dominates U , that is for each vertex $u \in U$, either $u \in DS(U)$ or u has a neighbor in $DS(U)$. Notice that $U \subseteq N_G[DS(U)]$ and $DS(U) \subseteq N_G[U]$. It should be emphasized that in this chapter, the symbol N refers to the neighbor set in the remaining graph, while N_G refers to the neighbor set in the original graph G . The algorithm is described in [Algorithm 8](#). As it has been emphasized, $N^r[u]$ is computed with respect to the remaining graph, while the dominating set $DS(\cdot)$ is computed with respect to the whole input graph G .

The partition resulted from [Algorithm 8](#) is $V(G) = N^{r_1+2}[u_1] \cup \dots \cup N^{r_K+2}[u_K]$, where K is the number of iterations and u_i and r_i are the vertex and the integer found in the i th iteration, respectively. Clearly, the output $D = \bigcup_{i=1}^K DS(N^{r_i+2}[u_i])$ is a dominating set of G .

Since $DS(N^{r_i+2}[u_i]) \subseteq N_G(N^{r_i+2}[u_i])$, it can be seen that vertices in $N_G(N^{r_i+2}[u_i]) \setminus N^{r_i+2}[u_i]$ might be used more than once as dominators. However, the repetition can be controlled to be small by the condition

$$|DS(N^{r+2}[u])| \leq (1 + \varepsilon)|DS(N^r[u])| \quad (15)$$

in Line 3 of [Algorithm 8](#).

Let D^* be a minimum dominating set of G , and let $D_i^* = D^* \cap N_G[N^{r_i}[u_i]]$. Since any vertex in $N^{r_i}[u_i]$ has its dominators in $N_G(N^{r_i}[u_i])$, it can be seen that D_i^* is a dominating set of $N^{r_i}[u_i]$, and thus

$$|DS(N^{r_i}[u_i])| \leq |D_i^*|.$$

Lemma 12 *The sets $N_G[N^{r_1}[u_1]], \dots, N_G[N^{r_K}[u_K]]$ are mutually disjoint.*

Proof Denote $U_0 = V(G)$ and $U_i = U_{i-1} \setminus N^{r_i+2}[u_i]$. By observing that $N_G[U_i] \cap N_G[N^{r_j}[u_j]] = \emptyset$ for $j < i$, the lemma follows by induction on i . \square

As a consequence of [Lemma 12](#),

$$|D^*| \geq \sum_{i=1}^K |D_i^*|.$$

Now, the performance ratio follows from

$$|D| = \sum_{i=1}^K |DS(N^{r_i+2}[u_i])| \leq (1 + \varepsilon) \sum_{i=1}^K |DS(N^{r_i}[u_i])|$$

Algorithm 8 Growing PTAS for MDS

Input: A unit ball graph G and a positive real number $\varepsilon < 1$.

Output: A dominating set D of G .

- 1: Set $U = V(G)$, $D = \emptyset$.
 - 2: **while** $U \neq \emptyset$ **do**
 - 3: Choose a vertex $u \in U$. Compute the minimum integer r such that $|DS(N^{r+2}[u])| \leq (1 + \varepsilon)|DS(N^r[u])|$.
 - 4: Set $D = D \cup DS(N^{r+2}[u])$, $U = U \setminus N^{r+2}[u]$.
 - 5: **end while**
 - 6: Output D .
-

$$\leq (1 + \varepsilon) \sum_{i=1}^K |D_i^*| \leq (1 + \varepsilon)|D^*|.$$

The analysis of time complexity is similar to the previous subsection and is omitted here.

7.3 Minimum Connected Dominating Set

To implement growing partition method on CDS problem, one first looks at condition (13) from another point of view and shows that it is essentially equivalent with

$$|MIS(N^{r+1}[u] \setminus N^r[u])| \leq \varepsilon|MIS(N^r[u])|. \quad (16)$$

To see the equivalence, first notice that the restriction of an independent set of $N^{r+1}[u]$ to $N^{r+1}[u] \setminus N^r[u]$ is still independent, so is the restriction of $N^{r+1}[u]$ to $N^r[u]$. Hence, $|MIS(N^{r+1}[u])| \leq |MIS(N^{r+1}[u] \setminus N^r[u])| + |MIS(N^r[u])|$, and thus condition (16) implies that $|MIS(N^{r+1}[u])| \leq (1 + \varepsilon)|MIS(N^r[u])|$.

However, condition (16) cannot be used directly to show that the running time is polynomial. The key is to find a constant c_0 such that the minimum integer r satisfying (16) can be found in at most c_0 steps (as in Lemma 11).

Lemma 13 *There is a constant $c_0 = O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ such that for any unit ball graph G and any vertex $u \in V(G)$, there is an integer $1 \leq r \leq c_0$ such that $|MIS(N^r[u] \setminus N^{r-1}[u])| \leq \varepsilon|MIS(N^{r-1}[u])|$.*

Proof Suppose this is not true. Then for any integer c , there exists a unit ball graph G and a vertex $u \in V(G)$ such that

$$|MIS(N^r[u] \setminus N^{r-1}[u])| > \varepsilon|MIS(N^{r-1}[u])| \quad (17)$$

holds for any integer $1 \leq r \leq c$. Since the union of an independent set of $N^r[u] \setminus N^{r-1}[u]$ and an independent set of $N^{r-2}[u]$ is an independent set of $N^r[u]$, one has $|MIS(N^r[u])| \geq |MIS(N^r[u] \setminus N^{r-1}[u])| + |MIS(N^{r-2}[u])|$. Combining this

with (17) and the observation $|MIS(N^{r-1}[u])| \geq |MIS(N^{r-2}[u])|$, one has

$$|MIS(N^r[u])| > (1 + \varepsilon)|MIS(N^{r-2}[u])|. \quad (18)$$

Recursively using inequality (18) and the observation $|MIS(N^0[u])| = 1$ and $|MIS(N^1[u])| \geq 1$, one has $|MIS(N^c[u])| > (1 + \varepsilon)^{\lfloor \frac{c}{2} \rfloor}$. By Lemma 10, one has

$$(2c + 1)^3 > (1 + \varepsilon)^{\lfloor \frac{c}{2} \rfloor}. \quad (19)$$

Since the left hand grows polynomially in c and the right hand grows exponentially in c , inequality (19) must be violated for some large c , a contradiction.

The order of c_0 can be calculated from inequality (19). \square

The implication of condition (16) is that in each iteration, the expansion of $N^r[u]$ halts when the boundary part is “thin” enough. This is why one does not lose too much at the boundary. This is especially important for the CDS problem, as suggested by the “compensation technique” in Sect. 5.4.

7.3.1 The Algorithm

For a connected vertex subset U , denote $CDS(U)$ the minimum connected subset of $V(G)$ which dominates U . Notice that such a *connected* set $CDS(U)$ exists since U is connected. Furthermore, $CDS(U) \subseteq N_G[U]$. As before, the symbol N refers to the neighbor set in the remaining graph, while the symbol N_G refers to the neighbor set in the original graph G . The algorithm is described in Algorithm 9.

7.3.2 Correctness

There are two major differences between Algorithm 9 and the previous ones. By Line 6, it can be seen that there is an overlap at the boundaries of the partition. By Line 4, it can be seen that except for the first vertex, the starting vertex u is not chosen arbitrarily, it is chosen to be “near” to the previous solutions. These two differences are made to ensure that the output is indeed a connected one.

Notice that $N^r[u]$ is always connected and $W \subseteq U$. In the following, u_i, r_i, U_i, W_i are used to denote the u, r, U, W obtained in the i^{th} iteration.

Algorithm 9 Growing PTAS for MCDS

Input: A connected unit ball graph G and a positive real number $\varepsilon < 1$.

Output: A connected dominating set D of G .

- 1: Set $U = V(G)$, $D = \emptyset$.
 - 2: Let u be an arbitrary vertex in U and set $W = \{u\}$.
 - 3: **while** $U \neq \emptyset$ **do**
 - 4: Choose $u \in W$.
 - 5: Compute the minimum integer r such that $|MIS(N_G[N^r[u]] \setminus N^r[u])| \leq \varepsilon |MIS(N^r[u])|$.
 - 6: Set $W = (W \cup N^{r+3}[u]) \setminus N^{r+2}[u]$, $D = D \cup CDS(N^{r+3}[u])$, $U = U \setminus N^{r+2}[u]$.
 - 7: **end while**
 - 8: Output D .
-

Lemma 14 *The output of Algorithm 9 is a connected dominating set of G .*

Proof Induction is used on i to show that the subset $C_i = \bigcup_{j=1}^i CDS(N^{r_j+3}[u_j])$ is connected. This is true for $i = 1$. Suppose C_i is connected. Since G is connected, it can be seen that as long as U_i is nonempty, the set W_i is nonempty. Hence, there exists a vertex $u_{i+1} \in W_i$. By the construction of W_i , the vertex $u_{i+1} \in N^{r_j+3}[u_j] \setminus N^{r_j+2}[u_j]$ for some $j < i$. Then u_{i+1} is dominated by a vertex $v \in CDS(N^{r_j+3}[u_j]) \subseteq C_i$. Thus, $C_{i+1} = C_i \cup CDS(N^{r_{i+1}+3}[u_{i+1}])$ is connected. \square

7.3.3 Performance

To analyze the performance ratio, it is first shown that the condition

$$|MIS(N_G[N^r[u]] \setminus N^r[u])| \leq \varepsilon |MIS(N^r[u])| \quad (20)$$

in Line 5 of Algorithm 9 guarantees a relationship which is similar to the condition (13) for the MIS problem and the condition (15) for the MDS problem.

Lemma 15 *There is a constant c_1 , such that $|CDS(N^{r_i+3}[u_i])| \leq (1 + c_1\varepsilon)|CDS(N^{r_i}[u_i])|$ holds for any $1 \leq i \leq K$.*

Proof The idea of the proof is to modify $CDS(N^{r_i}[u_i])$ to a connected dominating set of $N^{r_i+3}[u_i]$ by adding a controllable small number of vertices.

For convenience of statement, denote $C = CDS(N^{r_i}[u_i])$ and $M = MIS(N^{r_i+1}[u_i] \setminus N^{r_i}[u_i])$. Then $C \cup M$ is a dominating set of $N^{r_i+1}[u_i]$. Since each vertex $v \in M$ has a neighbor in $N^{r_i}[u_i]$, it can be seen that v can be connected to C by adding at most one vertex. For each vertex $v \in N^{r_i+3}[u_i] \setminus N^{r_i+1}[u_i]$, there is a vertex $w \in M$ which is at most 3 hops away from v . Thus, $N^{r_i+3}[u_i] \setminus N^{r_i+1}[u_i] \subseteq \bigcup_{w \in M} N^3[w]$. By further adding all the vertices in $\bigcup_{w \in M} CDS(N^3[w])$, one can obtain a connected dominating set of $N^{r_i+3}[u_i]$, denote it by \tilde{C} .

By Lemma 10 and Lemma 3, there is a constant c such that $|CDS(N^3[w])| \leq c$ holds for any w . Thus, $|\tilde{C}| \leq |C| + (2 + c)|M|$. By Corollary 1 and the condition (20), $|M| \leq |MIS(N_G[N^{r_i}[u_i]] \setminus N^{r_i}[u_i])| \leq \varepsilon |MIS(N^{r_i}[u_i])| \leq 12\varepsilon |CDS(N^{r_i}[u_i])| = 12\varepsilon |C|$. Taking $c_1 = 12(2 + c)$, one has $|CDS(N^{r_i+3}[u_i])| \leq |\tilde{C}| \leq (1 + c_1\varepsilon)|C|$. The lemma is proved. \square

Let D^* be a minimum connected dominating set of G . For $i = 1, 2, \dots, K$, let $D_i^* = D^* \cap N_G[N^{r_i}[u_i]]$. By Lemma 12, the sets $N_G[N^{r_1}[u_1]], \dots, N_G[N^{r_K}[u_K]]$ are mutually disjoint, and thus

$$|D^*| \geq \sum_{i=1}^K |D_i^*|.$$

Clearly, D_i^* is a dominating set of $N^{r_i}[u_i]$. However, D_i^* may be disconnected. Suppose $G[D_i^*]$ has k connected components. Similarly to the analysis in Sect. 5.4, adding at most two vertices reduces the number of components by one. Thus, one

can modify D_i^* into a connected dominating set of $N^{r_i}[u_i]$, denote it by \tilde{D}_i^* , such that

$$|\tilde{D}_i^*| \leq |D_i^*| + 2(k - 1).$$

Since D^* is connected, every connected component of $G[D_i^*]$ contains a vertex in $N_G[N^{r_i}[u_i]] \setminus N^{r_i}[u_i]$. These vertices must be independent since they belong to different connected components. Hence, $k \leq |\text{MIS}(N_G[N^{r_i}[u_i]] \setminus N^{r_i}[u_i])| \leq \varepsilon |\text{MIS}(N^{r_i}[u_i])| \leq 12\varepsilon |\text{DS}(N^{r_i}[u_i])| \leq 12\varepsilon |D_i^*|$. It follows that

$$|\tilde{D}_i^*| \leq (1 + 24\varepsilon) |D_i^*|.$$

Then the output D satisfies

$$\begin{aligned} |D| &\leq \sum_{i=1}^K |\text{CDS}(N^{r_i+3}[u_i])| \leq (1 + c_1\varepsilon) \sum_{i=1}^K |\text{CDS}(N^{r_i}[u_i])| \\ &\leq (1 + c_1\varepsilon) \sum_{i=1}^K |\tilde{D}_i^*| \leq (1 + c_1\varepsilon)(1 + 24\varepsilon) \sum_{i=1}^K |D_i^*| \leq (1 + \tilde{\varepsilon}) |D^*|. \end{aligned}$$

Similarly to the proof of [Lemma 13](#), it can be shown that there is a constant $c_0 = O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$ such that the condition (20) is met by some integer $r \leq c_0$, and thus the algorithm is polynomial.

7.3.4 A Remark

Observe that the key to achieving the above performance is the uniform upper bound for the MIS in a closed neighborhood. Hence all above algorithms apply to growth,-bounded graphs. For d -hop CDS problem, more vertices have to be added for connection. The main idea is that the added vertices can be compensated by those vertices in boundary region, which is thin enough to produce an ε .

8 Conclusion

This chapter focuses on partition method and its variations which are used to produce PTASs for geometric problems, including ball cover, maximum independent set, minimum dominating set, minimum connected dominating set, and minimum sum radii cover. Some techniques are introduced, including small compensation technique, multilayer partition technique, and growing partition technique. These methods are of methodological value and are applicable in an Euclidean space of any dimension. Whether there exists a constant approximation algorithm for minimum connected dominating set in disk graph is still an open problem.

Acknowledgements This work is supported by National Natural Science Foundation of China under grant (61222201) and Specialized Research Fund for the Doctoral Program of Higher Education (20126501110001). It is also supported by National Science Foundation of the USA under grants 1020 CNS0831579 and CCF0728851.

Cross-References

- ▶ [A Unified Approach for Domination Problems on Different Network Topologies](#)
 - ▶ [Algorithmic Aspects of Domination in Graphs](#)
 - ▶ [Connected Dominating Set in Wireless Networks](#)
 - ▶ [Coverage Problems in Sensor Networks](#)
 - ▶ [Packing Circles in Circles and Applications](#)
 - ▶ [Variations of Dominating Set Problem](#)
-

Recommended Reading

1. P.K. Agarwal, M. van Kreveld, S. Suri, Label placement by maximum independent set in rectangles. *Comput. Geom. Theory Appl.* **11**, 209–218 (1998)
2. K.M. Alzoubi, P. Wan, O. Frieder, Message-optimal connected dominating sets in mobile ad hoc networks, in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Lausanne, 2002
3. E.M. Arkin, M.M. Halldórrsson, R. Hassin, Approximating the tree and tour covers of a graph. *Inform. Process. Lett.* **47**, 275–282 (1993)
4. S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegegy, Proof verification and hardness of approximation problems, in *Proc. 33rd Annual Symp. on Foundations of Computer Science (FOCS 1992)*, Pittsburgh, PA, 1992, pp. 14–23
5. B.S. Baker, Approximation algorithms for NP-complete problems on planar graphs, in *Proceedings of the 24th Annual IEEE Symposium on Foundations of Computer Science*, Tucson, November 7–9. IEEE, New York, 1983, pp. 254–273. *J. ACM* **41**, 153–180 (1994)
6. R. Bar-Yehuda, S. Even, On approximating a vertex cover for planar graphs, in *STOC*, New York, NY, USA, 1982, pp. 303–309
7. R. Bar-Yehuda, S. Even, A local-ratio theorem for approximating the weighted vertex cover problem. *Inform. Process. Lett.* **47**, 275–282 (1993)
8. P. Berman, T. Fujito, On approximation properties of the independent set problem for degree 3 graphs, *Workshop on Algorithms and Data Structures. Lect. Note Comput. Sci.* **955**, 449–460 (1995)
9. P. Berman, B. Basgupta, S. Muthukrishnan, S. Ramaswami, Efficient approximation algorithms for tiling and packing problem with rectangles. *J. Algorithm* **41**, 178–189 (2001)
10. V. Bharghavan, B. Das, Routing in ad hoc networks using minimum connected dominating sets, in *International Conference on Communication*, Montreal, 1997
11. J. Blum, M. Ding, A. Thaeler, X.Z. Cheng, Connected dominating set in sensor networks and MANETs, in *Handbook of Combinatorial Optimization* (Kluwer Academic, Dordrecht, 2004), pp. 329–369
12. H. Breu, D.G. Kirkpatrick, Unit disk graph recognition is NP-hard. *Comput. Geometry Theor. Appl.* **9**, 3–24 (1998)
13. S. Butenko, O. Ursulenko, On minimum connected dominating set problem in unit-ball graphs, preprint. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.77.8336&rep=rep1&type=pdf>

14. M. Cadei, M.X. Cheng, X. Cheng, D. Du, Connected domination in ad hoc wireless networks, in *Proceedings of the Sixth International Symposium on Mobile Ad Hoc Networking and Computing*, Lausanne, 2002
15. T.M. Chan, Polynomial-time approximation schemes for packing and piercing fat objects. *J. Algorithm* **46**, 178–189 (2003)
16. Y.P. Chen, A.L. Liestman, Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks, in *Proceedings of the Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Lausanne, 2002
17. X. Cheng, X. Huang, D. Li, W. Wu, D. Du, A polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks. *Networks* **42**, 202–208 (2003)
18. B.N. Clark, C.J. Colbourn, D.S. Johnson, Unit disk graphs. *Discrete Math.* **86**, 165–177 (1990)
19. B. Das, V. Bharghavan, Routing in ad-hoc networks using minimum connected dominating sets, in *ICC'97*, Las Vegas, NV, USA, 1997, pp. 376–380
20. I. Dinur, S. Safra, On the hardness of approximating minimum vertex cover. *Ann. Math.* **162**, 439–485 (2005)
21. D.-Z. Du, Y. Zhang, Q. Feng, On better heuristic for Euclidean Steiner minimum trees, in *Proceedings 32nd IEEE Symposium on Foundations of Computer Science*, San Juan, Puerto Rico, 1991, pp. 431–439
22. T. Erlebach, K. Jansen, E. Seidel, Polynomial-time approximation schemes for geometric graphs, in *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms* (SODA'01), Philadelphia, PA, USA, 2001, pp. 671–679
23. T. Erlebach, K. Jansen, E. Seidel, Polynomial-time approximation schemes for geometric intersection graphs. *SIAM J. Comput.* **34**, 1302–1323 (2005)
24. B. Escoffier, L. Gourvès, J. Monnot, Complexity and approximation results for the connected vertex cover problem. *Graph Theor. Concept Comput. Sci. Lect. Note Comput. Sci.* **4769**, 202–213 (2007)
25. U. Feige, A threshold of $\ln n$ for approximating set cover, in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, (ACM, New York, 1996), pp. 314–318
26. T. Fujito, T. Doi, A 2-approximation NC algorithm for connected vertex cover and tree cover. *Inform. Process. Lett.* **90**, 59–63 (2004)
27. X.F. Gao, W. Wang, Z. Zhang, S.W. Zhu, W.L. Wu, A PTAS for minimum d -hop connected dominating set in growth bounded graphs. *Optim. Lett.* **4**, 321–333 (2010)
28. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1978)
29. B. Gfeller, E. Vicari, A Faster distributed approximation scheme for the connected dominating set problem for growth-bounded graphs, in *ADHOC-NOW 2007*, vol. 4686, ed. by E. Kranakis, J. Opatrný. LNCS (2007), pp. 59–73
30. C. Glaßer, S. Reith, H. Vollmer, The complexity of base station positioning in cellular networks. *Discrete Appl. Math.* **148**, 1–12 (2005)
31. S. Guha, S. Khuller, Approximation algorithms for connected dominating sets. *Algorithmica* **20**, 374–387 (1998)
32. D.S. Hochbaum, W. Maass, Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM* **32**, 130–136 (1985)
33. D.S. Hochbaum, *Approximation Algorithm for NP-Hard Problems* (PWS, Boston, 1997)
34. H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, R.E. Stearns, NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithm* **26**, 238–274 (1998)
35. K. Jansen, L. Porkolab, On preemptive resource constrained scheduling: polynomial-time approximation schemes. *Int. Programm. Combin. Optim. Lect. Note Comput. Sci.* **2337**, 329–349 (2006)
36. T. Jiang, E.B. Lawler, L.S. Wang, Aligning sequences via an evolutionary tree: complexity and algorithms, in *Proceedings 26th ACM Symposium on Theory of Computing*, New York, NY, USA, 1994, pp. 760–769
37. D.S. Johnson, Approximation algorithms for combinatorial problems. *JCSS* **9**, 256–278 (1974)

38. R.M. Karp, Probabilistic analysis of partitioning algorithms for the travelling-salesman problem in the plane. *Math. Oper. Res.* **2**, 209–224 (1977)
39. S. Khot, O. Regev, Vertex cover might be hard to approximate to within $2 - \varepsilon$. *J. Comput. Syst. Sci.* **74**, 335–349 (2008)
40. S. Khuller, U. Vishkin, N.A. Young, Primal-dual parallel approximation technique applied to weighted set and vertex cover, *Proceedings of the 3rd Integer Programming and Combinatorial Optimization Conference*, Erice, Italy, 1993, pp. 333–341
41. D. Kim, Z. Zhang, X.Y. Li, W. Wang, W.L. Wu, D.-Z. Du, A better approximation algorithm for computing connected dominating sets in unit ball graphs. *IEEE Trans. Mobile Comput.* **9**, 1108–1118 (2010)
42. P. Koebe, Kontaktprobleme der konformen Abbildung, Brichte über die Verhandlungen der Sächsischen Akademie der Wissenschaften, Leipzig. Math. Phys. Klasse **88**, 141–164 (1936)
43. J. Komolos, M.T. Shing, Probabilistic partitioning algorithms for the rectilinear steiner tree problem. *Networks* **15**, 413–423 (1985)
44. N. Lev-Tov, D. Peleg, Polynomial time approximation schemes for base station coverage with minimum total radii. *Comput. Network* **47**, 489–501 (2005)
45. C. Lund, M. Yannakakis, On the hardness of approximating minimization problems, in *Proc. 25th ACM Symp. on Theory of Computing (STOC 1993)*, San Diego, CA, 1993, pp. 286–293
46. E. Malesińska, Graph-theoretical models for frequency assignment problems, PhD thesis, Technische Universität Berlin, 1997
47. M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz, Simple heuristics for unit disk graphs. *Networks* **25**, 59–68 (1995)
48. M. Min, S.C.-H. Huang, J. Liu, E. Shragowitz, W. Wu, Y. Zhao, Y. Zhao, An approximation scheme for the rectilinear Steiner minimum tree in presence of obstructions, Novel Approaches to Hard Discrete Optimization, Fields Institute Communications Series. Am. Math. Soc. **37**, 155–163 (2003)
49. M. Min, H. Du, X. Jia, C.X. Huang, S.C. Huang, W. Wu, Improving construction for connected dominating set with Steiner tree in wireless sensor networks. *J. Global Optim.* **35**, 111–119 (2006)
50. B. Monien, E. Speckenmeyer, Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Inform.* **22**, 115–123 (1985)
51. T. Nieberg, J. Hurink, W. Kern, A robust PTAS for maximum weight independent sets in unit disk graphs, *WG 2004*, ed. by J. Hromkovič, M. Nagl, B. Westfechtel, LNCS, vol. 3353, 2004, pp. 214–221
52. T. Nieberg, J. Hurink, A PTAS for the minimum dominating set problem in unit disk graphs, *WAOA 2005*, ed. by T. Erlebach, G. Persiano, LNCS, vol. 3879, 2006, pp. 296–306
53. L. Ruan, H. Du, X. Jia, W. Wu, Y. Li, K. Ko, A greedy approximation for minimum connected dominating set. *Theor. Comput. Sci.* **329**, 325–330 (2004)
54. C. Savage, Depth-first search and the vertex cover problem. *Inform. Process. Lett.* **14**, 233–235 (1982)
55. W.D. Smith, D.C. Wormald, Geometric separator theorems and applications, in *Proc. 39th IEEE Sympos. Found. Comput. Sci.*, Palo Alto, California, USA, 1998, pp. 232–243
56. S.A. Vavasis, Automatic domain partitioning in three dimensions. *SIAM J. Sci. Stat. Comput.* **12**, 950–970 (1991)
57. P. Wan, K.M. Alzoubi, O. Frieder, Distributed construction of connected dominating set in wireless ad hoc networks, in *Proc. Infocom'2002*, New York, NY, USA, 2002
58. L.S. Wang, T. Jiang, An approximation scheme for some Steiner tree problems in the plane. *Networks* **28**, 187–193 (1996)
59. L.S. Wang, T. Jiang, E.L. Lawler, Approximation algorithms for tree alignment with a given phylogeny. *Algorithmica* **16**, 302–315 (1996)
60. L.S. Wang, T. Jiang, D. Gusfield, A more efficient approximation scheme for tree alignment, in *Proceedings 1st International Conference on Computational Biology*, New York, NY, USA, 1997, pp. 310–319

61. A. Wiese, E. Kranakis, Local PTAS for dominating and connected dominating set in location aware unit disk graphs. *Approx. Online Algorithm Lect. Note Comput. Sci.* **5426**, 227–240 (2009)
62. J. Wu, H.L. Li, On calculating connected dominating set for efficient routing in ad hoc wireless networks, in *Proceedings of the 3rd ACM International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, New York, NY, USA, 1999, pp. 7–14
63. Z. Zhang, X.F. Gao, W.L. Wu, D.-Z. Du, A PTAS for minimum connected dominating set in 3-dimensional wireless sensor networks. *J. Global Optim.* **45**, 451–458 (2009)
64. Z. Zhang, X.F. Gao, W.L. Wu, PTAS for connected vertex cover in unit disk graphs. *Theor. Comput. Sci.* **410**, 5398–5402 (2009)
65. C. Zong, *Sphere Packings* (Springer, New York, 1999)
66. F. Zou, X.Y. Li, D.Y. Kim, W.L. Wu, Construction of minimum connected dominating set in 3-dimensional wireless network, in *WASA 2008*. *Lect. Note Comput. Sci.* **5258**, 134–140 (2008)

Probabilistic Verification and Non-approximability

Mario Szegedy and Kashyap Kolipaka

Contents

1	Introduction	2626
1.1	Probabilistically Checkable Proofs	2627
1.2	A Brief History of Probabilistic Verification	2629
1.3	The Language of Cryptography	2630
1.4	The PCP Theorem	2631
2	Non-approximability Results	2632
2.1	A Brief History of Approximating NPO Problems	2634
2.2	The FGLSS Graph	2635
2.3	The Gap-3SAT Instance	2636
2.4	Refined Parameters of PCPs	2637
2.5	Amplification Techniques	2638
2.6	The Long Code	2646
2.7	Constraint Satisfaction Problems	2648
2.8	The Max Clique	2651
2.9	The Chromatic Number	2654
2.10	Set Cover	2657
2.11	Some More Non-approximability Results	2661
3	A Short Proof of the PCP Theorem	2662
3.1	The Three Sub-Steps of Dinur's Basic Reduction Step	2664
4	The Unique Games Conjecture	2667
4.1	Algorithms for Unique Games	2668
4.2	Variants of the Unique Games Conjecture	2669
4.3	Optimal Bounds Under the Unique Games Conjecture	2670
5	Structural Consequences of the PCP Theory	2674
5.1	Polynomial-Time Approximation Schemes	2675
5.2	Approximation Preserving Reductions	2675
5.3	APX Complete Problems	2677
6	Conclusion	2678
	Cross-References	2679
	Recommended Reading	2679

M. Szegedy (✉) • K. Kolipaka

Department of Computer Science, Rutgers, The State University of New Jersey, Piscataway, NJ,
USA

e-mail: szegedy@cs.rutgers.edu; kolipaka@cs.rutgers.edu

Abstract

This chapter describes some of the highlights of the theory of Probabilistically Checkable Proofs, from the milestone discovery of Feige et. al. through Dinur’s proof of the PCP theorem to the lately discovered consequences of the Unique Games Conjecture.

Our goal is to illuminate the major themes that run through the entire theory: probabilistic verification, proof recursion, gap enlarging reductions, code checking, constraint satisfaction problems. We have also discussed specific problems: set cover, independent set, graph coloring, etc. Since a completely comprehensive picture of this vast subject is not possible, we give pointers to a generous amount of literature.

1 Introduction

In 1991, Feige, Goldwasser, Lovász, Safra, and Szegedy [38] showed that results about multi-prover systems or equivalently PCPs imply the relative hardness of approximating the maximum clique number of a graph. Their discovery has joined the subjects of probabilistic verification and the theory of NP optimization, and a new subject emerged, called PCP theory. The goal of this chapter is to give an introduction and an overview of this theory. Since the key results alone would span several hundred pages, we have restricted ourselves to a representative selection.

Undoubtedly, the most fundamental part of the theory, with its numerous consequences, is the PCP theorem, which asserts that MAX3SAT is NP hard to approximate within a factor of $1 + \epsilon$ (for some $\epsilon > 0$). In Sect. 3 we sketch the short proof of it by Irit Dinur [33].

Reductions play a key part in proving hardness of approximation. They fall roughly into two types. Originally and traditionally relative hardness of approximating optimal values of NP optimization problems (NPO) was proven by gap-preserving reductions. In the more modern theory, we almost exclusively use Karp reduction between gap problems. We explain both types, as well as various amplification techniques, such as the parallel repetition.

We introduce the reader to probabilistic verification. The prover-verifier intuition has been the powerful force that exponentially accelerated the theory’s development and even today, when alternative interpretations are available, still proves to be indispensable when trying to obtain stronger results.

We shall talk about the non-approximability of classic NPO such as Max Clique, graph coloring, and set cover.

We shall explain the long code of Bellare, Goldreich, and Sudan, as it receives multiple applications throughout the theory.

A well-studied class of optimization problems is the class of Constraint Satisfaction Problems (CSP), which also happens to be the class, for which our non-approximability theory is the most successful. We devote a lot of attention to this class and sketch optimal non-approximability for its members.

Many of the optimal non-approximability results come from unique games, in particular, a beautiful and general result due to Raghavendra [83]. Unique

games-based hardness assumptions, initiated by Khot [61], might transform our views on complexity, by providing an alternative to the P versus NP paradigm. We devote two sections to unique games.

As we have indicated, we have made several omissions. We do not discuss several known non-approximability results. We do not mention efforts and techniques to optimize parameters of PCPs, such as proof size. Our chapter is concerned only with NPO. Probabilistic debate systems [26] and non-approximability of #P problems are out of the scope of this survey.

1.1 Probabilistically Checkable Proofs

The idea of probabilistic verification builds on the idea of deterministic verification. Recall that every language $L \in NP$ is deterministically verifiable in polynomial time, meaning that for every $L \in NP$, there is a polynomial-time machine V such that if $x \in L$ then there exists a polynomial size membership such that $V(x, P) = 1$ and if $x \notin L$ then for every P , $V(x, P) = 0$ (Fig. 1). The concept of probabilistic verification is analogous.

Definition 1 (Probabilistic Verification) For functions $f, g : \mathbf{N} \rightarrow \mathbf{N}$, a probabilistic polynomial-time verifier $V(x, P, r)$ is (f, g) -restricted if, for every input x of size n , it uses at most $f(n)$ random bits and examines at most $g(n)$ bits in the membership proof while checking it. Let Σ be an alphabet and $L \subseteq \Sigma^*$. V is an (f, g) -restricted polynomial-time probabilistic verifier for L with completeness q and soundness p if for every input x :

- If $x \in L$, then there exists a membership proof P such that

$$\text{Prob}_r(V(x, P, r) = 1) \geq q$$

- If $x \notin L$, then for any membership proof P ,

$$\text{Prob}_r(V(x, P, r) = 1) \leq p$$

If q and p are not stated explicitly, then, by definition, $q = 1$, and $p = 1 - \epsilon$ for some fixed $\epsilon > 0$.

Here we have to make some comments. First, it is best to think of V as a random access machine. Since Turing machines and random access machines are polynomial-time equivalent, we may also view V as a Turing machine, but even in the latter case, we need to assume that V has random access to the bits of P . Another issue is the *adaptivity* of V . Are the bits of P which V accesses determined in advance from x and r , or the location of the second bit read from P may depend on value of the first bit read from P , etc.? In the first case, we call the verifier *nonadaptive*, and in the second case, we call the verifier *adaptive*. Most verifier constructions in the PCP theory are nonadaptive.

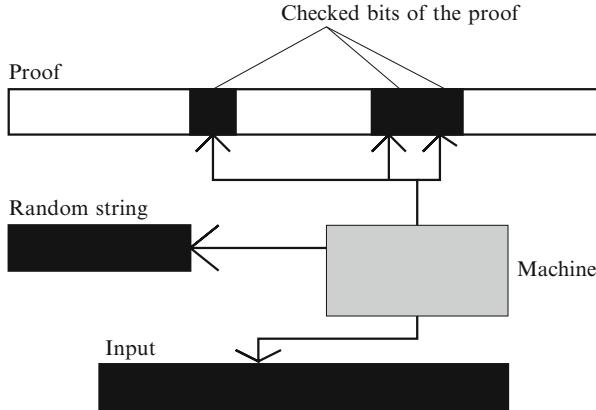


Fig. 1 A schematic picture of a probabilistic verifier. Only the black bits are read for a fixed random choice r

Definition 2 (PCP Classes[5]) A language L is in the class $PCP(f, g)$ iff there exists a constant $C > 0$ and an $(f(|x|^C), g(|x|^C))$ -restricted polynomial-time probabilistic verifier for L . For completeness and soundness values other than 1 and $1 - \epsilon$, we denote the corresponding PCP class by $PCP_{q,p}(f, g)$.

Clearly, $NP = PCP(0, |x|)$.

Lemma 1 If $L \in PCP_{q,p}(f, g)$ and L_1 polynomial time many-one reduces to L , then $L_1 \in PCP_{q,p}(f, g)$.

Proof Since L_1 many-one reduces to L , there is a polynomially computable T such that for every $x \in \Sigma^*$, $T(x) \in L$ iff $x \in L_1$. Let $V(x, P, r)$ be a probabilistic verifier for L . Then $V(T(x), P, r)$ is a probabilistic verifier for L_1 with the required parameters. \square

The PCP notation allows for a compact description of many known results:

Arora=A, Babai=B, Feige=Fe, Fortnow=F, Goldwasser=G, Levin=Le, Lovasz=Lo, Lund=Lu, Motwani=M, Safra=Safra, Sudan=Sudan, Szegedy=Sz	
$NEXP = PCP(n, n)$	BFLu [11]
$NP \subseteq PCP(\log n \log \log n, \log n \log \log n)$	BFLeSz [12]
$NP \subseteq PCP(\log n \log \log n, \log n \log \log n)$	FeGLoSaSz [38]
$NP = PCP(\log n, \sqrt{\log n})$	ASa [5]
$NP = PCP(\log n, 3)$	ALuMSuSz [4]

In Sect. 1.4 we are going to prove the last one [4] which we also call the basic PCP theorem because it is the basis of many further improvements. To describe these improvements we need to introduce new parameters such as free bit complexity,

amortized complexity, and their combinations. The various parameters will be discussed in details in Sect. 2.4.

1.1.1 Alphabet Size

Besides the four parameters of a probabilistic verifier (randomness, query size, completeness, soundness), sometimes its alphabet size may be an issue. Actually, for a machine $V(x, P, r)$, we have two different alphabets we may care about: that of input x and that of proof y . We will see cases, when the alphabet size of P is particularly important. We assume that alphabet of the input is either binary or it is understood from the context.

Definition 3 A language L is in the class $[\Sigma]PCP(f, g)$ iff there exists a constant $C > 0$ and an $(f(|x|^C), g(|x|^C))$ -restricted polynomial-time probabilistic verifier $V(x, P, r)$ for L , where $P \in \Sigma^*$.

Perhaps interestingly, $[\{0, 1\}]PCP_{1,p}(\log n, 2) = P$ for any $p < 1$.

1.2 A Brief History of Probabilistic Verification

The usual definition of NP uses the notion of a deterministic verifier that checks membership proofs of languages in polynomial time. Goldwasser, Micali, and Rackoff [49] and Babai [9, 13] were the first who allowed the verifier to be a probabilistic polynomial-time Turing machine that interacts with a “prover,” which is an infinitely powerful Turing machine trying to convince the verifier that the input x is in the language. A surprising result due to Lund, Fortnow, Karloff, and Nisan [10, 73] and Shamir [90] has shown that every language in PSPACE—which is suspected to be a much larger class than NP—admits such “interactive” membership proofs. Another variant of proof verification, due to Ben-Or, Goldwasser, Kilian, and Wigderson [17], involves a probabilistic polynomial-time verifier interacting with more than one mutually noninteracting provers. The class of languages with such interactive proofs is called MIP (for multi-prover interactive proofs). Fortnow, Rompel, and Sipser [43] gave an equivalent definition of MIP as languages that have a probabilistic polynomial-time oracle verifier that checks membership proofs (possibly of exponential length) using *oracle access* to the proof.

Babai, Fortnow, and Lund [11] showed that MIP is exactly NEXP, the class of languages for which membership proofs can be checked deterministically in exponential time. This result is surprising because NEXP is just the exponential analogue of NP, and its usual definition involves no notion of randomness or interaction. Therefore, researchers tried to discover if the MIP = NEXP result can be “scaled down” to say something interesting about NP. Babai, Fortnow, Levin, and Szegedy [12] introduced the notion of *transparent* membership proofs, namely, membership proofs that can be checked in polylogarithmic time, provided the input is encoded with some error-correcting code. They showed that NP languages have such proofs. Feige et al. [38] showed a similar result, but with a somewhat more efficient verifier. Arora and Safra [5] further improved the efficiency of checking

membership proofs for NP languages up to the point that they were able to give a new definition for NP this way.

They define a parameterized hierarchy of complexity classes called PCP (for probabilistically checkable proofs). This definition uses the notion of a “probabilistic oracle verifier” of Fortnow et al. [43] and classifies languages based on how efficiently such a verifier can check membership proofs for them. The notion of “efficiency” refers to the number of random bits used by the verifier as well as the number of bits it reads in the membership proof. Note that we count only the bits of the *proof* that are read—not the bits of the *input* which the verifier is allowed to read fully. The definition of a class very similar to PCP was implicit in the work of Feige et al. [38].

The paper of Arora, Lund, Motwani, Sudan, and Szegedy [4] in 1992 building on [5] was the first to decrease the number of check bits to constant. It is not just philosophically important that every transparently written proof can be checked with high accuracy by looking only at a constant number of places in the proof, but the construct of ALMSS also serves as a building block for nearly every construct that comes after it.

1.3 The Language of Cryptography

Articles in cryptography often describe mathematical objects such as information, knowledge, and secret through interactions between human characters. While the framework originally was used almost exclusively by cryptographers, it has become an integral part of the entire computer science culture.

In the remaining part of the text, we are going to need expressions such as “true prover,” “verifier,” and “cheating prover.” But what do they mean exactly? When we prove that a probabilistic verifier V recognizes a language L , our argument consists of two parts. In the first part we show that if $x \in L$, the verifier accepts some proof P with probability 1. In the second part we show that if $x \notin L$ then the verifier rejects every proof with probability at least ϵ .

The first part of the argument features the *true prover* who is responsible to provide a string P which is accepted by the verifier with probability one. In this formula, $\text{Prob}_r(V(x, P, r) = 1) = 1$. The presumed structure of P is expressed in terms of the actions of the true prover. For instance, when we say that “the good prover encodes string a using an encoding E ,” it means that the presumed structure of the proof is a code word $E(a)$.

When we investigate the $x \notin L$ case, the main character of the story is the *cheating prover*. By assumption he is an ultimately devious creature whose proof does not adhere to the structure of the true prover. He has to cheat somewhere, since no correct proof exists when $x \notin L$. This property is required by the axioms of a proof system for L . The goal of the cheating prover is to maximize the probability with which the verifier accepts P , the “proof” he presents. It is important to emphasize that the proof of the cheating prover is an arbitrary string formed from the letters of the given alphabet, typically $\Sigma = \{0, 1\}$. When we argue about the $x \in L$ case, we never need to talk about the cheating prover.

Like in the case of other formal proof systems in the case of PCPs, the proof itself is just a vehicle to indicate that a statement holds and itself is not interesting for the verifier at all. Only the mere existence or non-existence of its corrected version matters. The verifier does not want to learn the entire proof or to verify that the proof adheres to the structure of the good prover *everywhere*. It is sufficient for us to show that if $x \notin L$, no matter what string the cheating prover may write down, it is so far from being consistent either with the structure of the true prover or with other constraints defined by input x that the verifier is able to catch the error with probability ϵ .

1.4 The PCP Theorem

Theorem 1 (PCP Theorem) $NP \subseteq PCP(\log n, 3)$. Moreover, we can also assume that the verifier is nonadaptive and his accepting criterion is a disjunct (i.e., “OR”) of the literals made from the three bits he looks at.

Notice that by [Definition 2](#), the factor $\log n$ scales by a constant, but not the second argument. The following seemingly weaker variant of the above theorem is also often called the PCP theorem:

Theorem 2 $NP \subseteq [\Sigma]PCP(\log n, c)$ for some fixed constant c and for some fixed alphabet Σ .

This statement was first proven by Arora, Lund, Motwani, Sudan, and Szegedy [4], based on several previous works [11, 12, 38], in particular on a simultaneous work of Arora and Safra [5]. The original proof was based on several algebraic and combinatorial techniques. The proof had several variations, but they all had the basic design. We call this the classic PCP proof design. A new type of proof was given by Irit Dinur in 2005 [33]. She has proved the following:

Theorem 3 $NP \subseteq [\{0, 1\}^3]PCP(\log n, 2)$.

We give a sketch of her proof in [Sect. 3](#). In the rest of this section, we show that [Theorem 2](#) implies [Theorem 1](#) (the converse is straightforward).

Lemma 2 Let $\ell > 3$ be a constant and $f(n)$ be an arbitrary function from the positive integers to the positive integers. If there is a nonadaptive $(f(n), \ell)$ -restricted verifier V that checks a proof P with alphabet Σ and recognizes a language L with completeness 1 and soundness $1 - \epsilon$, then there is a non-adaptive $(f(n) + O(1), 3)$ -restricted verifier V' that checks a proof P' with alphabet $\{0, 1\}$ and recognizes L such that its checking criterion is always a disjunct of three literals made from the three bits it reads. Furthermore, V' has completeness 1 and soundness ϵ/K , where $K > 0$ is some constant dependent only on ℓ and Σ .

Proof We prove this lemma with a technique called *proof composition*. Proof composition is a very general principle stating that if we prove the existence of a proof for $x \in L$, then $x \in L$ follows (thus, instead of a proof, it suffices to have a proof that a proof exists). A proof composition technique in the PCP context was developed in [5]. Here we need a very simple version of it.

Let us fix x . For every fixed random choice r , the accepting criterion of V is a Boolean-valued function φ_r of the ℓ places of P examined by V . The composed proof contains the original proof and some extra information that helps the verifier. Since we want the composed proof to have a binary alphabet, we also encode the alphabet of the original proof in binary via some (arbitrary) injection $\Sigma \rightarrow \{0, 1\}^{\lceil \log |\Sigma| \rceil}$. The part of the new proof that is identical to P , with its letters encoded in binary is called the *core*. For every $r \in \{0, 1\}^{f(n)}$, we then create a constant size proof β_r that helps to convince the verifier that φ_r holds. The entire new proof is now the union of the core and $\bigcup_r \beta_r$. The details are as follows:

First recall from logic that every Boolean function $\varphi(\alpha)$ on v variables can be written in the form $(\exists \beta) \psi(\alpha, \beta)$, where ψ is a 3CNF formula. β is a sequence of at most $(v - 1)2^v$ auxiliary variables, and ψ has at most $2^v + (v - 1)2^{v+1}$ clauses, but for our purposes, we only care that these quantities are constants in terms of v .

Let $(\exists \beta_r) \psi_r(\alpha_r, \beta_r)$ be the 3CNF-equivalent of φ_r . Here α_r is the $\ell \lceil \log |\Sigma| \rceil$ bits of information that machine V reads from P under random bit r . The help β_r (of the true prover) is any evaluation of the auxiliary variables for which $\psi_r(\alpha_r, \beta_r) = 1$.

The new verifier, V' , picks a random r and a random clause of ψ_r and accepts if the selected clause evaluates to 1 under α_r and β_r of the composed proof (α_r is part of the core, β_r is the sequence of auxiliary bits that the prover provides for query r). Since ψ_r has constant number of clauses, the composed verifier needs at most constant number of additional random bits to perform its query.

If V accepts a proof P with probability 1, then for every r there exists a β_r such that $g_r(\alpha_r, \beta_r) = 1$, so there exists a composed proof which V' accepts with probability 1.

Assume now that $x \notin L$. We claim that no composed proof will be accepted with probability greater than $1 - \frac{\epsilon}{K}$ by V' , where K is the uniform upper bound on the number of clauses in $\psi_r(\alpha_r, \beta_r) = 1$ for all r s.

To see this, assume that a composed proof with core P is accepted with probability greater than $1 - \frac{\epsilon}{K}$. Then the fraction of r s for which the predicate $(\exists \beta_r) \psi_r(\alpha_r, \beta_r)$ is false is less than $K \times \frac{\epsilon}{K} = \epsilon$. Then V accepts P with probability greater than $1 - \epsilon$, a contradiction. \square

2 Non-approximability Results

NPO (see [29, 30, 56]) have one of the following forms:

- Find $\max_{|y| \leq q(|x|)} P(x, y)$ given input x . [maximization]
- Find $\min_{|y| \leq q(|x|)} P(x, y)$ given input x . [minimization]

Here $P(x, y)$ is a polynomially computable function that assigns non-negative rationals to pairs of strings, and q is a polynomial. (In the literature witness y is selected from an arbitrary polynomial-time computable set of strings not necessarily only of the form $\{y \mid |y| \leq q(|x|)\}$. Our definition is sufficient to describe all relevant problems and enables us to avoid anomalies that arise from the more general definition.)

While one can construct optimization problems that are not in NPO, the class plays a primary role in optimization theory with such well-known problems as coloring, allocation, scheduling, Steiner tree problems, TSP, linear and quadratic programming, knapsack, and vertex cover. The consequences of the PCP theory almost solely concern NPO problems. There are exceptions [26, 27], but we do not discuss them here.

Most NPO problems are NP-hard to compute exactly. A polynomial-time algorithm A is said to *approximate* an NP maximization problem P to within a factor $r(x) \geq 1$, if A outputs a witness y such that the optimal solution for input x lies within $\max_{|y| \leq q(|x|)} P(x, y)/r(x)$ and $\max_{|y| \leq q(|x|)} P(x, y)$. Here $r(x)$ is called *approximation ratio* and can be generalized also to minimization problems in an obvious way.

Below we give three examples to NPO problems:

MAX3SAT: Let x represent a 3CNF formula, y represent an assignment, and $P(x, y)$ be the number of clauses in x satisfied by y . [Maximization]

Set cover: Let x represent a polynomial size set system, y represent a selection of sets, and $P(x, y)$ be the function, which equals to the number of selected sets if the selected sets cover the universe, otherwise, the size of the entire set system. [Minimization]

PCP: Let V be an $(\log n, g(n))$ -restricted probabilistic verifier for a language L . x represents an input, y represents a proof, and $P(x, y)$ is the acceptance probability of proof y for input x . Since the verifier uses $\log n$ randomness, we can run V on all random strings and compute this probability in polynomial time. [Maximization]

If in the latest example, we take L to be an NP-complete language and V to be the verifier of [Theorem 1](#), then it is easy to see that if we could efficiently approximate the acceptance probability of V to within a factor better than $1 - \epsilon$ (for the ϵ of the theorem), then we could also solve the membership problem for L . This is an example to an argument about non-approximability. What is special about the optimization problem coming from [Theorem 1](#) is that the value of $P(x, y)$ is proportional to a number of very simple predicates that y satisfies, namely, each predicate depends only on three bits of y . This makes this NPO problem a *CSP*. Before the PCP theorem there were no non-approximability results for any CSP. The theory of PCP exploits the fact that probabilistic verifiers for NP-complete languages translate to provably hard approximation problems.

2.1 A Brief History of Approximating NPO Problems

The NP-completeness theory of Cook [28] and Levin [70] puts NPO problems into two categories: NP-hard or not NP-hard. The first alarm that this characterization is too coarse was sent off in an early paper of D. Johnson [56] entitled “Approximation Algorithms for Combinatorial Problems.”

Motivated by exact bounds on the performance of various bin packing heuristics of [45], Johnson gave algorithms for the subset sum, the set cover, and the MAX k -SAT problems with guarantees on their performances ($1 + o(1)$, $O(\log |S|)$, $2^k/(2^k - 1)$, respectively). He also gave non-approximability results, but unfortunately they referred only to specific algorithms. Nevertheless, he has brought up the issue of classifying NPO problems by the best approximation ratio achievable for them in polynomial time.

For years advances were very slow. Sahni and Gonzales [88] proved the non-approximability of the nonmetric traveling salesman and other problems, but their proofs were very similar to standard NP-completeness proofs, and they effected only problems where approximation algorithms were not explicitly sought for. A more promising approach was found by Garey and Johnson [46] who used graph products to show that the chromatic number of a graph cannot be approximated to within a factor of $2 - \epsilon$ unless $P = NP$ and an approximation algorithm for the Max Clique within *some* constant factor could be turned into an algorithm which approximates Max Clique within *any* constant factor.

The old landscape of approximation theory of NPO radically changed when in 1991 Feige, Goldvasser, Lovász, Safra, and Szegedy [38] for the first time used Babai, Fortnow, and Lund’s characterization of NEXP in terms of multi-prover interactive proof systems [11] to show that approximating the clique within any constant factor is hard to $\text{NTIME}(n^{1/\log \log n})$. Simultaneously Papadimitriou and Yannakakis defined a subclass of NPO what they called MAXSNP, in which problems have an elegant logical description and can be approximated within a constant factor. They also showed that if MAX3SAT, vertex cover, Max Cut, and some other problems in the class could be approximated with an arbitrary precision, then the same would hold for all problems in MAXSNP. They established this fact by reducing MAXSNP to these problems in an *approximation preserving* manner. Their original intention was most likely to build a theory of non-approximability via defining suitable reductions, analogous to those in the theory of NP, but new non-approximability results in [4] let them achieve much more. The theory of NP and the theory of non-approximability met, and research developed rapidly in three directions:

1. Non-approximability results for NPO problems
2. Construction of approximation algorithms achieving optimal or near-optimal ratios
3. Development of approximation preserving reductions and discovery of new (non)-approximability classes

Today, we have a precise idea about the non-approximability status of many NPO problems. The on-line compendium of Pierluigi Crescenzi and Viggo Kann [29] keeps track of the growing number of results and continuously updates data about most known NPO problems.

2.2 The FGLSS Graph

The first paper to construct a combinatorial object from a PCP was that of Feige et al. [38]. For deterministic verifiers an analogous construction cannot be meaningfully made (Fig. 2). The combinatorial object was simply a graph G , and the construction implied hardness of approximating the optimization problem

$$\text{MaxClique}(G) = \max_{S \subseteq V(G)} \{|S| \mid \text{for all } v, w \in S \text{ we have } (v, w) \in E(G)\}$$

Let V be an (f, g) -restricted verifier. The evaluation of the entries of the proof the verifier sees is called a *view*. An *accepting view* is a view the verifier accepts for some input x and random coin flip r .

By definition, an accepting view contains stars at positions where the verifier never looks at (for a particular random run) and exactly at these positions. Thus, an accepting view can be thought of as a partial proof that causes the verifier to accept. When the verifier is adaptive, the first bit read by him determines the location of the second bit, etc., so even for a fixed r , the locations of the non-stars may vary from view to view. Since the query size is upper bounded by $g(n)$, every accepting view contains at most $g(n)$ entries which are not stars. We call two accepting views consistent if they agree in locations where both are non-stars.

Let $ACC_V(x, r)$ be the set of all accepting views for a verifier V for input x and random string r , and let

$$ACC_V(x) = \{(r, W) \mid |r| = f(n), W \in ACC_V(x, r)\}.$$

Definition 4 (FGLSS Graph $G_V(x)$) Feige et al. [38] define a graph $G_V(x)$ on vertex set $ACC_V(x)$ such that (r, W) and (r', W') are connected if and only if $r \neq r'$ and W and W' are consistent.

Lemma 3 *Let V be an (f, g) -restricted verifier. Then:*

1. $G_V(x)$ has at most $2^{f(n)+g(n)}$ vertices.
2. $G_V(x)$ can be constructed in time polynomial in $\max\{n, 2^{f(n)+g(n)}\}$.
3. $\text{MaxClique}(G_V(x))$ is equal to $\max_P |\{r \mid V(r, P, x) = 1\}|$.

Proof One can easily check that even for an adaptive V , there are at most $2^{g(n)}$ different accepting views for a fixed r . Thus, the total number of accepting views is at most $2^{f(n)}2^{g(n)}$. In order to check whether a pair (r, W) is an accepting view or not, we can run V , which is by definition a polynomial-time machine. By comparing two views bit-wise, we can check in linear time if they form an edge in $G_V(x)$. As a result $G_V(x)$ can be constructed in time polynomial in $\max\{n, 2^{f(n)+g(n)}\}$.

The main observation of FGLSS is that there is a natural many-one map from the set of proofs (or proof candidates) to the set of cliques of $G_V(x)$ such that all maximal clique has at least one inverse image. Indeed, if we map a proof P to the set of all (r, W) pairs, with the property that $V(x, P, r) = 1$ and W is the verifier's view of P when reads r , then these pairs form a clique in $G_V(x)$, since their view

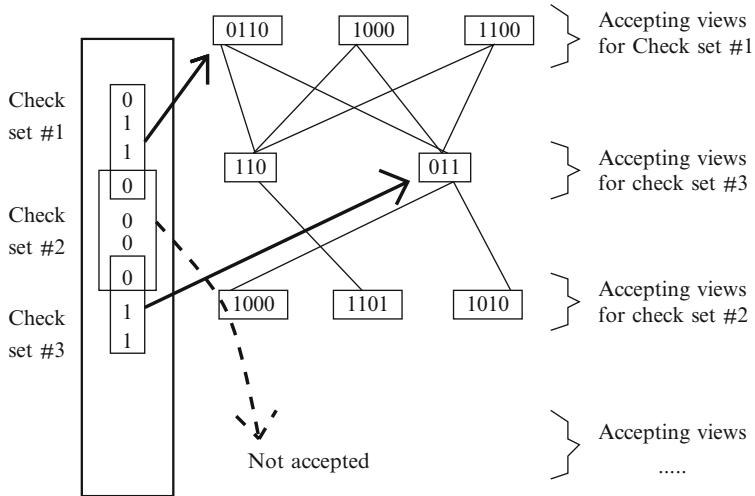


Fig. 2 Construction of the FGLSS graph. The set of all accepted views of a proof map to the set of vertices of a clique (the map is represented by the arrows). The reverse also holds as follows: every maximal clique of the graph corresponds to some proof

components are consistent with each other. Conversely, let C be a maximal clique of $G_V(x)$. There is a string (i.e., a proof) which is consistent with all elements of C . This can be easily seen, since a location is either star in all elements of C or uniquely defined by some elements of C . This string cannot be consistent with any view that is not in C , since C was maximal. Notice that in this correspondence, the size of C is exactly the number of those r s for which there is a W such that $(r, W) \in C$, since W , if exists, is unique for a fixed r . This number further equals to the number of different coin flips for which the verifier accepts any proof that maps to C . Point 3 of the lemma follows. \square

We shall apply Lemma 3 in Sect. 2.8.

2.3 The Gap-3SAT Instance

The gap-3SAT instance is a 3SAT formula, which is either satisfiable or at most $1 - \epsilon$ of its clauses are satisfiable.

Theorem 4 ([4]) *There exists an $\epsilon > 0$ such that for every $L \in NP$, there is polynomial-time computable map from Σ^* to 3SAT instances (mapping $x \in \Sigma^*$ to ϕ_x) such that:*

1. *If $x \in L$, then ϕ_x is satisfiable.*
2. *If $x \notin L$, then every assignment leaves at least ϵ clauses of ϕ_x unsatisfied.*

Indeed, build a PCP for L as in Theorem 1, and define

$$\phi_x = \wedge_r c_r,$$

where c_r is the verifier's accepting criterion for random string r and which by the assumption of [Theorem 1](#) is a disjunct of three literals. The variables are the bits of the proof. If $x \in L$ then all c_i 's evaluate to 1, but if $x \notin L$ then ϵ fraction of the clauses must remain unsatisfied for every assignment, i.e., for every proof of the prover.

The theorem has the immediate consequence that $MAX3SAT$ cannot be approximated in polynomial time to within a factor better than $1/(1-\epsilon)$; otherwise, we could use this algorithm to efficiently compute if $x \in L$. Since $MAX3SAT$ is in $MAXSNP$ (Papadimitriou and Yannakakis [80]), we get

Theorem 5 ([4]) *No $MAXSNP$ -complete problem can be efficiently approximated to within a factor arbitrarily close to 1 unless $P=NP$.*

[Theorem 5](#) does not give explicit non-approximability gaps. For explicit gaps see [Sect. 2.7](#).

2.4 Refined Parameters of PCPs

In [Sect. 1.1](#) we parameterized probabilistic verifiers with the number of random bits, $f(n)$, and the number of check bits $g(n)$ (see [Definition 1](#)). When we make a reduction from a PCP to an NPO problem, the exact non-approximability gap of the latter may depend on several other parameters of the PCP. Here we list some, starting with the most important ones:

1. The *completeness probability*, $q(n)$, which is the least probability of acceptance, when $x \in L$. Originally $q(n) = 1$, but Håstad [53, 54] proved that we can improve on other parameters if we allow $q(n)$ to be slightly less than 1.
2. The *soundness probability*, $p(n)$, which is the greatest probability of acceptance, when $x \notin L$.
3. The *free bit complexity*. In [Sect. 2.2](#) we defined $ACC_V(x, r)$. Using this notation,

$$free(n) = \max_{|x|=n} \log_2 \max_r |ACC_V(x, r)|.$$

4. The *average free bit complexity*:

$$free_{av}(n) = \max_{|x|=n} \log_2 \left(\frac{1}{2^{f(n)}} \sum_r |ACC_V(x, r)| \right).$$

Note that $\sum_r |ACC_V(x, r)|$ is the size of the FGLSS graph.

5. The alphabet size $|\Sigma|$. In most cases we assume that the witness \mathbf{w} is a string over the binary alphabet $\{0, 1\}$. Raz and Safra [87] showed the relevance of using alphabets which may contain up to n symbols.

The parameters we have discussed so far were defined from the model of PCP. We also consider *derived parameters* that can be expressed in terms of the parameters we have defined so far:

1. The *gap* function $gap(n) = q(n)/p(n)$.
2. The *amortized query bit complexity* defined as $\frac{g(n)}{\log_2 gap(n)}$.
3. The *amortized free bit complexity* defined as $\frac{free(n)}{\log_2 gap(n)}$, sometimes as $\frac{free_{av}(n)}{\log_2 gap(n)} = \max_{|x|=n} \log_{q/p} \left(\frac{1}{2^{f(n)}} \sum_r |ACC_V(x, r)| \right)$ [40].

The following notations for PCP classes with refined parameters are more or less standard in the literature:

$FPCP_{q,p}(f, free)$ denotes the class, where the query complexity is replaced with the free bit complexity. $\overline{FPCP}(f, \overline{free})$ is $\cup FPCP_{q,p}(f, free)$, where the union runs through for all choices of $q(n)$, $p(n)$, and $free(n)$ such that the amortized free bit complexity is at most $\overline{free}(n)$.

2.5 Amplification Techniques

Amplification techniques are transformations on probabilistically checkable proofs or on the FGLSS graph that improve on their parameters. An ancestor of these techniques was used in [46] to prove that the chromatic number cannot be approximated to within a factor of $2 - \epsilon$. In this section we review the naive repetition, the parallel repetition, the proof composition, and the randomized graph product (see also in Fig. 3). All of them are playing important roles in the theory of non-approximability.

2.5.1 Naive Repetition

In the case of naive repetition, the new verifier's query is the conjunct of k independently chosen queries of the original verifier. It increases the gap in between the completeness and the soundness probabilities as shown in the table below:

<i>Naive repetition</i>	Original verifier	Repeated verifier
Number of random bits	$f(n)$	$k f(n)$
Number of check-bits	$g(n)$	$k g(n)$
Completeness probability	$q(n)$	$q(n)^k$
Soundness probability	$p(n)$	$p(n)^k$

Perhaps the most troublesome aspect of the naive repetition is that it increases the number of random bits the verifier needs to use. Zuckerman [98] suggests a “less naive” repetition, where the verifier chooses its k -tuple from a set S of polynomially

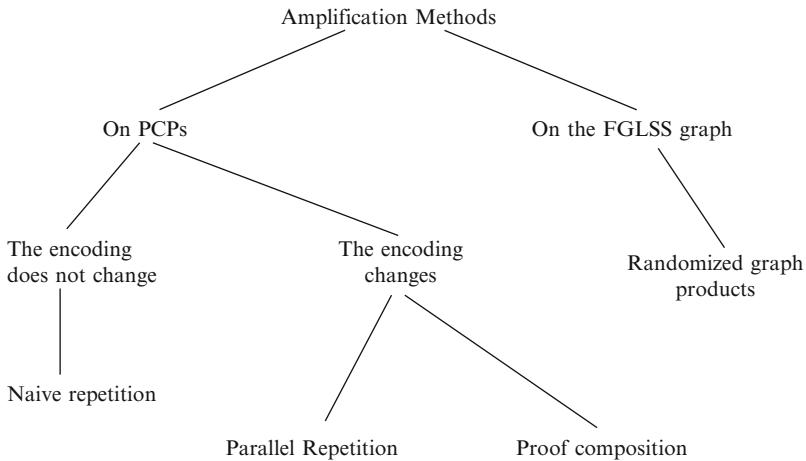


Fig. 3 Various types of amplifications

many k -tuples. Unfortunately, Zuckerman constructs S randomly. A PCP^S verifier is a PCP verifier that works with a (random) advise S . With high probability over a random S with $|S| = 2^{F(n)}$, the following holds for Zuckerman's verifier:

	Original verifier	Zuckerman's verifier
Number of random bits	$f(n)$	$F(n)$
Number of check bits	$g(n)$	$g(n)(F(n) + 2)$
Completeness probability	1	1
Soundness probability	$1/2$	$2^{f(n)-F(n)}$

Lars Engebretsen and Jonas Holmerin [36, Lemma 14] have computed the performance of the Zuckerman's verifier for all values of the completeness and soundness probabilities. This is what they got:

	Original verifier	EH-verifier
Number of random bits	$f(n)$	$F(n)$
Number of check bits	$g(n)$	$g(n)\frac{F(n)+2}{-\log p(n)} = g(n)D$
Completeness probability	$q(n)$	$q^D(n)/2$
Soundness probability	$p(n)$	$2^{f(n)-F(n)}$

In the case of perfect completeness, the EH verifier also has perfect completeness.

2.5.2 Parallel Repetition

Parallel repetition of PCPs, as opposed to naive repetition, requires to change not only the verifier's behavior but also the encoding. It increases the alphabet size in order to keep the query size only two and works as follows.

Assume that an (f, g) -restricted verifier V expects P to be segmented, i.e., P consists of blocks (usually of some fixed block size). Note that the segmentation is something that is only in the verifier's mind, so a cheating prover has no influence on this: segmented proof simply refers to the verifier's reading behavior. Assume that the verifier always accesses data only from two different blocks of the proof. A further restriction is that P consists of two sets of blocks, P_1 and P_2 , such that for every random string, V accesses one block from each set. A PCP with this structural restriction is called a *two-prover system*, since P is being thought to be provided by two different provers whose answers the verifier compares one with another.

Any naive repetition with $k > 1$ would alter the two-prover nature of the system, but parallel repetition keeps it. We shall denote a verifier V repeated k times in parallel by V^k . The true prover for V^k writes down all possible ordered k -tuples of segments from P_1 and all possible ordered k -tuples of segments from P_2 , and these k -tuples are the segments of the repeated provers. The verifier uses $kf(n)$ randomness to simulate k independent checks $(c_{1,1}, c_{2,1}), \dots (c_{1,k}, c_{2,k})$ of V by querying segments $(c_{1,1}, \dots, c_{1,k}) \in P_1^k$ and $(c_{2,1}, \dots, c_{2,k}) \in P_2^k$. V^k accepts iff V would accept all pairs $(c_{1,i}, c_{2,i})$ for $1 \leq i \leq k$.

It is easy to compute the acceptance probabilities if the proof has the P_1^k, P_2^k structure, but there is no guarantee that cheating provers adhere to this structure. Surprisingly we can still say something about the maximum acceptance probability for V^k .

Theorem 6 (Raz's Parallel Repetition Theorem [86]) *For every verifier V of a 2-prover PCP, there is a constant $0 \leq c \leq 1$:*

$$\max_{P'} \text{Prob}_{r_1, \dots, r_k}(V^k(x, P', r_1, \dots, r_k) = 1) \leq c^k, \quad (1)$$

where c depends only on $p(V, x) = \max_P \text{Prob}_r(V(x, P, r) = 1)$ and the maximum segment size for V and is strictly less than 1 if $p(V, x) < 1$. (In Eq. (1) P' ranges through all (possibly cheating) proofs for V^k .)

The properties of the repeated proof are summarized in the table below:

	Original verifier	With k repetitions
Number of random bits	$f(n)$	$kf(n)$
Number of check bits	$g(n)$	$kg(n)$
Completeness probability	$q(n)$	$q(n)^k$
Soundness probability	$p(n)$	c^k (see Theorem 6)

The price we pay for keeping the two-prover structure is a weaker bound on the soundness probability than in the case of naive repetition.

Two-prover systems were introduced by Ben-Or, Goldwasser, Kilian, and Wigderson [17] and have been studied by Rompel and Sipser [43], Feige and Lovász [41], and by many others.

In many PCP constructions, we use a certain folklore two-prover protocol. The verifier of this protocol, which we call $V_{3\text{SAT}}$, comes from the gap-3SAT instance of Sect. 2.3 and works as below.

Let ϕ_x (associated with $x \in \Sigma^*$) be a 3SAT instance, which is either completely satisfiable or at most $1 - \epsilon$ fraction of its clauses are satisfiable for some $\epsilon > 0$. By Theorem 4 we can assume that the language $L = \{x \mid \phi_x \text{ is satisfiable}\}$ is NP-complete and that ϕ_x is polynomial-time computable from x . The verifier $V_{3\text{SAT}}$ computes ϕ_x , picks a random clause in it, and asks the first prover to evaluate the three literals in the clause. If all the three literals evaluate to false, $V_{3\text{SAT}}$ rejects outright. Otherwise, he picks a random variable in the clause and asks the second prover to evaluate this variable (the two provers do not hear each other's answers, but they can agree in a strategy in advance, and they also know x). If the two provers evaluate the variable differently, the verifier rejects, otherwise accepts.

In order to see that the above informal description indeed defines a two-prover system in our former sense, identify P_i with the concatenation of all answers to all possible questions of the verifier to prover i . Each answer is one segment, and the number of segments in P_i equals to the number of different questions the verifier may ask from prover i . The segment size of P_1 is three bits, and the segment size of P_2 is one bit.

Next we show that the above protocol has a gap at least $\epsilon/3$. Clearly, when $x \in L$, i.e., when ϕ_x is satisfiable, both provers play consistently with the lexicographically first satisfying assignment of ϕ_x , and $V_{3\text{SAT}}$ accepts with probability 1. If $x \notin L$, every assignment fails to satisfy at least ϵ fraction of the clauses of ϕ_x . Observe that the strategy of P_2 corresponds to an assignment to the variables of ϕ_x . For the sake of simplicity, we call this assignment P_2 . This assignment must fail to satisfy at least ϵ fraction of the clauses of ϕ_x , and the verifier has probability ϵ that he finds one of these clauses. The first prover of course will lie and will not give the same evaluation to one of the variables in that clause as P_2 (to avoid sure rejection), but then, conditioned on the event that P_2 gets the clause in question, the verifier notices this inconsistency with probability at least $1/3$. Thus, in case $x \notin L$, $V_{3\text{SAT}}$ accepts with probability at most $1 - \epsilon/3$.

Feige [37] shows that we may also assume that in ϕ_x each variable appears exactly in 5 clauses and that each clause contains exactly 3 variables. By Feige's restriction, we may assume that $V_{3\text{SAT}}$ first selects a random variable and then chooses a clause randomly out of the five that contains this variable. Observe that the sequence in which the verifier asks the provers does not matter, and in the sequel we call the prover who provides the clauses the "second prover." We may also assume that the verifier decides at his output after he has heard both provers.

In all applications $V_{3\text{SAT}}$ is going to be repeated in parallel ℓ times, where ℓ depends on the application and ranges from a large constant to $\log n$. The lemma

below, which is a consequence of the PCP theorem and the parallel repetition theorem, summarizes the properties of $V_{3\text{SAT}}^\ell$.

Lemma 4 *Let $\ell = \ell(|x|)$ be an arbitrary function of $|x|$ (but most often a constant), and let $b = \{b_1, \dots, b_N\}$ be the set of variables of Feige's ϕ_x and $c = \{c_1, \dots, c_{5N/3}\}$ be the clauses of ϕ_x . (We say $b_i \in c_j$ if b_i or \bar{b}_i appears in c_j). Then $V_{3\text{SAT}}^\ell$ works as follows:*

1. *It randomly picks an ℓ tuple $U = (b_{i_1}, \dots, b_{i_\ell})$ and checks it against an ℓ tuple $W = (c_{j_1}, \dots, c_{j_\ell})$ such that $b_{i_v} \in c_{j_v}$ for $1 \leq v \leq \ell$. W is called a companion of U , and it is randomly picked from the set of all companions of U . The provers provide evaluations of all variables involved in the queries to them (in particular, the second prover gives the values of all variables in all clauses occurring in W) without listening to each other. The verifier accepts if $V_{3\text{SAT}}$ would accept (b_{i_v}, c_{j_v}) for every $1 \leq v \leq \ell$, i.e., all c_{j_v} are satisfied, and the assignments of the variables in U and W are consistent.*
2. *It has perfect completeness.*
3. *It has soundness at most c^ℓ for some fixed constant $c < 1$.*
4. *It uses $O(\ell \log n)$ random bits and $\ell + 3\ell$ check bits. (Here parameter n is the length of x , and it is in polynomial relation with N .) More importantly, the verifier reads only two segments of the proof. With segment sizes ℓ (bits) for prover one and 3ℓ (bits) for prover two, this is a two-query PCP.*

The above lemma amplifies the PCP theorem and implies approximation hardness of the *label cover* problem:

Label Cover Problem: An instance of the *label cover* problem is a tuple, $\mathcal{P} = (G, \Sigma_1, \Sigma_2, \Pi)$, where:

1. $G = (X, Y, E)$ is a bipartite graph with $V(G) = X \cup Y$ and $E = E(G) = E(X, Y)$.
2. Nodes in X, Y are assigned labels from Σ_1, Σ_2 , respectively.
3. $\Pi = \{\pi_{xy} \mid (x, y) \in E\}$ is a set of functions, $\pi_{xy} : \Sigma_1 \rightarrow \Sigma_2$.

The variables of \mathcal{P} , by definition, are the labels assigned to nodes in $X \cup Y$. For each edge $(x, y) \in E$, $\pi_{xy}(l(x)) = l(y)$ defines a binary “projection” constraint. Here, $l(x), l(y)$, are the labels assigned to x, y , respectively. The value of a label assignment is the fraction of constraints that are satisfied.

A $V_{3\text{SAT}}^\ell$ instance is a special label cover problem \mathcal{P} , where the vertices of X are the ℓ -tuples of clauses of a $V_{3\text{SAT}}$ instance Φ_x and the vertices right partition are the ℓ -tuples of the variables of Φ_x . Therefore, $\Sigma_1 = \{0, 1\}^{3\ell}$ and $\Sigma_2 = \{0, 1\}^\ell$.

Lemma 4 implies:

Theorem 7 *For any positive $\epsilon < 1$, there exist sufficiently large, but constant size alphabets $\Sigma_1 = \Sigma_1(\epsilon), \Sigma_2 = \Sigma_2(\epsilon)$, such that it is NP-hard to approximate label cover instances $\mathcal{P}(G, \Sigma_1, \Sigma_2, \Pi)$ within a factor smaller than $1/\epsilon$.*

Proof Choosing $\ell \in O(\log \frac{1}{\epsilon})$, the soundness of V_{3SAT}^ℓ can be made smaller than ϵ , giving the required hardness. \square

2.5.3 Proof Composition

The idea of composing probabilistic verifiers first appears in the paper of Arora and Safra [5] and is aimed at reducing the number of check bits of a probabilistic proof system.

To describe proof composition, consider a proof system with an (f, g) -restricted verifier V that recognizes a language L . From now on we call V the *outer verifier*. We would like to construct a new verifier V' which recognizes the exact same language as V but uses less check bits. Let $x \in L$, $|x| = n$. The composed proof for x consists of the old proof P , which we call the *core*, and a proof P_r for every choice of the random string r with $|r| = f(n)$ which we describe later. We shall compose proof systems only if their verifiers are non-adaptive. Let Q_r be the set of bits that V reads from P when we run it with random string r . The set of accepting views $ACC_V(x, r)$ can be viewed as a language L_r consisting of words only of length $|Q_r| = g(n)$. Let V_r be an (f', g') -restricted probabilistic verifier that recognizes L_r , where $f'(g(n))$ and $g'(g(n))$ are the number of random bits and check bits V_r needs for its verification. The f' and g' bounds must uniformly hold for each L_r (a more elegant formalism for composition was worked out in [92]). V_r is called the *inner verifier*, and it verifies that the view of the original verifier is accepting, with the help of some extra advise string φ_r , provided separately for every r in addition to the view.

The compound verifier works as follows: it picks a random r with $|r| = g(n)$ and a random string r' with $|r'| = f'(|Q_r|)$. It then runs V_r with random string r' on input Q_r , proof φ_r (reading $g'(|Q_r|)$ bits from the latter), and outputs $V_r(Q_r, \varphi_r, r')$.

Let us compute the parameters of the composed proof system. Assume that V has completeness probability $q(n)$ and soundness probability $p(n)$ and that each V_r has completeness probability $p'(n)$ and soundness probability $q'(n)$. Here is what we get:

	Original proof	Composed proof
Completeness probability	$q(n)$	$q(n)q'(n)$
Soundness probability	$p(n)$	$(1 - p(n))p'(n) + p(n)$
Number of check-bits	$g(n)$	$g(n) + g'(g(n))$

As we see, all parameters are getting worse. Why? The composed verifier uses $g(n) + g'(g(n))$ check bits. The second term is small, but the first term, $g(n)$, is

there, because although Q_r is input from the inner verifier's view point, it is in fact part of the composed proof. The situation would be different if V_r checked its input Q_r together with the associated proof φ_r instead of reading its entire input.

Babai, Fortnow, Levin, and Szegedy [12] have introduced a probabilistic verifier, which reads only a few bits from its input. The price of this efficiency is that the BFLS verifier can recognize a language only if it receives its input encoded. If now the outer verifier is *segmented*, i.e., the outer verifier reads k chunks of the proof, each of length $g(n)/k$, where k is a constant, we may use the BFLS verifier as the inner verifier if we replace the proof of the outer verifier with a one where each segment is encoded. (See Sect. 2.5.2 for an example to a segmented outer verifier.) In [12] it is shown:

Theorem 8 *For every language $L \in NP$ and every polynomial-time encoding function $E_n : \{0, 1\}^n \rightarrow \{0, 1\}^{\text{poly}(n)}$ that defines a code with relative distance ϵ , and associated decoding function D , there is a verifier V^{BFLS} that uses at most $f'(n) = O_\epsilon(\log n)$ random bits, reads at most $e(n) = (\log n)^{O_\epsilon(1)}$ bits of the input and $g'(n) = (\log n)^{O_\epsilon(1)}$ bits of the proof, and has the following properties:*

1. *If the input of the verifier is of the form $z = E(x)$ for some $x \in L$, then the verifier accepts with probability at least 1.*
2. *If V^{BFLS} accepts an input z with probability at most $\frac{1}{2}$, then z decodes to some $x \in L$.*

Assume now that the outer verifier's proof is segmented and that the verifier reads at most k segments for every r , where k is a constant. The composed prover encodes the segments (each individually) with an efficient error-correcting code E' . This will be the core of the composed proof. If we now apply the BFLS verifier of Theorem 8 as the inner verifier, then the last line of our earlier table will change to

	Original proof	Composed proof
Number of check bits	$g(n)$	$e(g(n)) + g'(g(n))$

The above composition technique was employed in [5] and in [4]. A serious effort in these articles had to be made to create segmented outer verifiers. Dinur's proof of the PCP theorem does not use the above exact composition technique, although it also uses proof composition.

2.5.4 Randomized Graph Products

Graph products amplify gaps in sizes of independent sets [47] and chromatic numbers [71]. There are many different types of graph products, but the one that will work for us here is the *inclusive graph product*.

Definition 5 (Inclusive Graph Product) Let G_1 and G_2 be graphs with vertex sets V_1 , resp., V_2 , and edge sets E_1 , resp. E_2 . We define the inclusive graph product of $G_1 \times G_2$ on the vertex set $V_1 \times V_2$ such that (x_1, x_2) is connected with (y_1, y_2) if and only if x_1 is connected with y_1 in G_1 or x_2 is connected with y_2 in G_2 .

Let $\alpha(G)$ be the maximum independent set size of G and $\omega(G)$ be the maximum clique size of G . Independent set sizes are multiplicative with respect to the inclusive graph product. For G^k , the k -wise inclusive graph product of G with itself, means

$$\begin{aligned}\alpha(G^k) &= \alpha(G)^k \\ \omega\left(\overline{G}^k\right) &= \omega(G)^k.\end{aligned}$$

To obtain the behavior of powers of G with respect to the chromatic number, we first define the fractional chromatic number of a graph:

Definition 6 (Fractional Chromatic Number) Let G be an undirected graph and I be a set of G . The indicator function X_I of I is map from $V(G)$ to the reals which assigns 1 to the elements of I and 0 to the elements of $V(G) \setminus I$. Let $Ind(G)$ be the collection of all independent sets of G . We define the fractional chromatic number of G by

$$\chi_f(G) = \min \sum_{I \in Ind(G)} \lambda_I$$

subject to

$$\lambda_I \geq 0 \text{ for all } I \in Ind(G)$$

$$\sum_{I \in Ind(G)} \lambda_I X_I \geq X_{V(G)} \text{ coordinate-wise.}$$

Clearly,

$$\chi(G) \geq \chi_f(G) \geq \max\left(\frac{V(G)}{\alpha(G)}, \omega(G)\right). \quad (2)$$

With respect to powering we have

$$\chi(G^k) \leq \chi(G)^k \quad (3)$$

$$\chi_f(G^k) = \chi_f(G)^k. \quad (4)$$

Equation (4) is observed by Lovász [72]. We wish to amplify constant gaps in the independence number and the fractional chromatic number to polynomial gaps by setting $k = c \log n$, but the size of the resulting graph becomes super-polynomial. In order to overcome this problem, Berman and Schnitger [18] developed a randomized

version of this graph product. Instead of taking G^k , they randomly, select an induced subgraph of it. Feige and Kilian [40] give the following lemma:

Lemma 5 *Vertex-induced subgraphs G' of G^k have the following properties:*

1. $\chi_f(G') \leq \chi_f^k(G)$.
2. *If $\alpha(G) \leq C$, then $\alpha(G') \leq k|V(G)|$ for nearly all subgraphs G' with $\frac{|V(G)|^k}{C^k}$ vertices.*
3. $\chi_f(G') \geq \frac{|V(G')|}{k|V(G)|}$, for all G' satisfying $\alpha(G') \leq k|V(G)|$.
4. $\alpha(G') \geq \frac{|V(G')|}{c^k}$ with high probability over a fixed sized subgraph G' , where $c = |V(G)|/\alpha(G)$.

The lemma lets us amplify the PCP theorem in a simple way to prove polynomial (n^ϵ) gap for the clique approximation problem. The reduction is randomized, however.

2.6 The Long Code

The long code was invented by Bellare, Goldreich, and Sudan [14], and it has become a standard tool to prove exact non-approximability bounds for various CSP. It is also used to prove the non-approximability of Max Clique to within a factor of $|V(G)|^{1-\epsilon}$ in [55].

Let U be an fixed finite set of so-called “labels.” The coordinates of the long code correspond to all possible functions $f : U \rightarrow \{0, 1\}$. When we encode an $\mathbf{x} \in U$, the coordinate corresponding to f takes the value $f(\mathbf{x})$:

$$\text{long}_U(\mathbf{x})_f = f(\mathbf{x}).$$

Since there are $2^{|U|}$ Boolean-valued functions on n inputs, the long code is a string of length $2^{|U|}$. Let A be a function that every $f : U \rightarrow \{0, 1\}$ associates an element $A_f \in \{0, 1\}$. Then A is a word of the long code if and only if there is an \mathbf{x} such that for every f : $A_f = f(\mathbf{x})$.

Alternatively, we can think of the long code as the composition of two encodings: first we encode $\mathbf{x} \in U$ to $\text{unary}(\mathbf{x}) \in \{0, 1\}^{|U|}$, where $\text{unary}(\mathbf{x})$ is one only at a single position, indexed with \mathbf{x} , and zero everywhere else. Then we encode $\text{unary}(\mathbf{x})$ with the *Hadamard encoding*. The latter contains all subset sums of the binary string it encodes, modulo two.

\mathbf{x}	$\text{unary}(\mathbf{x})$	Hadamard
1	0001	\rightarrow 000000011111111
2	0010	\rightarrow 0000111100001111
3	0100	\rightarrow 0011001100110011
4	1000	\rightarrow 0101010101010101

Sometimes we need a technique that is due to Bellare et. al. [14] called *folding*. If $A = \text{long}_U(\mathbf{x})$, then for every f , we have $A_{\neg f} = (\neg f)(\mathbf{x}) = \neg(f(\mathbf{x})) = \neg A_f$. Let $\mathbf{x}_0 \in U$ be a fixed (but otherwise arbitrarily chosen) element of U . For every f exactly one of $f(\mathbf{x}_0)$, $\neg f(\mathbf{x}_0)$ is zero. The folded long code long'_U is obtained by puncturing long_U , keeping only those entries of every word that belong to index f with $f(\mathbf{x}_0) = 0$. This is exactly half of all entries. We, however, think of the omitted entries that they are implicitly present, and for any f with $f(\mathbf{x}_0) = 1$, we define

$$A_f = \neg A_{\neg f}. \quad (5)$$

Unfolding of A makes sense even when A is not in long'_A , but rather an arbitrary 0–1 vector over the index set $\{f : U \rightarrow \{0, 1\} \mid f(\mathbf{x}_0) = 0\}$, if we “read” the entries that are not explicitly present by an application of Eq. (5). PCP constructs sometimes (or oftentimes) contain the folded variation.

2.6.1 Long Code Tests

Long code tests try to find out if in a prospective long code word A , there exists an \mathbf{x} such that A is close to $\text{long}_U(\mathbf{x})$, by the means of checking only a few randomly chosen entries of A . The checking procedure should have the following properties:

1. If $A = \text{long}_U(\mathbf{x})$ for some \mathbf{x} , then the test accepts with probability q ($q = 1$ or it is close to one, depending on the setting).
2. If no such \mathbf{x} is decodable from A and not even a short list of candidates, then the test accepts with probability at most p (in general, $p < q$, typically p is close to zero).

Perhaps the following arguments explain why we want to use such a wasteful encoding in our constructions:

- The long code is very efficiently testable, giving rise to efficient inner verifiers.
- From a long code word or from any word that is close to a long code word, we can efficiently decode the value that *any* Boolean-valued function takes over the encoded label.

Long code tests have either 1. perfect completeness or 2. imperfect completeness. Below we describe important examples to both ones. Our first example is a test with perfect completeness that checks long'_U and is due to Dinur [33]. In the test we

Dinur’s Folded Long Code Check [33].

1. Randomly select f_1, f_2, f_3 , conditioned on $f_1 \vee f_2 \vee f_3 = 1$. Retrieve $b_1 = A_{f_1}$, $b_2 = A_{f_2}$, $b_3 = A_{f_3}$.
2. Accept if $b_1 \vee b_2 \vee b_3 = 1$.

Remark 1 Note that for any word A of the long code, if $f_1 \vee f_2 \vee f_3 = 1$, then $A_{f_1} \vee A_{f_2} \vee A_{f_3} = 1$.

Lemma 6 *For the above checking procedure, the following holds:*

1. *If $A = \text{long}'_U(\mathbf{x})$, then A is accepted with probability 1.*
2. *If $\text{dist}(A, \text{long}'_U) > \delta$, then A is rejected with probability $\Omega(\delta)$.*

The other test is a test with imperfect completeness, and it is due to Håstad [54]. It has historical significance too: the first exact non-approximability results were obtained using this code. We also assume that the input is folded, i.e., Eq. (5) holds. Below \mathcal{F} denotes all functions of the form $f: U \rightarrow \{0, 1\}$.

Hastad's Folded Long Code Check [54]

1. Choose f_0 and f_1 from \mathcal{F} with uniform probability.
2. Choose a function $\mu \in \mathcal{F}$ by setting $\mu(\mathbf{x}) = 1$ with probability $1 - \epsilon$ and $\mu(\mathbf{x}) = 0$ otherwise, independently for every $\mathbf{x} \in U$.
3. Set $f_2 = f_0 \oplus f_1 \oplus \mu$, i.e., define f_2 for each $\mathbf{x} \in U$ by $f_2(\mathbf{x}) = f_0(\mathbf{x}) \oplus f_1(\mathbf{x}) \oplus \mu(\mathbf{x})$.
4. Accept if $A_{f_0} \oplus A_{f_1} \oplus A_{f_2} = 0$.

Lemma 7 *If A is the table of a word in the folded long code, the verifier accepts with probability at least $1 - \epsilon$. If a folded string A passes the test with probability at least $\frac{1+p}{2}$, then there is a B which is the modulo two sum of at most $\epsilon^{-1} \log 1/p$ words of the long code, and A and B agree in at least $\frac{1+p}{2}$ fraction of the $2^{|U|}$ bit positions.*

Remark 2 Notice the difference between the soundness conditions of Dinur's test and Hastad's test. If Dinur's test succeeds with high probability, \mathbf{x} is decodable from the code. In the case of Håstad, it is only list-decodable. Of course, in the case of Håstad "high probability" means $\frac{1+\text{tiny}}{2}$, while in the case of Dinur, it means $1 - \text{tiny}$. Dinur's objective was not to prove sharp results, but to prove the PCP theorem. Her long code test is part of her alphabet reduction step.

When we prove sharp results, the outer verifier plays a role too. Some generic outer verifiers, like the label cover instance of Sect. 2.5.2, lead to a variety of optimal non-approximability results, but the inner verifier is slightly more involved than the verifier of the long code alone. The inner verifier for the label cover problem needs to check two long codes at the same time together with a relation between them. We do not describe the test here, which is very similar to Håstad's long code test, and also appears in [54]. If we replace the label cover problem with the so-called unique game problem (Sect. 4), we get sharper results, but this outer verifier is not proven to be NP-hard.

2.7 Constraint Satisfaction Problems

A general CSP instance is a set

$$\Phi = \{f_i(x_{i,1}, \dots, x_{i,k})\}_{1 \leq i \leq m}$$

of k -variate functions acting on k -tuples of n variables, x_1, \dots, x_n , each ranging in a domain D . Often $D = \{0, 1\}$, but in general it can be any fixed finite set. The goal is to find an assignment to the variables such that all or in the maximization version the maximum number of f_i s is satisfied. Specific CSP have restrictions on the f_i s, namely, they have to come from a constraint family Γ , which is a subset of k -variate functions on D (in the most general setting, “ k -variate” has to be replaced with “at most k -variate”), where k is a fixed constant. CSP problems are in MAXSNP and therefore approximable within some constant factor. Here we discuss MAX k LIN, MAX k SAT and MAX k CSP. All of these problems are over the Boolean domain, i.e., $|D| = 2$.

Problem Name: MAX k LIN:

Instance: A set of linear equations over \mathbf{F}_2 , $L = \{x_{i,1} \oplus x_{i,2} \oplus x_{i,3} = c_i \mid 1 \leq i \leq m\}$ such that for $1 \leq i \leq m$, $1 \leq j \leq 3$: $x_{i,j} \in \{x_i\}_{1 \leq i \leq n}$, where x_i are variables in \mathbf{F}_2 and c_i ($1 \leq m$) are constants in \mathbf{F}_2 .

Witness: An assignment to the variables x_i ($1 \leq i \leq n$).

Objective: Maximize the number of equations set true in L .

Obviously, a random replacement on expectation satisfies half of the constraints. On the other hand, Håstad constructs a PCP reduction [55] which shows as follows:

Theorem 9 ([55]) *For any $\epsilon > 0$, $k \geq 3$ it is hard to approximate MAX k LIN to within a factor of $2 - \epsilon$.*

Håstad’s reduction [54] briefly works as follows: The label cover problem from Sect. 2.5.2 is the outer verifier (one may view it as a starting point of the reduction). A random string of the outer verifier leads to checking an edge (x, y) of the label cover instance for the relation $\pi_{xy}(l(x)) = l(y)$, where $l(x)$ and $l(y)$ are the labels assigned to these nodes. The compound prover encodes each label with the folded long code (with $long'_{\Sigma_1}$ or with $long'_{\Sigma_2}$ depending on the node), and upon the outer verifier picking edge (x, y) , the inner verifier checks the long codes associated with *both* nodes and *also* if the desired relation holds between the labels that these long codes encode. This seems like a lot of checking, but Håstad does the checking very economically, only with three query bits. It becomes important that π_{xy} is a projection. The verification process is very similar to the long code test of Håstad in Sect. 2.6, with a little extra twist, which we do not describe here. The composed verifier accepts with probability $1 - p$ for positive inputs and with probability at least $\frac{1-p}{2}$ for negative inputs, where p can be made arbitrarily small. Moreover, each checking criterion is a linear equation over \mathbf{F}_2 involving three variables. Theorem 9 follows.

Problem Name: MAX k SAT:

Instance: A set of disjuncts $\Phi = \{(t_{i,1} \vee t_{i,2} \vee t_{i,3}) \mid 1 \leq i \leq m\}$ such that for $1 \leq i \leq m$, $1 \leq j \leq 3$ $t_{i,j} \in \{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$, where x_i ($1 \leq i \leq n$) are Boolean variables.

Witness: A Boolean assignment to the variables x_i ($1 \leq i \leq n$).

Objective: Maximize the number of disjuncts set true in Φ .

Johnson [56] in 1974 showed that the MAX k SAT is approximable to within a factor of $\frac{1}{1-2^{-k}}$ in polynomial time as long as each clause contains exactly k literals. Trevisan, Sorkin, Sudan, and Williamson [95] show that MAX3SAT is approximable within 1.249 without any restriction, and Karloff and Zwick [57] show that it is approximable within a factor of 8/7 for satisfiable instances. Here we prove the non-approximability of MAX3SAT.

Theorem 10 ([55]) *For any $\epsilon > 0$, it is hard to approximate MAX3SAT to within a factor of $8/7 - \epsilon$.*

Proof We reduce the problem to the non-approximability of MAX3LIN. Let L be an instance of MAX3LIN such that either at least $1 - \epsilon_1$ fraction of its equations are satisfied or at most $1/2 + \epsilon_2/2$ fraction of its equations are satisfied. We replace every equation $x \oplus y \oplus z$ of L with a set of four clauses $(x \wedge y \wedge \bar{z})$, $(x \wedge \bar{y} \wedge z)$, $(\bar{x} \wedge y \wedge z)$, and $(\bar{x} \wedge \bar{y} \wedge \bar{z})$. An assignment that satisfies the linear equation satisfies all the clauses, while an assignment that does not satisfies the linear equation satisfies three of the four clauses. The MAX3SAT instance Φ we construct for L is the multiset union of these replacements. It is easy to see that either $1 - \epsilon_1$ fraction of the clauses of Φ are satisfied or at most $7/8(1 + \epsilon_2)$ fraction. \square

Håstad has proven the more general statement that MAX k SAT is not approximable within $\frac{1}{1-2^{-k}} - \epsilon$ for any k and $\epsilon > 0$, even if every clause consists of exactly k literals.

Theorem 10 is proven via a specific *gadget reduction*, i.e., where each constraint of a CSP is replaced with a set of constraints from another CSP, called *gadget*. These in general may contain auxiliary variables. Trevisan, Sorkin, Sudan, and Williamson [97] use linear programs to systematically construct gadgets that give optimal gap-CSPs starting from gap-E3LIN2. For a large class of CSPs, they prove that optimal gadgets exist. They arrive at these gadgets via linear programming.

Problem Name: MAX k CSP:

Instance: A set $S = \{f_i \mid 1 \leq i \leq n\}$ of Boolean expressions (constraints), each depending on at most k of the Boolean variables x_i $1 \leq i \leq k$.

Witness: A Boolean assignment to the variables x_i ($1 \leq i \leq n$).

Objective: Maximize the number of expressions set true in S .

The best known algorithm for the MAX k CSP problem has an approximation ratio 2^{k-1} [94]. Samorodnitsky and Trevisan [89] showed that for any $\delta > 0$ and any NP-complete language L , there is a PCP with amortized query complexity $1 + \delta$ (!). Their proof leads to the following:

Theorem 11 ([89]) *For every k the MAX k CSP problem is NP-hard to approximate to within $2^{k-O(\sqrt{k})}$.*

2.8 The Max Clique

Instance: Undirected graph $G = (V, E)$.

Witness: A clique $C \subseteq V$ in G : $(x, y \in C) \wedge (x \neq y) \rightarrow (x, y) \in E(G)$

Objective: $\omega(G) = \max\{|C| : C \text{ is a clique in } G\}$.

Boppana and Halldórsson [20] in 1992 proved that the maximum clique problem can be approximated within a factor of $O(|V|/(\log |V|)^2)$. The hope for a big improvement faded away as non-approximability results arose. Feige, Goldwasser, Lovász, Safra and Szegedy [38] in 1991 made the following connection between clique approximation and proof checking:

Theorem 12 (FGLSS) *Let $NP \subseteq PCP_{q,p}(f, g)$. Then if MaxClique of a graph of size $h(n) = 2^{f(n)+g(n)}$ can be approximated within a factor better than $q(n)/p(n)$ in time polynomial in $h(n)$, then $NP \subseteq DTIME(poly(h(n)))$.*

Proof Let L be an NP -complete language. Let $V(r, P, x)$ be a probabilistic verifier that recognizes L using $f(n)$ query bits, and $g(n)$ random bits, and which accepts for at least $q(n)2^{f(n)}$ choices of r for some P , if $x \in L$ and accepts for at most $p(n)2^{f(n)}$ choices of r for every P , if $x \notin L$.

Consider the FGLSS graph $G_V(x)$ defined in Sect. 2.2. Conditions 1–3 of Lemma 3 imply the theorem, since if we could approximate the Max Clique of $G_V(x)$ within a factor better than $q(n)/p(n)$ in time $poly(h(n))$, then we would be able to use this algorithm to tell whether $x \in L$ or not. \square

Let $f(n) = \log n \log \log n$. Feige et al. show that $NP \subseteq PCP_{1,1/2}(f, f)$. This argument used the scaled-down version of the two-prover interactive protocol by Babai Fortnow and Lund [11] and provided the first rudimentary PCP construction. If we apply $\log^k n$ independent naive repetitions on the above verifier (see Sect. 2.5.1) we obtain

$$NP \subseteq PCP_{1,1/2^{\log^k n}}(f(n) \log^k n, f(n) \log^k n).$$

By Theorem 12 this implies that for every fixed $\epsilon > 0$, Max Clique cannot be approximated within a factor of $2^{\log^{1-\epsilon}|V|}$ unless NP has quasi-polynomial-time algorithms.

The next important step was made by Arora and Safra [5] who showed the NP -hardness of Max Clique up to an approximation factor $2^{\sqrt{\log|V|}}$. This result was the first to show the non-approximability of Max Clique under the natural $P \neq NP$ assumption and provided the first PCP that characterized NP .

Arora, Lund, Motwani, Sudan, and Szegedy [4] proved that $NP = PCP_{1,1/2}(\log n, O(1))$ (see Theorem 1). An amplification method [25] based on the “expander walk” technique of Ajtai, Komlós, and Szemerédi [1] yields that $NP = PCP_{1,1/n}(\log n, \log n)$. Then Theorem 12 gives that there is an ϵ that it is NP -hard to approximate MaxClique up to a factor of $|V|^\epsilon$. Alon, Feige,

Wigderson, and Zuckerman [3] showed how to obtain the same result starting with FGLSS graph constructed from [Theorem 1](#) by derandomizing the graph product of Berman and Schnitger [18].

The constant ϵ was made explicit by Bellare, Goldwasser, Lund, and Russel [15]. They also slightly improved on the proof methods of [4] and coupled it with a new amplification technique of [18, 98] to show that Max Clique is hard to approximate within a factor of $|V|^{1/25}$ unless NP is contained in $co-R\tilde{P}$. Here \tilde{P} stands for the quasi-polynomial time.

Notice that in the last result the hardness condition is different than before, because it involves the randomized class, $co-R\tilde{P}$. BGLR replace the condition $NP \subseteq PCP_{q(n), p(n)}(f(n), g(n))$ of [Theorem 12](#) with $NP \leq_R PCP_{q(n), p(n)}(f(n), g(n))$, where \leq_R stands for “randomly reducible.” Although this weakens the hardness condition, it allows better amplification methods and thus better parameters.

Feige and Kilian [39] have observed that [Theorem 12](#) can be replaced with the following lemma (for the definitions see [Sect. 2.4](#)):

Lemma 8 [39] *Let $NP \subseteq FPCP_{q,p}(f, \text{free})$. Then if Max Clique of a graph of size $h(n) = 2^{f(n)+\text{free}(n)}$ can be approximated within a factor better than $q(n)/p(n)$ in time polynomial in $h(n)$, then $NP \subseteq \text{DTIME}(\text{poly}(h(n)))$.*

From the above lemma they showed that Max Clique cannot be approximated to within a factor of $|V|^{1/15-\epsilon}$ unless $NP = coRP$. Bellare and Sudan in [16] introduced the amortized free bit complexity, $\overline{\text{free}}$ (see [Sect. 2.4](#)), and notice that the best available amplification techniques give that

Lemma 9 *If there is a PCP which uses $O(\log n)$ randomness and has amortized free bit complexity $\overline{\text{free}}$, then Max Clique cannot be approximated within a factor better than $n^{\frac{1}{1+\overline{\text{free}}}}$ unless $NP \subseteq coR\tilde{P}$.*

They prove that Max Clique cannot be approximated to within a factor better than $|V|^{1/4}$ unless $NP \subseteq coR\tilde{P}$. The next major step was made by Bellare, Goldreich, and Sudan [14], who discovered the long code (see [Sect. 2.6](#)), and building it into their PCP immediately could reduce the free bit complexity. They prove the non-approximability of Max Clique to within a factor better than $|V|^{1/3}$ unless $NP \subseteq coRP$.

Although the constant in the exponent gradually improved, it seemed that theoretical limitations bar the improvement of the non-approximability exponent beyond $1/2$. It has turned out that the limitations could be overcome by dropping the perfect completeness property of the PCP. Relying on a new test for the long code of Bellare et. al., Håstad was able to show the following:

Theorem 13 [54] *Max Clique cannot be approximated to within a factor of $|V|^{1-\epsilon}$ for any $\epsilon > 0$ unless $NP \subseteq ZPP$.*

Arora=A, Babai=B, Bellare = Be, Feige=Fe, Goldwasser=G, Goldreich = Go, Håstad =H, Kilian=K, Khot = Kh, Lovász=Lo, Lund=Lu, Motwani=M, Russel = R, Safra=Sa, Sudan=Su, Szegedy=Sz, Zuckerman = Z Engebretsen=En, Holmerin=Ho		
Due to	Factor	Assumption
FeGLoSaSz	$2^{\log^{1-\epsilon} V }$ for any $\epsilon > 0$	$NP \not\subseteq \tilde{P}$
ASa	$2\sqrt{\log V }$	$NP \neq P$
ALuMSuSz	$ V ^\epsilon$ for some ϵ	$NP \neq P$
BeGLuR	$ V ^{1/30}$	$NP \neq coRP$
BeGLuR	$ V ^{1/25}$	$NP \not\subseteq coR\tilde{P}$
FeK	$ V ^{1/15}$	$NP \neq coRP$
BeSu	$ V ^{1/6}$	$NP \neq P$
BeSu	$ V ^{1/4}$	$NP \not\subseteq coR\tilde{P}$
BeGoSu	$ V ^{1/4}$	$NP \neq P$
BeGoSu	$ V ^{1/3}$	$NP \neq coRP$
H	$ V ^{1-\epsilon}$ for any $\epsilon > 0$	$NP \not\subseteq ZPP$
Kh	$ V /2^{\log^{1-\epsilon} V }$ for any $\epsilon > 0$	$NP \not\subseteq ZPP$
EnHo	$ V ^{1-O(\sqrt{\log \log n})}$	$NP \subseteq ZP(2^{O(\log n(\log \log n)^{3/2})})$
Z	$ V ^{1-\epsilon}$ for any $\epsilon > 0$	$NP \neq P$
Z	$ V /2^{\log^{1-\epsilon} V }$ for any $\epsilon > 0$	$\tilde{N}P \neq \tilde{P}$

Fig. 4 A summary table on the history of clique non-approximability

In the proof of this theorem, Håstad uses the same probabilistically checkable proof construction as he does for CSP, but the verifier works differently, and the analysis is different. He is able to achieve amortized free bit complexity ϵ for any $\epsilon > 0$.

Sudan and Trevisan [91] and Samorodnitsky and Trevisan [89] simplified the involved analysis of Håstad and constructed a PCP which has the additional benefit of having amortized query bit complexity $1 + \epsilon$. Building on [91] and [89] Lars Engebretsen and Jonas Holmerin [36] showed that unless $NP \subseteq ZPTIME(2^{O(\log n(\log \log n)^{3/2})})$, Max Clique cannot be approximated to within a factor of $|V|^{1-O(\sqrt{\log \log n})}$. Under the same assumption, Khot [60] was able to increase the non-approximability ratio to $|V|/2^{\log^{1-\epsilon}|V|}$. Using new construction of dispersers, Zuckerman de-randomized the PCP constructions of Håstad and Khot [99]. Fig. 4 summarizes developments on the Clique non-approximation problem.

2.9 The Chromatic Number

Instance: Undirected graph $G = (V, E)$.

Witness: An integer k and a k -coloring F of $V(G)$ with the property that $F : V(G) \rightarrow [1, k]$ and for every $(a, b) \in E(G) : F(a) \neq F(b)$.

Objective: $\chi(G) = \min k$ such that a k -coloring F of $V(G)$ exists.

The best-known lower bound for approximating the chromatic number was obtained by Halldórsson in 1993 [52]. Given a graph G on n nodes, he finds a coloring of G in polynomial time with at most

$$\chi(G) \times O\left(n \frac{(\log \log n)^2}{\log^3 n}\right)$$

colors. From the other direction, an early result of Garey and Johnson [46] shows that $\chi(G)$ is NP-hard to approximate to within any constant less than 2. In 1992 Lund and Yannakakis found a reduction from approximating the Max Clique of the FGLSS graph of the ALMSS verifier to the problem of approximating the chromatic number.

Theorem 14 (Lund and Yannakakis [74]) *There is an ϵ such that it is NP-hard to approximate $\chi(G)$ within a factor of $|V(G)|^\epsilon$.*

The proof of Lund and Yannakakis works in three steps:

1. Construct the FGLSS graph $G_V(x)$ for the ALMSS verifier V of an NP-complete language L . Observe that $\overline{G}_V(x)$ has bounded degree and:
 - a. If $x \in L$ then $\alpha(\overline{G}_V(x)) = R$ (for some $R = n^\epsilon$).
 - b. If $x \notin L$ then $\alpha(\overline{G}_V(x)) \leq R(1 - \epsilon)$.
2. Use the randomized graph products of Berman and Schnitger [18] and Blum [19] to obtain a graph G_1 from $\overline{G}_V(x)$, which has maximum independence number n^{ϵ_1} for positive instances and maximum independence number n^{ϵ_2} for negative instances, for some $\epsilon_1 > \epsilon_2 > 0$.
3. Apply another transformation which turns G_1 into a graph G_2 such that $\chi(G_2) \geq n^{\epsilon_3}$ for positive instances and $\chi(G_2) \leq n^{\epsilon_4}$ for negative instances, for some $\epsilon_3 > \epsilon_4 > 0$.

Improving upon this result and its subsequent sharpening [16, 58], Fürer [44] gives a randomized reduction which shows that if $\omega(G)$ cannot be approximated to within $|V(G)|^{\frac{1}{free+1}}$, then $\chi(G)$ cannot be approximated to within $|V(G)|^\delta$, where $\delta = \min\left(\frac{1}{2}, \frac{1}{2free+1}\right) - o(1)$. His reduction assumes the FGLSS graph structure. Feige and Kilian [40] have taken a different route and building on the Max Clique result of Håstad [54] show:

Theorem 15 [40] *Unless $NP \subseteq ZPP$, it is hard to approximate $\chi(G)$ to within $|V(G)|^{1-\epsilon}$ for any constant $\epsilon > 0$. Here ZPP denotes the class of languages that are*

solvable with a randomized algorithm that makes no error and on expectation runs in polynomial time.

The proof of this theorem gives more, namely, that it is NP-hard (under randomized reductions) to distinguish between graphs with $\alpha(G) \leq |V(G)|^\epsilon$ and graphs with $\chi(G) \leq |V(G)|^\epsilon$. To understand their proof scheme, we need to introduce a new parameter for a PCP which requires the notion of fractional chromatic number of Sect. 2.5.4 (Definition 6).

Definition 7 (Covering Parameter) The covering parameter of a PCP with verifier V is

$$\rho = \min_{x \in L} \frac{1}{\chi_f(\overline{G}_V(x))}. \quad (6)$$

Here $G_V(x)$ is the FGLSS graph for V and x (see Definition 4).

Since for every graph G we have $\chi_f(G) \geq \omega(G)$, the covering parameter of a PCP for input x is at most $2^{-\text{free}(|x|)}$. The notion of covering parameter first appears in [40] in the context of RPCPs. RPCPs are PCPs such that for every $x \in L$, there is a probability distribution (possibly different for different inputs) on all the proofs. For a fixed $x \in L$, this distribution gives rise to a probability distribution on $\text{Ind}(\overline{G}_V(x))$, which in turn can be used to give a lower bound on the covering parameter of the PCP. This lower bound is what Feige and Kilian call the covering parameter of the RPCP. Whether we talk about RPCPs or immediately define the covering parameter of a PCP is a matter of taste. RPCPs were introduced because their definition suggests an elegant connection to zero knowledge proof systems.

Feige and Kilian do their crucial transformation directly on the FGLSS graph of the PCP theorem, and *then* they amplify it with the randomized graph product. This is the reverse of how Lund and Yannakakis do it. Here is what Feige and Kilian rely on.

Assume that for an NP-complete language L , there is a polynomial-time transformation that sends a word x into a graph $G(x)$ such that for some constants $0 < c_2 < c_1 < 1$:

$$\text{If } x \in L \text{ then } \chi_f(G(x)) \leq 1/c_1. \quad (7)$$

$$\text{If } x \notin L \text{ then } \alpha(G(x)) \leq c_2|V(G(x))|. \quad (8)$$

Then, if we apply the randomized graph product construction of [18] on $G(x)$ with $k = c_3 \log_{c_2^{-1}} |V(G(x))|$, where $c_3 > 1$ is a constant, and select a random subgraph G' of G^k with size c_2^{-k} , then from Lemma 5 of Sect. 2.5.4,

$$\chi_f(G') \leq |V(G')|^{\frac{\log c_1}{\log c_2}} \quad \text{if } x \in L, \quad (9)$$

$$\alpha(G') \leq |V(G')|^\epsilon \quad \text{if } x \notin L, \quad (10)$$

where ϵ can be made arbitrarily small if c_3 is large enough. We immediately obtain the following:

Lemma 10 *Assume that for an NP-complete language L there is a polynomial-time transformation that sends a word x into a graph $G(x)$ such that for some constants $0 < c_2 < c_1 < 1$ conditions (7) and (8) hold. Then for every $\epsilon > 0$, we can construct a graph G' in randomized polynomial time such that*

$$\chi(G') \leq |V(G')|^{\frac{\log c_1}{\log c_2} + \epsilon} \text{ if } x \in L, \quad (11)$$

$$\chi(G') \geq |V(G')|^{1-\epsilon} \text{ if } x \notin L. \quad (12)$$

Indeed the Inequality (11) follows from Inequality (10) and Inequality (12) is implied by Inequality (9) and by the following result of Lovász [72]:

$$\chi(G) \leq \chi_f(G) \log(1 + \alpha(G)). \quad (13)$$

If we combine Lemma 10 with Definition 7 of the covering parameter, we obtain the following:

Lemma 11 *Suppose that one can generate a PCP for NP with $f(n)$ random bits, soundness probability $p = p(n)$, average free bit complexity $f_{av} = free_{av}(n)$, and $\rho = \rho(n)$. Assume that p , f_{av} , and ρ are constants and that the size of the FGSSS graph, $2^{f(n)+f_{av}}$, is polynomially bounded as n grows to infinity. Then it is hard to approximate $\chi(G)$ to within $|V(G)|^{\frac{f_{av}-\log \rho + \log \rho}{f_{av}-\log p} - \epsilon}$, where ϵ is an arbitrarily small positive constant, assuming $NP \not\subseteq ZPP$.*

Starting from the construction of Håstad [55], Feige and Kilian show the existence of PCPs for an NP-complete problem such that $\log \rho/(f_{av} - \log p)$ is arbitrarily small. Theorem 15 is implied now by Lemma 11.

We can also use Lemma 10 to derive the Lund Yannakakis result from the non-approximability result from the MAXSNP-hardness of the MAX-3-coloring problem proved by Petrank [82]. This example was given by Feige and Kilian, and it is so easy that we can give their entire proof here:

Proof (Theorem 14) Let H be the graph of Petrank's construction such that it has m edges and either it has a valid three-coloring or every three-coloring of its vertices miscolor at least qm edges, where $q > 0$ is some fixed constant. From H we construct a graph G , which has $6m$ vertices of the form (e, c) , where e ranges over the m edges of H and c ranges over the 6 valid 3-colorings of the two endpoints of an edge. Two vertices (e_1, c_1) and (e_2, c_2) are connected by an edge if e_1 and e_2 intersect at a vertex of H and c_1 and c_2 disagree on the coloring of this vertex. A valid 3-coloring C of H induces an independent set of size m in G . Let π be a permutation of the three colors. As π ranges over its 6 possibilities,

the composition of C with π induces 6 independent sets in G of size m . Furthermore, these independent sets are disjoint and cover $V(G)$. Hence, G has a chromatic number of 6. If H is not 3-colorable (and hence qm of its edges are miscolored) then the largest independent set of G is of size at most $m(1 - q)$. If we apply Lemma 10 for G with $c_1 = 1/6$, $c_2 = (1 - q)/6$, we get Theorem 14. \square

The above non-approximability of $\chi(G)$ was derandomized by Zuckerman [99] to get the same non-approximability ratio conditioned on $P \neq NP$.

Finally we mention another type of non-approximability result for $\chi(G)$. S. Khanna, N. Linial, and S. Safra [58] have shown that it is NP-hard to tell apart 3-chromatic graphs from 5 chromatic graphs. But the following question is open and would be very interesting to resolve:

Problem 1 Prove that for any fixed $k > 5$, it is NP -hard to tell apart a k -chromatic graph from a three-chromatic graph.

Remark 3 Recently, Dinur, Mossel, and Regev have shown [35]: For any two fixed integers $Q > q > 2$, it is hard to decide whether for an input graph G we have $\chi(G) \leq q$ or $\chi(G) \geq Q$. The hardness is under the 2-1 conjecture, if $q \geq 4$, and under a similar, but slightly more complicated “fish shaped” variant of this conjecture, if $q = 3$ (see the definition of 2-1 conjecture in Sect. 4).

2.10 Set Cover

Instance: A collection $F = \{S_1, \dots, S_s\}$ of subsets of $S = \{1, \dots, n\}$.

Witness: A subcollection F' of F such that $\cup_{S_i \in F'} S_i = S$.

Objective: $v(F) = \min |F'|$.

In 1974 D. Johnson [56] showed that the greedy algorithm finds a collection F' for the problem such that $|F'|$ is to within $\log n$ factor optimal (here the \log is based on $e = 2.71\dots$). Chvatal [24] extended this algorithm to the weighted version, and Lovász [72] has studied a linear programming relaxation with logarithmic (in terms of the hypergraph degree) integrality gap.

The first hardness result is that of Lund and Yannakakis [74]. They show using a construction coming from the PCP theory that set cover cannot be approximated to within a factor of $\frac{\log n}{4}$ unless $NP \subseteq TIME(n^{\text{polylog} n})$ and within a factor of $\frac{\log n}{2}$ unless $NP \subseteq ZTIME(n^{\text{polylog} n})$. Here $ZTIME(t)$ denotes the class of problems for which there is a probabilistic algorithm that makes no error and runs in expected time t .

Subsequent works went in two different directions. The best results up to date that represent these directions are that of Feige [37] which proves that set cover cannot be approximated efficiently to within a factor of $(1 - o(1)) \log n$ unless $NP \subseteq TIME(n^{O(\log \log n)})$ and that of Raz and Safra [87] which proves that set cover is NP-hard to approximate within a factor of $\epsilon \log n$ for some fixed $\epsilon > 0$. Until now

there is no result which would achieve both optimal non-approximability ratio and hardness condition $P \neq NP$.

A predecessor of [37] was the result of Bellare et.al. [15], which proved that the set cover cannot be approximated within any constant ratio unless $P = NP$ and that it cannot be approximated within a factor of $\frac{\log n}{4}$ unless $NP \subseteq \text{TIME}(n^{O(\log \log n)})$. Arora and Sudan [6] followup on [87] and achieve the same bound using elegant and difficult algebraic techniques such as Hilbert's Nullstellensatz.

We cannot give here the technically very involved result of Raz and Safra, but we present a complete proof of Feige's result. The proof is a modification of a simplified argument due to Håstad (A Rewriting of Feige's Proof for the Setcover, Unpublished). At many places we use his original wording. We ought to remark that simplification is formal to a large extent: the key components of the proof are the same as those in [37].

Theorem 16 [37] *The set cover cannot be approximated efficiently to within a factor of $(1 - o(1)) \log n$ unless $NP \subseteq \text{TIME}(n^{O(\log \log n)})$. The logarithm is $e = 2.71\dots$ based.*

Proof A partition system $B(m, p, k, d)$ [37] is a system of p partitions, $(p_i)_{i=1}^p$, on a basis set B with $|B| = m$ such that:

1. Each p_i ($1 \leq i \leq p$) consists of k parts. The parts of p_i are denoted $p_{i,j}$ ($1 \leq j \leq k$).
2. No collection $(p_{i_l, j_l})_{l=1}^d$ of d partition segments with all i_l distinct covers the universe.

In other words if we only use sets from different partitions to cover, we need at least d sets, while a small cover is given by the k sets of one p_i . \square

Lemma 12 (Feige [37]) *For any $c > 0$, there exists a $B(m, p, k, d)$ partition system with $p \leq (\log m)^c$, k an arbitrary constant, and $d = (1 - f(k))k \ln m$, where $f(k)$ tends to 0 when k tends to infinity and m is large enough compared to k and c .*

Feige shows that a random partition system satisfies Properties 1–2 with high probability. Since $d = O(\log m)$, we can quickly check if the Properties 1–2 hold. Deterministic constructions are also known. Next we describe the PCP reduction to the set-cover problem.

Our starting point is the probabilistic verifier V_{3SAT}^ℓ of Lemma 4 in Sect. 2.5.2 for an NP-complete language L with the choice of $\ell = c_0 \log \log n$, where we shall choose $c_0 > 0$ to be large enough. Recall that this verifier is associated with a gap-3SAT instance ϕ_x with N Boolean variables. Note that $|x| = n$ is in polynomial relationship with N . On the other hand, the instance of set cover we construct will have slightly super-polynomial size in n .

The verifier sends a subset U of variables with $|U| = \ell$ to the first prover who answers with an assignment σ_U to these variables. Independently, he sends a W companion of U to the second prover, i.e., a set of ℓ clauses containing these

variables. The second prover answers with the evaluation of the 3ℓ variables in W . The verifier rejects if not all clauses are satisfied in W or if σ_U is not equal to the projection $\pi_U(\sigma_W)$, i.e., the assignment that σ_W gives to U . What is important for our purposes is that if $x \notin L$, the verifier accepts with probability at most c_1^ℓ for some fixed $0 < c_1 < 1$. The structure of ϕ_x determines all companion pairs U, W and all the projections π_U , and we can compute these in $DTIME(n^{O(\ell)})$ from x . From now on, when we talk about a U, W pair, we always mean a pair, where W is a companion of U .

Let $k > 0$ be a constant such that $f(k)$ of Lemma 12 is less than ϵ . For each U and each k -tuple $W_1, W_2 \dots W_k = \vec{W}$ of possible W companions of this U , we construct a separate partition system on a new set of m points. We have N^ℓ different U and, given the choice of U , each W_i can be chosen in 5^ℓ ways. We thus have $R = N^\ell 5^{k\ell}$ partition systems, and with the choice of $m = R^{\frac{1}{\epsilon}}$, the total size of the universe is $R^{1+\frac{1}{\epsilon}} = m^{1+\epsilon}$. We set the further parameters of the partition systems as $p = 2^\ell, k$, and $d = (1-f(k))k \ln m$. Since $p = 2^{c_0 \log \log n} = (\log n)^{c_0}$ and $\log m > \log n$, the conditions of Feige's lemma hold, and the existence of such partition systems is guaranteed. Notice that $m = n^{O(\log \log n)}$ for a fixed ϵ . A particular set in one of the partition systems has an index given by $U, \vec{W}, \alpha \in \{0, 1\}^\ell$ and $i \in [k]$, and we denote it by $S_{U, \vec{W}, \alpha, i}$.

The sets in our set-cover instance are indexed by W and β where $\beta \in \{0, 1\}^W$ should be thought of as an answer from the second prover on the question W . We denote it by $T_{W, \beta}$ and it is a union of $S_{U, \vec{W}, \alpha, i}$ which satisfy

$$(W_i = W) \wedge (\pi_U(\beta) = \alpha),$$

where we of course assume that β satisfies the clauses used to construct W .

Let $Q = (5N/3)^\ell$ be the number of different W . We first have the simple lemma telling us what happens when $x \in L$.

Lemma 13 *If $x \in L$, the associated set system has a cover of size Q .*

Proof We cover the universe with $\{T_{W, \sigma_W}\}_W$, where σ_W is the answer of the second prover to question W . Since the answers are consistent with those of the first prover, for each U and \vec{W} , there is some σ_U so that the chosen sets contain $S_{U, \vec{W}, \sigma_U, i}$ for all i . This σ_U is simply the answer of the first prover to question U . By definition, the system $\{S_{U, \vec{W}, \sigma_U, i}\}_{1 \leq i \leq k}$ covers the m points of the associated U, \vec{W} pair. \square

Thus, we just need to prove the lemma below (and choose c_0).

Lemma 14 *If $x \notin L$, the associated set system has a no cover of size $dQ / (k(1 + \epsilon))$.*

Proof Suppose we have a cover of size rQ . For each W there is a set A_W of assignments β on W such that $T_{W, \beta}$ belong to the cover. By assumption

$$\sum_W |A_W| = rQ. \quad (14)$$

We note that none of the A_W is empty since we need to cover the partition systems with $W_i = W$ for all i . Let us call a W *good* if $|A_W| \leq r(1 + \epsilon)$. We claim that for appropriate choice of c_0 either $r > \log m$ or there is U, \vec{W} such that

1. $\pi_U(A_{W_i})$ ($1 \leq i \leq k$) are pairwise disjoint.
2. W_i ($1 \leq i \leq k$) are good.

Let us recall a simple lemma in set theory:

Lemma 15 *Let A_1, \dots, A_q be sets of size at most s with the property that no k of them are pairwise disjoint. Then there is an element which is contained in at least $\frac{1}{(k-1)s}$ fraction of the sets.*

Proof Without loss of generality we may assume that A_1, \dots, A_l are pairwise disjoint, but for all $j > l$, the set A_j intersects $A = \cup_{i=1}^l A_i$. Note that by our assumption $l \leq k - 1$; hence, $|A| \leq (k - 1)s$. Since every A_i intersects A , there must be an element of A which is contained at least $\frac{1}{(k-1)s} \leq \frac{1}{|A|}$ of the sets. \square

Let

$$S_U = \{\pi_U(A_W) \mid W \text{ is a companion of } U, \text{ and } W \text{ is good}\}$$

be a set system on the assignments for U , and let σ_U be an assignment which is contained in as large fraction of elements of S_U as possible (If S_U is empty then σ_U is arbitrary). Consider the strategy of the two provers when to question U the first prover answers σ_U , and to question W the second prover answers with a random $\sigma_W \in A_W$. (So the strategy of the second prover is randomized. An averaging argument shows that there is an equally good deterministic strategy.)

If for a fixed U , there is no \vec{W} such that both Conditions 1 and 2 hold, then among the good companions of U , there are no W_1, \dots, W_k such that $\pi_U(A_{W_i})$ ($1 \leq i \leq k$) are pairwise disjoint. By applying Lemma 15 to S_U , we obtain that there is an assignment to U which occurs in at least $\frac{1}{(k-1)r(1+\epsilon)}$ fraction of the sets in S_U , and if fact σ_U is such. In what follows we assume that Conditions 1 and 2 do not hold for any U, \vec{W} .

By Eq. (14) the expected value of $|A_W|$ for a random W is r , so for at least $1 - (1 + \epsilon)^{-1}$ fraction of W s, we have $|A_W| \leq r(1 + \epsilon)$. The verifier chooses such a W with probability at least $1 - (1 + \epsilon)^{-1}$. Conditioned on the intersection of this event (i.e., that W is good) and the event that the verifier picks some fixed U , by our earlier remark, the verifier chooses a W such that $\pi_U(A_W)$ contains σ_U with probability at least $\frac{1}{(k-1)r(1+\epsilon)}$. Hence, the probability of the event that the verifier picks a U, W pair such that W is good and $\sigma_U \in \pi_U(A_W)$ is at least

$$\frac{1 - (1 + \epsilon)^{-1}}{(k - 1)r(1 + \epsilon)}. \quad (15)$$

In the case of this event, there is a $\frac{1}{|A_W|} \geq \frac{1}{r(1+\epsilon)}$ chance over $\sigma_W \in A_W$ that $\pi_U(\sigma_W) = \sigma_U$. Combining this with (15), we get that V_{3SAT}^ℓ accepts the described strategy of the two provers with probability at least

$$\frac{1 - (1 + \epsilon)^{-1}}{(k - 1)r(1 + \epsilon)} \frac{1}{r(1 + \epsilon)} = \Omega(r^{-2}). \quad (16)$$

Either $r > \log m$ or since $\log m = O(c_0 \log n \log \log n)$, we can choose a large enough c_0 that depends on ϵ, k, c_1 , and $\log_n N = O(1)$, such that $2^{-c_1 \ell} = 2^{-c_1 c_0 \log \log n}$ becomes smaller than Expression (16), and we arrive at a contradiction.

It follows from the contradiction that there exist an U and \vec{W} such that Conditions 1 and 2 hold. Fix this choice of U and \vec{W} and consider how the elements from the corresponding partition system are covered. Since $|W_i| \leq r(1 + \epsilon)$ for $1 \leq i \leq k$, the cover system has at most $kr(1 + \epsilon)$ sets. By the disjointness of $\pi_U(A_{W_i})$ s, these sets all come from distinct partitions. Using the property of the cover system, we conclude that $kr(1 + \epsilon) > d$.

In the case $r > \log m$, since we have $d \leq (1 - f(k))kr$, we conclude the same. We are done, since the cover size $|rQ| > dQ/k(1 + \epsilon)$, as claimed. \square

Since the universe for the set-cover instance has total size $t = m^{1+\frac{1}{\epsilon}}$, we get that the quotient of the minimum cover sizes when $x \notin L$ versus when $x \in L$ is

$$\frac{1 - f(k)}{1 + \epsilon} \log m = \frac{1 - f(k)}{(1 + \epsilon)^2} \log t,$$

which tends to $\log t$ when $k \rightarrow \infty$ and $\epsilon \rightarrow 0$.

2.11 Some More Non-approximability Results

The label cover problem from Sect. 2.5.2 and its variants have served as the starting point for several known optimal non-approximability results assuming $P \neq NP$. The table below describes some further known approximation hardness results based on the NP -hardness of the label cover. Let γ_p be the p^{th} moment of the Gaussian random variable with expectation 0 and variance 1.

Problem	Known approx.	Inapprox.	Ref.
Max Cut	≈ 1.1389 .	$\frac{17}{16} - \epsilon$	[48, 55]
L_p Groth. Problem	γ_p^2	$\gamma_p^2 - \epsilon$	[51, 67]
L_p Subspace approx.	γ_p	$\gamma_p - \epsilon$	[32, 51]

The L_p Grothendieck problem is a maximization problem where the input is a symmetric $n \times n$ matrix $A = (a_{ij})$ with zero diagonal entries. The objective

is to maximize the quadratic multilinear function $\sum_{i,j \in [n]} a_{ij} x_i x_j$ subject to $\sum_{i=1}^n |x_i|^p \leq 1$ for $x_i \in \mathbb{R}$. The L_p subspace approximation problem is a generalization of the well-known *least squares regression* problem. The input to the problem is a set of m points a_1, \dots, a_m in \mathbb{R}^n . The objective of the problem is to find a k -dimensional subspace H such that $\sum_{i=1}^m dist(H, a_i)^p$ is minimized. Here $dist(H, a_i)$ is the ℓ_2 distance from a_i to H . The non-approximability results for the L_p Grothendieck problem and subspace approximation problem are from a recent manuscript by Guruvanam, Raghavendra, Saket, and Wu [51] who use a stronger, *smooth* variant of the label cover from [42].

3 A Short Proof of the PCP Theorem

Before [33] and [34], several different proofs have been made for the PCP theorem, but their rough structure did not differ much. In 2005, Dinur published a combinatorial proof for the PCP theorem. One feature of Dinur's proof is that it can be fully explained without mentioning PCPs and talking only about gap-CSPs. The only hint to the old PCP ideas is the use of the long code.

Definition 8 An instance of a $[k, \Sigma]$ CSP problem, where Σ is a constant size alphabet, consists of a set x_1, \dots, x_n of variables that take their values from Σ and a set of m constraints, where each constraint is a k -ary relation for a subset of k variables. The instance is satisfiable if there is an assignment to the variables which satisfies all constraints. We identify $[k, \Sigma]$ CSP with the language of satisfiable instances.

The gap version of the problem is analogous to the gap- k -SAT problem:

Definition 9 $Gap([k, \Sigma]CSP, p, q)$ for $0 \leq p < q \leq 1$ is the problem, where we output 1 on $[k, \Sigma]$ CSP instances, that have an assignment satisfying at least q fraction of the constraints and 0 on those instances with no assignment satisfying more than p fraction of the constraints. On any other instances the output is arbitrary.

Fact 1 Every PCP for a language L , with completeness q , soundness p , query size $k = O(1)$, and witness-alphabet Σ can be interpreted as a Karp reduction from L to $Gap([k, \Sigma]CSP, p, q)$ and vice-versa.

Proof Recall that a Karp reduction from a language $L \subseteq \Sigma^*$ to the gap version of a maximization problem OPT , with lower and upper thresholds p, q , respectively, is a polynomial-time computable function f from Σ^* to instances of OPT such that:

1. If $x \in L$, then $OPT(f(x)) \geq q$
2. If $x \notin L$, then $OPT(f(x)) \leq p$.

Let now $V(x, P, r)$ be the verifier of a PCP for L , which queries at most k bits nonadaptively. For fixed x and r , there is a k -ary relation $\varphi_{x,r}$ expressing if the verifier accepts or rejects the entries of the proof P it views. We have

$$\max_y \text{Prob}(V(x, P, r) = 1) = \max_P |\{r | \varphi_{x,r}(P) = 1\}| / 2^{|r|}. \quad (17)$$

The problem on the right-hand side of (17) is a gap $[k, \Sigma]$ CSP problem, and the completeness and soundness conditions of the PCP translate to the gap requirements. The converse is also straightforward: Let Φ_x be a $[k, \Sigma]$ CSP instance that we get from x by the Karp reduction for L that we have assumed to exist. This is a PCP when we treat the assignment to the variables of Φ_x as proof P . The verifier of this PCP checks if P satisfies a random clause of Φ_x . The completeness of this system is q and the soundness is p . \square

Till the end of the section we are going to focus only on gap-CSP instances.

Theorem 17 (Dinur's Theorem) *Let $\Sigma = \{0, 1\}^3$. Then there exists $\epsilon > 0$ such that $[2, \Sigma]$ CSP Karp reduces to Gap ($[2, \Sigma]$ CSP, $1 - \epsilon$, 1).*

Remark 4 Let $\Sigma = \{0, 1\}^3$. Then the $[2, \Sigma]$ CSP is NP-complete, and the PCP theorem follows.

For the rest of the section, we fix $\Sigma = \{0, 1\}^3$.

Instance Size and Satisfiability Gap

For a $[2, \Sigma]$ CSP instance $\Phi = \bigwedge_{i=1}^m \Phi_i$ (in Sect. 2.7 Φ_i was denoted by f_i), we define the *instance size* as $|\Phi| = m$. The *satisfiability gap* of Φ is $\overline{\text{sat}}(\Phi) = \min_P |\{i | \Phi_i(P) = 0\}| / m$ (one minus the maximal fraction of the simultaneously satisfiable constraints by any assignment P).

Let Φ be a $[2, \Sigma]$ CSP instance. Either $\overline{\text{sat}}(\Phi) = 0$ (the formula is satisfiable) or $\overline{\text{sat}}(\Phi) \geq 1/|\Phi|$ (the formula is not satisfiable). For unsatisfiable instances, the satisfiability gap cannot be smaller than $1/|\Phi|$, since if the formula is not satisfiable, then at least one component of Φ is not satisfied under any assignment. Dinur constructs a reduction that enlarges this tiny gap to a constant. The same reduction, if applied to a satisfiable instance, leaves the resulting instance satisfiable. The latter property will be straightforward, so we shall focus on unsatisfiable instances. The reduction proceeds in $O(\log_2 m)$ steps, making small progresses at a time.

It is sufficient to show that there exists a (sufficiently large) constant $C > 0$ and an $\epsilon > 0$ such that any $[2, \Sigma]$ CSP instance Φ can be reduced in polynomial time to a $[2, \Sigma]$ CSP instance Φ' that has the following properties:

1. If Φ is satisfiable, then Φ' is satisfiable.
2. $|\Phi'| \leq C|\Phi|$.
3. $\overline{\text{sat}}(\Phi') \geq \min\{2 \overline{\text{sat}}(\Phi), \epsilon\}$.

To get [Theorem 17](#), we start from the original instance and apply the above reduction on it $\log_2 m$ times. The size of the final instance is polynomially bounded by that of the original instance (i.e., by m). To construct the reduction sequence takes polynomial time.

3.1 The Three Sub-Steps of Dinur's Basic Reduction Step

We obtain Φ' from Φ of the previous section in three steps: (1) structural Improvement, (2) gap amplification, (3) alphabet reduction. We say that a $[2, \Sigma]$ CSP instance is d -regular, expanding, if the graph we obtain by replacing each constraint with the corresponding pair of variables is a d -regular expander. Let Φ be an arbitrary $[2, \Sigma]$ CSP. Let $d = 11$ and t be a large enough constant to be determined later. The steps are as follows:

$$\begin{aligned} \Phi &\in [2, \Sigma]\text{CSP} & \rightarrow & \text{(Structural improvement)} \\ \Phi_{\text{reg}} &\in d\text{-regular, expanding } [2, \Sigma]\text{CSP} & \rightarrow & \text{(Gap amplification)} \\ \Phi_{\text{big}} &\in [2, \Sigma^{(d+1)^{\lceil \frac{t}{2} \rceil}}]\text{CSP} & \rightarrow & \text{(Alphabet reduction)} \\ \Phi' &\in [2, \Sigma]\text{CSP} \end{aligned}$$

Furthermore:

- If Φ is satisfiable, then so are Φ_{reg} , Φ_{big} and Φ' .
- There are constants C' and C'' such that

$$|\Phi_{\text{reg}}| \leq C' |\Phi|, \quad |\Phi_{\text{big}}| = (d + 1)^{\lceil \frac{t}{2} \rceil - 1} |\Phi_{\text{reg}}|, \quad |\Phi'| \leq C'' |\Phi_{\text{big}}|.$$

- There are $\epsilon > 0$, $D > 1$, and $\delta > 0$ with $D\delta \geq 20$ such that the satisfiability gaps change as follows:

$$\begin{aligned} \overline{\text{sat}}(\Phi_{\text{reg}}) &\geq 0.1 \overline{\text{sat}}(\Phi) \\ \overline{\text{sat}}(\Phi_{\text{big}}) &\geq \min\{D \overline{\text{sat}}(\Phi_{\text{reg}}), \epsilon/\delta\} \\ \overline{\text{sat}}(\Phi') &\geq \delta \overline{\text{sat}}(\Phi_{\text{big}}). \end{aligned}$$

Notice that from Φ to Φ' the satisfiability gap increases by a factor of at least $0.1D\delta \geq 2$, unless it is already ϵ . Only the $\Phi_{\text{big}} \rightarrow \Phi'$ reduction requires classic PCP ideas, namely, the long code.

3.1.1 Structural Improvement

The $\Phi \rightarrow \Phi_{\text{reg}}$ reduction is a fairly standard transformation, which starts with creating $\deg v$ clones of every node v of the constraint graph of Φ . We distribute the outgoing edges among the clones, making every clone an end-point of exactly one outgoing edge. Then for every clone group (associated with the same original node),

we place a degree 5 expander on the members of the clone group (each member is one node of the expander), and we put equality constraints on the new edges. This way the entire graph will be a 6-regular graph, and it may not be an expander itself, since we assume nothing about the structure of Φ . To ensure the expanding property, we now add new edges with empty constraints on them in such a way that the new edges form a degree 5 expander on all nodes. The final graph is $d = 11$ -regular, and it is an expander (since expander + any graph = expander). Throughout the whole construction, we preserve multiple edges (possibly with different constraints). The parameter changes are easy to calculate.

3.1.2 Gap Amplification

The second reduction ($\Phi_{\text{reg}} \rightarrow \Phi_{\text{big}}$) is a wonderful new addition to *PCP* theory, and this is the one that gains us the gap. We define an operation on binary constraint systems called *powering*. Let G be a constraint graph and $t > 1$ be an integer. First we add a loop to each node (with an empty constraint). We denote the resulting graph with $G + I$. Then we construct $(G + I)^t$ in such a way that:

- The vertices of $(G + I)^t$ are the same as the vertices of G .
- Edges: u and v are connected by k edges in $(G + I)^t$ iff the number of paths of length t from u to v in $G + I$ (if the path includes a loop, it also counts towards the length) is exactly k .
- Alphabet: The alphabet of $(G + I)^t$ is $\Sigma^{(d+1)^{\lceil t/2 \rceil}}$, where every vertex (when the prover is honest) specifies values for all of its neighbors reachable in $\lceil t/2 \rceil$ steps.
- Constraints: The constraint associated with an edge (u, v) of $(G + I)^t$ is satisfied iff the assignments for u and v are consistent with an assignment that satisfies all of the constraints induced by the union of the $\lceil t/2 \rceil$ neighborhoods of u and v .

If G is satisfiable, then $(G + I)^t$ is satisfiable as well. To see what happens with the negative instances, Dinur shows that powering has the following gap-enlarging property:

Lemma 16 (Amplification Lemma [33]) *Let Σ be an arbitrary constant size alphabet. There exists a constant $\gamma = \gamma(d, |\Sigma|) > 0$ such that for any $t > 0$ and for any d -regular expanding constraint graph G ,*

$$\overline{\text{sat}}((G + I)^t) \geq \gamma \sqrt{t} \min \left\{ \overline{\text{sat}}(G), \frac{1}{t} \right\}.$$

We define $\Phi_{\text{big}} = (\Phi_{\text{reg}} + I)^t$. Parameter t has to be chosen so that we get a large enough $\overline{\text{sat}}(\Phi_{\text{big}})/\overline{\text{sat}}(\Phi_{\text{reg}})$ ratio to compensate for the loss in the satisfiability gap in the first and third transformations and even gaining a factor of two over that. In other words, **Lemma 16** ensures that we can choose D to be sufficiently large.

3.1.3 Alphabet Size Reduction

$(\Phi_{\text{big}} \rightarrow \Phi')$: The gap amplification step has increased the alphabet size from Σ to Σ_{big} , and now we have to reduce the alphabet to Σ without losing much of the gap we have gained.

Since $\Sigma = \{0, 1\}^3$, we can identify $\Sigma_{\text{big}} \stackrel{\text{def}}{=} \Sigma^{(d+1)^{\lceil \frac{l}{2} \rceil}}$ with $\{0, 1\}^{3(d+1)^{\lceil \frac{l}{2} \rceil}}$. If Φ_{big} is satisfiable, the true prover encodes each variable of the satisfying assignment (i.e., element of Σ_{big}) with some encoding function $E : \{0, 1\}^{3(d+1)^{\lceil \frac{l}{2} \rceil}} \rightarrow \{0, 1\}^{10 \times 3(d+1)^{\lceil \frac{l}{2} \rceil}}$ that corrects constant fraction of errors (from coding theory we know that such encoding exists). To represent the bits of the code word, we shall use Σ -valued variables, somewhat wastefully $10 \times 3(d+1)^{\lceil \frac{l}{2} \rceil}$ of them. The necessity of the error-correcting encoding will be explained later.

These are, however, not all the variables the true prover provides. If v_i and v_j are two variables on which Φ_{big} has a constraint, we install a Dinur assignment tester for the (v_i, v_j) pair. This means an additional constant number of Σ -valued variables per every constraint of Φ_{big} . For a constraint on (v_i, v_j) Dinur's tester checks, using the additional information given by the prover, if the desired relation holds between the (encoded) labels of v_i and v_j . The details are as follows:

Lemma 17 (Assignment Tester of Dinur) *Let h be an arbitrary positive integer, $\mathcal{T} \subseteq \{0, 1\}^h$. Then there are positive integers l and f , and a 2-query verifier V , that uses a random string $r \in \{0, 1\}^f$ accesses a pair of oracles (theorem, $P \in \{0, 1\}^h \times \Sigma^l$) and satisfies:*

1. *If $\text{theorem} \in \mathcal{T}$, then there is $P \in \Sigma^l$ such that $\text{Prob}_r(V(\text{theorem}, P, r) = 1) = 1$.*
2. *If $z \in \{0, 1\}^h$ is η -far from all words in \mathcal{T} , then for every $P \in \Sigma^l$, we have $\text{Prob}_r(V(z, P, r) = 0) \geq \eta/100$.*

It is crucial, that the verifier does not read theorem that it verifies.

We do not prove Lemma 17 here, only mention, that to construct P of the true prover, Dinur employs the folded long code together with her test described in Sect. 2.6 and standard techniques.

For every constraint (v_i, v_j) of Φ_{big} , Dinur installs an assignment tester to verify the property:

$$\mathcal{T}_{i,j} = \{(E(\sigma_i), E(\sigma_j)) | (\sigma_i, \sigma_j) \in \Sigma_{\text{big}}^2 \text{ satisfy all constraints of } \Phi_{\text{big}} \text{ on } v_i, v_j\}.$$

Why is error-correcting encoding E necessary at all, when Dinur's assignment tester in Lemma 17 does not require encoded input? Notice that the success of the test depends on the distance of the assignment from \mathcal{T} . The tester should not accept its input with high probability, unless there is an underlying assignment for Φ_{big} that satisfies the constraint that the tester tests. If $\mathcal{T}_{i,j}$ was simply defined as the collection of all (σ_i, σ_j) laid out in binary (without any encoding) that satisfy all constraints of Φ_{big} on v_i, v_j , then if Σ_{big} is large, changing a single bit in the pair

could make the constraint of Φ_{big} fail on (σ_i, σ_j) , yet the assignment tester would still accept it with high probability.

In contrast, if we change one bit of $E(\sigma_i) \cup E(\sigma_j)$, σ_i is still decodable from $E(\sigma_i)$ and σ_j from $E(\sigma_j)$, using the closest code-word decoding. (It is crucial that the decoding procedure depends on the members of the pair individually and not on the entire pair globally.) In fact, one has to change a constant fraction of the bits of $E(\sigma_i) \cup E(\sigma_j)$ to make either σ_i or σ_j un-decodable. Thus, when the constraints of Φ_{big} are replaced with assignment testers and the average acceptance probability of all assignment testers is $1 - \epsilon'$, the closest distance decoding decodes to an assignment for Φ_{big} that has satisfiability gap at most ϵ'/δ , where δ is some fixed constant.

Summarizing the above, the true prover transforms the proof $\sigma_1 \dots \sigma_n$ of Φ_{big} as follows:

$$\begin{array}{ll} \sigma_1 \dots \sigma_n & \rightarrow \quad (\text{Encodes the alphabet}) \\ E(\sigma_1) \dots E(\sigma_n) & \rightarrow \quad (\text{Adds assignment testers}) \\ E(\sigma_1)' \dots E(\sigma_n)' P_{i_1, j_1} \dots P_{i_{m'}, j_{m'}}. & \end{array}$$

Here $m' = |\Psi_{\text{big}}|$, and $P_{i,j}$ is the proof of the assignment tester for constraint (i, j) . The apostrophes in $E()$ ' refer to the embedding of every bit of $E()$ into an element of Σ : $0 \rightarrow 000$ and $1 \rightarrow 100$. The new tests are those of the assignment testers, combined, appropriately weighted.

It is easy to see that when Φ_{big} is satisfied and the prover is faithful to the protocol, all tests are accepting. Assume now that Φ_{big} is a negative instance. As was sketched a little earlier, in this case the satisfiability gap of Φ' is at least $\delta_{\text{sat}}(\Phi)_{\text{big}}$ for some fixed δ , *independently of* $|\Sigma_{\text{big}}|$ (independence of $|\Sigma_{\text{big}}|$ is crucial, since we need the freedom to set t in the $\Phi_{\text{reg}} \rightarrow \Phi_{\text{big}}$ step to achieve a gap enlargement that more than compensates us for the cumulative loss in the $\Phi_{\text{big}} \rightarrow \Phi'$ and the $\Phi \rightarrow \Phi_{\text{reg}}$ steps). The transformation blows up the instance size by only a constant factor (the constant depends on $|\Sigma_{\text{big}}|$, but it is all right). Since the tester looks at binary constraints over the alphabet $\Sigma = \{0, 1\}^3$, by Fact 1 it corresponds to a $[2, \{0, 1\}^3]CSP$.

4 The Unique Games Conjecture

Amplification of the PCP theorem via parallel repetition results in the non-approximability of the label cover problem. When the label cover is composed with the long code inner verifier, Hastad's Fourier analytic technique yields optimal non-approximability bounds for problems like MAX3SAT [55], MAX3LIN [55], and Max Clique [54]. It is not known, however, how to prove such results for 2CSPs like vertex cover, MAX2LIN, MIN2SAT Deletion. The primary barrier is that it is not possible to efficiently perform both the code-word and consistency tests for projection constraints with just two queries. The barrier gets larger, when the image of the projection becomes much smaller than the domain. This state of affairs compelled Subhash Khot [61] to investigate what happens if we substitute the

$P \neq NP$ assumption with the assumption that label cover with bijective constraints are hard to approximate for large label size. Optimally, this could be NP-hard too, but as of now no one can prove this.

An instance of the *unique games* (\mathcal{UG}) problem (*bipartite unique games problem*) is a tuple $\mathcal{U} = (G, \Sigma, \Pi)$, where:

1. $G = (V, E)$ is a graph ($G = (V, W, E)$ is a bipartite graph with bipartition V, W) with edge set E .
2. Nodes in V (in both V and W) are assigned labels from Σ .
3. $\Pi = \{\pi_{vw} \mid (v, w) \in E\}$, is a set of bijections (permutations), $\pi_{vw} : \Sigma \rightarrow \Sigma$.

An assignment $l : V \rightarrow \Sigma$ ($l : V \cup W \rightarrow \Sigma$) is a labeling of the nodes with the elements of Σ . An edge (v, w) is satisfied iff $\pi_{vw}(l(v)) = l(w)$. Notice that unique games are special label cover instances, where projections are bijections.

Conjecture 1 (Unique Games Conjecture (UGC) [61]) *For every $\epsilon, \delta > 0$, there exists a fixed alphabet Σ , such that $\text{Gap}([2, \Sigma]\mathcal{UG}, \delta, 1 - \epsilon)$ is NP-hard.*

A weaker version of the conjecture is that $\text{Gap}([2, \Sigma]\mathcal{UG}, \delta, 1 - \epsilon)$ is not in polynomial time.

Remark 5 Khot, Kindler, Mossel, and O'Donnell [63] proved that the UGC is equivalent to the special case where each projection constraint is a linear constraint modulo q , i.e., $\Sigma = [q]$ and π_{uv} is defined via $l(u) = l(v) + c_{uv} \pmod q$. Khot and Regev [65] proved that it is enough to consider constraint graphs that are *left regular*.

Conditioned on the conjectured hardness of unique games, [61] was able to prove optimal bounds for problems including MIN2SAT Deletion and MAX2LIN.

4.1 Algorithms for Unique Games

Before discussing further remarkable consequences of the UGC, let us investigate in what sense the unique games problem is different from (easier than) the general label cover problem. Several algorithms solve the unique games in polynomial time for special cases, but none is strong enough to disprove the conjecture. In general, these algorithms impose relations between the parameters ϵ, δ , and $|\Sigma|$ or constraints on the underlying graph, and their existence tells us that one has to be careful with the parameters and the structure of the instance, when trying to prove the conjecture.

It is in P to tell if a unique games instance is perfectly satisfiable. It is also easy to see that a random assignment of labels to the nodes satisfies at least $\frac{1}{|\Sigma|}$ fraction of edges on expectation, implying $\delta > \frac{1}{|\Sigma|}$.

From the result of Charikar, Makarychev, and Makarychev [22], it follows, that if $\epsilon \in O\left(\frac{1}{\log |\Sigma|}\right)$, the UGC is in polynomial time (regardless of δ).

In order to disprove the UGC, it is enough to have a polynomial-time algorithm that finds an assignment that satisfies $\delta(\epsilon)$ fraction of the constraints, given a $1 - \epsilon$

satisfiable unique games instance and $\delta \rightarrow c$ as $\epsilon \rightarrow 0$, where $c > 0$ is some universal constant. Importantly, δ should not depend on $|\Sigma|$.

Motivated by sparsest cut, researchers have considered the special case, where the unique games graph, G , is a good *spectral expander*. If $\lambda = \lambda_2(G)$ is the second smallest eigenvalue of the Laplacian of G , Arora, Khot, Kolla, Steurer, Tulsiani, and Vishnoi [8] (improved by Makarychev and Makarychev [75]) give a polynomial-time algorithm with $\delta(\epsilon) = 1 - \frac{\epsilon}{\lambda} \log \frac{\lambda}{\epsilon}$.

In a recent result Arora, Barak, and Steurer [7] give an algorithm with $\delta(\epsilon) = 1 - \epsilon^\alpha$ that runs in $\exp(n^{\epsilon^\alpha})$ time for a fixed constant $\alpha > 0$. Underlying their analysis is a graph decomposition theorem, stating that the vertex set of a graph can be partitioned in such a way that each resulting subgraph has few large eigenvalues ($\leq n^{O(\epsilon)}$) and there are at most a constant fraction (dependent on ϵ) of edges that go across the parts. The result involves applying this decomposition to the unique games graph and then solving the unique games problem for the induced subgraphs in the partition, using similar methods to the one used by Kolla [68].

We refer the reader to Khot's survey [62], which has an excellent discussion of many of the details.

4.2 Variants of the Unique Games Conjecture

There are several non-approximability results that have been proved starting from variants of Unique Games. Here we mention some of the variants.

A $d \rightarrow 1$ game is a special case of the label cover problem, and it is a generalization of unique games, where $|\Sigma_1| = d|\Sigma_2|$ and each projection constraint π is a $d \rightarrow 1$ map, i.e., $|\pi^{-1}(x)| \leq d$ for every $x \in \Sigma_2$. When $\Sigma_1 = [dq]$ and $\Sigma_2 = [q]$, we denote such a game by $[2, q]\mathcal{PG}_{d \rightarrow 1}$. For every positive integer $d \geq 2$, Khot [61] makes the following conjecture:

Conjecture 2 ($d \rightarrow 1$ -conjecture [61]) *For every $\epsilon > 0$, there exists a $q = q(\epsilon)$ such that $\text{Gap}([2, q]\mathcal{PG}_{d \rightarrow 1}, \epsilon, 1)$ is NP-hard.*

The $d \rightarrow 1$ -conjecture has the perfect completeness property which is useful in some situations. It is shown by O'Donnell and Wu [78] that given a satisfiable instance of a Boolean 3CSP, it is $2 \rightarrow 1$ -hard to find an assignment that satisfies more than $\frac{5}{8}$ fraction of the constraints.

Although the unique games is easy if the underlying graph is a good expander, it may be still hard if the graph is mildly expanding. A variant of the UGC states exactly this:

Conjecture 3 (UGC with expansion, [8]) *There exists a universal constant $1/2 < t < 1$ with the following properties: For every $\epsilon, \delta > 0$, there exists a $\Sigma(\epsilon, \delta)$ such that given a unique games instance $\mathcal{U} = (G, \Sigma, \Pi)$, it is NP-hard to distinguish between the following cases:*

- \mathcal{U} is at least $1 - \epsilon$ -satisfiable.
- \mathcal{U} is at most δ -satisfiable and for every partition $A \dot{\cup} B \dot{\cup} C$ of the vertex set of \mathcal{U} , where $|A| \leq 0.001|V|$ and $|B|$ and $|C| \geq 0.1|V|$, the (B, C) cut has at least ϵ^t fraction of the edges across.

Among the consequences of the above conjecture is that the balanced separator problem, and therefore the sparsest cut problem, is hard to approximate within any constant factor.

4.3 Optimal Bounds Under the Unique Games Conjecture

Assuming the unique games conjecture, exact non-approximability bounds can be proved for several classes of problems. These include max-CSPs [83], strict CSPs [69], ordering problems [50], and clustering problems [64]. There are also problems for which the UGC does not give optimal bounds but still better ones than the $P \neq NP$ assumption alone. These include graph partitioning [2, 23, 66] and graph coloring [35, 60]. Some of the results, like those for graph coloring, are based on variants of the UGC. There are several excellent surveys that have details regarding these problems and the UGC [62, 96]. In this section, we discuss some outstanding consequences of the UGC, including Raghavendra's general hardness result for CSPs.

4.3.1 Optimal Non-approximability for Max Cut

Given a graph G , the Max Cut problem is to find a bipartition of $V(G)$ such that the fraction of edges crossing between the parts is maximized. In [55] an $\frac{17}{16} - \epsilon$ non-approximability bound was shown under $P \neq NP$. The best polynomial-time algorithm, due to Goemans and Williamson [48] approximates Max Cut within a factor of ≈ 1.1389 and employs breakthrough *semidefinite programming (SDP)* techniques.

The celebrated Goemans–Williamson (GW) approximation bound, which is precisely $\frac{\pi}{2} \max_{0 < \theta \leq \pi} \frac{1 - \cos \theta}{\theta}$, at first was not conjectured to be optimal. In a surprise move, however, Khot, Kindler, Mossel, and O'Donnell [63] proved that assuming UGC, the GW ratio is essentially the best possible. Their result was later beautifully extended by Raghavendra [83], who showed that a natural class of semidefinite programs solves all CSP optimally under UGC. We shall sketch the ideas in KKMO in this section and its generalization in Sect. 4.3.3.

For proving the non-approximability of Max Cut, we build our PCP from a bipartite unique games instance $\mathcal{U} = (G, \Sigma, \Pi)$ with $G = (V, W, E)$ as the outer verifier. By Remark 5 we will assume that the instance is left regular, i.e., that all $v \in V$ have the same degree. Depending on how closely we want to get to the GW bound, we choose the alphabet size, $|\Sigma| = R$, of \mathcal{U} to be an appropriately large constant. Next, we need to set a parameter $-1 < \rho < 0$ optimally. For any ρ in this range, we get that to tell instances apart that are $\frac{1-\rho}{2}$ satisfiable (positive instances) from instances that are slightly more than $\frac{\arccos \rho}{\pi}$ satisfiable (negative instances) is unique games hard. The choice for ρ that gives the largest relative gap leads to

the GW bound. The inner verifier is designed a little differently from the usual reductions, where the outer verifier is the label cover problem:

- The (true) prover encodes the labels, $l(w)$, of every $w \in W$ with the long code long_R . (But it does not use the labels of $v \in V$ anyhow!)
- The verifier picks a vertex $v \in V$ at random and two of its neighbors $w, w' \in W$ at random. Let $\pi = \pi_{vw}$ and $\pi' = \pi_{vw'}$ be the respective bijections for edges (v, w) and (v, w') .

At this point we apply the following inner verifier:

1. Let y^w and $y^{w'}$ be the supposed long codes of the labels of w and w' , respectively.
2. Pick $f : R \rightarrow \{0, 1\}$ uniformly and randomly.
3. Pick $\mu : R \rightarrow \{0, 1\}$ by choosing each function value independently to be 0 with probability $\frac{1}{2} + \frac{1}{2}\rho$ and 1 with probability $\frac{1}{2} - \frac{1}{2}\rho$.
4. Accept iff $y_g^w \neq y_{g'}^{w'}$, where $g = f \circ \pi$ and $g' = (f \circ \pi') \oplus \mu$. Here operation \oplus means the exclusive or of two Boolean-valued function.

It is easy to see that the completeness of the above test is almost (where the “almost” comes from the imperfect completeness of the outer verifier) at least $\frac{1}{2} - \frac{1}{2}\rho$. Analyzing the soundness requires Fourier analytic techniques and the brilliant majority is stablest theorem of Mossel, O’ Donnell, and Oleszkiewicz [77]. If the soundness of the above PCP is slightly bigger than $(\arccos \rho)/\pi$, then a solution for \mathcal{U} exists satisfying a small, but constant fraction of the constraints (independently of M), which is a contradiction, if M was chosen large enough (which makes the soundness of the outer verifier small enough). Without giving the details of the calculation, we just hint that reason for this is that for those $v \in V$, whose average “neighbor-pair” inner verifiers have average success rate at least $(\arccos \rho)/\pi + \text{small constant}$, a “probabilistic” (supposed) long code word exists, namely, $\frac{1}{\deg v} \sum_{w \sim v} \pi_{v,w} y^w$ (notice, we needed to shuffle the coordinates of y^w), from which a label l for v can be decoded, which correlates with the (supposed) long codes for v ’s neighbors as follows: The individual (supposed) long codes, y^w , of a constant fraction of the neighbors w of v have the property that from them we can decode a constant length list of labels such that at least one from the list is $\pi_{v,w}(l)$. A standard random assignment technique (which picks a random label from each list) now gives a labeling for \mathcal{U} , which satisfies a constant fraction of the constraints. Important was that the list for each w was recovered only from the supposed long code of w , without looking at v .

4.3.2 Semidefinite Programming and Integrality Gap

Semidefinite programming has resulted in the best known approximation ratios for various classes of optimization problems. For the performance analysis of semidefinite programs, it is crucial to understand the notion of *integrality gap*.

Fix instance Φ of a maximization problem Λ (think of Λ as Max Cut). The first step in designing an SDP-based algorithm for Φ is to translate it to a quadratic program QP_Φ , such that the optimal solutions for Φ and QP_Φ are the same. Then we *relax* the variables, allowing them to take any real vector values. Multiplications between the variables of QP_Φ become scalar products ($xy \rightarrow (\vec{x}, \vec{y})$) in the corresponding semidefinite programming instance, SDP_Φ .

An optimal solution, $SDOPT_\Phi$, to SDP_Φ is then computed, which is *rounded* to a solution of QP_Φ . Let OPT_Φ be the optimal value for Φ (and QP_Φ). Suppose the value of the rounded solution for SDP_Φ is $ROUND_\Phi$. To prove an approximation ratio of r_Φ , it is sufficient to have $r_\Phi ROUND_\Phi \geq SDOPT_\Phi$. The approximation ratio follows because

$$SDOPT_\Phi \geq OPT_\Phi \geq ROUND_\Phi \geq \frac{1}{r_\Phi} SDOPT_\Phi. \quad (18)$$

A barrier to this rounding technique is the *integrality gap* of the relaxation, $IGap_\Lambda = \max_\Phi \frac{SDOPT_\Phi}{OPT_\Phi}$. A lower bound on the integrality gap is usually proved by exhibiting a suitable sequence of instances Φ with integrality gap arbitrarily close to $IGap_\Lambda$.

4.3.3 Exact Non-approximability for All CSPs

In a remarkable recent result, Raghavendra [83] proved that assuming the UGC, SDP rounding algorithms yield essentially the best possible approximation ratios achievable in polynomial time. This result has confirmed former beliefs about the extraordinary strength of semidefinite programming. For every CSP Λ , Raghavendra explicitly constructs semidefinite programs whose maximal integrality gap is the optimal non-approximability bound for Λ .

We will describe the SDP for any 2-variable CSP maximization problem. Let Λ be a 2-variable CSP over the alphabet $\Sigma = [q]$. We can think of each instance Φ of Λ as a directed graph $G = (V, E)$ with nodes denoting variables and edges denoting constraints. Each edge (u, v) has a corresponding *pay-off* function $P_{uv} : \Sigma^2 \rightarrow [0, 1]$ (generalized from $\{0, 1\}$). Without loss of generality, we can assume that the constraints have nonnegative weights w_{uv} summing up to one, which is a probability distribution over E . Our goal is to find an assignment $l : V \rightarrow \Sigma$ such that

$$\sum_{(u,v) \in E} w_{uv} P_{uv}(l(u), l(v))$$

is maximized. When we turn this into a quadratic program, for every node u of G and for every element $i \in \Sigma$, we introduce a variable u_i that takes value one, if u gets label i in the optimal solution, and zero otherwise. The quadratic program is the same in appearance as the analogous semidefinite relaxation, where the u_i 's are now vectors and the “dot” means scalar product:

SDP_Λ

$$\text{Maximize } \sum_{(u,v) \in E} w_{uv} \left(\sum_{i,j \in [q]} P_{uv}(i, j) u_i \cdot v_j \right)$$

Subject to,

$$\forall u, v \in V, i, j \in [q] : u_i \cdot u_j = 0, u_i \cdot v_j \geq 0$$

$$\forall u \in V : \sum_{i \in [q]} |u_i|^2 = 1$$

$$\forall u, v \in V : \sum_{i,j \in [q]} u_i \cdot v_j = 1$$

Raghavendra [83] creates a long code test $Test_A(\Phi)$ from instance Φ , whose completeness and soundness ratio is the integrality gap of Φ . The long code test is a generalization of the folded long code, where $\{0, 1\}$ valued functions are replaced with $[q]$ valued functions. The key idea is that to define the query distribution of $Test_A(\Phi)$, Raghavendra uses an optimal solution of the above SDP. For each edge $(u, v) \in E$ and for each node $u \in V$, we define distributions $D_{uv} : Prob[i, j] = u_i \cdot v_j$ and $D_u : Prob[i] = |u_i|^2$, respectively. The following is the long code test for $z \in [q]^{q^R}$:

$Test_A(\Phi, \epsilon)$ for $z \in [q]^{q^R}$

Choose an edge $(u, v) \in E$ according to w_{uv}

Pick R independent copies from D_{uv} to obtain $x, y \in [q]^R$

Form \hat{x} from x by replacing each x_i independently, with probability ϵ , from the distribution D_u

Form \hat{y} from y by replacing each y_i independently, with probability ϵ , from the distribution D_v

Return “yes” with probability $P_{uv}(z_{\hat{x}}, z_{\hat{y}})$

From the definition of the test it is fairly straightforward that the completeness of the test $Test_A(\Phi)$ is $SDOPT_\Phi - o_\epsilon(1)$. Recall that completeness means the probability with which a word of the long code is accepted, minimized over all long code words.

To talk about soundness, Raghavendra first defines what it means that a word is far away from all words of the long code. In his definition, this happens when the word to be checked is pseudorandom, which he expresses in terms of the Fourier coefficients of the word. The soundness analysis is nontrivial and crucially depends on Mossel’s invariance principle of [76]. It turns out that the soundness cannot be much larger than OPT_Φ . More precisely Raghavendra shows that any word w that is sufficiently pseudo-random gives a randomized rounding procedure round_w to round the optimal solution of the semidefinite program $SDP_A(\Phi)$, yielding an integral solution (i.e. a solution to the original Φ), whose value is close to the success probability of $Test_A(\Phi)$ on w . Here “close” means in terms of ϵ and the pseudorandomness parameters. To understand this beautiful duality better we refer the reader to [83].

Using the unique games outer verifier of [63] in the manner as described in Sect. 4.3.1 for the special case of Max Cut, we obtain the following result:

Theorem 18 *Let Φ be an instance of the CSP A . Then for every $\gamma > 0$, there exist $\epsilon, \delta > 0$ such that there is a polynomial-time reduction from $\text{Gap}(\mathcal{UG}, \delta, 1 - \epsilon)$ to $\text{Gap}(A, SDOPT_\Phi - \gamma, OPT_\Phi + \gamma)$.*

4.3.4 Small Set Expansion and Unique Games

The unique games conjecture has contributed to the great progress in understanding optimal non-approximability bounds. Since unique games conjecture is so central,

it is natural to look for equivalent ways of stating it. Finding natural approximation-equivalent problems to unique games, however, have eluded us for a long time. To remedy this situation, Raghavendra and Steurer [84] consider the *small set expansion (SSE)* problem, a natural and widely applicable graph partitioning problem. They observe that the current techniques do not seem to be able to solve the problem and hence propose the *small set expansion conjecture (SSEC)*; that SSE is NP-hard. They proved that the small set expansion conjecture implies the unique games conjecture. In the rest of the section, we will state the small set expansion conjecture and briefly discuss the recent results related to this conjecture.

Let $G = (V, E)$ be a d -regular undirected graph, $E(A, B)$ for $A, B \subseteq V$, $A \cap B = \emptyset$ be the set of edges between A and B , and μ be the uniform measure on V . The *edge expansion* of a subset of vertices S is defined as $\Phi_G(S) = \frac{|E(S, V \setminus S)|}{d|S|}$ and $\Phi_G(\delta) = \min_{\mu(S) \approx \delta} \Phi_G(S)$, where $0 \leq \delta \leq 1$, and $a \approx b$ means $1/2 \leq a/b \leq 2$. (We have opted to use Φ , which is a standard notation for conductance, and has nothing to do with the Φ of earlier sections, where it denotes two-query PCP instances.) Then $\text{Gap}(SSE, \eta, \delta)$ problem is distinguish between the cases $\Phi_G(\delta) \geq 1 - \eta$ and $\Phi_G(\delta) \leq \eta$. Raghavendra and Steurer make the following conjecture:

Conjecture 4 (SSE Conjecture) *For every $\eta > 0$, there exists a $\delta > 0$ such that $\text{Gap}(SSE, \eta, \delta)$ is NP-hard.*

Small set expansion comes up in several contexts in relation to unique games. If a unique games instance \mathcal{U} with alphabet Σ satisfies at least $1 - \epsilon$ fraction of the edges, then the *label extended* graph has a less-than- ϵ -expanding set of measure $\frac{1}{|\Sigma|}$. In the first step of their sub-exponential algorithm for unique games, [7] use a graph decomposition scheme that partitions the unique games graph into almost-non-expanding small sets. Also, all known integrality gap instances for unique games and various other CSPs have very good small set expansion properties.

The small set expansion conjecture is interesting for several reasons. As is the case with the unique games conjecture, it is in a way a “win–win” situation; a positive resolution would of course prove the unique games conjecture and a negative resolution would solve a very important natural problem, likely with powerful new techniques. Since it is a stronger conjecture, it could potentially lead to optimal non-approximability results for problems like sparsest cut which have so far been out of the reach of the unique games conjecture. In fact, in a recent result, Raghavendra, Steurer, and Tulsiani [85] prove better non-approximability bounds for problems including balanced separator and min linear rearrangement, than those that are known via UGC. Another interesting line of research is to have a unifying conjecture by reducing SSEC to the variants of UGC [62].

5 Structural Consequences of the PCP Theory

What makes some NPO problems hard to approximate, while others easy? This natural question had already been raised by Johnson in [56] long before the PCP theory was developed: “Is there some stronger kind of reducibility than the simple

polynomial reducibility that will explain [approximation] results, or are they due to some structural similarity between the problems as we define them?" In this section we present two completeness results that are consequences of the basic PCP theorem. These results give an evidence that different problems in the same non-approximability classes have the same reason for their hardness of approximation.

5.1 Polynomial-Time Approximation Schemes

Not all natural NP-hard optimization problems are hard to approximate. Easy classes include PTAS and FPTAS. For the definition of these classes and for that of the class APX, see the table below.

	FPTAS	PTAS	APX
Quantifiers	$\exists c \forall \epsilon > 0 \exists A_\epsilon$	$\exists c(\epsilon) \forall \epsilon \exists A_\epsilon$	$\exists c \exists C > 1 \exists A$
Algorithm	A_ϵ	A_ϵ	A
Running Time	$(x + 1/\epsilon)^c$	$ x ^{c(\epsilon)}$	$ x ^c$
Approximability Factor	$1 + \epsilon$	$1 + \epsilon$	C

Clearly, the following inclusions hold as follows:

$$FPTAS \subseteq PTAS \subseteq APX \subseteq NPO.$$

These inclusions are strict if $P \neq NP$. Cesati and Trevisan [21] also introduce the class *EPTAS* which differs from *FPTAS* in that the bound on the running time is $f(\epsilon)n^c$, where $f(\epsilon)$ is an arbitrary function of ϵ and $c > 0$ is an arbitrary constant.

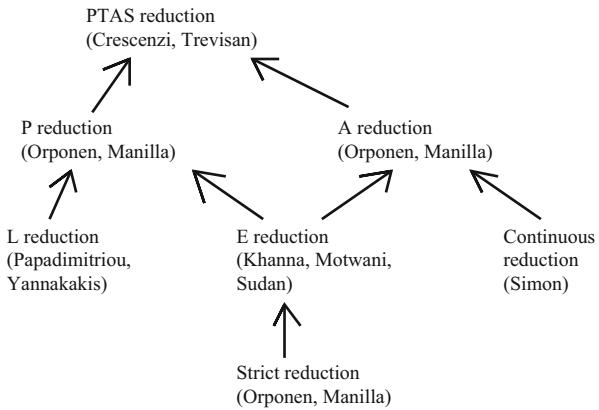
An optimization problem is practically tractable if it has a fully polynomial-time approximation scheme or somewhat weaker, an EPTAS. If a problem has a PTAS, but not EPTAS, then there is ϵ such that the $|x|^{c(\epsilon)}$ running time practically prohibits approximating it within a factor better than $1 + \epsilon$.

5.2 Approximation Preserving Reductions

Approximation preserving reductions in between NPO problems are used to show that if a problem P_1 is easy to approximate, then any problem P_2 is also easy to approximate which reduces to P_1 . Since the easiness of an approximation problem is associated with its membership in *PTAS*, almost all approximation preserving reductions preserve membership in *PTAS* (Fig. 5).

The first paper which defines an *approximation preserving reduction* was that of Orponen and Mannila [79]. Up to the present time, there are at least eight notions

Fig. 5 The taxonomy of approximation preserving reducibilities (Courtesy of Crescenzi et. al. [30])



of approximation preserving reductions in use with a similar overall scheme. This scheme is the following:

Let $P_1(x, y)$ and $P_2(x', y')$ be two polynomial-time functions that are to be optimized for y and y' (maximized or minimized in an arbitrary combination). Let $OPT_1(x)$ and $OPT_2(x')$ be the optimum of these problems. A *reduction* assumes two maps:

1. A map f to transform instances x of P_1 into instances $x' = f(x)$ of P_2 [instance transformation]
2. A map g to transform (input, witness) pairs (x', y') of P_2 into witnesses y of P_1 [witness transformation]

Let $OPT_1 = OPT_1(x)$, $OPT_2 = OPT_2(h(x))$, $APPR_1 = P_1(x, g(h(x), y'))$, and $APPR_2 = P_2(h(x), y')$. The centerpiece of any approximation scheme is a relation which is required to hold between these four quantities. This relation must roughly say: “If $APPR_2$ well approximates OPT_2 , then $APPR_1$ well approximates OPT_1 .” To see that indeed this is what we need, assume that we have a PTAS for OPT_2 and that P_1 reduces to P_2 . In order to get a good approximate solution for $OPT_1(x)$, where x is an arbitrary input instance of P_1 , first we construct $h(x)$ and find a witness y' such that $P_2(h(x), y')$ approximates $OPT_2(h(x))$ well. By the central assumption of the reduction, $P_1(x, y)$ well approximates $OPT_1(x)$ for $y = g(h(x), y')$. For the above argument to hold, f and g must be computable in polynomial time.

Different reductions differ in the relation required in between the four quantities. The L -reduction of Papadimitriou and Yannakakis [81] requires that OPT_2 is upper bounded by $c_1 OPT_1$ and that $|APPR_1 - OPT_1|$ is upper bounded by $c_2 |APPR_2 - OPT_2|$ for some constants c_1 and c_2 . It follows from the next lemma that L -reduction preserves PTAS.

Lemma 18 *A reduction scheme preserves PTAS iff it enforces that $|APPR_1 - OPT_1|/OPT_1 \rightarrow 0$ whenever $|APPR_2 - OPT_2|/OPT_2 \rightarrow 0$.*

5.3 APX Complete Problems

APX is the class of constant factor approximable NPO problems, and APX-PB is its subclass with problems that have a polynomial bound on their objective function. Our goal in this section is to give complete problems for *APX* and *APX – PB*.

Completeness proofs assume an underlying reduction. Even though in the theory of *APX*, the *L*-reduction is the most widely used reduction, in [31] it has been shown that it does not allow to reduce some problems which are known to be easy to approximate to problems which are known to be hard to approximate. Another flaw of the *L*-reduction is that it is too weak in another sense, namely, it is not always approximation preserving unless $P = NP \cap co-NP$ [30]. Therefore, we cannot hope for completeness results for *APX* or *APX – PB* with respect to the *L*-reduction. Instead, S. Khanna, R. Motwani, M. Sudan, and U. Vazirani [59] define the *E* reduction which, using the notation of the previous section, requires that

$$\max \left\{ \frac{APPR_1}{OPT_1}, \frac{OPT_1}{APPR_1} \right\} \leq 1 + c \left(\max \left\{ \frac{APPR_2}{OPT_2}, \frac{OPT_2}{APPR_2} \right\} - 1 \right)$$

for some $c > 0$. It can be easily shown that the *E* reduction preserves PTAS. Using the basic PCP theorem, Khanna et al. show the following:

Theorem 19 [59] *The MAX SAT problem is complete for APX – PB with respect to the E reduction. Also,*

$$APX - PB = \overline{\text{MAX SNP}} = \overline{\text{MAX NP}},$$

where the closure means closure under the *E* reduction.

The *E* reducibility is still somewhat too strict. In [31] it has been shown that natural PTAS problem exists, such as *MAX KNAPSACK*, which are not *E* reducible to polynomially bounded *APX* problems such as *MAX3SAT*. This drawback is mainly due to the fact that an *E* reduction preserves optimum values (see [31]). Crescenzi, Kann, Silvestri, and Trevisan [30] develop a reduction where functions f and g (see previous section) are allowed to depend on the *performance ratio*, where the performance ratio of an *NPO* problem A is defined as the function

$$R_A(x, y) = \max \left\{ \frac{A(x, y)}{OPT_A(x)}, \frac{OPT_A(x)}{A(x, y)} \right\}.$$

Definition 10 (AP Reduction [30]) Let A and B be two *NPO* problems. A is said to be *AP reducible* to B , if two functions f and g and a positive constant α exist such that:

1. For any x and for any $r > 1$, $f(x, r)$ is computable in time $t_f(|x|, r)$.
2. For any x and for any $r > 1$ and for any y , $g(x, y, r)$ is computable in time $t_g(|x|, |y|, r)$.

-
3. For any fixed r , both $t_f(., r)$ and $t_g(., ., r)$ are bounded by a polynomial.
 4. For any fixed n , both $t_f(n, .)$ and $t_g(n, n, .)$ are nonincreasing functions.
 5. For any x and any $r > 1$ and for any y , $R_B(f(x, r), y) \leq r$ implies

$$R_A(x, g(x, y, r)) \leq 1 + \alpha(r - 1).$$

The triple (f, g, α) is said to be an LP reduction from A to B .

In [30] the following is claimed:

Theorem 20 *MAX SAT is APX complete with respect to the AP reduction.*

6 Conclusion

The major challenge of PCP theory is to build PCP reductions. The intuition to these reductions came from Arthur–Merlin games and zero knowledge proofs. From a combinatorial point of view, PCP reductions correspond to isoperimetric inequalities. From a complexity theoretical standpoint, they are pseudorandom constructs. We cannot, however, identify a single tool for building them. PCP theory takes its tools from many different branches of mathematics and computer science: complexity theory, polynomials, coding theory, combinatorics, pseudorandomness, probability theory, and geometry (for semidefinite programming). PCP reductions serve as the atoms of the theory, from which its proofs are composed.

The theory of PCP consists of the PCP theorem and its consequences. The PCP theorem states that every proof has a “transparent” version that admits a quick Monte Carlo verification algorithm with a constant number of check bits. Its current shortest proof, due to Irit Dinur, differs structurally from the previous proofs. The transformation of an ordinary proof into a transparent one is a PCP reduction itself.

The greatest consequence of PCP theory is the conditional hardness of approximating NPO. The theory of NP had cast serious doubt that for many important problems in NP we will ever find a polynomial-time solution, but hope still existed that algorithms will be available that output increasingly good approximate solutions. This hope has been shattered by the PCP theory: we know now that our long-term inability to design approximation algorithms for classical optimization problems such as *MAX CLIQUE*, *SET COVER*, *COLORING*, and *METRIC TSP* is due to the NP-hardness of the associated gap problems. PCP theory had a very major effect on the theory of CSP. Not only that the PCP theorem itself is an inapproximability statement of a CSP problem but due to the search for sharper inapproximability bounds, our understanding of CSPs has gone through an exponential acceleration. Much of this investigation has been centered around the so-called long code.

The most important spin-off of PCP theory is hardness studies based on the unique game conjecture (UGC). The UGC allows us to fully characterize the approximation bounds of all CSPs.

PCP theory has also provided a wealth of new algebraic and combinatorial ideas, like the notion of checkable codes and Raz's parallel repetition theorem for two-prover games.

Although PCP reductions have mostly replaced the approximation preserving reductions of the pre-PCP era, the latter still may be interesting when we study syntactically defined classes of optimization problems.

In spite of the tremendous advances, many questions remain open, like the NP-hardness of the unique games problem itself, the behavior of chromatic number under approximation for small chromatic graphs, and whether asymmetric TSP is constant factor approximable. These problems will give exciting projects for future generations of complexity theorists.

Cross-References

- ▶ [Algorithms for the Satisfiability Problem](#)
 - ▶ [Binary Unconstrained Quadratic Optimization Problem](#)
 - ▶ [Complexity Issues on PTAS](#)
 - ▶ [Fractional Combinatorial Optimization](#)
 - ▶ [Max-Coloring](#)
-

Recommended Reading

1. M. Ajtai, J. Komlós, E. Szemerédi, Deterministic simulation in logspace, *Proceedings of 19th Annual Symp. on the Theory of Computing*, ACM, 1987, pp. 132–140
2. C. Ambühl, M. Mastrolilli, O. Svensson, Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling, in *FOCS*, 2007, pp. 329–337
3. N. Alon, U. Feige, A. Wigderson, D. Zuckerman, Derandomized graph products. *Comput. Complex.* 60–75 (1995)
4. S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and the intractability of approximation problems. *J. ACM* **45**(3), 501–555 (1998). Preliminary version in *Proc. of FOCS'92*
5. S. Arora, S. Safra, Probabilistic checking of proofs: a new characterization of NP. *J. ACM* **45**(1), 70–122 (1998). Preliminary version in *Proc. of FOCS'92*
6. S. Arora, M. Sudan, Improved low degree testing and its applications, in *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997, pp. 485–495
7. S. Arora, B. Barak, D. Steurer, Subexponential algorithms for unique games and related problems, in *FOCS*, 2010, pp. 563–572
8. S. Arora, S. Khot, A. Kolla, D. Steurer, M. Tulsiani, N.K. Vishnoi, Unique games on expanding constraint graphs are easy: extended abstract, in *STOC*, 2008, pp. 21–28
9. L. Babai, Trading group theory for randomness, in *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing*, ACM, 1985
10. L. Babai, E-mail and the unexpected power of interaction, in *Proc. 5th IEEE Structure in Complexity Theory conf.*, Barcelona, 1990, pp. 30–44
11. L. Babai, L. Fortnow, C. Lund, Non-deterministic exponential time has two-prover interactive protocols. *Comput. Complex.* **1**, 3–40 (1991)

12. L. Babai, L. Fortnow, L. Levin, M. Szegedy, Checking computations in polylogarithmic time, in *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing*, ACM, 1991
13. L. Babai, S. Moran, Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.* **36**, 254–276 (1988)
14. M. Bellare, O. Goldreich, M. Sudan, Free bits, PCPs and non-approximability—towards tight results. To appear *SIAM J. Comput.* **27**(3), 804–915 (1998). Preliminary version in *Proc. of FOCS'95*. Full version available as TR95-024 of ECCC, the *Electronic Colloquium on Computational Complexity*, <http://www.eccc.uni-trier.de/eccc/>
15. M. Bellare, S. Goldwasser, C. Lund, A. Russell, Efficient probabilistically checkable proofs, in *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing*, ACM, 1993. (See also Errata sheet in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994)
16. M. Bellare, M. Sudan, Improved non-approximability results, in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994, pp. 184–193
17. M. Ben-Or, S. Goldwasser, J. Kilian, A. Wigderson, Multi-prover interactive proofs: how to remove intractability assumptions, in *Proceedings of the Twentieth Annual Symposium on the Theory of Computing*, ACM, 1988
18. P. Berman, G. Schnitger, On the complexity of approximating the independent set problem. *Inform. Comput.* **96**, 77–94 (1992)
19. A. Blum, Algorithms for Approximate Graph Coloring, Ph.D. thesis, Massachusetts Institute of Technology, 1991 (MIT/LCS/TR-506, June 1991)
20. R. Boppana, M.M. Halldorsson, Approximating maximum independent sets by excluding subgraphs. *Bit Numer. Math.* **32**, 180–196 (1992)
21. M. Cesati, L. Trevisan, On the efficiency of polynomial time approximation schemes. *Inform. Process. Lett.* **64**(4), 165–171 (1997)
22. M. Charikar, K. Makarychev, Y. Makarychev, Near-optimal algorithms for unique games, in *STOC*, 2006, pp. 205–214
23. S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, D. Sivakumar, On the hardness of approximating multicut and sparsest-cut. *Comput. Complex.* **15**(2), 94–114 (2006)
24. V. Chvatal, A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **4**, 233–235 (1979)
25. A. Cohen, A. Wigderson, Dispersers, deterministic amplification, and weak random sources, in *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science*, IEEE, 1989
26. A. Condon, J. Feigenbaum, C. Lund, P. Shor, Probabilistically checkable debate systems and approximation algorithms for PSPACE-hard functions, in *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing*, ACM, 1993
27. A. Condon, J. Feigenbaum, C. Lund, P. Shor, Random debaters and the hardness of approximating stochastic functions. *SIAM J. Comput.* **26**(2), 369–400 (1997)
28. S. Cook, The complexity of theorem-proving procedures, in *Proceedings of the Third Annual Symposium on the Theory of Computing*, ACM, 1971
29. P. Crescenzi, V. Kann, A compendium of NP optimization problems. Technical Report, Dipartimento di Scienze dell'Informazione, Università di Roma “La Sapienza”, SI/RR-95/02, 1995. The list is updated continuously. The latest version is available by anonymous ftp from <nada.kth.se> as *Theory/Viggo-Kann/compendium.ps.Z*. Web address: <http://www.nada.kth.se/~viggo/problemList/compendium.html>
30. P. Crescenzi, V. Kann, R. Silvestri, L. Trevisan, *Structure in Approximation Classes*, Electronic Colloquium on Computational Complexity (ECCC)(066): (1996); *SIAM J. Comput.* **28**(5), 1759–1782 (1999)
31. P. Crescenzi, L. Trevisan, On approximation scheme preserving reducibility and its applications. *Theor. Comput. Syst.* **33**(1), 1–16 (2000)
32. A. Deshpande, M. Tulsiani, N.K. Vishnoi, Algorithms and hardness for subspace approximation, in *SODA*, 2011

33. I. Dinur, The PCP theorem by gap amplification, in *Proceedings of the 31st ACM Symposium on Theory of Computing*, 2006, pp. 29–40
34. I. Dinur, O. Reingold, Assignment testers: towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.* **36**(4), 975–1024 (2006)
35. I. Dinur, E. Mossel, O. Regev, Conditional hardness for approximate coloring. *SIAM J. Comput.* **39**(3), 843–873 (2009)
36. L. Engebretsen, J. Holmerin, Clique is hard to approximate within $n^{1-o(1)}$. Manuscript
37. U. Feige, A threshold of $\ln n$ for set cover, in *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing*, ACM, 1996. Journal Version: *J. ACM* **45**(4), 634–652 (1998)
38. U. Feige, S. Goldwasser, L. Lovász, S. Safra, M. Szegedy, Interactive proofs and the hardness of approximating cliques. *J. ACM* **43**(2), 268–292 (1996)
39. U. Feige, J. Kilian, Two prover protocols—low error at affordable rates, in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing*, ACM, 1994
40. U. Feige, J. Kilian, Zero knowledge and chromatic number, in *Proceedings of the Eleventh Annual Conference on Complexity Theory*, IEEE, 1996, pp. 172–183
41. U. Feige, L. Lovász, Two-prover one-round proof systems: their power and their problems, in *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing*, ACM, 1992
42. V. Feldman, V. Guruswami, P. Raghavendra, Yi. Wu, Agnostic learning of monomials by halfspaces is hard. *CoRR*, abs/1012.0729 (2010)
43. L. Fortnow, J. Rompel, M. Sipser, On the power of multi-prover interactive protocols. *Theor. Comput. Sci.* **134**(2), 545–557 (1994)
44. M. Fürer, Improved hardness results for approximating the chromatic number, in *Proceedings of the Thirty Sixth Annual Symposium on the Foundations of Computer Science*, IEEE, 1995
45. M.R. Garey, R.L. Graham, J.D. Ullman, Worst case analysis of memory allocation algorithms, in *Proceedings in the 4th Annual ACM Symposium on the Theory of Computing*, 1972, pp. 143–150
46. M. Garey, D. Johnson, The complexity of near-optimal graph coloring. *J. ACM* **23**, 43–49 (1976)
47. M. Garey, D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, 1979)
48. M. Goemans, D. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **42**(6), 1115–1145 (1995)
49. S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems. *SIAM J. Comput.* **18**(1), 186–208 (1989)
50. V. Guruswami, R. Manokaran, P. Raghavendra, Beating the random ordering is hard: inapproximability of maximum acyclic subgraph, in *FOCS*, 2008, pp. 573–582
51. V. Guruswami, P. Raghavendra, R. Saket, Yi. Wu, Bypassing ugc from some optimal geometric inapproximability results. *Electronic Colloquium on Computational Complexity*, 2010
52. M. Halldórsson, A still better performance guarantee for approximate graph coloring. *Inform. Process. Lett.* **46**, 169–172
53. J. Håstad, Testing of the long code and hardness for clique, in *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing*, ACM, 1996
54. J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, in *Proceedings of the Thirty Seventh Annual Symposium on the Foundations of Computer Science*, IEEE, 1996, pp. 627–636
55. J. Håstad, Some optimal inapproximability results. *J. ACM* **48**(4), 798–859 (2001)
56. D. Johnson, Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.* **9**, 256–278 (1974)
57. H. Karloff, U. Zwick A 7/8-approximation for MAX 3SAT?, in *Proc. 38th Ann. IEEE Symp. on Foundations of Comput. Sci.* (IEEE Computer Society, 1997), pp. 406–415
58. S. Khanna, N. Linial, S. Safra, On the hardness of approximating the chromatic number, in *Proceedings of the Second Israel Symposium on Theory and Computing Systems*, 1993

59. S. Khanna, R. Motwani, M. Sudan, U. Vazirani, On syntactic versus computational views of approximability. *SIAM J. Comput.* **28**, 164–191 (1998). Preliminary Version: Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1994, pp. 819–830. (with S. Khanna, M. Sudan, U. Vazirani) Also available as: ECCC Report No. TR95-023, Electronic Colloquium on Computational Complexity, <http://www.eccc.uni-trier.de/eccc/>, 1995
60. S. Khot, Improved inapproximability results for maxclique, chromatic number and approximate graph coloring, in *FOCS*, 2001, pp. 600–609
61. S. Khot, On the power of unique 2-prover 1-round games, in *STOC*, 2002, pp. 767–775
62. S. Khot, On the unique games conjecture (invited survey), in *IEEE Conference on Computational Complexity*, 2010, pp. 99–121
63. S. Khot, G. Kindler, E. Mossel, R. O’Donnell, Optimal inapproximability results for max-cut and other 2-variable csp? *SIAM J. Comput.* **37**(1), 319–357 (2007)
64. S. Khot, A. Naor, Sharp kernel clustering algorithms and their associated grothendieck inequalities, in *SODA*, 2010, pp. 664–683
65. S. Khot, O. Regev, Vertex cover might be hard to approximate to within 2-epsilon. *J. Comput. Syst. Sci.* **74**(3), 335–349 (2008)
66. S. Khot, N.K. Vishnoi, The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into l_1 , in *FOCS*, 2005, pp. 53–62
67. G. Kindler, A. Naor, G. Schechtman, The UGC hardness threshold of the p grothendieck problem. *Math. Oper. Res.* **35**(2), 267–283 (2010)
68. A. Kolla, Spectral algorithms for unique games, in *IEEE Conference on Computational Complexity*, 2010, pp. 122–130
69. A. Kumar, R. Manokaran, M. Tulsiani, N.K. Vishnoi, On lp-based approximability for strict csp, in *SODA*, 2011
70. L. Levin, Universal’nye perebornyie zadachi (Universal search problems: in Russian). Problemy Peredachi Informatsii **9**(3), 265–266 (1973). A corrected English translation appears in an appendix to Trakhtenbrot [93]
71. N. Linial, U. Vazirani, Graph products and chromatic numbers, in *Proc. 30th IEEE Symp. on Foundations of Computer Science*, 1989, pp. 124–128
72. L. Lovász, On the ratio of optimal integral and fractional covers. *Discrete Math.* **13**, 383–390 (1975)
73. C. Lund, L. Fortnow, H. Karloff, N. Nisan, Algebraic methods for interactive proof systems. *J. ACM* **39**, 859–868 (1992)
74. C. Lund, M. Yannakakis, On the hardness of approximating minimization problems. *J. ACM* **41**(5), 960–981 (1994)
75. K. Makarychev, Y. Makarychev, How to play unique games on expanders, in *WAOA*, 2010, pp. 190–200
76. E. Mossel, Gaussian bounds for noise correlation of functions and tight analysis of long codes, in *FOCS*, 2008, pp. 156–165
77. E. Mossel, R. O’Donnell, K. Oleszkiewicz, Noise stability of functions with low influences invariance and optimality, in *FOCS*, 2005, pp. 21–30
78. R. O’Donnell, Yi. Wu, Conditional hardness for satisfiable 3-csp, in *STOC*, 2009, pp. 493–502
79. P. Orponen, H. Manilla, On approximation preserving reductions: complete problems and robust measures. Technical Report C-1987-28, Department of Computer Science, University of Helsinki, 1987
80. C. Papadimitriou, M. Yannakakis, Optimization, approximation and complexity classes. *J. Comput. Syst. Sci.* **43**, 425–440 (1991)
81. C. Papadimitriou, M. Yannakakis, The traveling salesman problem with distances one and two. *Math. Oper. Res.* (1992)
82. E. Petrank, The hardness of approximation: gap location. *Comput. Complex.* **4**, 133–157 (1994)
83. P. Raghavendra, Optimal algorithms and inapproximability results for every csp? in *STOC*, 2008, pp. 245–254

84. P. Raghavendra, D. Steurer, Graph expansion and the unique games conjecture, in *STOC*, 2010, pp. 755–764
85. P. Raghavendra, D. Steurer, M. Tulsiani, Reductions between expansion problems (2010)
86. R. Raz, A parallel repetition theorem, in *Proceeding of the 27th STOC*, 1995, pp. 447–456. Journal version in: *SIAM J. Comput.* **27**(3), 763–803 (1998)
87. R. Raz, S. Safra, A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP, in *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing*, ACM, 1997, pp. 475–484
88. S. Sahni, T. Gonzales, P-complete approximation problems. *J. ACM* **23**, 555–565 (1976)
89. A. Samorodnitsky, L. Trevisan, A pcp characterization of np with optimal amortized query complexity, in *STOC*, 2000, pp. 191–199
90. A. Shamir, IP = PSPACE. *J. ACM* **39**(4), 869–877 (1992)
91. M. Sudan, L. Trevisan, Probabilistically checkable proofs with low amortized query complexity, in *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, 1998
92. M. Szegedy, Many valued logics and holographic proofs, in *ICALP*, 1999
93. B. Trakhtenbrot, A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Ann. Hist. Comput.* **6**, 384–400 (1984)
94. L. Trevisan, Recycling queries in PCPs and in linearity tests, in *Proceedings of the 30th ACM Symposium on Theory of Computing*, 1998
95. L. Trevisan, G.B. Sorkin, M. Sudan, D.P. Williamson, Gadgets, approximation, and linear programming, in *Proceedings of the 37th Symposium on Foundations of Computer Science*, (IEEE, 1996), pp. 617–626
96. L. Trevisan, Inapproximability of combinatorial optimization problems. *Electron. Colloq. Comput. Complex.* (065) (2004)
97. L. Trevisan, G.B. Sorkin, M. Sudan, D.P. Williamson, Gadgets, approximation, and linear programming. *SIAM J. Comput.* **29**(6), 2074–2097 (2000)
98. D. Zuckerman, On unapproximable versions of NP-complete problems. *SIAM J. Comput.* **25**(6), 1293–1304 (1996)
99. D. Zuckerman, Linear degree extractors and the inapproximability of max clique and chromatic number. *Theor Comput.* **3**(1), 103–128 (2007)

Protein Docking Problem as Combinatorial Optimization Using Beta-Complex

Deok-Soo Kim

Contents

1	Introduction.....	2686
2	Brief on Molecular Binding Process.....	2690
2.1	Force Field.....	2690
2.2	Funnel in Energy Landscape.....	2692
3	Definition of Docking Problem: Search + Scoring.....	2692
3.1	Search Subproblem.....	2694
3.2	Scoring Subproblem.....	2698
3.3	Two Primary Uses of Heuristics in Docking.....	2699
3.4	Grid: Computational Acceleration.....	2700
3.5	Input Data to Docking Simulation.....	2700
4	Flexibility of Molecules.....	2701
4.1	Ligand Flexibility.....	2702
4.2	Receptor Flexibility.....	2703
5	Previous Studies on Docking.....	2704
6	Formalization of Topology Structures.....	2708
6.1	Primal Structures.....	2708
6.2	Dual Structures.....	2711
6.3	Molecular Structure.....	2713
7	Pocket Extraction Using Beta-Shape.....	2718
7.1	Previous Works.....	2719
7.2	Extraction of Pocket Using Beta-Shape.....	2720
8	Generation of Initial Ligand Conformations.....	2724
8.1	Superposition of Two Point Sets with Identical Cardinalities.....	2725
9	Search of Global Minima.....	2727
9.1	Genetic Operators.....	2728
9.2	Genetic Iteration.....	2730
10	Conclusion.....	2731
	Cross-References.....	2731
	Recommended Reading.....	2731

D.-S. Kim

Department of Industrial Engineering, Hanyang University, Seongdong-ku, Seoul, South Korea
e-mail: dskim@hanyang.ac.kr

Abstract

The completion of the Human Genome Project has launched the post-sequencing phase of genome research. Once a genome information is identified, protein structures become most important in life sciences. Drug design is one immediate example, and docking, a computational method to understand molecular binding between protein and compound, is one of its fundamental implementation methods.

This chapter presents a docking problem between a receptor and a ligand in the framework of combinatorial optimization and is an effort to formalize the problem into a mathematically and computationally rigorous theory so that follow-up studies can safely rely on the concepts presented in this chapter. Docking problem is a mathematical and computational problem and a highly complicated, intractable problem. In addition, this problem is very much interdisciplinary encompassing biology, biochemistry, biophysics, medicinal chemistry, and computational disciplines such as optimization, mathematics, and geometric computations. Therefore, it is desirable to develop a common language. To accomplish this objective, this chapter is based on geometric constructs called the beta-complex and beta-shape of molecules.

This chapter has three parts. The first part introduces basic concepts and theory of molecular binding/recognition process and tries to formalize the transformation of the docking problem to a combinatorial optimization problem. This part will be particularly useful for people from mathematical and computational disciplines who wants to learn about the fundamentals of docking problem in a nutshell. The second part introduces the topology representation of molecules and the fundamental building blocks for the computational process with a slightly mathematical rigor. The Voronoi diagram, the spatial tessellations (triangulations), and the beta-complex and beta-shape are primary geometric objects introduced. The third part shows how to formulate and solve a protein-ligand docking problem as combinatorial optimization problems using the beta-complex and beta-shape, as it is done in BetaDock – the docking software developed by author's group.

1 Introduction

An interaction between a protein and another molecule is important for understanding the biological function of the protein and is essential for *structure-based drug design* because it helps to identify lead compounds. For example, *imatinib mesylate* (also called Gleevec® or STI571¹) binds to a tyrosine kinase, the protein related

¹STI stands for *signal transduction inhibitor*.

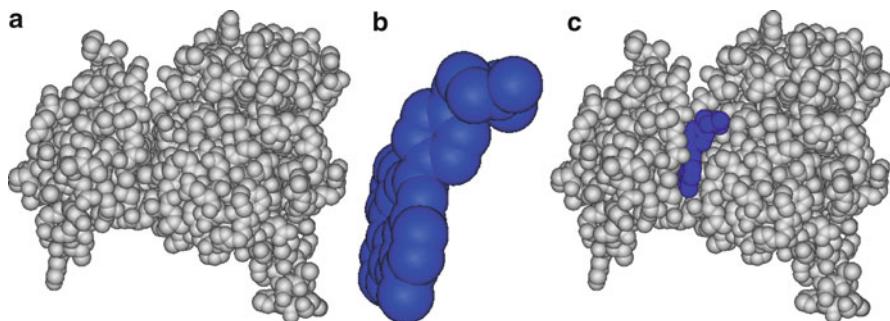


Fig. 1 Tyrosine kinase and imatinib. (a) Tyrosine kinase, (b) Imatinib, and (c) Tyrosine kinase with imatinib

to leukemia, at an *active site*² so that the drug effectively inhibits or inactivates the function of the tyrosine kinase [6, 160].³ Figure 1a, b show tyrosine kinase and imatinib, respectively. Figure 1c shows the result of the binding between these two molecules. Here, the tyrosine kinase is called a *receptor* and the imatinib is called a *ligand*. Tyrosine kinase is also called a *target protein* in that it is responsible for disease. The binding between receptor and ligand occurs due to the physicochemical interaction between the two molecules such as van der Waals forces among all atoms, electrostatic forces among charged or polar atoms, and hydrogen bonds between hydrogen donors and acceptors [114, 140]. Binding phenomenon is frequently formulated as an energy minimization process because a ligand is believed to bind to a receptor at its lowest potential energy. Gleevec® indeed opened a new medication model called *target therapy* which blocks the growth of cancer cells by interfering with specific target protein.

Protein-ligand docking is a computational method that attempts to find the *best binding* or the *best match* between a receptor and a ligand, given chemical bonding constraints of the molecules. There are other kinds of docking: protein-protein docking, protein-DNA docking, protein-RNA docking, etc. In this chapter, “docking” refers to the protein-ligand docking, unless otherwise stated.

Docking assumes that the structure of both receptor and ligand is known where the molecular structure implies its atomic coordinates and biochemical properties. The objective of docking is to predict the *ligand pose* with respect to a receptor as precisely and as quickly as possible through a computational method. “Ligand pose” is the position and orientation of ligand with respect to receptor and its conformation

²A substrate binds at (usually one or at most a few) specific (not random) sites of an enzymatic protein. Such binding sites are called an *active site*.

³Gleevec® was developed by rational drug design at Novartis in the late 1990s. With high-throughput screening, researchers identified 2-phenylaminopyrimidine which was then modified introducing methyl and benzamide groups to add enhanced binding properties. Receiving FDA approval in May 2001, it made the cover of TIME magazine as the *magic bullet* to cure cancer.

determined by the prescribed dihedral angle of rotatable bonds. In general, the best binding occurs when the potential energy of a ligand minimizes with respect to a receptor where the potential energy is usually formulated as a nonlinear function in a high-dimensional space. The potential energy function contains many local minima where each corresponds to a locally stable ligand pose. Therefore, finding the best binding becomes solving a nonlinear optimization problem to find the global minimum of the energy function. Receptor is usually a protein, and thus the problem is often called a protein-docking problem. Ligand is usually a compound of small molecular weight or sometimes another protein.

Computational method is useful only when it is accurate, robust, and sufficiently efficient. This statement applies to any application area. In docking, the number of local minima grows exponentially with respect to the size of the molecules (i.e., the number of atoms of both receptor and ligand) leading the problem to NP-hard [33, 119, 188]. Small proteins usually have hundreds of atoms, and some larger ones have hundreds of thousands of atoms. Hence, classical nonlinear optimization techniques such as the conjugate gradient method suffer [67]. For example, Gehlhaar et al. presented a docking example having more than 10^{30} local minima [70]. Therefore, docking is computationally challenging and usually deemed to be intractable [145, 189] which inevitably invites heuristic approaches to solution process. If conformational flexibility is also allowed for molecules, the docking problem becomes even harder.

It is generally agreed that binding usually occurs at molecular boundaries in spite that there are important exceptional cases due to significant conformational flexibility of molecules. Hence, there is no doubt that effectively recognizing molecular boundary contributes to the efficient docking simulation. In this respect, the recent theory of the beta-complex and beta-shape defined for sphere set contributes to docking problem and others [101, 103, 105, 182].

A docking simulation is usually done between a particular protein related to a disease of interest and many compounds in chemical databases. Hence, the computational speed and the solution quality are critical in *virtual screening* (VS) [10, 207] which is to identify new drug candidates, called *lead compounds*, from compound database through protein vs. compound docking simulation. For example, the Cambridge Structural Database (CSD) has 541,748 compound structures (as of Jan. 2011) [4, 27], the Available Chemicals Directory (ACD) has over 3,870,000 non drug-like compounds [9], and MACCS-II Drug Data Report (MDDR) has over 150,000 drug-like compounds [139]. *High-throughput screening* (HTS) is a method based on scientific experimentation in drug discovery using robotics, sensors, miniaturization technologies, etc., to quickly perform millions of biochemical or pharmacological tests. HTS quickly processes to identify active compounds or antibodies which modulate a particular biomolecular pathway, and the results provide starting points for drug design. Starting with 96-well plates in mid-1990s [24], current HTS technology supports 1,536-well microplate format for compound handling [113, 142]. VS and HTS are two very important components of modern drug discovery research. VS and HTS have been traditionally considered

distinct approaches in that VS is theoretical, computational whereas HTS is experimental. However, they are now being considered very much complementary to each other and beginning to be merged [11]. The success of the fusion of VS and HTS will largely lie in the representation of the three-dimensional molecular structure.

It is not the intention of this chapter to try to provide a comprehensive review about protein docking. The objective of this chapter is two-fold at the cost of sacrificing other aspects of docking:

- Understanding the rationale behind docking problem and its mathematical/computational formulation
- Understanding the combinatorial optimization aspect of docking problem

Significant emphasis is focused on the formalization of docking problem because many previous studies from communities of biology, chemistry, medicine, etc., tend to ignore or lack mathematical rigor, computational efficiency, and sufficient level of reproducibility. Good review papers on different aspects of docking problem are already available [19, 39, 151, 154, 170, 189, 190, 194].

This chapter consists of three parts:

Part (1) the fundamentals of molecular binding process and the mathematical/computational formulation of docking problem ([Sects. 2–5](#))

Part (2) the theory of topology representation for biomolecules based on the Voronoi diagram, the quasi-triangulation, and the beta-complex ([Sect. 6](#))

Part (3) the combinatorial optimization formulation for the new, powerful docking method based on the beta-complex and its implementation in BetaDock⁴ ([Sects. 7–9](#))

[Section 2](#) briefly introduces the basic biophysics about the force field theory and is important for correctly understanding the molecular binding and docking process. [Section 3](#) defines what the docking problem is and how the problem is modeled for computation. This chapter is important in that the theoretical justification of the combinatorial optimization formulation of docking problem is presented. [Section 4](#) discusses the flexibility issue related with the conformation changes of molecules during docking process. [Section 5](#) presents a literature survey related to docking with a particular emphasis on the historical development of docking concept. Important docking softwares are also briefly described. [Section 6](#) presents the formalization of the topology representation of geometric constructors related to docking problem and provides a rigorous theoretical, mathematical, and computational basis for not only docking but also many other molecular modeling and analysis problems. In particular, the important geometric constructs called the beta-complex and beta-shape are introduced. [Section 7](#) presents the algorithm, including a short survey of related previous works, for extracting pockets on the boundary of receptors using the beta-shape of the receptor. [Section 8](#) presents

⁴BetaDock and BetaMol are freely available at Voronoi Diagram Research Center (VDRC, <http://voronoi.hanyang.ac.kr>).

an algorithm generating initial conformations of ligands to launch the genetic algorithm to find the global optima on the energy surface. [Section 9](#) presents the genetic algorithm used in BetaDock. [Section 10](#) concludes the paper. All three-dimensional figures are generated from BetaMol, the molecular modeling software based on the beta-complex theory and available at VDRC [204].

2 Brief on Molecular Binding Process

To correctly understand the receptor-ligand binding process and to formulate the computational problem of docking, it is necessary to have a good understanding about the physicochemical aspect of the binding process, in particular the forces involved.

2.1 Force Field

The free energy of a molecular system is the amount of work that the system can perform. It can be estimated most precisely by solving quantum mechanics that explicitly represents the electron distribution to derive the properties of the molecular system. However, most problems in quantum mechanics are too large to be correctly solved by current computing technology. *Molecular mechanics* (also known as *force field method*), ignoring the contribution of electrons, calculates the free energy as a function of the nuclear coordinates in the molecular system. The function is often a convenient form of a rational polynomial. Being computationally much less expensive than quantum mechanics, molecular mechanics can handle molecules with a significant number of atoms.

In molecular biology, force field refers to the mathematical function of nuclear coordinate of atoms in the molecular system and the parameters describing the function. The function and its parameters are derived from both experimentation and quantum mechanical calculations. It is called the “force field” because forces on individual atoms are related to the gradient of this function. Force field is empirical. This implies that it is not appropriate to question about the correctness of a particular force field. One force field function may perform better for a given molecular system than another depending on the condition of the problem of interest. The mathematical form of force field is usually devised considering the trade-off between accuracy and computational efficiency. Studies on protein force field began in early 1980s. Even if the force field approach is empirical, NP-hardness is still a challenge for efficiently solving a docking problem using the force field method. Another class of *semiempirical* approach called the ab initio [80, 167] is also computationally prohibitive in many practical problems [217].

A simple, popular example of force field that can be used for a single or a set of molecules is as follows:

$$\begin{aligned}
E = & \sum_{\text{bonds}} \frac{1}{2} k_b (l_b - l_b^*)^2 + \sum_{\text{angles}} \frac{1}{2} k_a (\theta_a - \theta_a^*)^2 + \sum_{\text{torsions}} \frac{1}{2} k_t (1 + \cos(t\omega - \gamma_t))^2 \\
& + \sum_{i < j} \left\{ \frac{A_{ij}}{d_{ij}^{12}} - \frac{B_{ij}}{d_{ij}^6} \right\} + \sum_{i < j} \frac{q_i q_j}{\varepsilon(d_{ij}) \cdot d_{ij}} + \sum_{i < j} \cos^4 \theta \left\{ \frac{C_{ij}}{d_{ij}^{12}} - \frac{D_{ij}}{d_{ij}^{10}} \right\}. \quad (1)
\end{aligned}$$

In the first term, l_b and l_b^* are the *instance length* and the *ideal length* of the b -th bond, respectively. The “instance length” denotes the bond length observed in the protein structure file such as those in PDB, and the “ideal length” denotes the bond length that the two atoms in the bond are theoretically most stable. The values of ideal length for bonds between different atom types are available [17]. k_b is the force constant of the b -th bond. The first term models the interaction between pairs of bonded atoms and is approximated by Hooke’s law to measure the deformation of each bond due to either stretch or compression. In the second term, θ_a and θ_a^* are the instance angle and the ideal angle, respectively, and k_a is a force constant of the a -th angle. The second term models the interaction among triplets of bonded atoms and measures the angular deformation among pairs of bonds incident to each atom. In the third term, γ_t is the phase angle and k_t is a force constant of the t -th torsion, respectively. The third term models a torsional potential and measures the twist of two planes intersecting at each bond. Hence, these three terms altogether model the internal energy of a molecular system due to chemical bond and thus denoted by E_{bonded} [114]. E_{bonded} measures intramolecular interactions. The fourth term is the well-known (12,6)-Lennard-Jones potential function which models the van der Waals interaction between pairs of atoms. The van der Waals interaction consists of two forces: an attractive force and a repulsive force which are represented by d^{-6} term and d^{-12} term, respectively. The fifth term is the Coulomb’s law between two point charges which models the electrostatic interaction between pairs of atoms. The last term models the hydrogen-bonding interaction between pairs of atoms lying on the boundary of molecules. Note that, in the binding environment around proteins, only three atom pairs, OH, NH, and SH, can define hydrogen bonding. A_{ij} , B_{ij} , C_{ij} , and D_{ij} are all constants depending on the type of two atoms i and j ; d_{ij} is the distance between the centers of two atoms. q_i and q_j are two point charges; $\varepsilon(d_{ij})$ is the dielectric permittivity of the medium where the two charges are placed and is given as a function of distance; θ is the angle defined at the hydrogen atom with two nearby atoms (where one is the hydrogen donor and the other is the hydrogen acceptor) when the hydrogen bonding may be defined and is frequently taken as 180°. The last three terms are all defined between nonbonded atoms and thus altogether denoted as $E_{\text{nonbonded}}$. $E_{\text{nonbonded}}$ models intermolecular interactions. In fact, Eq.(1) is very close to the one actually used in AutoDock, one of the most popular docking software. Then, Eq.(1) can be rewritten in a simple form as

$$E = E_{\text{bonded}} + E_{\text{nonbonded}}. \quad (2)$$

AMBER (Assisted Model Building and Energy Refinement) [114, 212, 213] and CHARMM (Chemistry at HARvard using Molecular Mechanics) [20] are the two most popular molecular mechanics force fields. In certain situations, extremely simplified representations of force field can be used from computational efficiency point of view. For example, Gehlhaar et al. used a piecewise linear approximation with five line segments for the van der Waals forces [70]. For the review of various force fields, see [110, 174].

2.2 Funnel in Energy Landscape

Levinthal observed the following paradox, called the *Levinthal paradox* in 1968 [132]: “Proteins fold into native state extremely quickly in spite that there exist exponential number of local minima in the energy surface with respect to the number of rotatable bonds in proteins.” As a possible resolution of Levinthal paradox, in 1992, Leopold et al. proposed the concept of *protein-folding funnel* as a kinetic mechanism for understanding protein folding [22, 131]. A protein-folding funnel is a set of downhill kinetic pathways that converge to a single minimum on the energy surface leading to a unique, stable, and native conformation. In the world of proteins, it has long been agreed that the potential energy surface of protein is biased toward the global minimum so that a single deep funnel leads proteins to fold to their native state, despite very large configurational spaces [53, 205].

As the physicochemical principle of protein binding is similar to that of protein folding, it is natural to extend the concept of funnel to intermolecular phenomenon such as binding. Frauenfelder et al. presented the idea for connecting ligand binding to the funnel of energy landscapes by showing that the energy surface for small molecule binding to myoglobins could be viewed as the energy landscape for the protein-folding theory [68]. Others also used the funnel concept in molecular binding or recognition [196, 198, 200, 205].

3 Definition of Docking Problem: Search + Scoring

Having modeled the free energy E of a molecular system as Eq. (2), solving docking problem is then equivalent to solving the optimization problem given as

$$\text{Minimize } E. \quad (3)$$

In formulating docking problem, there are usually four assumptions in the model of binding as stated by Miller and Dill [150]:

- *Steric fit*: a binding is determined mainly by a shape complementarity.
- *Native binding*: ligands mainly bind to native state of receptors.
- *Locality*: ligands perturb protein structures mainly at the binding site.
- *Continuity*: small changes in ligand or protein structure lead to small changes in binding affinity.

Note that three out of the four assumptions (i.e., the steric fit, the locality, and the continuity) are related with “geometry.” However, be aware that there are cases that these assumptions are not preserved. For example, some ligands bind to highly denatured states more tightly than native states. The function E is frequently replaced by $E_{\text{nonbonded}}$, ignoring the E_{bonded} terms, yielding the following formula:

$$\begin{aligned} \text{Min } E_{\text{nonbonded}} = & \sum_{l \in L} \sum_{r \in R} \left\{ \frac{A_{lr}}{d_{lr}^{12}} - \frac{B_{lr}}{d_{lr}^6} \right\} + \sum_{l \in L} \sum_{r \in R} \frac{q_l q_r}{\varepsilon(d_{lr}) \cdot d_{lr}} \\ & + \sum_{l \in L} \sum_{r \in R} \cos^4 \theta \left\{ \frac{C_{lr}}{d_{lr}^{12}} - \frac{D_{lr}}{d_{lr}^{10}} \right\} \end{aligned} \quad (4)$$

where L and R are ligand and receptor, respectively. This formulation is mainly due to the assumption that the conformational flexibility of molecules in the binding process can be separately reflected.

Equation (4) is an unconstrained nonlinear minimization problem, still in a high-dimensional space even when both receptor and ligand have a moderate number of atoms. For example, the PDB model 1t46 contains both a tyrosine kinase (2,381 heavy atoms) and an imatinib (37 heavy atoms). Therefore, the van der Waals force between the tyrosine kinase and the imatinib consists of 157,176 terms and the electrostatic force consists of 78,588 terms. There may be up to 37 hydrogen bonds between tyrosine kinase and imatinib because the number of ligand atoms is the upper bound of the number of hydrogen bonds between the receptor and the ligand. Hence, there may be between 0 and 74 terms for hydrogen bond. Therefore, there are altogether more than 235,764 nonlinear terms in this minimization problem. Hence, efficiently finding the global minimum of Eq.(4) is still a challenge.

In addition, in solving a docking problem, Eq.(4) needs to be solved several times. After solving Eq.(4) once, the ligand is transformed in the space with respect to a fixed receptor to result in a new pose with respect to the receptor. Then, a new instance of Eq.(4) is formulated to be solved again so that the ligand is to be transformed again. This process repeats until the termination condition, which may be defined differently depending on the employed docking method, is encountered. Hence, the efficiency of the algorithm searching for the global minimum of an entire docking problem is very important.

It is known that the energy function in Eq.(4) usually has an exponential number of minima with respect to the number of atoms in the system. This leads the docking problem to NP-hard and therefore, in search of the global minimum, it is inevitable and a common practice to employ a heuristic method. Based on this observation, docking is usually viewed as a problem consisting of two major subproblems of search and scoring. To emphasize,

$$\text{Docking} = \text{Search} + \text{Scoring}. \quad (5)$$

The search subproblem is to align a ligand to the receptor boundary, and the scoring subproblem is to estimate their binding affinity represented by a mathematical function such as Eq.(4). However, the two subproblems are not completely independent but they are closely related in the following sense: If a scoring function is not sufficiently accurate, the computational effort required in the search for a good ligand pose may be wasted or a bad pose may be selected as a solution. Hence, devising good scoring functions is critical for the success of good docking method. This chapter will mainly focus on efficiently exploring the search space based on the scoring function in Eq.(4) is employed.

Exploring the search space is inherently based on the combinatorics among the atoms in the system. Therefore, this chapter discusses the “combinatorial optimization” of Eq.(4) whose combinatorial space is from the all atoms in both receptor and ligand.

3.1 Search Subproblem

Consider a pair of atoms. For example, consider an oxygen O and a hydrogen H where O is from a receptor and H is from a ligand. Assume that both molecules are rigid. Depending on condition, O and H may or may not define a hydrogen bond (See Sect. 3.5). Let E_{vdW}^{OH} , E_{es}^{OH} , and E_{hb}^{OH} be the forces of van der Waals, electrostatic, and hydrogen bonding, respectively. Then, from Eq.(4), E_{vdW}^{OH} and E_{hb}^{OH} are given as

$$E_{vdW}^{OH} = \frac{38,919.64}{d^{12}} - \frac{124.0492}{d^6} \quad (6)$$

$$E_{hb}^{OH} = \cos^4 \theta \left\{ \frac{75,570}{d_{lr}^{12}} - \frac{23,850}{d_{lr}^{10}} \right\}. \quad (7)$$

The coefficients for E_{vdW}^{OH} and E_{hb}^{OH} are borrowed from AutoDock 1.0 which adopted these parameters from AMBER. θ is considered to be 180° here because the hydrogen bonding is most strong at this angle. Figure 2a–c shows the graphs of E_{vdW}^{OH} , E_{es}^{OH} , and E_{hb}^{OH} , respectively, where the horizontal axis denotes the *interatomic distance* between the centers of two atoms and the vertical axis denotes the function value.

E_{es}^{OH} needs a little bit of explanation. Cornell et al. published an important paper which contained the point-charges of atoms in residues (i.e., amino acids) [43]. According to that paper, the charge of a particular atom type is not fixed but varies depending on the amino acid type and the location of the atom in the amino acid. The minimum and maximum values of the charges for the five popular atom types are given as follows [43]: C: [-0.4121 (LEU), 0.076 (ARG)]; N: [-0.0407 (GLN), -0.2548 (PRO)]; O: [-0.8188 (GLU), -0.5579 (TYR)]; S: [-0.3199 (CYS), -0.2731 (MET)]; H: [-0.0425 (GLU), 0.4478 (ARG)]. The three-letter words in parentheses denote the amino acid type from which the charges are defined.

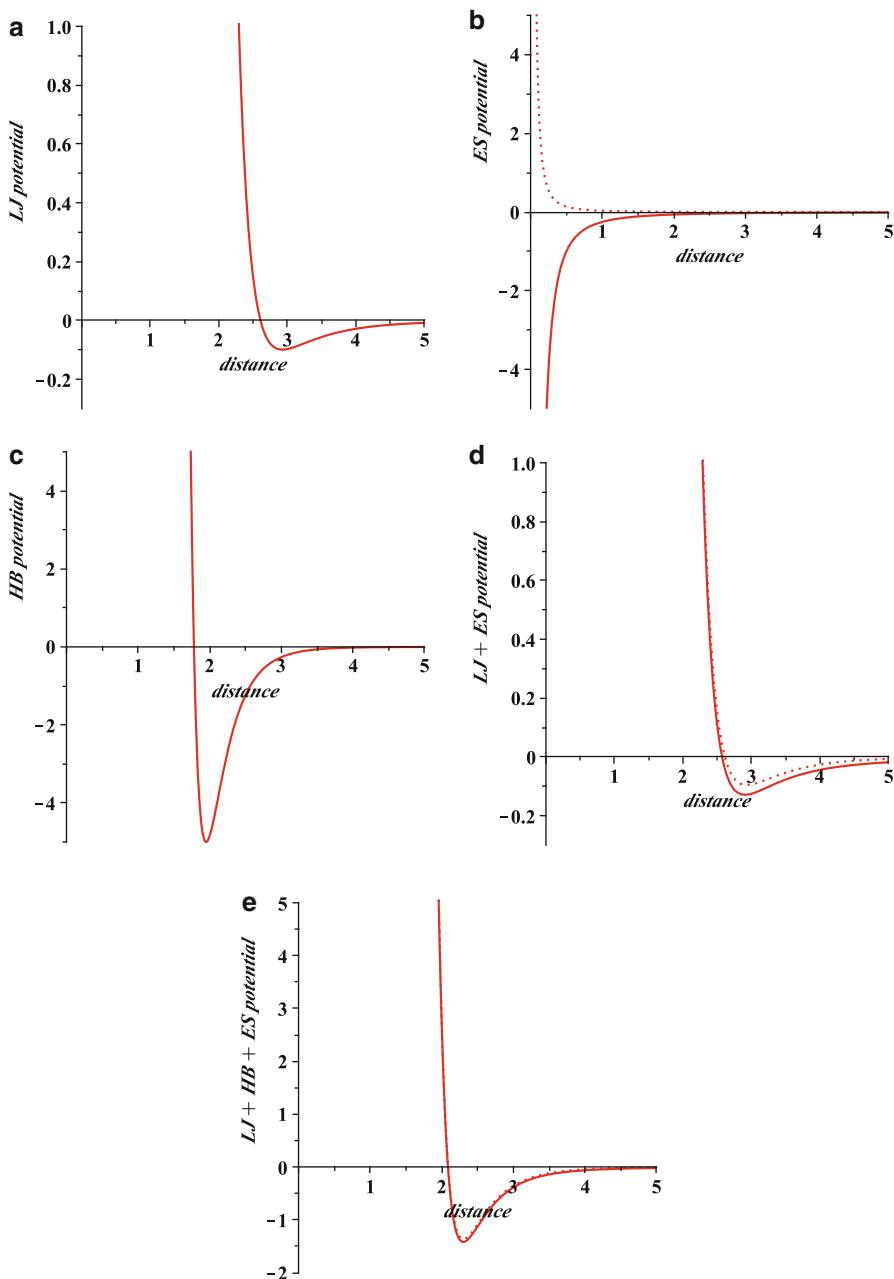


Fig. 2 Nonbonded force terms between O and H using AMBER force field (where the hydrogen bonding may exist). (a) The (12,6)-Lennard-Jones potential term (E_{vdW}^{OH}), (b) the electrostatic term (E_{es}^{OH}), (c) the hydrogen bonding term (E_{hb}^{OH}), (d) (a)+(b), and (e) (a)+(b)+(c)

For example, “LEU” denotes leucine, “ARG” denotes arginine, etc. Let \underline{E}_{es}^{OH} be the curve of the electrostatic force between O and H where their charges are both defined by the minimum values. Similarly, let \overline{E}_{es}^{OH} be defined by the maximum values of both O and H. Then, they define the lower and upper bound of the electrostatic force given as follows:

$$\underline{E}_{es}^{OH} = \frac{(-0.5579) \cdot (0.4478)}{d_{lr}^2}. \quad (8)$$

$$\overline{E}_{es}^{OH} = \frac{(-0.8188) \cdot (-0.0425)}{d_{lr}^2} \quad (9)$$

In Fig. 2b, the solid curve denotes \underline{E}_{es}^{OH} and the broken curve denotes \overline{E}_{es}^{OH} .

An example of the interpretation of these graphs is as follows: In the van der Waals force point of view (Fig. 2a), the two atoms O and H will be located and stable at the interatomic distance where E_{vdW}^{OH} minimizes, unless another force affects the system. Similar explanation can be made for the hydrogen bond (Fig. 2c). Therefore, the minimum of each curve is of interest. In Fig. 2a, the minimum is at (2.926, -0.099). In Fig. 2c, the minimum is at (1.950, -5.002). In the case of Fig. 2b, both curves for the upper and lower bound of the electrostatic force are monotonic.

Figure 2d shows $E_{vdW}^{OH} + E_{es}^{OH}$ and Fig. 2e shows $E_{vdW}^{OH} + E_{es}^{OH} + E_{hb}^{OH}$. Each of both figures has two curves because E_{es}^{OH} has two curve instances. In Fig. 2d, the minimum of the upper-bound curve is at (2.929, -0.095) and one of the lower-bound curves is at (2.903, -0.128). In Fig. 2e, the minimum of the upper-bound curve is at (2.302, -1.364) and one of the lower-bound curves is at (2.300, -1.418).

The claim to be made is as follows. Consider the interatomic distance in the example. The most commonly used van der Waals radii of atoms in proteins are those reported by Bondi as follows (unit: Å) [17]: C(1.70), H(1.20), O(1.52), N(1.55), and S(1.80). Without the consideration of hydrogen bonding, Fig. 2d says that the two atoms O and H are most stable where the inter-atomic distance is between 2.903 and 2.929, and this forms an interval of length 0.026 Å where the minimum of the curve in Fig. 2d exists. Because the van der Waals radii of O and H are 1.52 and 1.20, respectively, this interval is in the free space (i.e., O and H do not intersect). The boundaries of O and H are apart from each other: at least 0.183 (= 2.903 - (1.52 + 1.20)) Å and at most 0.209 (= 2.929 - (1.52 + 1.20)) Å away from each other. With hydrogen bonding, however, Fig. 2e says that O and H are most stable where the interatomic distance is between 2.300 and 2.302. In this case, the interatomic distance is between -0.420 and -0.418. This implies that two atoms slightly overlap when they are stable. Recall that the hydrogen bonding can be defined on the boundaries of two molecules only when certain conditions are satisfied and does not occur very often. This observation, therefore, strongly asserts that O and H, in general, stay stable together around their contact distance.

Consider another pair of atoms, O and N, that cannot define a hydrogen bonding. Let E_{vdW}^{ON} and E_{es}^{ON} be the van der Waals potential and electrostatic force between

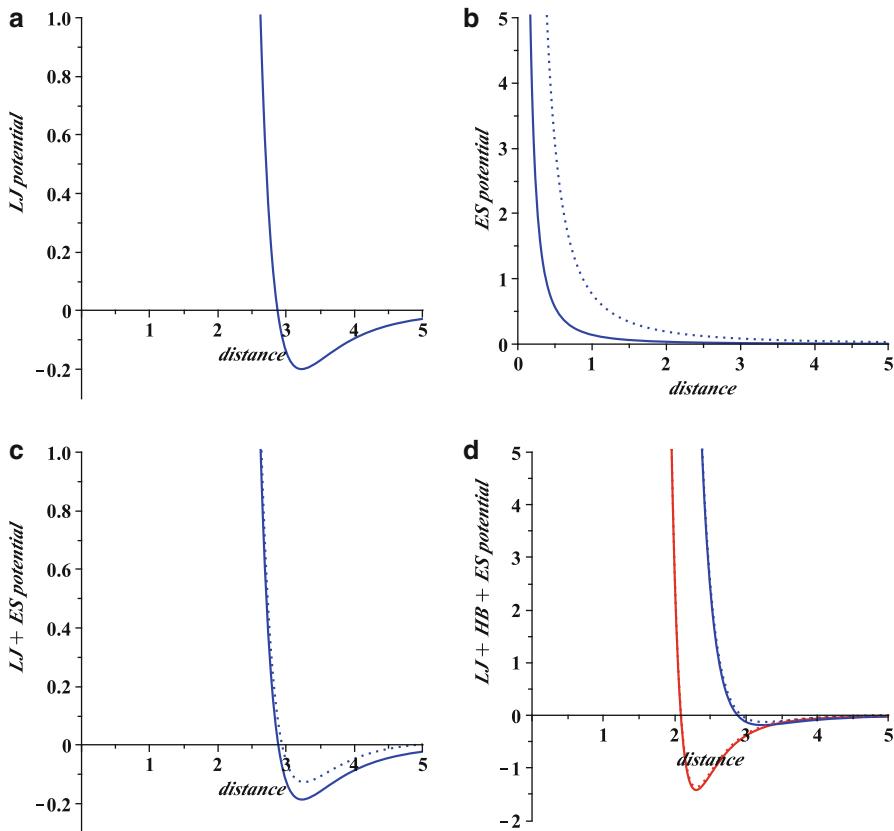


Fig. 3 Nonbonded force terms between O and N using AMBER force field (where the hydrogen bonding does not exist). (a) The (12,6)-Lennard-Jones potential term (E_{vdW}^{ON}), (b) the electrostatic term (E_{es}^{ON}), (c) (a) + (b), and (d) (c) and Fig. 2e together

O and N. Formulae similar to Eqs. (6) and (7) can be defined for this pair, and Fig. 3a, b shows E_{vdW}^{ON} and E_{es}^{ON} , respectively. Figure 3c shows $E_{vdW}^{ON} + E_{es}^{ON}$. The analysis similar to above shows the following: O and N are most stable where their interatomic distance is between 3.229 and 3.259. Subtracting the radii of O and N, the distance is 0.157 and 0.189. This observation also states that there is a slight free space between the two atoms when they are stable. Investigation of other pairs of atoms frequently found in proteins (i.e., C, H, O, N, and S) shows similar phenomena. Therefore, it can be safely said that the binding between two molecules are very stable around their boundaries. In addition, note that there are very few terms in E_{hb} because the hydrogen bonding is only defined on the boundary of receptor. This observation therefore asserts the important consensus that binding usually occurs at the boundary of molecules. However, there are two reservations in this explanation: (i) In a live body, the molecular binding actually occurs in

solvent, and therefore the condition of the above discussion may differ slightly; (ii) the rest atoms in the system may push the atoms on the boundary so that the free space may shrink. **Figure 3d** shows the curves for both OH and ON: For OH, $E_{vdW}^{OH} + E_{es}^{OH} + E_{hb}^{OH}$ is shown in red; for ON, $E_{vdW}^{ON} + E_{es}^{ON}$ is shown in blue.

Therefore, it is very useful to have an effective and efficient way to recognize and manipulate the “molecular boundary.” Then, the search space of the combinatorial optimization problem of [Eq. \(4\)](#) significantly reduces as its combinatorial space is from the boundary atoms in receptor and ligand. As stated earlier, the unavoidable combinatorial explosion nature of docking problem necessarily requires to adopt a heuristic method into the solution process. Frequently used heuristics are genetic algorithm, Monte Carlo (the simulated annealing) method, tabu search, or some combination of these heuristics. Local search method, such as the conjugate gradient method, is also sometimes combined with heuristic method [156]. The generation of some initial solutions is usually the precondition to launch the solution process.

3.2 Scoring Subproblem

Binding affinity between receptor and ligand is measured by the scoring functions whose role is to estimate the binding free energy that estimates the fitness between the molecules. The accuracy of the scoring function has a major impact on the quality of docking results because the final docked pose is usually selected according to their scores. There are three categories of scoring functions [19, 186]:

- *First-principle method* [62, 90, 156]

This method is also called the *force field method* because binding affinity is estimated by the mathematical function of the force field used in molecular mechanics. For instance, [Eq. \(4\)](#) itself is an example of the scoring function of this type. Compared to other two methods below, this approach is usually computationally expensive while the solution quality is better.

- Empirical (or statistical) scoring functions [13–15, 60, 70, 177, 177, 180, 193, 209]

This method estimates binding affinity by a weighted sum of statistically uncorrelated, additive terms where the formula is devised through a multivariate regression analysis based on a set of carefully chosen learning set of structure files from PDB. For instance, the scoring function used by LUDI takes into account hydrogen bonds, ionic interaction, the lipophilic protein-ligand contact surface, and the number of rotatable bonds in the ligand [14]. Hence, taking other types of molecular features into consideration introduces different empirical scoring functions. While the computational speed is very fast, the main disadvantage of this approach is its dependence on the data set used in the regression analysis and the regression model used in the parameter estimation. To improve the precision of estimation, the solvent accessible surface is sometimes taken into consideration [13, 14].

- Knowledge-based scoring functions [49, 50, 73, 85, 159, 201]

This method estimates binding affinity by primarily observing the frequencies of the contacts between protein-ligand atom pairs in the database of

three-dimensional structures of protein-ligand complexes via a procedure based on statistical mechanics [85]. This type is built from statistical analysis of experimentally determined structures in PDB assuming that interatomic distances occurring more often than some average value should represent favorable contacts. Usually this method uses very simple potential function and allows very efficient screening of large compound databases.

Combination of more than one type of scoring functions is sometimes used in the name of *consensus scoring* to reduce a false positive [30, 37, 210]. Depending on the problem at hand, different scoring function needs to be used by compromising accuracy and computational requirement. For example, virtual screening requires to use a computationally efficient scoring function because it has to scan a large compound database to quickly locate a set of lead compounds. Examples of well-known chemical databases are as follows: Available Chemicals Directory [9], Cambridge Structural Database [4, 27], and MACCS-II Drug Data Report [139]. The performance of a scoring function can be evaluated by checking how closely the predicted conformation resembles the one observed in the experimentally determined structure by X-ray crystallography or NMR. Devising a good scoring function still remains an important problem of future study. It is important to note that the “empirical scoring function” is a misnomer in that all scoring functions are indeed empirical in principle. Instead, the *statistical (regression) scoring function* may be more appropriate to denote this type of scoring functions. A comparative evaluation of 11 scoring functions is given in [210]. Other reviews on the scoring function are also available [72, 110, 152, 190, 195, 210].

3.3 Two Primary Uses of Heuristics in Docking

There are two main locations in docking simulation where heuristics are used:

- In the search of the global minimum of the energy function of Eq. (4), and
- the conformation change by assigning a new dihedral angle for each rotatable bond.

The heuristics used in docking software for finding the global minima should also be discussed. In much software, the classical concept of GA is sometimes modified to fit special requirements. For example, Lamarckian GA used in AutoDock or the tabu search-enhanced GA in PSI-DOCK have been reported as exploring the conformational space of ligand efficiently [156, 169]. Most heuristics in docking algorithms require some initial ligand conformations in the vicinity of receptor at the early phase of the algorithm execution. Recent studies suggested locating the initial positions of the ligand around the predicted binding site [42, 158]. Good reviews on the heuristics in docking are available as follows: Westhead et al. reported a comparison of four heuristic algorithms for molecular docking: the simulated annealing, the genetic algorithm, the evolutionary programming, and the tabu search [215]. Baxter et al. reported a docking algorithm using tabu search as the meta-heuristic [12]. In these works, the starting position and orientation of the ligand were randomized around the bounding box containing a pocket, without

explaining how the pocket was computed. Diller and Verlinde reported a comparison of four heuristic algorithms for molecular docking: the random walk, the simulated annealing, the stochastic approximation with smoothing, and the terminal repeller unconstrained subenergy tunneling [52]. In this work, the authors also distributed the initial ligand uniformly around the bounding box. In [110], the use of GA for the docking problem was described in detail.

3.4 Grid: Computational Acceleration

Because of heavy computational requirement in docking simulation, acceleration techniques are important. To reduce computation time, various types of preprocessing are usually performed before main computational process launches for docking. One of the most common and effective techniques is to use a grid system as proposed by Eyck to approximate the force field around the Euclidean space where docking simulation is conducted [63].

Usually a rectangular bounding box containing receptor is first created and the space within the box is transformed into a grid system with a prescribed granularity. Then, the potential energy of ligand is computed and assigned at each grid point so that the grid system is to be used as a lookup table for the rapid evaluation of potential energy via interpolation. Suppose that the ligand is placed at an arbitrary location within the bounding box during docking simulation. Then, the values at the nearby grid points are retrieved and appropriately interpolated to estimate the energy of the ligand at the location. Since the evaluation of the potential energy can be expensive, this interpolation-based technique can save a significant amount of computation. Particularly when a complicated force function is used, this type of preprocessing is effective and therefore most programs like DOCK and AutoDock use variations of such a grid system. For example, see [75, 76]. Without such a supplementary device, the computation time ordinarily increases rather quickly with respect to the product of the numbers of atoms in receptor and ligand.

3.5 Input Data to Docking Simulation

For docking simulation, it is a prerequisite to have the three-dimensional structure (i.e., the atomic coordinates) of molecules. Usually X-ray diffraction, NMR, or homology modeling are used to determine the three-dimensional structure of proteins. Once the structure is solved⁵, the structure data are usually uploaded in a public database called Protein Data Bank (PDB) [179] from which they can be freely

⁵In the structural biology community, the term *solved* or *determined* means that the atomic coordinates of molecules are determined by interpreting data obtained from X-ray crystallography, NMR, or other means such as computational homology modeling.

downloaded. Each PDB file is a text file containing geometric data, physicochemical data, and biological data of a protein.

A PDB file usually contains a protein. However, a PDB file may contain other molecules bound to protein. Water molecules are frequently found. There are also many PDB files containing a protein bound by chemical compounds, DNA, and/or RNA. Depending on application that use PDB file, it is usually necessary to preprocess the file. For example, water molecules, ligands, or other molecules like DNA or RNA may have been removed from the file before the protein is processed. Each PDB file is associated with a resolution which denotes the precision level of the file. A PDB file may not have a complete information about the corresponding protein. Depending on the condition that the protein structure is determined, a subset of the protein structure may be missing from the PDB file.

Each distinct protein has a unique sequence defining its primal structure. Once a protein sequence is identified, it is usually uploaded to UniPROT [199], a public website maintaining protein sequence database. On the other hand, PDB catalogues the structure data of proteins (possibly with other molecules). A distinct protein in UniPROT may correspond to several files (i.e., structures) in PDB because different researcher may solve the structure of the identical protein and upload to the PDB site. For example, the famous P53 (also known as *protein 53* or *tumor protein 53*) corresponds to more than 40 PDB files. Therefore, it is important to choose a good PDB file for the study of particular protein because different PDB files have different file characteristics.

PDB files mostly contain only heavy atoms and ignore most hydrogen atoms because of the limitation of current technology to determine structure. Hence, it may be necessary to generate those missing hydrogen atoms before using in an application. There are rules to generate hydrogen atoms and the following is an example. Suppose that x , y , and z denote the center of atoms. Let $d(x, y)$ be the Euclidean distance between x and y and $\angle xyz$ be the smaller angle at y . Let H be the hydrogen atom, D be the hydrogen donor, and A be the hydrogen acceptor. It is generally agreed that a hydrogen-bond is defined between H and A if all of the following conditions are simultaneously satisfied [143]: (i) $d(H, A) \leq 2.5 \text{ \AA}$, (ii) $d(A, D) \leq 3.9 \text{ \AA}$, and (iii) $\angle AHD > 90^\circ$ (A , H , and D denote their centers). HBPLUS [143] implements this rule for identifying hydrogen bonds between molecules [86, 149, 218]. There are only three cases of atom pairs that hydrogen bonding can be defined: OH, NH, and SH.

4 Flexibility of Molecules

In 1958, Koshland proposed the *induced fit* theory which states that live proteins change their conformation within solvent and ligands also change their conformation when they bind to proteins [115]. Koshland indeed emphasized the notion of “shape.” Hence, an induced fit denotes the change of molecular shape during binding. This idea adds the flexibility of both receptor and ligand where the flexibility comes from rotatable bonds of both molecules. If the flexibility is to

be considered during docking simulation, the computational requirement increases rather quickly. If both molecules are rigid and do not change their conformations during the process of docking simulation, the docking method is called the *rigid body docking*. If one of both parties (usually the ligand) is flexible, it is called the *semi-flexible docking*. If both parties are flexible, it is called the *flexible docking*.

It can easily be verified that the number of possible conformation of flexible ligand increases exponentially in the number of rotatable bonds. A compound with m rotatable bonds has c^m distinct conformations if there are c local minima for each bond. It can be easily verified that $m = O(n)$ in the worst case where n represents the number of atoms in a ligand (or a receptor). This is because protein is a linear sequence of amino acids and each amino acid has a side chain consisting of atoms. Note that there is an upper bound in the number of atoms in side chains. Recall that rigid-body docking problem is already NP-hard. Let $T(\text{RigidDock})$ be the time complexity of an algorithm solving a rigid-body docking. Let $C(\text{FlexLigand})$ and $C(\text{FlexReceptor})$ be the combinatorial complexity of the flexible ligand and flexible receptor, respectively. Then, the time complexity $T(\text{FlexDock})$ of an algorithm solving the flexible docking problem can be given as

$$T(\text{FlexDock}) = C(\text{FlexLigand}) \cdot C(\text{FlexReceptor}) \cdot T(\text{RigidDock}). \quad (10)$$

This implies that the flexible docking problem is much harder than its rigid-body counterpart. Both the computational difficulty and the computational requirement increase in the order of rigid-body docking, semi-flexible docking, and flexible docking.

4.1 Ligand Flexibility

There are three major methods to incorporate ligand flexibility in docking: *stochastic*, *incremental construction*, and *multi-conformer*. The stochastic method is to use randomness in finding new ligand conformations, and the genetic algorithm or the Monte Carlo algorithm is usually used. Suppose that a ligand has a set of m rotatable bonds where the corresponding dihedral angles are represented as $\Theta = \{\theta_1, \theta_2, \dots, \theta_m\}$. Then, Monte Carlo algorithm assigns each $\theta_i \in \Theta$ at random. As usual convention, the Monte Carlo algorithm adopts the annealing concept in that the first cycle is run at high temperature and the later cycles are done at lower temperatures [26, 75, 76, 191, 211]. ICM and the earlier version of AutoDock use the Monte Carlo method for ligand flexibility. Genetic algorithm is also popular for this purpose. In the genetic algorithm, a new conformation is created by the mutation and crossover of the dihedral angles among the population of ligand conformations [70, 90, 156, 168]. DOCK, GOLD, and the later version of AutoDock use the genetic algorithm.

The multi-conformer method is to create multiple instances of ligand conformations of low energy. Then, the rigid-body docking is tried for each ligand

conformation. In other words, this approach uses rigid-body docking several times [36, 125]. This approach tends to produce an exponential number of ligand conformations with respect to rotatable bonds in the ligand. To avoid the combinatorial explosion, the incremental construction method was devised.

The incremental construction method first divides a ligand into a set of fragments at each rotatable bond and each fragment is independently docked within the binding pocket. Then, fragments are incrementally assembled back to form the entire ligand [13, 14, 45, 126, 185, 214]. The idea was first introduced by Goodford [74] and is now very popular for incorporating ligand flexibility. Hammerhead [214], DOCK4.0, and FlexX use this approach. This method is also used for de novo design of ligand [153]. In the process of fragment assembly, the dynamic programming is frequently used [79]. Minimization using a quasi-Newton method is used to avoid unfavorable steric contacts between adjacent atoms in a molecule as much as possible [111]. The comparison among these approaches are discussed in [25].

4.2 Receptor Flexibility

During the induced fit, conformational change also occurs for receptors. In the reflection of receptor flexibility, it is usual to distinguish two locations of receptor from which the flexibility is induced: the side chain and the backbone. While receptor conformational change usually occurs around the side chains, large backbone movements are also sometimes observed. For example, there is a report for seven test cases that caused significant conformational change for more than half of the interface backbone residues [32]. To cope with the receptor flexibility, the approaches above for ligand can also be used. However, care should be exercised because the blind use of these approaches may find the computational requirement prohibitive.

The most significant approach for receptor flexibility is to use *rotamer library* which is a set of rotamers (meaning *rotational isomer*) where each is a side chain of amino acid found in the crystal structure of proteins [125, 183] or may exist in the native state of may-be-found proteins. Hence, a rotamer library can be considered as a discretization of the large, continuous conformation space of side chains into a finite, small number of probable instances.

A rotamer library can be backbone-independent or backbone-dependent depending on whether the dihedral angles of the rotamers (and their frequencies) depend on the backbone conformation or not [91]. The backbone-independent rotamer library is calculated from the structures of side chains without the reference to the backbone conformation, whereas the backbone-dependent rotamer library presents the side-chain conformations (and frequencies) that are dependent on the backbone conformation. If the information about secondary structure such as helix or sheet is used in calculating a rotamer library, it is called secondary-structure dependent. The first rotamer library was reported by Ponder and Richards [175]. The most widely

used are the backbone-dependent one developed by Dunbrack's group [181] and the backbone-independent one developed by Lovell's group [138].⁶

Given a rotamer library, algorithms are designed to find the most favorable receptor conformation by exploring the entire combinatorial space of all rotamer instances for all amino acids of the receptor. Computing the most favorable protein conformation is also another combinatorial optimization problem and is known to be NP-hard [172]. There are two categories of algorithms for this problem: stochastic and deterministic. Stochastic algorithm obvious includes approaches using the genetic algorithm and Monte Carlo simulation with the randomness incorporated. An important stochastic algorithm is FASTER (Fast and Accurate Side-chain Topology and Energy Refinement) developed by Desmet [48]. FASTER is a greedy algorithm iteratively optimizing single protein conformation given that the rotamers of the other positions are fixed. Deterministic algorithm includes those based on the optimization techniques such as the linear programming, the semi-definite programming, or the mixed-integer programming [5, 31, 69, 77, 109, 202, 219]. An important deterministic algorithm is DEE (Dead-End Elimination) developed by Desmet [47, 123]. DEE is one of the most effective algorithms for pruning the search space of rotamers by attempting to eliminate as many rotamers as possible that are guaranteed incompatible with the global minima of energy function. Originally developed as a method for side-chain optimization problem, DEE is now the most popular method for various other problems as well, such as protein design problem.

However, fully incorporating the receptor flexibility into docking problem is still far beyond current technology. It is interesting to find that the concept of side-chain optimization was already used early in 1975 by Levinthal et al. when they initially proposed the computational approach to understand molecular interaction [133]. There were a few reported cases of employing receptor flexibility (even at a limited sense) [21, 184]. Carlson and McCammon presented a good review of approaches to receptor flexibility [28]. Note: It may seem possible and be even better to create a set of random rotamers because a random library may enable to explore a larger search space. However, it turns out that most members of randomly constructed libraries do not fold into stable protein-like structures.

5 Previous Studies on Docking

In 1890, Emil Fischer, the Nobel Laureate for chemistry in 1902, proposed the theory of *lock-and-key* [64, 65] that became the initial proposition of both the molecular binding and the *shape complementarity*. In 1953, Francis Crick, the Nobel Laureate in physiology or medicine in 1962, suggested the idea of a computational approach to the binding between two small molecules through their surfaces in 1953 (the same year that the double-helix structure of DNA was published in

⁶Both rotamer libraries are freely available at: <http://dunbrack.fccc.edu/bbdep/bbdebformat.php> and <http://kinemage.biochem.duke.edu/databases/rotamer.php>.

Nature by Watson and Crick) and called *knob-and-hole* [44]. Crick suggested that shape complementarity in the helical coiled coil could be modeled as knobs fitting into holes. In 1958, Koshland proposed the *induced-fit* theory explaining the mutual adaptation of both molecules in a certain degree during binding [115–117]. Therefore, the induced-fit theory is a generalization of the initial proposition of the lock-and-key concept by adding the concept of flexibility. Both theories of lock-and-key and induced fit assert that molecular binding is largely shape-determined. In 1963, Koshland reported the strong correlation of the function of an enzyme with its structure and proposed that the geometry of enzyme surface must be a critical feature of the enzymatic process [116].

In 1931, Lennard-Jones explained the possibility of modeling the attractive and repulsive forces between two atoms in the van der Waals fields using the inverse powers of the distance between the centers of atoms [130] among other kinds of empirical functions [23,88]. The original form was the form of (13,7)-powers of the distance, and it was later modified to the currently popular function of (12,6)-powers (as shown in Eq.(1)) to accelerate the function evaluation. This change was because the major portion of the computation in the docking (and the molecular dynamics simulation) was spent in the evaluation of the energy function.⁷

The report [173] by Platzer et al., in 1972, from Scheraga's group, Cornell University, seems to be one of the earliest works of computational docking which was applied to an enzyme against substrates. The study already discussed the two distinct aspects of docking: (i) the prediction of location and orientation of a substrate and (ii) the minimization of the estimated free energy. In 1975, Levinthal et al. used computerized molecular modeling to study the arrangement of hemoglobin molecules within a sickling cell by analyzing the geometric contacts between molecules after geometric transformations are made [133]. This study employed a purely geometric approach in analyzing the relation between molecules by using fully determined structures and attempted to maximize the “size” of the contact region in terms of the number of interacting atomic groups.

In 1978, Wodak and Janin first presented an algorithm for docking that solely used the empirical potential energy function for the nonbonded interaction between proteins [216]. In this very first work, they used the simplified (8,6)-Lennard-Jones potential function to refine the relative configuration to allow more interpenetrations (i.e., more intersections) between receptor atoms and ligand atoms.⁸ In the same

⁷ Note that r^{-12} can be computed as $(r^{-6}) * (r^{-6})$ where r^{-6} can be computed by $r^{-2} * r^{-2} * r^{-2}$. Hence, for one function evaluation, r^{-12} (used in the (12,6)-form) requires four multiplications whereas r^{-13} (used in the (13,7)-form) requires five multiplications, both in floating-point arithmetic. Therefore, one multiplication is saved in each function evaluation by the modification. Recall that this function evaluation is performed many times in docking simulation.

⁸ In (8,6)-Lennard-Jones form, r^{-8} term represents the repulsive force. Note that r^{-8} grows slower than r^{-12} as r , the interatomic distance, decreases. Hence, (8,6)-Lennard-Jones form allows more intersections between receptor atoms and ligand atoms than (12,6)-Lennard-Jones form does.

year, Greer and Bush discussed an algorithm for computing the boundary⁹ [78] and proposed the possibility of applying the algorithm for detecting the “shape complementarity” between receptor and ligand. The study by Wodak and Janin in [216] was based on the force function whereas the one by Greer and Bush in [78] was purely geometric.

In 1982, Kuntz’s group, University of California San Francisco, observed the possibility of better using the shape complementarity between a receptor and a ligand [121]. They first extracted a pocket on the receptor boundary and primarily used the geometric fit or shape matching between the pocket and the ligand. They first generated a set of artificial spheres on the surface of the receptor and collected the spheres in each presumptive binding site that corresponded to a pocket. Then, they tried the shape matching between the sphere set in the pocket and the atoms in the ligand by comparing the distribution of intersphere distances for the pocket and the distribution of the interatomic distances for the ligand. They continued the study to speed up and explore more search space [46, 61, 187, 188]. Later in 1993, they also included the minimization of the (12,6)-Lennard-Jones potential energy function [146] and incorporated the flexibility of the ligand [62, 158]. Hence, the approach by Kuntz’s group eventually evolved to a hybrid approach starting from a purely geometric approach. For the shape matching, they used bipartite graph matching, etc. The effort by Kuntz’s group has resulted in the software DOCK whose core part is still being actively used in some softwares. Connolly also suggested using the shape complementarity for docking [41]. There have also been many other studies following the idea of using the shape complementarity in docking [33, 34, 41, 66, 81, 87, 93, 129, 149, 161–164, 187, 206, 208].

An approach solely based on the energy minimization immediately follows. In 1985, Goodford reported an approach using the energy function including the (12,6)-Lennard-Jones potential function, the electrostatic function, and the hydrogen bond function [74]. In 1990, Olson’s group, Scripps in California, extended Goodford’s approach adapting the Monte Carlo method [147, 148] in the search of the global minimum of the energy function [75] based on the AMBER force field. Later, Olson’s group generalized their study to cope with ligand flexibility [76, 155]. In 1998, they adopted the genetic algorithm to replace the Monte Carlo method in the search of the global minimum [156]. The continuing research of this group has resulted in the popular software AutoDock [84, 157]. It is interesting to note that this group is solely based on the energy function minimization and does not take advantage of shape complementarity at all. AutoDock produces initial ligand conformations by placing ligands on the boundary of a bounding box around a receptor. AutoDock is reported to be the most frequently cited docking software in publications [190]. Scheraga’s group is another example of solely using the

⁹Specifically speaking, the boundary was the Lee-Richards (accessible) surface in structural molecular biology term or the offset surface in geometer’s term. For the clear-cut explanation about these terms, see [106].

energy minimization [71, 127, 128, 135, 137, 165] which led to the docking software PRODOCK in 1999 [197].

Starting with the concept of “shape complementarity” by Fischer, there were many studies emphasizing either one of the shape or potential energy. In 1991, S. Kim’s group verified that both geometric and energetic complementarity on molecular boundary are important in docking and proposed *soft docking* concept [87]. In the soft docking, they used both shape and energy: They first used geometric complementarity to effectively and efficiently reduce the solution space and then used the minimization of energy function. Since then, there were many studies and softwares which follows the idea of soft docking. In 1995, the group of Jones and Glen, University of Sheffield and Wellcome Research Lab, respectively, reported GOLD (Genetic Optimization for Ligand Docking) which is a hybrid approach using both shape complementarity and energy function minimization [89, 90]. After extracting a protein pocket (without an explanation), they placed ligand within the pocket using a least-square fitting technique. Then, they applied the minimization of the energy function consisting of (12,6)-Lennard-Jones potential and the hydrogen bonding. They used a genetic algorithm in the search of the global minima of the energy function.

In 1994, Abagyan’s group, European Molecular Biology Laboratory at Heidelberg, reported ICM¹⁰ which is based on the internal coordinate system [1, 3]. In ICM, both the search of global minimum and the ligand flexibility are found by Monte Carlo simulation.

In 1995, Rarey’s group, German National Research Center (GMD) for Computer Science, reported FlexX which emphasizes the explicit exploitation of ligand flexibility in docking. They used the incremental construction algorithm to handle the ligand flexibility [118, 176–178]. They reported FlexE as the derivative of FlexX to add the receptor flexibility using a pre-created, superimposed ensemble of receptor structures [38]. FlexE uses the same scoring function as FlexX.

Sousa et al. reported a good review about the docking software currently used [190]. According to the statistics reported in [190], the five most popular docking softwares are AutoDock, GOLD, FlexX, DOCK, and ICM in the given order.

In the context of this chapter, it may be necessary to mention about the research and development effort for molecular docking simulation from author’s group. Author’s group has developed the theory of beta-complex based on the Voronoi diagram of atoms in molecules (which is briefly explained in the section). Based on the beta-complex theory, a new, powerful docking method has recently been developed [108]. Later sections explain docking problem formulated as the combinatorial optimization problem based on the beta-complex theory. The presented theory was entirely implemented into a software BetaDock and is freely available from the Voronoi Diagram Research Center (VDRC), Hanyang University [204].

¹⁰Initially, ICM was short for Internal Coordinate Modeling [3] and it is now for Internal Coordinate Mechanics [2].

6 Formalization of Topology Structures

As explained earlier, the recognition of molecular boundary is very helpful for facilitating the effective and efficient combinatorial search of ligand location and orientation with respect to receptor. Recognized molecular boundary is in general two-dimensional manifold, and therefore it is useful to have its own two-dimensional topology representation. This is because applications frequently require topology information for efficient query processing. There are different definitions of molecular boundary depending on context [106]. Regardless its definition, it is necessary to have the representation of proximity among molecular atoms for the effective and efficient recognition of molecular boundary. Earlier studies on the molecular shape, however, rarely used the concept of topology.

This section introduces the theory about topology among (spherical) particles, including molecular atoms, in \mathbb{R}^3 from three different perspectives: *primal structures*, *dual structures*, and *molecular structures*. Despite that the primary interest is spherical atoms with different radii in \mathbb{R}^3 , points are also discussed as they may represent atoms with equal radii. The combinatorial complexity of all structures in this section are $O(n^2)$ in the worst case and $O(n)$ on average in \mathbb{R}^3 for n generating particles [7, 8, 166]. For convenience of discussion, all particles are assumed to be in general position so as to avoid computational degeneracy.

6.1 Primal Structures

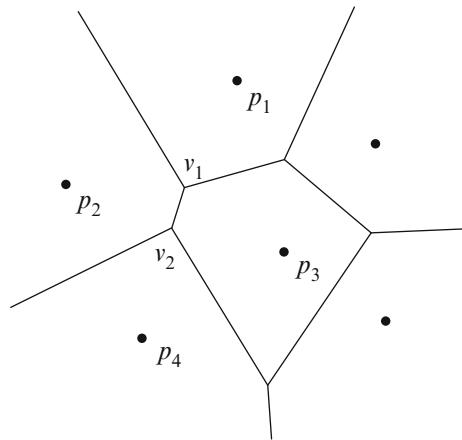
6.1.1 Ordinary Voronoi Diagram of Points

Let $P = \{p_1, p_2, \dots, p_n\}$ be a finite set of point sites in \mathbb{R}^d , $d = 1, 2, \dots$. An *ordinary Voronoi diagram* $VD(P)$ of P tessellates the space \mathbb{R}^d such that each cell in the tessellation corresponds to one of the sites and all the locations in the cell are closer to the corresponding point site than to any other site. Formally, a *Voronoi cell* $VC(p_i)$ of a site p_i is defined as

$$VC(p_i) = \{x \in \mathbb{R}^d \mid d(x, p_i) \leq d(x, p_j) \text{ for } i \neq j\} \quad (11)$$

where $d(x, p)$ is the Euclidean distance between two points $x, p \in \mathbb{R}^d$. Voronoi cell $VC(p)$ is always a convex polyhedron. The ordinary Voronoi diagram $VD(P)$ can be defined as $VD(P) = \{VC(p_1), VC(p_2), \dots, VC(p_n)\}$ where the connectivity among the topological entities (such as vertices, edges, faces, and higher-dimensional cells) of the ordinary Voronoi diagram is appropriately represented. $VD(P)$ is a cell complex and the most compact, concise proximity representation for points in Euclidean space. For the review on the theory, algorithms, and applications of the ordinary Voronoi diagram, see [8, 166]. Efficient and robust codes for computing $VD(P)$ in \mathbb{R}^2 and \mathbb{R}^3 are available [29, 144, 192]. Figure 4 shows an example of the ordinary Voronoi diagram of six points in the plane. Note that the

Fig. 4 An example of the ordinary Voronoi diagram of six points in the plane



circle passing through the three points p_1 , p_2 , and p_3 is centered at the Voronoi vertex v_1 and never contains p_4 inside.

6.1.2 Power Diagram of Weighted Points

Suppose that each point site $p_i \in P$ is assigned with a real number $w_i \geq 0$, where w_i denotes the weight of p_i . The weighted points $p_i^w = (p_i, w_i)$ form a set $P^w = \{p_1^w, p_2^w, \dots, p_n^w\}$. Let $\text{pow}(x, p_i^w) = d(x, p_i)^2 - w_i$ be the power distance. Then, a *power cell* $\text{PC}(p^w)$ is defined as

$$\text{PC}(p_i^w) = \{x \in \mathbb{R}^d \mid \text{pow}(x, p_i^w) \leq \text{pow}(x, q_j^w) \text{ for } i \neq j\}. \quad (12)$$

The power diagram $\text{PD}(P^w)$ can be defined as $\text{PD}(P^w) = \{\text{PC}(p_1^w), \text{PC}(p_2^w), \dots, \text{PC}(p_n^w)\}$ where the connectivity among the topological entities is appropriately represented [7]. $\text{PD}(P^w)$ is also sometimes called the Laguerre diagram or the Dirichlet cell complex. Bisectors in the power diagram are planar. Hence, the power cells in the power diagram are bounded by planar facets and form a cell complex. The power distance corresponding to a point $p \in \mathbb{R}^d$ and the related weight w has a nice geometric interpretation as follows. A weighted point $p^w = (p, w)$ can be viewed as a sphere $s = (p, \sqrt{w})$ where p and \sqrt{w} are the center and radius of s , respectively. Then, $\text{pow}(x, p^w)$ can be interpreted as a tangential distance from point x to sphere s . Hence, the topology of the power diagram $\text{PD}(P^w)$ is not necessarily identical to the topology of the ordinary Voronoi diagram $\text{VD}(P)$. Figure 6 shows an example of the power diagram for six circles in the plane where the circle centers are at the points in Fig. 4. Note that the empty tangent circle defined by the three circles a_1 , a_2 , and a_3 is not centered at v_3 in general and may intersect the other circle a_4 .

Power diagram reflects the size of spheres in the tessellation at a certain level, while the ordinary Voronoi diagram of points does not. Its computation is based on the solution of a linear system like the ordinary Voronoi diagram of points. A power bisector between two spheres s_i and s_j always passes through the intersection of the boundaries of the spheres, if s_i and s_j do intersect. Hence, power diagram correctly represents the intersection events among the intersecting spheres. However, it should be emphasized that power diagram has a few shortcomings for Euclidean space applications: (i) the Euclidean proximity among nonintersecting spheres may not be correctly represented in the topology of the power diagram, and (ii) a power cell $\text{PC}(p^w)$ may not contain the center p of its generating sphere. Aurenhammer [7] gives an excellent review on the power diagram.

6.1.3 Voronoi Diagram of Spheres

Consider a d -dimensional sphere set $S = \{s_1, s_2, \dots, s_n\}$ where $s_i = (p_i, r_i)$ is a sphere (or an atom) with the center p_i and radius r_i . Hence, $s_i = \{q \in \mathbb{R}^d \mid \|q - p_i\| \leq r_i\}$. Assume that $s_i \not\subseteq s_j$ for all $i \neq j$, but $s_i \cap s_j \neq \emptyset$ for some $i \neq j$. Let $\text{VC}(s_i)$ denote a *Voronoi cell* for s_i defined as follows:

$$\text{VC}(s_i) = \{x \in \mathbb{R}^d \mid d(x, p_i) - r_i \leq d(x, p_j) - r_j \quad \text{for } i \neq j\}. \quad (13)$$

Then, the Voronoi diagram $\text{VD}(S)$ for the sphere set S is defined as $\text{VD}(S) = \{\text{VC}(s_1), \text{VC}(s_2), \dots, \text{VC}(s_n)\}$ where the connectivity among the topological entities are appropriately represented. In \mathbb{R}^3 , the topological entities are vertices (V-vertices), edges (V-edges), faces (V-faces), and cells (V-cells) whose sets are denoted as $V^V = \{v_1^V, v_2^V, \dots\}$, $E^V = \{e_1^V, e_2^V, \dots\}$, $F^V = \{f_1^V, f_2^V, \dots\}$, and $C^V = \{c_1^V, c_2^V, \dots, c_n^V\}$, respectively. By definition, a V-vertex v^V is the center of an empty sphere tangent to the boundaries of four nearby atoms, and a V-edge e^V is a locus of points equidistant from the boundaries of three nearby atoms. A V-face f^V is the mid-surface between the boundaries of two nearby atoms. While both $\text{VD}(P)$ and $\text{PD}(P^w)$ are cell complexes, $\text{VD}(S)$ may not be a cell complex because a Voronoi face in $\text{VD}(S)$ may contain holes. The Voronoi diagram of spheres in \mathbb{R}^3 can be computed in $O(n^3)$ time in the worst case [98] and in $O(n)$ time on average [35], where n is the number of spheres in S . Figure 5 shows an example of the Voronoi diagram for the six circles in Fig. 6. Verify the topology difference in the two figures. Note that the empty tangent circle defined by the three circles a_1, a_2 , and a_4 is centered at the Voronoi vertex v_5 and does not intersect the other circle a_3 .

The Voronoi diagram of spheres is also known as an additively weighted Voronoi diagram in computational geometry [166]. However, from the application point of view (particularly for structural molecular biology point of view), the name of “Voronoi diagram of spheres (or spherical atoms)” is preferred because the interpretation of the weights associated with point sites are more intuitive when they are considered as the radii of spheres or atoms in the Euclidean space. For the details of the Voronoi diagram of spheres, see [95, 98].

Fig. 5 An example of the ordinary Voronoi diagram of six points in the plane

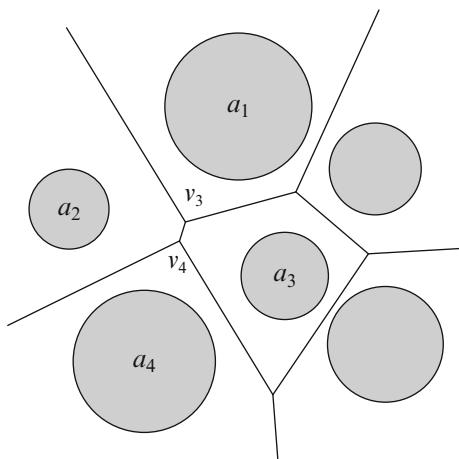
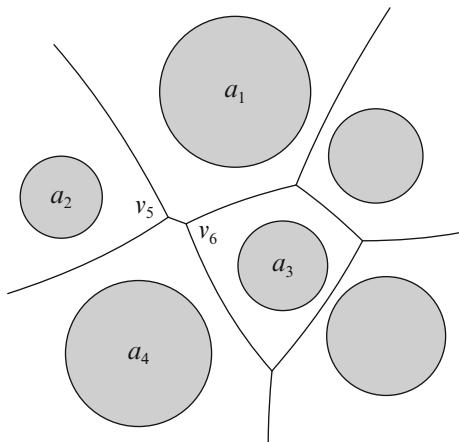


Fig. 6 An example of the power diagram of six circles in the plane

6.2 Dual Structures

Dual structure is usually more convenient than the corresponding primal structure for reasoning the spatial proximity among particles. If a dual structure is a simplicial complex, it can be usually more compactly stored in the memory than its primal structure while the query performance is maintained at the same level of efficiency. In addition, algorithms for topology traversal in a simplicial complex are much simpler.

6.2.1 Dual Mapping

The topological dual mapping between the primal and the dual structures has been recently reported in [100, 104]. Let \mathbb{D} be a *dual operator* which transforms \mathcal{VD} to \mathcal{QT} which is defined as follows:

- $\mathbb{D} : c^{\mathcal{V}} \Rightarrow v^{\mathcal{Q}}$ is a mapping from a V-cell $c^{\mathcal{V}} \in C^{\mathcal{V}}$ to a q-vertex $v^{\mathcal{Q}} \in V^{\mathcal{Q}}$. The coordinate of the q-vertex $v^{\mathcal{Q}}$ is the center p of the atom a corresponding to the V-cell $c^{\mathcal{V}}$.
- $\mathbb{D} : f^{\mathcal{V}} \Rightarrow e^{\mathcal{Q}}$ is a mapping from a V-face $f^{\mathcal{V}} \in F^{\mathcal{V}}$ to a q-edge $e^{\mathcal{Q}} \in E^{\mathcal{Q}}$. The q-edge $e^{\mathcal{Q}}$ is a line segment bounded by $v_i^{\mathcal{Q}}$ and $v_j^{\mathcal{Q}}$, for distinct i and j , where the atoms a_i and a_j define the V-face $f^{\mathcal{V}}$. The orientation of $e^{\mathcal{Q}}$ can be arbitrary.
- $\mathbb{D} : e^{\mathcal{V}} \Rightarrow f^{\mathcal{Q}}$ is a mapping from a V-edge $e^{\mathcal{V}} \in E^{\mathcal{V}}$ to a q-face $f^{\mathcal{Q}} \in F^{\mathcal{Q}}$. The q-face $f^{\mathcal{Q}}$ is a triangle whose vertices are $v_i^{\mathcal{Q}}$, $v_j^{\mathcal{Q}}$, and $v_k^{\mathcal{Q}}$, for distinct i , j , and k , where the atoms a_i , a_j , and a_k define the V-edge $e^{\mathcal{V}}$. The orientation of $f^{\mathcal{Q}}$ should be consistent with the orientation of $e^{\mathcal{V}}$.
- $\mathbb{D} : v^{\mathcal{V}} \Rightarrow c^{\mathcal{Q}}$ is a mapping from a V-vertex $v^{\mathcal{V}} \in V^{\mathcal{V}}$ to a q-cell $c^{\mathcal{Q}} \in C^{\mathcal{Q}}$. The q-cell $c^{\mathcal{Q}}$ is a topological tetrahedron bounded by four vertices $v_i^{\mathcal{Q}}$, $v_j^{\mathcal{Q}}$, $v_k^{\mathcal{Q}}$, and $v_l^{\mathcal{Q}}$, for distinct i , j , k , and l , which correspond to the four atoms defining the V-vertex $v^{\mathcal{V}}$. The orientation among the four q-vertices is identical to the orientation of the four tangent points between the four atoms and the empty tangent sphere corresponding to the V-vertex. Each pair of q-vertices defines a q-edge $e^{\mathcal{Q}}$ of $c^{\mathcal{Q}}$, and each triplet of $v^{\mathcal{Q}}$'s defines a q-face $f^{\mathcal{Q}}$ of $c^{\mathcal{Q}}$, where the orientation of $e^{\mathcal{Q}}$ and $f^{\mathcal{Q}}$ are maintained as stated above.

6.2.2 Delaunay Triangulation

Given a set P of d -dimensional points, the Delaunay triangulation $\text{DT}(P)$ is the set of simplexes $d\text{-}\sigma$ such that the $(d + 1)$ vertices of $d\text{-}\sigma$ are in P and the interior of $d\text{-}\sigma$ does not contain any point of P . It is well-known that $\text{DT}(P)$ is the topological dual of the Voronoi diagram $\text{VD}(P)$ of P obtained by applying the dual operator \mathbb{D} to $\text{VD}(P)$. Due to several convenient properties, $\text{DT}(P)$ has been extensively studied and used. Robust and efficient algorithms and codes for $\text{DT}(P)$ in \mathbb{R}^2 and \mathbb{R}^3 are available [29, 144, 192]. The transformation between $\text{VD}(P)$ and $\text{DT}(P)$ takes linear time with respect to the number of topological entities involved. $\text{VD}(P)$ is a cell complex and $\text{DT}(P)$ is a simplicial complex¹¹ if the points are in general position [7, 8, 16].

6.2.3 Regular Triangulation

The process for obtaining regular triangulation is similar to the process for Delaunay triangulation. The regular triangulation can be obtained by applying the dual operator \mathbb{D} to the power diagram [141]. The regular triangulation is also a simplicial complex and the transformation between the power diagram and the regular triangulation can be done in linear time with respect to the number of topological entities involved [166].

¹¹A *simplicial complex* \mathcal{C} is a finite set of simplexes such that (1) any face of a simplex in \mathcal{C} is also a simplex in \mathcal{C} and (2) if two simplexes in \mathcal{C} intersect, they intersect at a lower-dimensional simplex which is a face of both simplexes [16].

6.2.4 Quasi-triangulation

While the Voronoi diagrams of circles in \mathbb{R}^2 and spheres in \mathbb{R}^3 have been discussed since the eighties, the topological dual of the Voronoi diagram has never been discussed until the notion of the quasi-triangulation was proposed by author's group in 2006 [100]. Recall that the Voronoi diagram of spheres may not be a cellcomplex. It turns out that the dual structure of the Voronoi diagram of spheres is not necessarily a simplicial complex. In fact, the dual structure may not even satisfy the conditions for being a complex.

Suppose that the dual operator \mathbb{D} is applied to the Voronoi diagram $VD(S)$ of a sphere set S . Then, a Voronoi diagram $VD(S)$ is mapped to a dual structure called the *quasi-triangulation* $QT(S)$ where $QT(S) = \{\tau_1, \tau_2, \dots, \Delta_1, \Delta_2, \dots\}$, where τ and Δ denote a tetrahedron and a triangle, respectively, which are the dual cells in the quasi-triangulation $QT(S)$. Its topology is assumed to be appropriately represented. Let $V^Q = \{v_1^Q, v_2^Q, \dots, v_n^Q\}$, $E^Q = \{e_1^Q, e_2^Q, \dots\}$, $F^Q = \{f_1^Q, f_2^Q, \dots\}$, and $C^Q = \{c_1^Q, c_2^Q, \dots\}$ denote the sets of vertex simplexes, edge simplexes, face simplexes, and cell simplexes in $QT(S)$ in \mathbb{R}^3 , respectively. There are n vertices in V^Q since a vertex v_i^Q corresponds to a generator sphere s_i . For the details of the quasi-triangulation, see [96, 97, 100].

Even though a quasi-triangulation is not a valid triangulation, its topological entities are still vertices, edges, triangles, and tetrahedra. $QT(S)$ is called a quasi-“triangulation” since “most” of the tetrahedra in the dual structure of the Voronoi diagram of spheres satisfy the “local” condition for a simplicial complex. It is called a “quasi”-triangulation since there may be some, “mostly very few,” tetrahedra which altogether do not form a complex and their influences are local in the dual structure. The quasi-triangulation of most protein models reveals this characteristics.

The concept of quasi-ness has been devised due to the following two important reasons:

- The proximity among particles which is stored in the topology of the Voronoi diagram can be most compactly stored in the dual structure.
- To solve application problems more effectively and efficiently using the proximity information, it is necessary to have a well-defined dual structure stored in an appropriate data structure. The computation of the beta-complex and beta-shape is such an example.

6.3 Molecular Structure

6.3.1 Alpha-Shape and Weighted Alpha-Shape

The theory of the alpha-shape is very powerful for the spatial reasoning among point clouds. The initial concept of the alpha-shape was proposed for \mathbb{R}^2 in 1983 [56], and its development for \mathbb{R}^3 was published in 1994 [55] by Edelsbrunner's group. The following intuitive description is adopted from [55]. Suppose that α is a real number such that $0 \leq \alpha \leq \infty$.

Consider a three-dimensional space filled with a soft matter with a set P of solid points scattered within the matter. Carving out the matter with an omnipresent spherical eraser with the radius α will result in a shape called an *alpha-hull*. Since the eraser is omnipresent, there can be interior voids as well. The spherical eraser is called a probe. If the alpha-hull is straightened by substituting straight edges for the circular ones and triangles for the spherical caps, the resulting object bounded by the edges and triangles is called the *alpha-shape* of P .

If $\alpha = \infty$, the alpha-shape is identical to the convex hull of P . For $\alpha = 0$, the alpha-shape reduces to the point set P itself. In general, alpha-shapes can be concave and disconnected. Alpha-shapes can contain two-dimensional patches of triangles and one-dimensional strings of edges. An alpha-shape may have handles and interior voids. Each instance of an alpha-shape for a given value of α can be efficiently computed by checking if each simplex in the Delaunay triangulation should remain in the alpha-shape. The theory of alpha-shape is based on the Delaunay triangulation of points and therefore does not account for the size variations among atoms centered at the points in P at all. The subset of the Delaunay triangulation contained within or on the boundary of an alpha-shape is called the alpha-complex.

To reflect the size variation among atoms, the alpha-shape was extended to the weighted alpha-shape using the regular triangulation [54]. Concepts and algorithms for the weighted alpha-shape are very close to the ordinary alpha-shape except that the distance measure used is now the power distance. Hence, a regular triangulation for the weighted points is first computed, and the edges and faces which are considered to be larger than a spherical eraser, in the power distance sense, are removed from the regular triangulation. Hence, the weighted alpha-shape reflects the size variations among spheres into consideration at a certain level. The weighted alpha-shapes have been used in biology to recognize the shapes of proteins since the atoms in the proteins usually have different sizes. However, the distance metric, the power distance, is not very convenient in many aspects. While the power diagram has correct information about the intersection event between atoms, the spatial proximity between nonintersecting atoms cannot be made correctly unless a global search is performed in the worst case. The weighted alpha-complex inherits this shortcoming. Therefore, for many spatial reasoning problems among nonintersecting atoms, the weighted alpha-shape algorithm requires to recompute the power diagram after an input atom set is modified based on the problem formulation in terms of intersecting atoms. Difficulty to comprehend the behavior of the weighted alpha-shape is another significant drawback. It is not easy to intuitively visualize and anticipate how the theory may operate in the Euclidean space. The discussions of the theoretical aspects of the weighted alpha-shapes are not widely available, very few [54].

6.3.2 Beta-Complex and Beta-Shape

To fully incorporate the size differences of spheres in the Euclidean space, the concept of the beta-complex and beta-shape has been proposed by author's group [101, 105]. The beta-complex is very powerful in reasoning the spatial

properties of a sphere set S in the ordinary, intuitive Euclidean distance sense. The beta-complex is an “abstraction” of the set S as a set of vertices, edges, faces, and cells with an appropriate representation of its topology among the simplexes. On the other hand, the beta-complex is also an “exact representation” of the set S itself. Hence, the beta-complex is not an ordinary approximation of S . Due to these properties, the beta-complex can be used for efficient processing of both the global reasoning about the shape of S and the local computation such as mass properties like the volume and area of the union of spheres in S . The beta-shape defines the topology among atoms on the boundary and facilitates the efficient computation of the Connolly surface [182], the extraction of pockets [99], etc.

Intuitive explanation of the beta-shape follows. Consider a soft matter filled \mathbb{R}^3 with some spherical, metallic balls of varying radii scattered within the matter. Carving out the soft matter with an omnipresent spherical eraser with the radius β will result in a shape called the *beta-hull*. Hence, the boundary of the resulting beta-hull contains the *blending surface* created by the spherical eraser as the blending ball in addition to the subset of ball boundaries. There can be interior voids as well. The spherical eraser is called a β -probe.

Given a beta-hull, suppose that the beta-hull is straightened as follows. Replace each spherical cap on the boundary of the beta-hull by a triangle whose vertices are the centers of balls touching the cap. Then, the resulting shape consists of those triangular faces, the edges shared by the faces, and the vertices corresponding to the centers of balls touching the spherical caps. The straightened object bounded by the triangular, planar facets is the *beta-shape* of A . It is called a beta-shape because the shape is defined by adopting the concept of β -lending in geometric modeling. No explicit computation for blending operation is performed during the computation but the existence of such blending patches on the boundary of beta-hull is only checked by referring to the appropriate simplexes in the quasi-triangulation. The computations of the beta-complex and the beta-shape are well-defined and very efficient algorithms are available.

Explanation for the beta-complex follows. Consider again the spherical rocks, or atoms, $A = \{a_1, a_2, \dots, a_n\}$ inside the soft matter filled \mathbb{R}^3 where $a_i \cap a_j = \emptyset, i \neq j$. Let a spherical eraser b of size 0 be located at a point $x \in \mathbb{R}^3$ where $x \notin \forall a \in A$. Increase the radius of the eraser b from zero until it touches one rock a_i . While keeping its tangency with a_i , increase the radius of b until it touches another rock $a_j, i \neq j$. In some cases, b may not touch a_j even though its radius increases to ∞ . If it touches two rocks a_i and a_j , increase the radius until it touches $a_k, i \neq j \neq k$, while the tangent contact is maintained. Increase the radius of b until it reaches β . When a_i is touched, a vertex simplex is defined at the center of a_i . When a_i , and a_j are touched, an edge simplex is also defined, in addition to another new vertex simplex corresponding to a_j , between the centers of two rocks. When a_i, a_j , and a_k are touched, a face simplex is defined by the three centers in addition to one more vertex and two more edge simplexes. If a_i, a_j, a_k and a_l are touched, a tetrahedral cell simplex is similarly defined. Hence, during this process starting from a location x , four vertex simplexes, six edge simplexes, four face simplexes, and one tetrahedral cell simplex are created. By repeating this

process for all possible locations in \mathbb{R}^3 , we get the beta-complex of A . Hence, a beta-complex, once computed, can represent the interior topology of the atom set very nicely, while a beta-shape is very convenient for representing the topology among the boundary atoms. The beta-complex is the subset of the quasi-triangulation that is topologically contained within or on the boundary of the beta-shape.

It is important to mention the following. The beta-complex can be computed very efficiently from the quasi-triangulation of a given molecule. The computation is done simply by checking each simplex in the quasi-triangulation if it should remain in the beta-complex for a given value of β . Because the Voronoi diagram of a given molecule is unique, the quasi-triangulation is also unique. This property implies that only one Voronoi diagram (and therefore only one quasi-triangulation) needs to be computed for the beta-complex for any value of β . Therefore, any kind of problems in Euclidean distance can be solved by one Voronoi diagram and therefore one quasi-triangulation. Almost all application problems are Euclidean. The beta-shape can also be computed efficiently from the beta-complex because the boundary of the beta-shape is the boundary of the beta-complex.

[Figure 7](#) illustrates the concept of the Voronoi diagram, the quasi-triangulation, the beta-complex, and beta-shape in the plane. [Figure 7a](#) shows a two-dimensional molecule S consisting of two-dimensional atoms, and [Fig. 7b](#) shows the boundary of S . Note that the boundary contains two internal voids, a few shallow depressions, and a deep depression on the exterior boundary. [Figure 7c](#) is the beta-hull of S corresponding to a small β -probe (shown in the black circle). Note that only one interior void is left in [Fig. 7c](#), which corresponds to the larger void in [Fig. 7b](#). The void corresponding to the smaller one has disappeared. [Figure 7d](#) shows the beta-shape corresponding to the beta-hull of [Fig. 7c](#) where the shaded region is the interior of the beta-shape. The beta-shape in [Fig. 7d](#) has an interior void corresponding to the void of the beta-hull and a dangling edge corresponding to a pair of atoms that are exposed to or touched by the probe. The dangling edge corresponds to the deep depression in the external boundary of the beta-hull. [Figure 7e](#) shows the beta-complex corresponding to the beta-shape in [Fig. 7d](#). Note that each vertex of the beta-shape and beta-complex corresponds to an atom. [Figure 7f, g](#) shows the beta-shape and beta-complex corresponding to a larger probe, respectively. Note that both the dangling edge and the internal void in the beta-shape of [Fig. 7d](#) have now disappeared. The dotted line segments in [Fig. 7g](#) form the quasi-triangulation of the molecule S together with the beta-complex. [Figure 7h](#) shows a three-dimensional protein model, and [Fig. 7i](#) shows its beta-shape corresponding to a probe of the radius 1.4 Å. The white spot in the middle of the beta-shape in [Fig. 7i](#) in fact denotes a tunnel passing through the beta-hull of the protein.

[Figure 8a](#) shows an atomic complex of HIV(PDB accession code: 5hvp). [Figure 8b](#) shows the beta-hull (also frequently called the Connolly surface) of the molecule corresponding to a probe with a radius of 1.4 Å which approximates a water molecule. [Figure 8c-f](#) is the (boundary of) beta-shapes corresponding to the probes of sizes 0, 1.4, 5, and ∞ Å, respectively.

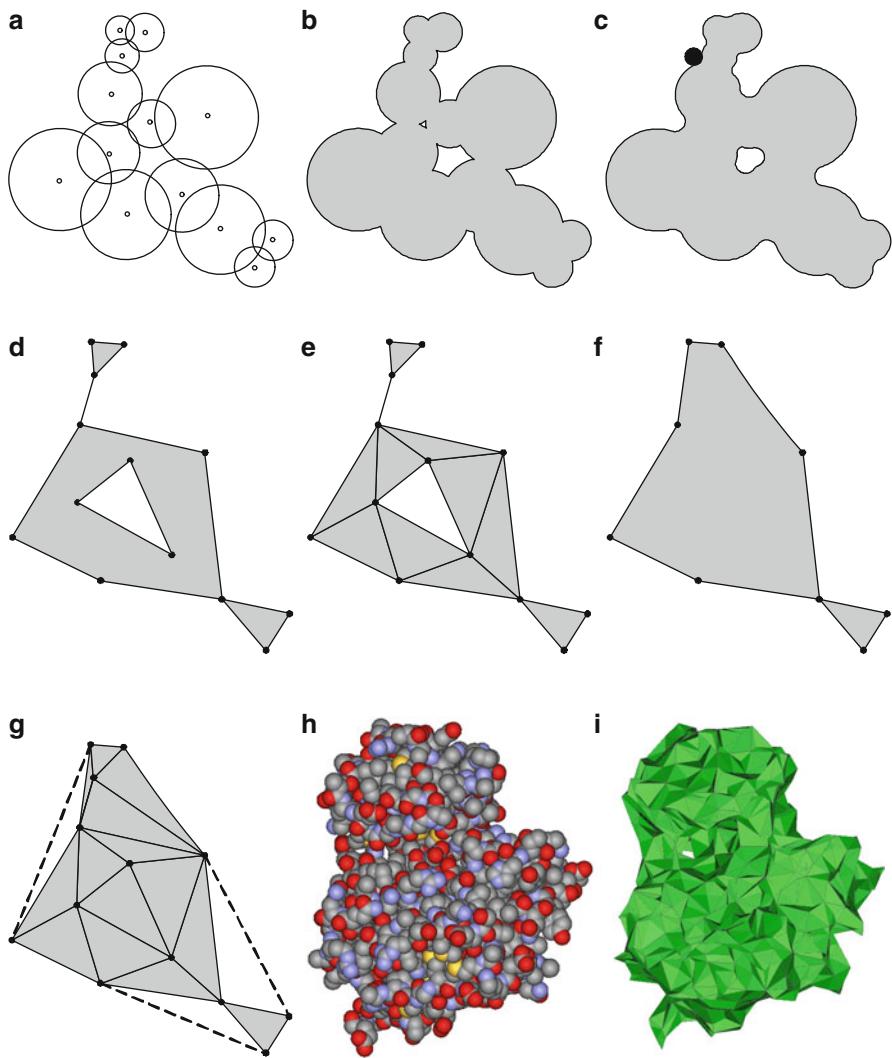


Fig. 7 Protein, the boundaries of the protein, the beta-shape, and the beta-complex. (a) A set of 13 atoms in \mathbb{R}^2 , (b) the vdw-boundary, (c) the molecular surface corresponding to a small probe, (d) the corresponding beta-shape, (e) the corresponding beta-complex, (f) a beta-shape corresponding to a larger probe, (g) the corresponding beta-complex with simplexes removed from the quasi-triangulation, (h) the protein tyrosine kinase (1xba, 2,068 atoms), (i) the beta-shape of 1xba corresponding to $\beta = 1.42 \text{ \AA}$

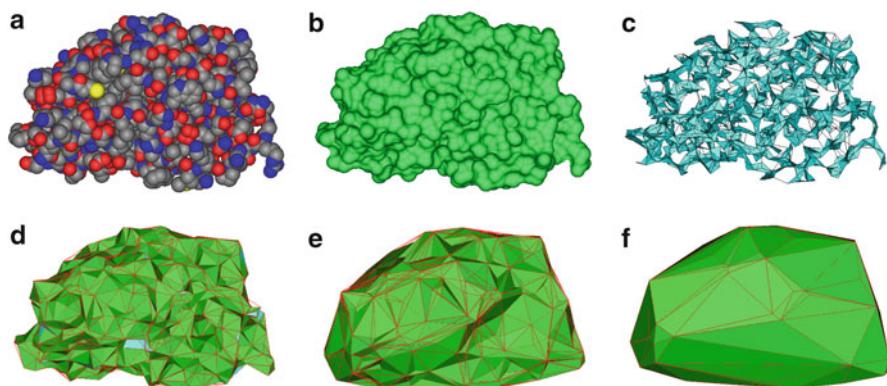


Fig. 8 The beta-hull and different beta-shapes defined by different probes from an atom set. (a) An atomic complex of HIV (PDB accession code: 5hvp), (b) the beta-hull (or the Connolly surface) obtained by a probe with the radius 1.4 Å. (i.e., a water molecule), (c)–(f) the beta-shape corresponding to different values of β : (c) $\beta = 0 \text{ \AA}$, (d) $\beta = 1.4 \text{ \AA}$, (e) $\beta = 5 \text{ \AA}$, (f) $\beta = \infty \text{ \AA}$. Note that (b) and (d) corresponds to the same probe

7 Pocket Extraction Using Beta-Shape

As previously explained, the interaction between receptor and ligand occurs at their boundaries. Assuming the rigidity of molecules (i.e., ignoring flexibility of their conformations), this is obviously true. More importantly, the concept of funnel in the binding energy landscape naturally leads to the concept of geometric and physicochemical complementarity between receptor and ligand. It is generally believed that the bottom of a funnel in the energy landscape corresponds to the depressed shape, called *pocket*, on receptor boundary. The shape complementarity is indeed the fundamental concept of lock-and-key proposed by Fischer in 1890 [64]. Since then, many studies repeatedly have shown that the shape complementarity is of primary importance in docking. In fact, the solution obtained from direct formulation of docking problem as a single, global energy minimization problem is usually very close to the solution obtained by the approach using pocket. Hence, it is not a surprise to find that there were several studies on docking purely based on geometry only [40, 46, 121].

The concept of the shape complementarity naturally leads to the concept of locating a probable docking site, pocket, on the boundary of receptor that a ligand may bind with a high probability. Shape complementarity has long been used in docking. In the past, however, there was no framework of formal theory for properly defining and representing molecular boundary but some ad hoc approaches. Having the powerful theory of the beta-complex and beta-shape to detect molecular boundary, the docking problem can be transformed to a matching problem for sets of points, sets of edges, sets of faces, or their combination. This transformation

naturally leads the docking problem to a combinatorial optimization problem among these discrete entities representing the continuous molecular boundaries. Sections 7–9 present this transformation as used in the docking method and software, called BetaDock, developed by author's group [107, 108].

7.1 Previous Works

There are three approaches for recognizing pockets on molecular boundary: a grid-based approach, a sphere-coating approach, and an approach based on the proximity among atoms on the receptor boundary. Being both conceptually and computationally straightforward, the grid-based approach was the first method of choice that used a three-dimensional spatial lattice of the space occupied by the receptor to reason the relations among the grid points in the lattice [83, 134, 203]. The grid points, associated with some attributes, were then used to extract the exterior boundary of receptor and to recognize the depressed regions on the surface. The sphere-coating approach was to place a set of artificial spherical balls not intersecting the receptor in the space and reasons about the spatial property of the spheres [18, 121, 122]. After making some efforts to use the concept of filling small spheres around a receptor and separating some meaningful chunks of spheres, the mainstream of this approach proceeded to use the mathematically rigorous, computationally efficient, and robust representation of the atomic complex.

Compared to the approaches above, the Voronoi diagrams and their derivative structures provided a formal, theoretical framework to the pocket extraction problem. The alpha-shape was the first generation of this effort and it was used to reason about the proximity among atoms on the boundary of molecules, assuming that all atoms were identically sized. Liang et al. presented a discrete-flow algorithm for extracting pockets [136] based on the definition of a pocket using a alpha-shape [57]. The pocket recognized in this approach was equivalent to the invagination in [112]. Peters et al. reported an algorithm using the alpha-shape to construct the topology among atoms on the surface of a protein [171]. There was an effort to extend the idea of the ordinary alpha-shape to the weighted alpha-shape to cope with the size variation among atoms in proteins.

Kim et al. reported an algorithm to extract pockets of protein based on the theory of the beta-shape. The idea was to use two instances of beta-shapes where one was defined by a small probe (corresponding to water molecule) and the other was defined by a large probe (corresponding to a ligand) [99]. The beta-shape defined by a small β value was called an *inner beta-shape*, and one defined by a large β was called an *outer beta-shape*. Then, the difference between the outer beta-shape and the inner beta-shape defined pockets on the protein boundary. To facilitate the computational efficiency, the boundary of the beta-shape was represented by the winged-edge data structure so that the traversal among simplexes could be done very efficiently. To overcome the non-manifoldness of the beta-shape, manifoldization technique was used [102].

7.2 Extraction of Pocket Using Beta-Shape

Let \mathcal{S} be the beta-shape where $\beta = 1.4$ which corresponds to the probe (1.4-probe) with the radius 1.4 \AA representing water molecule. Proteins usually exist in a cell which is mostly filled with water. Hence, the smallest molecular unit interacting with proteins is usually water. It is well-known that the shape of water molecule is nonsymmetric. However, the interaction of a real water molecule with a biomolecule is too complex to analytically, precisely analyze. Furthermore, the interaction is stochastic rather than deterministic. Hence, it is usual practice to approximate the water molecule with a spherical probe with the radius 1.4 \AA .

Let Σ be the set of all simplexes (i.e., vertices, edges, and triangular faces) of the boundary $\partial\mathcal{S}$ of \mathcal{S} . Let \mathcal{S}^* be the beta-shape corresponding to a threshold β -value $\beta^* > 1.4$. Consider a vertex simplex $\sigma \in \Sigma$ such that σ is exposed to 1.4-probe but not exposed to β^* -probe. This implies that σ , which is a vertex exposed to air in \mathcal{S} , is now internal to $\partial\mathcal{S}^*$. Such a vertex σ on the boundary of \mathcal{S} is called a *potential pocket vertex* and denoted as the *pp-vertex*. The atom corresponding to a pp-vertex is called the *potential pocket atom* and denoted as the *pp-atom*. Let V^{pp} be the set of pp-vertices and A^{pp} be the set of pp-atoms. Suppose that $f \subset \partial\mathcal{S}$ be a face such that all three vertices of f are pp-vertices. Then, f is called the *potential pocket face*, denoted as the *pp-face*. Let F^{pp} be the set of all pp-faces. If $f_i \in F^{pp}$ and $f_j \in F^{pp}$ share an edge in common, they are said to be *edge-connected*.

Let \mathcal{F}^{\max} be a maximal set of edge-connected pp-faces in F^{pp} that is the set of edge-connected faces with the largest number of faces. Hence, for a given F^{pp} , \mathcal{F}^{\max} can be easily found. Let $F^{pp} = F^{pp} - \mathcal{F}^{\max}$. Repeating this operation until F^{pp} is empty, F^{pp} can be decomposed into a set of edge-connected components $\mathcal{F}^{\text{MAX}} = \{\mathcal{F}_1^{\max}, \mathcal{F}_2^{\max}, \dots\}$. Suppose that the elements in \mathcal{F}^{MAX} are in a sorted order such that $|\mathcal{F}_i^{\max}| \geq |\mathcal{F}_j^{\max}|$ if $i < j$, where $|\mathcal{F}_i^{\max}|$ denotes the number of faces in \mathcal{F}_i^{\max} . Hence, \mathcal{F}_i^{\max} does not have fewer atoms than \mathcal{F}_j^{\max} , and \mathcal{F}_i^{\max} is said bigger than \mathcal{F}_j^{\max} , $i < j$. Each component $\mathcal{F}_i^{\max} \in \mathcal{F}^{\text{MAX}}$ is called a *raw pocket*. Let \mathcal{V}_i^{\max} and \mathcal{A}_i^{\max} be the sets of all distinct pp-vertices and the corresponding pp-atoms collected from the faces in the raw pocket \mathcal{F}_i^{\max} , respectively. The computation of the raw pockets is very fast.

[Figure 9a](#) illustrates the PDB model 1t46 which is a combined complex consisting of tyrosine kinase (in gray) with imatinib (in blue) which was found via X-ray crystallography. This type of model, the protein structure with bound compounds, is called a *bound* model. The compound in a bound model is usually regarded as the optimal solution of the computational docking simulation. Note that the most part of the imatinib boundary is buried in the receptor. [Figure 9b](#) shows the tyrosine kinase with a partial transparency. [Figure 9c](#) shows the beta-shape \mathcal{S} of the tyrosine kinase which corresponds to the probe of radius 1.4 \AA . [Figure 9d](#) shows the imatinib. The minimum sphere S_{\min} enclosing the imatinib is computed with the radius 8.13 \AA .

[Figure 10](#) illustrates some examples of the atom set \mathcal{A}_1^{\max} , the biggest raw pocket of the model 1t46 for the various values of β . The blue set is the bound ligand, the imatinib. In all figures of [Fig. 10a–f](#), the imatinib is positioned at the identical location in the receptor. The red set in [Fig. 10](#) is the largest raw pocket \mathcal{A}_1^{\max}

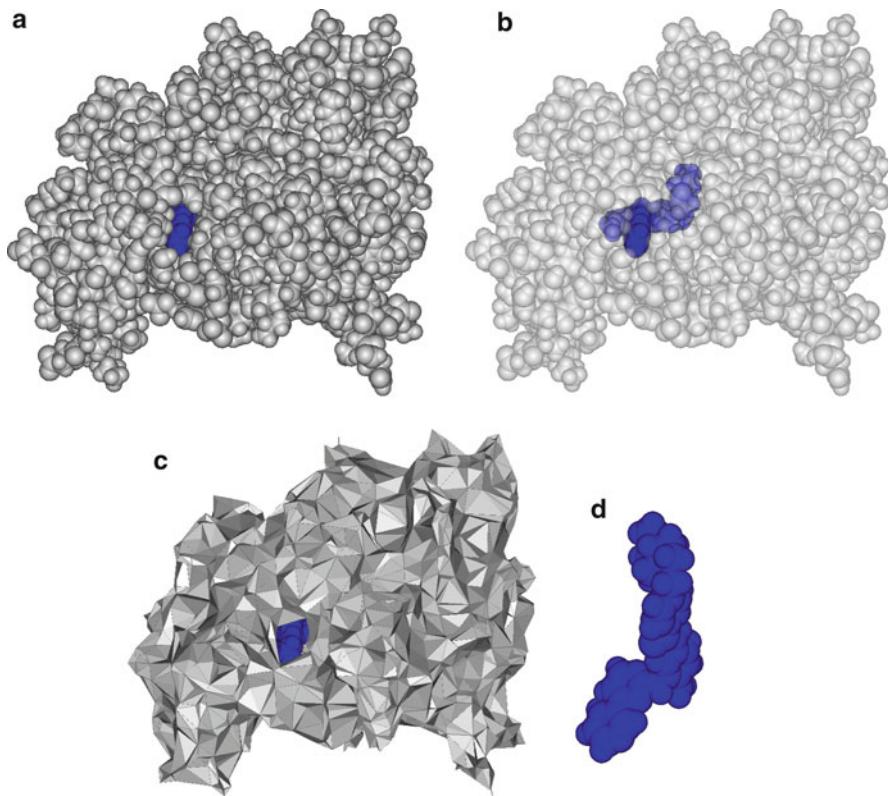


Fig. 9 The protein 1t46 (tyrosine kinase) with a bound ligand (imatinib): (a) space-filling model, (b) transparency applied model, (c) beta-shape for tyrosine kinase corresponding to the probe of water molecule, and (d) imatinib

corresponding to various threshold values of β^* . Figure 10a shows A_1^{\max} where $\beta^*=1.5 \text{ \AA}$. Obviously, 1.5 \AA is significantly smaller than 8.13 \AA , the radius of the minimum sphere of the ligand, and therefore this pocket surely cannot contain the ligand. Therefore, A_1^{\max} corresponding to 1.5 \AA is either located far away from the ligand or interacts with the ligand at a relatively small portion of the interface between the ligand and the receptor. Figure 10b–f shows A_1^{\max} where β^* is 2.0, 3.0, 4.0, 5.0, and 5.71 \AA , respectively. Note that the size of A_1^{\max} increases as β^* increases. When $\beta^* \geq 3.0 \text{ \AA}$, it seems that A_1^{\max} almost completely contains the ligand. Deciding the optimal value of β^* is a challenge and left for future study.

A raw pocket extracted by the procedure above may have noise and therefore the raw pocket goes through a refining process called the pocket refinement. Figure 11 shows the largest pocket of tyrosine kinase extracted by the explained algorithm viewed from three different orientations. In this figure, the raw pocket is visualized as a set of faces on the beta-shape boundary. There are red faces and blue faces in

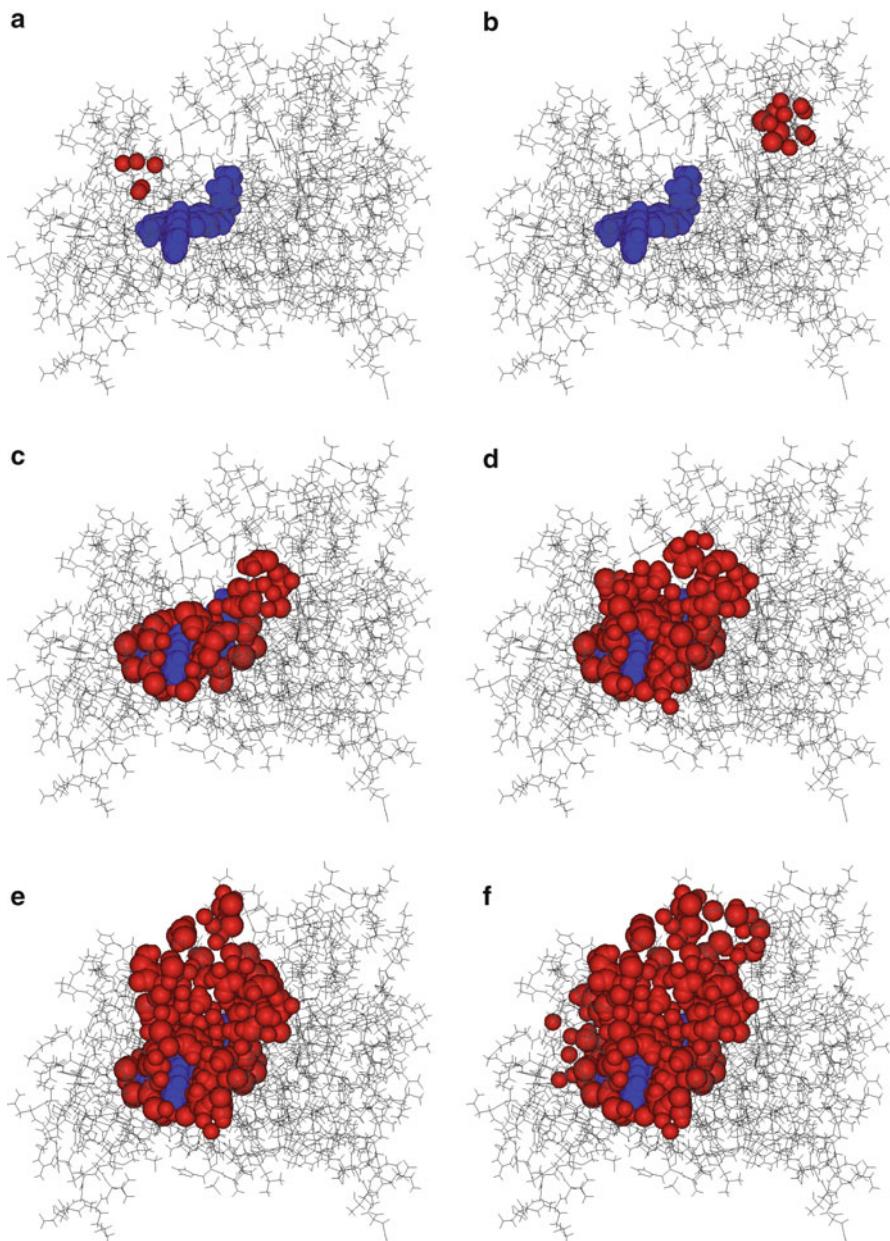


Fig. 10 Atoms in the largest raw pocket \mathcal{F}_l^{\max} of 1t46 corresponding to a particular threshold value. The blue is the ligand (imatinib), and the red are the pockets corresponding to β^* as follows: (a) $\beta^* = 1.5 \text{ \AA}$, (b) $\beta^* = 2.0 \text{ \AA}$, (c) $\beta^* = 3.0 \text{ \AA}$, (d) $\beta^* = 4.0 \text{ \AA}$, (e) $\beta^* = 5.0 \text{ \AA}$, and (f) $\beta^* = 5.71 \text{ \AA}$

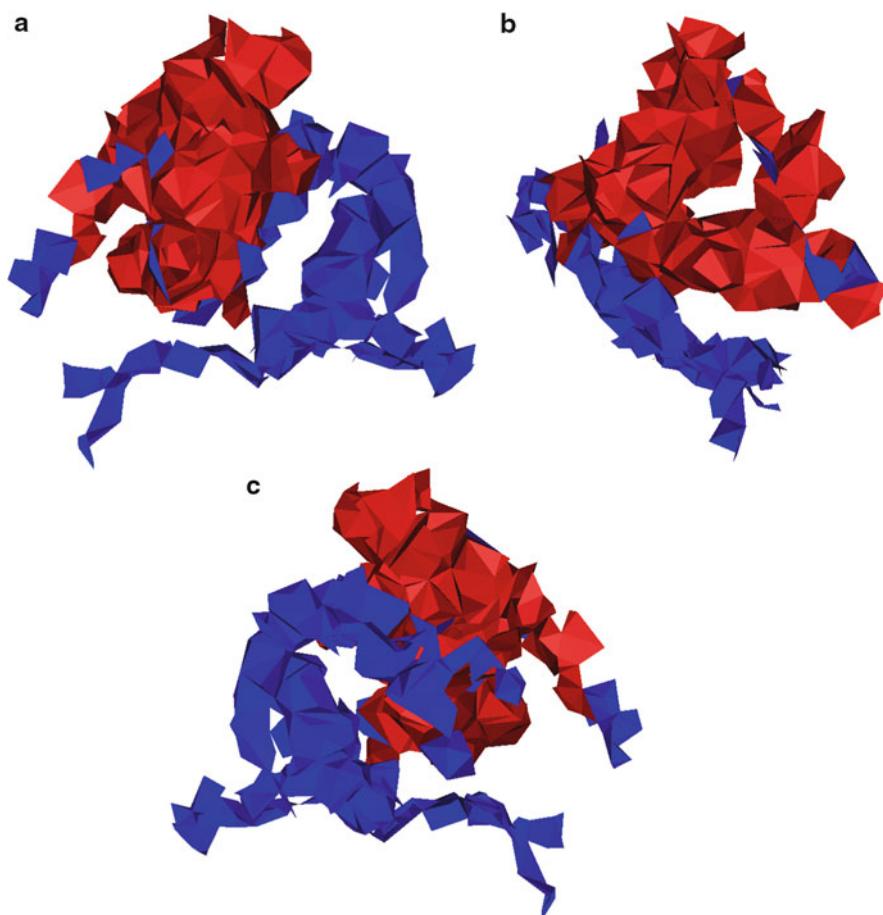


Fig. 11 The largest raw pocket of tyrosine kinase viewed from three different orientations. (a), (b), and (c) 1t46 where $\beta_\theta = 5.71 \text{ \AA}$

the figure. The blue faces are considered as noise and have to be removed from the raw pocket. Therefore, the red face set is the eventually recognized, largest pocket.

There may be other raw pockets: the second largest \mathcal{F}_2^{\max} , the third largest \mathcal{F}_3^{\max} , etc. Figure 12a shows the top three largest potential pockets \mathcal{F}_1^{\max} , \mathcal{F}_2^{\max} , and \mathcal{F}_3^{\max} (to be more precise, \mathcal{A}_1^{\max} , \mathcal{A}_2^{\max} , and \mathcal{A}_3^{\max} , respectively) for the threshold β^* -value 2.0 Å. The red set, the green set, and the brown set denote the largest, the second largest, and the third largest pocket, respectively. Figure 12b, c shows the equivalent pockets corresponding to the threshold β^* -values of 4.0 and 5.71 Å, respectively.

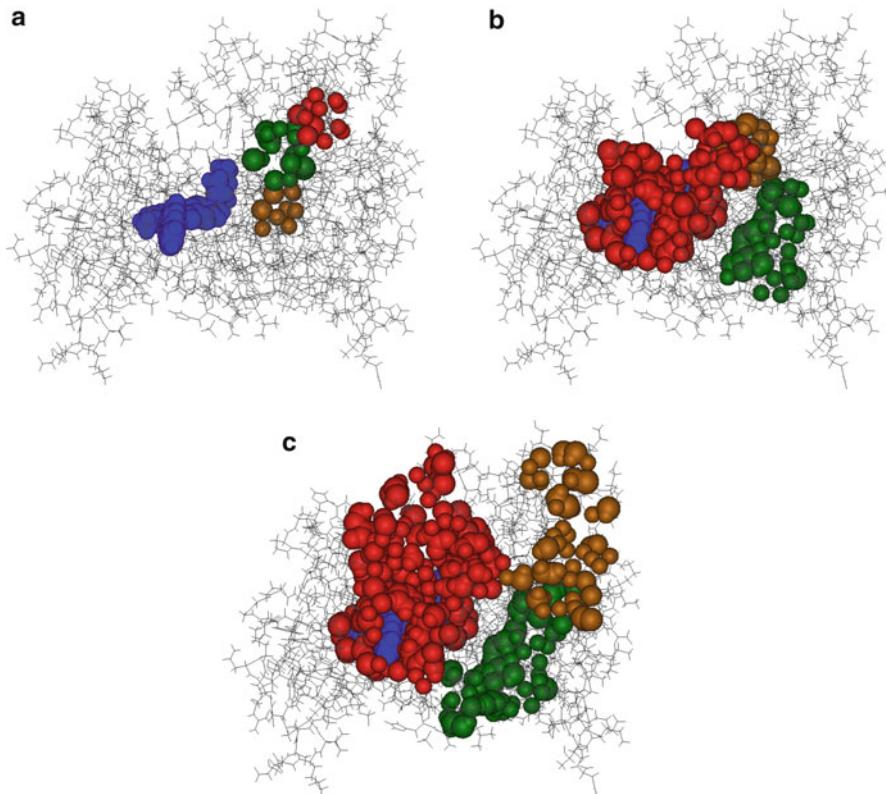


Fig. 12 Three largest raw pockets of 1t46 corresponding to three threshold values. Blue atoms constitute the ligand (imatinib). The red, green, and brown atoms constitute \mathcal{F}_1^{\max} , \mathcal{F}_2^{\max} , and \mathcal{F}_3^{\max} . The values of β^* are as follows: (a) $\beta^* = 2.0 \text{ \AA}$, (b) $\beta^* = 4.0 \text{ \AA}$, and (c) $\beta^* = 5.71 \text{ \AA}$

8 Generation of Initial Ligand Conformations

Finding an optimal pose of ligand in a docking problem is NP-hard and a heuristic is inevitable. Before launching the search of the global optimum via the genetic algorithm, it is necessary to compute an initial population $\Xi^0 = \{\xi_1^0, \xi_2^0, \dots, \xi_q^0\}$ of chromosomes where each chromosome represents an initial ligand conformation with respect to a receptor. Because the best position of the ligand is most probably in the vicinity of a pocket on the boundary of a receptor, the generation of Ξ^0 uses the shape complementarity between the recognized pocket and the ligand L . For this purpose, we use the beta-shapes of the receptor R corresponding to $\beta = 1.4 \text{ \AA}$, which corresponds to water molecule.

Each chromosome consists of six *genes*: three for the location and another three for the orientation of the ligand with respect to the pocket of R , assuming that R is fixed and centered at the origin. The set of chromosomes is called the *population*.



Fig. 13 Placement of ligand into a pocket

Hence, the initial ligand conformations constitute an initial population Ξ^0 where $\xi \in \Xi^0$ is represented as $\xi^0 = (\theta_x, \theta_y, \theta_z, x, y, z)$. For the notational convenience, $\xi = (g_1, g_2, \dots, g_6)$ is also used when it is convenient. This representation of the chromosome was used by many previous studies in docking.

Figure 13 shows a two-dimensional analogy of the process. The cluster of shaded circles denote the atoms constituting a pocket on the receptor, and the cluster of white circles denote the atoms of a ligand. First, some offset points (shown as black dots) are computed from the shaded circles toward the outside of the receptor, as shown in Fig. 13a. An offset point is computed as the center of a sphere, shown as the broken circle, tangent to the shaded atoms where the tangent sphere represents a typical atom in the ligand. The offset amount can be determined in various ways by reflecting the radius of atoms. Suppose that O is the set of offset points. Second, a subset $O' \subset O$ of consecutive offset points is computed where $|O'| = |L|$ for the ligand L . In this example, $|O'| = 3$ and Fig. 13b shows the leftmost three offset points selected. Then, the ligand L is transformed by optimally superposing the centers of atoms in L to O' , where the distance between L and O' is minimized. Then, the transformed ligand is considered as an initial chromosome. Figure 13c shows another chromosome produced from the set of the rightmost three offset points.

8.1 Superposition of Two Point Sets with Identical Cardinalities

The superposition between the centers of atoms in L with O' is formulated as the following optimization problem. Let L' be the superposed conformation of the ligand after the rigid body transformation and considered as an initial chromosome $\xi^0 \in \Xi^0$. Assume that the pocket has m faces. Then, the offsetting operation produces m offset points. Let $O = \{o_1, o_2, \dots, o_m\}$ be the set of the offset points. The following computation is repeated as many times as the number of the required initial chromosomes.

Let $\Lambda = \{c_1, c_2, \dots, c_m\}$ be the set of points where c_i represents the center of an atom $l_i \in L$. Note that $|\Lambda| = |O|$. Then, an optimal superposition of Λ onto O is attempted as follows. Suppose that O is fixed. Then, Λ is moved via a rigid-body

transformation to superpose onto O . Then, the optimal superposition of Λ onto O is formulated as the following integer linear programming problem:

Problem 1 Find the transformation T between Λ and O such that

$$\text{Minimize } \mathcal{D} = \sum_{o_i \in O, c_j \in \Lambda} ||o_i - T * c_j|| x_{ij} \quad (14)$$

$$\text{Subject to } \sum_i x_{ij} = 1, \text{ for all } j, 1 \leq j \leq m$$

$$\sum_j x_{ij} = 1, \text{ for all } i, 1 \leq i \leq m$$

$$x_{ij} = 0 \text{ or } 1$$

In other words, the transformation T is such that the distance \mathcal{D} between O and the transformed Λ is minimized, where \mathcal{D} is the sum of the Euclidean distances in all pairwise assignments (satisfying the constraints) between points $o_i \in O$ and $c_j \in \Lambda$, $1 \leq i, j \leq |\Lambda|$. Finding such an optimal transformation is hard even though $|O| = |\Lambda| = m$ [124]. Hence, iterative heuristic algorithms are usually employed based on an approach frequently used in the structure superposition problem as follows [51, 82, 92, 94].

First, an association between O and Λ is defined as follows. Select one point $o_i \in O$ and one point $c_j \in \Lambda$. After associating o_i and c_j , they are removed from O and Λ , respectively. If there is any remaining element in O and Λ , this procedure is repeated. Then, this procedure yields a set of associations $E(O, \Lambda) = \{(o_1, c_1), (o_2, c_2), \dots, (o_m, c_m)\}$ which is called the *equivalence* between O and Λ . Secondly, the following problem is solved.

Problem 2 Find the transformation T such that

$$\text{Minimize } \mathcal{D}' = \sum_{i=1}^m (o_i - T * c_i)^2 \quad (15)$$

Hence, Eq. (15) finds the optimal transformation so that the distance \mathcal{D}' between O and Λ is minimized with the equivalence constraint $E(O, \Lambda)$. Based on the observation in [59], the singular value decomposition (SVD) solves Eq. (15) $O(1)$ time in the worst case. The details of SVD for the superposition of two sets with an identical number of points is skipped. However, the solution to Eq. (15) does not guarantee the global minimum of Eq. (14). Hence, the procedure iterates as follows. Let $\Lambda' = \{c'_1, c'_2, \dots, c'_m\}$ be the set of transformed points by T computed from Eq. (15). Then, the new equivalence $E(O, \Lambda')$ is computed after the transformation.

Let $E(O, \Lambda') = \{(o_{i1}, c_{j1}), (o_{i2}, c_{j2}), \dots, (o_{ir}, c_{jr})\}$ be a new equivalence between O and Λ' so that $\sum_{k=1}^m |o_{ik} - c_{jk}|$ is minimized. Then, finding the new

equivalence $E(O, \Lambda')$ can be formulated as an assignment problem, a special case of an integer linear programming problem. Let $d_{ij} = dist(o_i, c_j)$ which denotes the Euclidean distance between o_i and c_j . Note that all d_{ij} for $1 \leq i \leq m$ and $1 \leq j \leq m$ are known after the transformation via T . Then, the assignment problem is defined as follows:

Problem 3 Solve the following problem.

$$\begin{aligned} & \text{Minimize} \quad \sum_i \sum_j d_{ij} x_{ij} \\ & \text{Subject to} \quad \sum_i x_{ij} = 1, \text{ for all } j, 1 \leq j \leq m \\ & \quad \sum_j x_{ij} = 1, \text{ for all } i, 1 \leq i \leq m \\ & \quad x_{ij} = 0 \text{ or } 1 \end{aligned} \tag{16}$$

Then, the new equivalence $E(O, \Lambda')$ is optimal between O and Λ' . The assignment problem can be efficiently solved by the Hungarian method initially proposed by Kuhn [120] and can be solved by the Edmond and Karp algorithm in $O(n^3)$ time in the worst case where n represents $|O| = |\Lambda'|$ [58]. Therefore, iterating the solution process of [Problems 2](#) and [3](#) a few times yields a solution of [Problem 1](#) (which may not be optimal) that gives one of the chromosomes in Ξ .

9 Search of Global Minima

Given an initial population (i.e., a set of initial chromosomes), the genetic algorithm is applied in the search of the optimal solution. While there may be variations, this chapter presents an approach employed by authors group. The fitness function used is a slight variation of [Eq.\(4\)](#) where $\varepsilon(d_{lr}) = d_{lr}$ and $\cos \theta = 1$ (implying $\theta = 180^\circ$). Therefore, the fitness function of BetaDock is as follows:

$$F = \sum_{l \in L} \sum_{r \in R} \left\{ \frac{A_{lr}}{d_{lr}^{12}} - \frac{B_{lr}}{d_{lr}^6} \right\} + \sum_{l \in L} \sum_{r \in R} \frac{q_l q_r}{d_{lr}^2} + \sum_{l \in L} \sum_{r \in R} \left\{ \frac{C_{lr}}{d_{lr}^{12}} - \frac{D_{lr}}{d_{lr}^{10}} \right\} \tag{17}$$

where L and R are ligand and receptor, respectively. BetaDock uses AMBER parameters [43] for the van der Waals force and the electrostatic force and AutoDock parameters for the hydrogen-bond interaction [84, 157].

9.1 Genetic Operators

Suppose that the k -th generation $\Xi^k = \{\xi_1^k, \xi_2^k, \dots, \xi_q^k\}$ of q chromosomes is available and each chromosome $\xi_i^k \in \Xi^k$ is associated with a fitness function value F_i^k from Eq.(17). The process proceeds to the $(k+1)$ -th generation Ξ^{k+1} by applying the three standard operators of genetic algorithm: selection, crossover, and mutation.

9.1.1 Selection

Suppose that the q chromosomes in Ξ^k are ordered so that $F_i^k \leq F_j^k$, $0 < i < j \leq q$. First, π_{select} percent of chromosomes is selected from the front of Ξ^k and copied into Ξ^{k+1} . Hence, these chromosomes have lower energy level than the others. Recall that the lower the value of F_i^k is, the better the chromosome is in docking simulation. These chromosomes are stored in another temporary storage Q_{temp} for later use after the crossover and the mutation operations are also completed. Let $|\Xi^{k+1}|$ be the number of current chromosomes in Ξ^{k+1} . Then, $q - |\Xi^{k+1}|$ chromosomes are selected from Ξ^k so that one with lower energy has a higher chance to be selected, and thus selected chromosomes are copied into Ξ^{k+1} . Hence, a new chromosome set Ξ^{k+1} is eventually created completely when $|\Xi^{k+1}| = |\Xi^k|$. Note that some chromosomes in Ξ^{k+1} can be redundant. The redundancy in Ξ^{k+1} has two important implications. A chromosomes with a better fitness function value has a higher probability to have redundancy. This implies that the neighbors of a good chromosome get a higher probability to be explored. On the other hand, by allowing redundancy, the search space itself is actually reduced by the reduction of the gene variety in the chromosome set Ξ^{k+1} , thus reducing the probability of finding a better solution.

9.1.2 Crossover

The crossover operator produces new chromosomes in Ξ^{k+1} by interchanging and/or mixing genes between chromosomes in the same population. Recall that all chromosomes are already located in a relatively narrow region in a pocket. From Ξ^{k+1} , a subset $\widehat{\Xi} \subset \Xi^{k+1}$ is selected with the crossover probability π_{cross} . If $|\widehat{\Xi}|$ is an odd number, we select one more chromosome at random to make $|\widehat{\Xi}|$ even. From $\widehat{\Xi}$, $\frac{1}{2}|\widehat{\Xi}|$ pairs of chromosomes are created in the order of selection. Then, each pair of chromosomes in $\widehat{\Xi}$ produces a pair of new chromosomes in Ξ^{k+1} . Three variants of crossover operators are studied as follows.

Two-Point Crossover: Consider a chromosome pair $\phi = (\xi_i, \xi_j) \in \widehat{\Xi}$. Recall that a chromosome ξ has six genes represented as $\xi_i = (\theta_x^i, \theta_y^i, \theta_z^i, x^i, y^i, z^i) = (g_1^i, g_2^i, \dots, g_6^i)$. First, two genes g_{left}^i and g_{right}^i are located at random from the six genes in ξ_i , where $\text{left} < \text{right}$. This is equivalent to selecting two elements from the array of ξ_i (which consists of six elements). Let Δ_i be the subset of ξ_i consisting of g_i^1 and g_i^2 and those other genes located between g_i^1 and g_i^2 on ξ_i . For example, suppose that $g_{\text{left}}^i = g_2^i$ and $g_{\text{right}}^i = g_5^i$. Then, $\Delta_i = \{g_2^i, g_3^i, g_4^i, g_5^i\}$. Let Δ_j be the set of genes on ξ_j corresponding to Δ_i . In the example, $\Delta_j = \{g_2^j, g_3^j, g_4^j, g_5^j\}$.

Then, Δ_i and Δ_j are switched between ξ_i and ξ_j to complete the crossover operation for the pair ϕ . This rule is called the standard *two-point rule* for crossover in the genetic algorithm.

Interpolated Crossover: Given a pair of chromosomes ξ_i and ξ_j , both in $\widehat{\Xi}$, each pair of corresponding genes from both chromosomes is interpolated. Suppose that $i < j$. Then, $F_i \leq F_j$. The interpolation is defined between two chromosomes as follows:

$$g'_k = (0.75 * g_k^i) + (0.25 * g_k^j), \quad 1 \leq k \leq 6 \quad (18)$$

$$g''_k = (0.25 * g_k^i) + (0.75 * g_k^j), \quad 1 \leq k \leq 6 \quad (19)$$

Then, $\{g'_1, g'_2, g'_3, g'_4, g'_5, g'_6\}$ and $\{g''_1, g''_2, g''_3, g''_4, g''_5, g''_6\}$ are the new chromosomes in Ξ^{k+1} . The motivation for this interpolated crossover is to increase the number of distinct genes in the entire chromosomes in Ξ^{k+1} .

Annealed Interpolated Crossover: Adding a simple simulated annealing concept to Eqs. (18) and (19) yields the following:

$$g'_k = (0.75 * u_1 * g_k^i) + (0.25 * u_2 * g_k^j), \quad 1 \leq k \leq 6 \quad (20)$$

$$g''_k = (0.25 * u_3 * g_k^i) + (0.75 * u_4 * g_k^j), \quad 1 \leq k \leq 6 \quad (21)$$

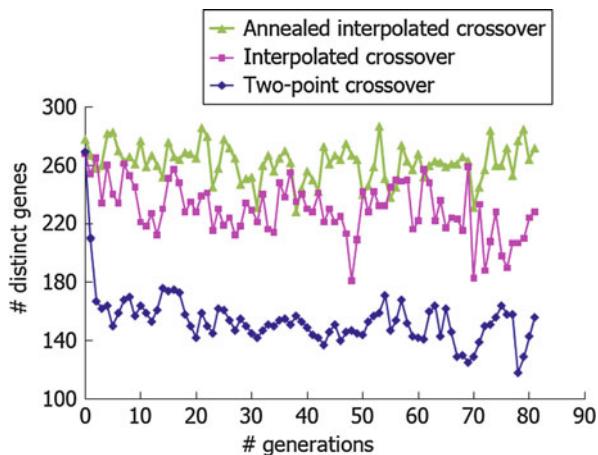
where u_1, u_2, u_3 , and u_4 are random numbers between 0 and 1. The rationale behind Eqs. (20) and (21) is the following: It is preferred to have a larger search space than that could be provided by the standard two-point crossover scheme which simply switches some genes in two chromosomes without changing any gene value. The motivation of the annealed interpolated crossover is to attempt to maximize gene diversity in Ξ^{k+1} .

To verify the behavior of the three crossover operators, an experiment was conducted as follows. Initially, the genetic algorithm was launched with 50 distinct initial chromosomes. Therefore, there were initially a little less than 300 distinct genes because there existed some redundancy in the initial gene set. Figure 14 shows the distribution of the number of distinct genes in Ξ^k with respect to the generation number k . The lowest curve with blue diamonds corresponds to the traditional, standard two-point crossover. Note that the curve decays quite rapidly. The middle curve with red rectangles corresponds to the interpolated crossover. The topmost curve with green triangles corresponds to the annealed interpolated crossover. Note that the gene diversity is maximally maintained in the annealed interpolated crossover.

9.1.3 Mutation

For each chromosome in Ξ^{k+1} , each gene is mutated with the mutation probability π_{mutate} . Suppose that a gene g is selected to be mutated where $g \in \{g_1, g_2, g_3\}$. Then, g is a rotation gene and a random number is produced between 0 and 360 to be added to the current value of g . Suppose that g is a translation gene

Fig. 14 Distribution of the number of distinct genes in each generation. The *blue diamond* at the bottom is the standard two-point crossover. The *red rectangle* in the middle is the interpolated crossover. The *green triangle* at the top is the annealed interpolated crossover



(i.e., $g \in \{g_4, g_5, g_6\}$). Then, the value of g is changed by the amount selected from the radius of the minimum enclosing sphere of the ligand. The direction of the move is also determined at random.

9.2 Genetic Iteration

Given a new set of chromosomes $\Xi^{k+1} = \{\xi_1^{k+1}, \xi_2^{k+1}, \dots, \xi_q^{k+1}\}$, its fitness is measured by evaluating the fitness function values F_i^{k+1} for all chromosomes in Ξ^{k+1} . Then, Q_{temp} is concatenated to Ξ^{k+1} and sorted according to the fitness function value. This is the restoration of the chromosomes with good fitness function values because they may disappear during the crossover and the mutation operation. Taking the first q chromosomes from the sorted set as the set Ξ^{k+1} , one iteration of the genetic algorithm is completed. This process repeats until a termination condition is encountered.

The genetic iteration process terminates when one of the following two conditions is encountered. First, the iteration terminates when the a priori defined maximum number of generations N_{maxgen} is encountered. Second, there is the maximum number of consecutive iterations N_{stable} where the best energy function value is stable and does not improve. Hence, if the genetic iteration does not improve the currently best solution during N_{stable} consecutive generations, the algorithm terminates.

The presented docking method was implemented into the software BetaDock and tested using a standard benchmark data set. It turned out that the performance of BetaDock is very powerful despite that the current version is an intermediate result of an ongoing research. For the details of the benchmark result, please refer to [107, 108].

10 Conclusion

There has long been a general consensus that protein structure is the most important in determining protein function. In particular, it is generally agreed that structural genomics, the discipline to study protein structure in relation with the entire genome, will be one of the main foci of research in life science after the complete identification of genome. Designing new drugs is an immediate example because information about protein structure is critically important in designing drugs. Docking is one of the most important technologies in the drug design process.

This chapter presented docking problem in the framework of combinatorial optimization with a slight level of mathematical and computational rigor. Because docking problem is a mathematical problem and is highly complicated, and highly interdisciplinary, the effort given in this chapter will be worth if it has succeeded to develop a common language that future scientists and engineers from different disciplines can rely on. It is expected that the optimization theory along with the powerful geometric constructs, the Voronoi diagram, the quasi-triangulation, and the beta-complex and beta-shape, will be the basis of future advancement of the structural molecular biology.

Acknowledgements This research was supported by the National Research Lab grant, the National Research Foundation, Korea (No. R0A-2007-000-20048-0).

Cross-References

- ▶ [Advanced Techniques for Dynamic Programming](#)
 - ▶ [Combinatorial Optimization Algorithms](#)
 - ▶ [Graph Searching and Related Problems](#)
 - ▶ [Quadratic Assignment Problems](#)
 - ▶ [Tabu Search](#)
-

Recommended Reading

1. R. Abagyan, M. Totrov, Biased probability monte carlo conformational searches and electrostatic calculations for peptides and proteins. *J. Mol. Biol.* **235**(3), 983–1002 (1994)
2. R. Abagyan, M. Totrov, High-throughput docking for lead generation. *Curr. Opin. Chem. Biol.* **5**(4), 375–382 (2001)
3. R. Abagyan, M. Totrov, D. Kuznetsov, Icm: a new method for protein modeling and design: applications to docking and structure prediction from the distorted native c. *J. Comput. Chem.* **15**(3), 488–506 (1994)
4. F.H. Allen, The Cambridge structural database: a quarter of a million crystal structures and rising. *Acta Crystallogr.* **B58**(1), 380–388 (2002)
5. E. Althaus, O. Kohlbacher, H.-P. Lenhof, P. Müller, A combinatorial approach to protein docking with flexible side chains. *J. Comput. Biol.* **9**(4), 597–612 (2002)
6. S. Atwell, J.M. Adams, J. Badger, M.D. Buchanan, I.K. Feil, K.J. Froning, X. Gao, J. Hendle, K. Keegan, B.C. Leon, H.J. Müller-Dieckmann, V.L. Nienaber, B.W. Noland, K. Post, K.R.

- Rajashankar, A. Ramos, M. Russell, S.K. Burley, S.G. Buchanan, A novel mode of gleevec binding is revealed by the structure of spleen tyrosine kinase. *J. Biol. Chem.* **279**(53), 55827–55832 (2004)
7. F. Aurenhammer, Power diagrams: properties, algorithms and applications. *SIAM J. Comput.* **16**, 78–96 (1987)
8. F. Aurenhammer, Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surv.* **23**(3), 345–405 (1991)
9. Available Chemicals Directory (ACD), <http://www.sympyx.com/products/databases/sourcing/acd/index.jsp>
10. J. Bajorath, Selected concepts and investigations in compound classification, molecular descriptor analysis, and virtual screening. *J. Chem. Inf. Comput. Sci.* **41**(2), 233–245 (2001)
11. J. Bajorath, Integration of virtual and high-throughput screening. *Nat. Rev. Drug Dis.* **1**, 882–894 (2002)
12. C.A. Baxter, C.W. Murray, D.E. Clark, D.R. Westhead, M.D. Eldridge, Flexible docking using tabu search and an empirical estimate of binding affinity. *Proteins* **33**, 367–382 (1998)
13. H.-J. Böhm, LUDI: rule-based automatic design of new substituents for enzyme inhibitor leads. *J. Comput. Aided Mol. Des.* **6**(6), 593–606 (1992)
14. H.-J. Böhm, The development of a simple empirical scoring function to estimate the binding constant for a protein-ligand complex of known three-dimensional structure. *J. Comput. Aided Mol. Des.* **8**(3), 243–256 (1994)
15. H.-J. Böhm, Prediction of binding constants of protein ligands: a fast method for the prioritization of hits obtained from de novo design or 3D database search programs. *J. Comput. Aided Mol. Des.* **12**(4), 309–323 (1998)
16. J.-D. Boissonnat, M. Yvinec, *Algorithmic Geometry* (Cambridge University Press, Cambridge, 1998)
17. A. Bondi, van der Waals volumes and radii. *J. Phys. Chem.* **68**, 441–451 (1964)
18. G.P. Brady Jr., P.F.W. Stouten, Fast prediction and visualization of protein binding pockets with PASS. *J. Comput. Aided Mol. Des.* **14**, 383–401 (2000)
19. N. Brooijmans, I.D. Kuntz, Molecular recognition and docking algorithms. *Ann. Rev. Biophys. Biomol. Struct.* **32**, 335–373 (2003)
20. B.R. Brooks, R.E. Brucolieri, B.D. Olafson, D.J. States, S. Swaminathan, M. Karplus, CHARMM: a program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem.* **4**(2), 187–217 (1983)
21. H.B. Broughton, A method for including protein flexibility in protein-ligand docking: Improving tools for database mining and virtual screening. *J. Mol. Graph. Model.* **18**(3), 247–257 (2000)
22. J.D. Bryngelson, J.N. Onuchic, N.D. Socci, P.G. Wolynes, Funnels, pathways and the energy landscape of protein folding: a synthesis. *Proteins* **21**(3), 167–271 (1995)
23. R.A. Buckingham, The classical equation of state of gaseous helium, neon and argon. *Proc. R. Soc. Lond.* **168**(933), 264–283 (1938)
24. J.J. Burbaum, N.H. Sigal, New technologies for high-throughput screening. *Curr. Opin. Chem. Biol.* **1**(1), 72–78 (1997)
25. B.D. Bursulaya, M. Totrov, R. Abagyan, C.L. Brooks III, Comparative study of several algorithms for flexible ligand docking. *J. Comput. Aided Mol. Des.* **17**, 755–763 (2003)
26. A. Caflisch, P. Niederer, M. Anliker, Monte Carlo docking of oligopeptides to proteins. *Proteins* **13**(3), 223–230 (1992)
27. Cambridge Structural Database (CSD), <http://www.ccdc.cam.ac.uk/products/csd/>
28. H.A. Carlson, J.A. McCammon, Accommodating protein flexibility in computational drug design. *Mol. Pharmacol.* **57**(2), 213–218 (2000)
29. CGAL User and Reference Manual: All Parts, Release 3.2.1, July 2006
30. P.S. Charifson, J.J. Corkery, M.A. Murcko, W.P. Walters, Consensus scoring: a method for obtaining improved hit rates from docking databases of three-dimensional structures into proteins. *J. Med. Chem.* **42**(25), 5100–5109 (1999)

31. B. Chazelle, C. Kingsford, M. Singh, A semidefinite programming approach to side chain positioning with new rounding strategies. *INFORMS J. Comput.* **16**(4), 380–392 (2004)
32. R. Chen, J. Mintseris, J. Janin, Z. Weng. A protein-protein docking benchmark. *Proteins* **52**, 88–91 (2003)
33. J. Cherfils, J. Janin, Protein docking algorithms: Simulating molecular recognition. *Curr. Opin. Struct. Biol.* **3**(2), 265–269 (1993)
34. J. Cherfils, S. Duquerroy, J. Janin, Protein-protein recognition analyzed by docking simulation. *Proteins* **11**(4), 271–280 (1991)
35. Y. Cho, D. Kim, H.C. Lee, J.Y. Park, D.-S. Kim, Reduction of the search space in the edge-tracing algorithm for the voronoi diagram of 3d balls, in *Proceeding of the International Conference on Computational Science and Applications (ICCSA 2006)*. Volume 3980 of Lecture Notes in Computer Science (Springer, Berlin/New York, 2006), pp. 111–120
36. V. Choi, Yucca: an efficient algorithm for small-molecule docking. *Chem. Biodivers.* **2**, 1517–1524 (2005)
37. R.D. Clark, A. Strizhev, J.M. Leonard, J.F. Blake, J.B. Matthew, Consensus scoring for ligand/protein interactions. *J. Mol. Graph. Model.* **20**(4), 281–295 (2002)
38. H. Claussen, C. Buning, M. Rarey, T. Lengauer, FLEXE: Efficient molecular docking considering protein structure variations. *J. Mol. Biol.* **308**, 377–395 (2001)
39. J.C. Cole, C.W. Murray, J.W.M. Nissink, R.D. Taylor, R. Taylor, Comparing protein-ligand docking programs is difficult. *Proteins* **60**, 325–332 (2005)
40. M.L. Connolly, Measurement of protein surface shape by solid angles. *J. Mol. Graph.* **4**(1), 3–6 (1986)
41. M.L. Connolly, Shape complementarity at the hemoglobin $\alpha_1\beta_1$ subunit interface. *Biopolymers* **25**(7), 1229–1247 (1986)
42. C.R. Corbeil, P. Englebienne, N. Moitessier, Docking ligands into flexible and solvated macromolecules. 1. development and validation of FITTED 1.0. *J. Chem. Inf. Model.* **47**, 435–449 (2007)
43. W.D. Cornell, P. Cieplak, C.I. Bayly, I.R. Gould, K.M. Merz Jr., D.M. Ferguson, D.C. Spellmeyer, T. Fox, J.W. Caldwell, P.A. Kollman, A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.* **117**, 5179–5197 (1995)
44. F. Crick, The packing of α -Helices: Simple coiled-coils. *Acta Crystallogr.* **6**, 689–697 (1953)
45. R.L. DesJarlais, R.P. Sheridan, J.S. Dixon, I.D. Kuntz, R. Venkataraghavan, Docking flexible ligands to macromolecular receptors by molecular shape. *J. Med. Chem.* **29**(11), 2149–2153 (1986)
46. R.L. DesJarlais, R.P. Sheridan, G.L. Seibel, J.S. Dixon, I.D. Kuntz, R. Venkataraghavan, Using shape complementarity as an initial screen in designing ligands for a receptor binding site of known three-dimensional str. *J. Med. Chem.* **31**(4), 722–729 (1988)
47. J. Desmet, M.D. Maeyer, B. Hazes, I. Lasters, The dead-end elimination theorem and its use in protein side-chain positioning. *Nature* **356**, 539–542 (1992)
48. J. Desmet, J. Spijriet, I. Lasters, Fast and accurate side-chain topology and energy refinement (FASTER) as a newmethod for protein structure optimization. *Proteins* **48**(1), 31–43 (2002)
49. R.S. DeWitte, E.I. Shakhnovich, SMoG: de Novo design method based on simple, fast, and accurate free energy estimates. 1. methodology and supporting evidence. *J. Am. Chem. Soc.* **118**(47), 11733–11744 (1996)
50. R.S. DeWitte, A.V. Ishchenko, E.I. Shakhnovich, SMoG: de Novo design method based on simple, fast, and accurate free energy estimates. 2. case studies in molecular design. *J. Am. Chem. Soc.* **119**(20), 4608–4617 (1979)
51. R. Diamond, On the comparison of conformations using linear and quadratic transformations. *Acta Crystallogr. Sect. A* **A32**(1), 1–10 (1976)
52. D.J. Diller, C.L.M.J. Verlinde, A critical evaluation of several global optimization algorithms for the purpose of molecular docking. *J. Comput. Chem.* **20**(16), 1740–1751 (1999)
53. J.P.K. Doye, M.A. Miller, D.J. Wales, The double-funnel energy landscape of the 38-atom Lennard-Jones cluster. *J. Chem. Phys.* **110**(14), 6896–6906 (1999)

54. H. Edelsbrunner, Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1992.
55. H. Edelsbrunner, E.P. Mücke, Three-dimensional alpha shapes. *ACM Trans Graph.* **13**(1), 43–72 (1994)
56. H. Edelsbrunner, D.G. Kirkpatrick, R. Seidel, On the shape of a set of points in the plane. *IEEE Trans. Inf. Theory* **29**(4), 551–559 (1983)
57. H. Edelsbrunner, M. Facello, J. Liang, On the definition and the construction of pockets in macromolecules. *Discrete Appl. Math.* **88**, 83–102 (1998)
58. J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM* **19**(2), 248–264 (1972)
59. D. Eggert, A. Lorusso, R. Fisher, Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Mach. Vis. Appl.* **9**, 272–290 (1997)
60. M.D. Eldridge, C.W. Murray, T.R. Auton, G.V. Paolini, R.P. Mee, Empirical scoring functions: I. the development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes. *J. Comput. Aided Mol. Des.* **11**, 425–445 (1997)
61. T.J. Ewing, I.D. Kuntz, Critical evaluation of search algorithms for automated molecular docking and database screening. *J. Comput. Chem.* **18**, 1175–1189 (1997)
62. T.J.A. Ewing, S. Makino, A.G. Skillman, I.D. Kuntz, DOCK 4.0: search strategies for automated molecular docking of flexible molecule databases. *J. Comput. Aided Mol. Des.* **15**, 411–428 (2001)
63. L.F.T. Eyck, Crystallographic fast fourier transforms. *Acta Crystallogr. Sect. A* **29**(2), 183–191 (1973)
64. E. Fischer, Synthesen in der zuckergruppe. Berichte der deutschen chemischen Gesellschaft **23**(2), 2114–2141 (1890)
65. E. Fischer, in *Syntheses in the Purine and Sugar Group*. Nobel Lectures in Chemistry 1901–1921 (Elsevier, Amsterdam, 1966)
66. D. Fischer, S.L. Lin, H.L. Wolfson, R. Nussinov, A geometry-based suite of molecular docking processes. *J. Mol. Biol.* **248**, 459–477 (1995)
67. R. Fletcher, C.M. Reeves, Function minimization by conjugate gradients. *Comput. J.* **7**(2), 149–154 (1964)
68. H. Frauenfelder, S.G. Sligar, P.G. Wolynes The energy landscapes and motions of proteins. *Science* **254**(5038), 1598–1603 (1991)
69. H. Fung, S. Rao, C. Floudas, O. Prokopyev, P. Pardalos, F. Rendl, Computational comparison studies of quadratic assignment like formulations for the In silico sequence selection problem in De Novo protein design. *J. Comb. Optim.* **10**(1), 41–60 (2005)
70. D.K. Gehlhaar, G.M. Verkhivker, P.A. Rejto, C.J. Sherman, D.B. Fogel, L.J. Fogel, S.T. Freer, Molecular recognition of the inhibitor AG-1343 by HIV-1 protease: conformationally flexible docking by evolutionary programm. *Chem. Biol.* **2**(5), 317–324 (1995)
71. K.D. Gibson, H.A. Scheraga, Minimization of polypeptide energy. i. preliminary structures of bovine pancreatic ribonuclease S-peptide. *Proc. Natl. Acad. Sci. U. S. A.* **58**(2), 420–427 (1967)
72. H. Gohlke, G. Klebe, Statistical potentials and scoring functions applied to protein-ligand binding. *Curr. Opin. Struct. Biol.* **11**(2), 231–235 (2001)
73. H. Gohlke, M. Hendlich, G. Klebe, Knowledge-based scoring function to predict protein-ligand interactions. *J. Mol. Biol.* **295**(2), 337–356 (2000)
74. P.J. Goodford, A computational procedure for determining energetically favorable binding sites on biologically important macromolecules. *J. Med. Chem.* **28**, 849–857 (1985)
75. D.S. Goodsell, A.J. Olson, Automated docking of substrates to proteins by simulated annealing. *Proteins* **8**(3), 195–202 (1990)
76. D.S. Goodsell, G.M. Morris, A.J. Olson, Automated docking of flexible ligands: applications of autodock. *J. Mol. Recognit.* **9**, 1–5 (1996)
77. D. Gordon, G.K. Hom, S.L. Mayo, N.A. Pierce, Exact rotamer optimization for protein design. *J. Comput. Chem.* **24**(2), 232–243 (2002)

78. J. Greer, B.L. Bush, Macromolecular shape and surface maps by solvent exclusion. *Proc. Natl. Acad. Sci. U. S. A.* **75**(1), 303–307 (1978)
79. K. Gulukota, S. Vajda, C. Delisi, Peptide docking using dynamic programming. *J. Comput. Chem.* **17**(4), 418–428 (1996)
80. C. Hardin, T.V. Pogorelov, Z. Luthey-Schulten, *Ab Initio* protein structure prediction. *Curr. Opin. Struct. Biol.* **12**, 176–181 (2002)
81. M. Helmer-Citterich, A. Tramontano, Puzzle: a new method for automated protein docking based on surface shape complementarity. *J. Mol. Biol.* **235**(3), 1021–1031 (1994)
82. W.A. Hendrickson, Transformations to optimize the superposition of similar structures. *Acta Crystallogr. Sect. A* **A35**(1), 158–163 (1979)
83. C.M. Ho, G.R. Marshall, Cavity search: an algorithm for the isolation and display of cavity-like binding regions. *J. Comput. Aided Mol. Des.* **4**, 337–354 (1990)
84. R. Huey, G.M. Morris, A.J. Olson, D.S. Goodsell, A semiempirical free energy force field with charge-based desolvation. *J. Comput. Chem.* **28**(6), 1145–1152 (2007)
85. A.V. Ishchenko, E.I. Shakhnovich, SMall molecule growth 2001 (SMoG2001): an improved knowledge-based scoring function for protein-ligand interactions. *J. Med. Chem.* **45**(13), 2770–2780 (2002)
86. D.J. Jacobs, A.J. Rader, L.A. Kuhn, M.F. Thorpe, Protein flexibility predictions using graph theory. *Proteins* **44**, 150–165 (2001)
87. F. Jiang, S.H. Kim, Soft docking: Matching of molecular surface cubes. *J. Mol. Biol.* **219**(2), 79–102 (1991)
88. J.E. Jones, On the determination of molecular fields. ii. from the equation of state of a gas. *Proc. R. Soc. Lond.* **106**(738), 463–477 (1924)
89. G. Jones, P. Willett, R.C. Glen, Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *J. Mol. Biol.* **245**, 43–53 (1995)
90. G. Jones, P. Willett, R.C. Glen, A.R. Leach, R. Taylor, Development and validation of a genetic algorithm for flexible docking. *J. Mol. Biol.* **267**, 727–748 (1997)
91. R.L. Dunbrack Jr., Rotamer libraries in the 21st century. *Curr. Opin. Struct. Biol.* **12**(4), 431–440 (2002)
92. W. Kabsch, A solution for the best rotation to relate two sets of vectors. *Acta Crystallogr. Sect. A* **A32**(5), 922–923 (1976)
93. E. Katchalski-Katzir, I. Shariv, M. Eisenstein, A.A. Friesem, C. Aflalo, I.A. Vakser, Molecular surface recognition: Determination of geometric fit between proteins and their ligands by correlation techniques. *Proc. Natl. Acad. Sci. U. S. A.* **89**, 2195–2199 (1992)
94. S.K. Kearsley, Structural comparisons using restrained inhomogeneous transformations. *Acta Crystallogr. Sect. A* **A45**(9), 623–635 (1989)
95. D. Kim, D.-S. Kim, Region-expansion for the Voronoi diagram of 3D spheres. *Comput. Aided Des.* **38**(5), 417–430 (2006)
96. D.-S. Kim, D. Kim, K. Sugihara, Voronoi diagram of a circle set from Voronoi diagram of a point set: I. topology. *Comput. Aided Geom. Des.* **18**, 541–562 (2001)
97. D.-S. Kim, D. Kim, K. Sugihara, Voronoi diagram of a circle set from Voronoi diagram of a point set: II. geometry. *Comput. Aided Geom. Des.* **18**, 563–585 (2001)
98. D.-S. Kim, Y. Cho, D. Kim, Euclidean Voronoi diagram of 3D balls and its computation via tracing edges. *Comput. Aided Des.* **37**(13), 1412–1424 (2005)
99. D.-S. Kim, C.-H. Cho, D. Kim, Y. Cho, Recognition of docking sites on a protein using β -shape based on Voronoi diagram of atoms. *Comput. Aided Des.* **38**(5), 431–443 (2006)
100. D.-S. Kim, D. Kim, Y. Cho, K. Sugihara, Quasi-triangulation and interworld data structure in three dimensions. *Comput. Aided Des.* **38**(7), 808–819 (2006)
101. D.-S. Kim, J. Seo, D. Kim, J. Ryu, C.-H. Cho, Three-dimensional beta shapes. *Comput. Aided Des.* **38**(11), 1179–1191 (2006)
102. D. Kim, C. Lee, Y. Cho, D.-S. Kim, Manifoldization of β -shapes by topology operators, in *Proceedings of the Geometric Modeling and Processing (GMP 2008)*. Volume 4975 of Lecture Notes in Computer Science (Springer, Berlin/Heidelberg, 2008), pp. 505–511

103. D.-S. Kim, J. Seo, D. Kim, Y. Cho, J. Ryu, The β -shape and β -complex for analysis of molecular structures, in *Generalized Voronoi Diagrams: A Geometry-based Approach to Computational Intelligence* ed. by M. Gavrilova. The Book Series on Computational Intelligence, Springer-Verlag Berlin Heidelberg, ISBN 978-3-642-09883-3 (2008)
104. D.-S. Kim, Y. Cho, K. Sugihara, Quasi-worlds and quasi-operators on quasi-triangulations. *Comput. Aided Des.* **42**(10), 874–888 (2010)
105. D.-S. Kim, Y. Cho, K. Sugihara, J. Ryu, D. Kim, Three-dimensional beta-shapes and beta-complexes via quasi-triangulation. *Comput. Aided Des.* **42**(10), 911–929 (2010)
106. D.-S. Kim, C.-I. Won, J. Bhak, A proposal for the revision of molecular boundary typology. *J. Biomol. Struct. Dyn.* **28**(2), 277–287 (2010)
107. C.-M. Kim, C.-I. Won, J.-K. Kim, J. Ryu, J. Bhak, D.-S. Kim, Protein-ligand docking based on Beta-shape. *Trans. Comput. Sci. IX, LNCS* **6290**, 123–138 (2010)
108. D.-S. Kim, C.-M. Kim, C.-I. Won, J.-K. Kim, J. Ryu, Y. Cho, C. Lee, J. Bhak, BetaDock: Shape-priority docking method based on Beta-complex. *J. Biomol. Struct. Dyn.* **29**(1), 219–242 (2011)
109. C.L. Kingsford, B. Chazelle, M. Singh, Solving and analyzing side-chain positioning problems using linear and integer programming. *Struct. Bioinf.* **21**(7), 1028–1036 (2005)
110. D.B. Kitchen, H. Decornez, J.R. Furr, J. Bajorath, Docking and scoring in virtual screening for drug discovery: Methods and applications. *Nat. Rev. Drug Discov.* **3**, 935–949 (2004)
111. G. Klebe, T. Mietzner, A fast and efficient method to generate biologically relevant conformations. *J. Comput. Aided Mol. Des.* **8**(5), 583–606 (1994)
112. G.J. Kleywegt, T.A. Jones, Detection, delineation, measurement and display of cavities in macromolecular structures. *Acta Crystallogr. Sect. D* **50**, 178–185 (1994)
113. M. Klumpp, A. Boettcher, D. Becker, G. Meder, J. Blank, L. Leder, M. Forstner, J. Ottl, L.M. Mayr, Readout technologies for highly miniaturized kinase assays applicable to high-throughput screening in a 1536-well format. *J. Biomol. Screen.* **11**(6), 617–633 (2006)
114. P. Kollman, Free energy calculations: Applications to chemical and biochemical phenomena. *Chem. Rev.* **93**, 2395–2417 (1993)
115. D.E. Koshland Jr., Application of a theory of enzyme specificity to protein synthesis. *Proc. Natl. Acad. Sci.* **44**, 98–104 (1958)
116. D.E. Koshland Jr., Correlation of structure and function in enzyme action. *Science* **142**(3599), 1533–1541 (1963)
117. D.E. Koshland Jr., The key-lock theory and the induced fit theory. *Angew. Chem. Int. Ed.* **33**, 2375–2378 (1994)
118. B. Kramer, M. Rarey, T. Lengauer, Evaluation of the flexx incremental construction algorithm for protein-ligand docking. *Proteins* **37**, 228–241 (1999)
119. F. Kuhl, G.M. Crippen, D.K. Friesen, A combinatorial algorithm for calculating ligand binding. *J. Comput. Chem.* **5**(1), 24–34 (1984)
120. H.W. Kuhn, The hungarian method for the assignment problem. *Naval Res. Logist. Quart.* **2**(1–2), 83–97 (1955)
121. I.D. Kuntz, F.M. Blaney, S.J. Oatley, A geometric approach to macromolecule-ligand interactions. *J. Mol. Biol.* **161**, 269–288 (1982)
122. R.A. Laskowski, N.M. Luscombe, M.B. Swindells, J.M. Thornton, Protein clefts in molecular recognition and function. *Protein Sci.* **5**, 2438–2452 (1996)
123. I. Lasters, M.D. Maeyer, J. Desmet, Enhanced dead-end elimination in the search for the global minimum energy conformation of a collection of protein side chains. *Protein Eng.* **8**(8), 815–822 (1995)
124. R.H. Lathrop, The protein threading problem with sequence amino acid interaction preferences is np-complete. *Protein Eng.* **7**(9), 1059–1068 (1994)
125. A.R. Leach, Ligand docking to proteins with discrete side-chain flexibility. *J. Mol. Biol.* **235**(1), 345–356 (1994)
126. A.R. Leach, I.D. Kuntz, Conformational analysis of flexible ligands in macromolecular receptor sites. *J. Comput. Chem.* **13**(6), 730–748 (1992)

127. J. Lee, H.A. Scheraga, S. Rackovsky, New optimization method for conformational energy calculations on polypeptides: Conformational space annealing. *J. Comput. Chem.* **18**(9), 1222–1232 (1997)
128. K. Lee, J. Sim, J. Lee, Study of protein-protein interaction using conformational space annealing. *Proteins* **60**, 257–262 (2005)
129. T. Lengauer, M. Rarey, Computational methods for biomolecular docking. *Curr. Opin. Struct. Biol.* **6**, 402–406 (1996)
130. J.E. Lennard-Jones, Cohesion. *Proc. Phys. Soc.* **43**, 461–482 (1931)
131. P.E. Leopold, M. Montal, J.N. Onuchic, Protein folding funnels: a kinetic approach to the sequence-structure relationship. *Proc. Natl. Acad. Sci. U. S. A.* **89**, 8721–8725 (1992)
132. C. Levinthal, Are there pathways for protein folding? *J. Chem. Phys.* **65**(1), 44–45 (1968)
133. C. Levinthal, S.J. Wodak, P. Kahn, A.K. Davivianian, Hemoglobin interaction in sickle cell fibers i: Theoretical approaches to the molecular contacts. *Proc. Natl. Acad. Sci. U. S. A.* **72**(4), 1330–1334 (1975)
134. D. Levitt, L. Banaszak, POCKET: a computer graphics method for identifying and displaying protein cavities and their surrounding amino acids. *J. Mol. Graph.* **10**, 229–234 (1992)
135. Z. Li, H.A. Scheraga, Monte Carlo-minimization approach to the multiple-minima problem in protein folding. *Proc. Natl. Acad. Sci.* **84**(19), 6611–6615 (1987)
136. J. Liang, H. Edelsbrunner, C. Woodward, Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. *Protein Sci.* **7**(9), 1884–1897 (1998)
137. A. Liwo, M. Pincus, R. Wawak, S. Rackovsky, H. Scheraga, Prediction of protein conformation on the basis of a search for compact structures: Test on avian pancreatic polypeptide. *Protein Sci.* **2**, 1715–1731 (1993)
138. S.C. Lovell, J.M. Word, J.S. Richardson, D.C. Richardson, The penultimate rotamer library. *Proteins* **40**(3), 389–408 (2000)
139. MACCS-II Drug Data Report (MDDR), <http://www.sympyx.com/products/databases/bioactivity/mddr/index.jsp>.
140. J. MacKerell, A.D. Empirical force fields for biological macromolecules: overview and issues. *J. Comput. Chem.* **25**, 1584–1604 (2004)
141. T. Masada, H. Imai, K. Imai, Enumeration of regular triangulations, in *Proceedings of the 12th Annual Symposium on Computational Geometry (SoCG'96)* (ACM, New York, 1996), pp. 224–233
142. L.M. Mayr, P. Fuerst, The future of high-throughput screening. *J. Biomol. Screen.* **13**(6), 443–448 (2008)
143. I.K. McDonald, J.M. Thornton, Satisfying hydrogen bonding potential in proteins. *J Mol. Biol.* **238**, 777–793 (1994)
144. K. Mehlhorn, S. Näher, *LEDA: A Platform for Combinatorial and Geometric Computing* (Cambridge University Press, New York, 1999)
145. R. Mendez, R. Leplae, L.D. Maria, S.J. Wodak, Assessment of blind predictions of protein-protein interactions: current status of docking methods. *Proteins* **52**, 51–67 (2003)
146. E.C. Meng, D.A. Gschwend, J.M. Blaney, I.D. Kuntz, Orientational sampling and rigid-body minimization in molecular docking. *Proteins* **17**(3), 266–278 (1993)
147. N. Metropolis, The beginning of the MONTE CARLO METHOD. *Los Alamos Sci. (Special Issue)*, 125–130 (1987)
148. N. Metropolis, S. Ulam, The monte carlo method. *J. Am. Stat. Assoc.* **44**(247), 335–341 (1949)
149. M. Meyer, P. Wilson, D. Schomburg, Hydrogen bonding and molecular surface shape complementarity as a basis for protein docking. *J. Mol. Biol.* **264**, 199–210 (1996)
150. D.W. Miller, K.A. Dill, Ligand binding to proteins: the binding landscape model. *Protein Sci.* **6**, 2166–2179 (1997)
151. V. Mohan, A.C. Gibbs, M.D. Cummings, E.P. Jaeger, R.L. DesJarlais, Docking: successes and challenges. *Curr. Pharm. Des.* **11**, 323–333 (2005)

152. N. Moitessier, P. Englebienne, D. Lee, J. Lawandi, C. Corbeil, Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go. *Br. J. Pharm.* **153**, S7–S26 (2008)
153. J.B. Moon, W.J. Howe, Computer design of bioactive molecules: a method for receptor-based de novo ligand design. *Proteins* **11**(4), 314–328 (1991)
154. I.S. Moreira, P.A. Fernandes, M.J. Ramos, Protein-protein docking dealing with the unknown. *J. Comput. Chem.* **31**(2), 317–342 (2010)
155. G.M. Morris, D.S. Goodsell, R. Huey, A.J. Olson, Distributed automated docking of flexible ligands to proteins: parallel applications of AutoDock 2.4. *J. Comput. Aided Mol. Des.* **10**(4), 293–304 (1996)
156. G.M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, A.J. Olson, Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J. Comput. Chem.* **19**(14), 1639–1622 (1998)
157. G.M. Morris, D.S. Goodsell, R. Huey, W.E. Hart, S. Halliday, R. Belew, A.J. Olson, User's guide AutoDock version 3.0.5. Technical report, The Scripps Research Institute, Molecular Graphics Laboratory, Department of Molecular Biology, USA, 11, 2001.
158. D.T. Moustakas, P.T. Lang, S. Pegg, E. Pettersen, I.D. Kuntz, N. Brooijmans, R.C. Rizzo, Development and validation of a modular, extensible docking program: DOCK 5. *J. Comput. Aided Mol. Des.* **20**, 601–619 (2006)
159. I. Muegge, Y.C. Martin, A general and fast scoring function for protein-ligand interactions: a simplified potential approach. *J. Med. Chem.* **42**(5), 791–804 (1999)
160. B. Nagar, c-Abl tyrosine kinase and inhibition by the cancer drug imatinib (Gleevec/STI-571)^{1–4}. *J. Nutr.* **137**, 1518S–1523S (2007)
161. R. Norel, D. Fischer, H.J. Wolfson, R. Nussinov, Molecular surface recognition by a computer vision-based technique. *Protein Eng.* **7**(1), 39–46 (1994)
162. R. Norel, S.L. Lin, H.J. Wolfson, R. Nussinov, Shape complementarity at protein-protein interfaces. *Biopolymers* **34**(7), 933–940 (1994)
163. R. Norel, S.L. Lin, H.J. Wolfson, R. Nussinov, Molecular surface complementarity at protein-protein interfaces: the critical role played by surface normals at well placed, spa. *J. Mol. Biol.* **252**, 263–273 (1995)
164. R. Norel, D. Petrey, H.J. Wolfson, R. Nussinov, Examination of shape complementarity in docking of unbound proteins. *Proteins* **36**, 307–317 (1999)
165. D. Oberlin Jr., H.A. Scheraga, B-spline method for energy minimization in grid-based molecular mechanics calculations. *J. Comput. Chem.* **19**(1), 71–85 (1998)
166. A. Okabe, B. Boots, K. Sugihara, S.N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, 2nd edn. (Wiley, Chichester, 1999)
167. D.J. Osguthorpe, *Ab Initio* protein folding. *Curr. Opin. Struct. Biol.* **10**, 146–152 (2000)
168. C. Oshiro, I. Kuntz, J.S. Dixon, Flexible ligand docking using a genetic algorithm. *J. Comput. Aided Mol. Des.* **9**(2), 113–130 (1995)
169. J. Pei, Q. Wang, Z. Liu, Q. Li, K. Yang, L. Lai, PSI-DOCK: towards highly efficient and accurate flexible ligand docking. *Proteins* **62**, 934–946 (2006)
170. E. Perola, W.P. Walters, P.S. Charifson, A detailed comparison of current docking and scoring methods on systems of pharmaceutical relevance. *Proteins* **56**, 235–249 (2004)
171. K.P. Peters, J. Fauck, C. Frömmel, The automatic search for ligand binding sites in protein of known three dimensional structure using only geometric criteria. *J. Mol. Biol.* **256**, 201–213 (1996)
172. N.A. Pierce, E. Winfree, Protein design is NP-hard. *Protein Eng.* **15**(10), 779–782 (2002)
173. K.E.B. Platzer, F.A. Momany, H.A. Scheraga, Conformational energy calculations of enzyme-substrate interactions. *Int. J. Pept. Protein Res.* **4**(3), 201–219 (1972)
174. J.W. Ponder, D.A. Case, Force fields for protein simulations. *Adv. Protein Chem.* **66**, 27–86 (2003)
175. J.W. Ponder, F.M. Richards, Tertiary templates for proteins: use of packing criteria in the enumeration of allowed sequences for different structural classes. *J. Mol. Biol.* **193**(4), 775–791 (1987)

176. M. Rarey, B. Kramer, T. Lengauer, Time-efficient docking of flexible ligands into active sites of proteins. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* **3**, 300–308 (1995)
177. M. Rarey, B. Kramer, T. Lengauer, G. Klebe, A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.* **261**, 470–489 (1996)
178. M. Rarey, S. Wefing, T. Lengauer, Placement of medium-sized molecular fragments into active sites of proteins. *J. Comput. Aided Mol. Des.* **10**(1), 41–54 (1996)
179. RCSB Protein Data Bank (2009), <http://www.rcsb.org/pdb/>
180. D. Rognan, S.L. Lauemøller, A. Holm, S. Buus, V. Tschinker, Predicting binding affinities of protein ligands from three-dimensional models: application to peptide binding to class i major histocompatibility proteins. *J. Med. Chem.* **42**(22), 4650–4658 (1999)
181. R.L. Dunbrack Jr., F.E. Cohen, Bayesian statistical analysis of protein side-chain rotamer preferences. *Protein Sci.* **6**(8), 1661–1681 (1997)
182. J. Ryu, R. Park, D.-S. Kim, Molecular surfaces on proteins via beta shapes. *Comput. Aided Des.* **39**(12), 1042–1057 (2007)
183. L. Schaffer, G.M. Verkhivker, Predicting structural effects in HIV-1 protease mutant complexes with flexible ligand docking and protein side-chain optimization. *Proteins* **33**(2), 295–310 (1998)
184. M. Schapira, B.M. Raaka, H.H. Samuels, R. Abagyan, Rational discovery of novel nuclear hormone receptor antagonists. *Proc. Natl. Acad. Sci. U. S. A.* **97**(3), 1008–1013 (2000)
185. D. Schneidman-Duhovny, Y. Inbar, R. Nussinov, H.J. Wolfson, Geometry-based flexible and symmetric protein docking. *Proteins* **60**, 224–231 (2005)
186. T. Schulz-Gasch, M. Stahl, Scoring functions for protein-ligand interactions: a critical perspective. *Drug Discov. Today* **1**(3), 231–239 (2004)
187. B.K. Shoichet, I.D. Kuntzt, Protein docking and complementarity. *J. Mol. Biol.* **221**, 327–346 (1991)
188. B.K. Shoichet, I.D. Kuntzt, D.L. Bodian, Molecular docking using shape descriptors. *J. Comput. Chem.* **13**(3), 380–397 (1992)
189. G.R. Smith, M.J. Sternberg, Prediction of protein-protein interactions by docking methods. *Curr. Opin. Struct. Biol.* **12**, 28–35 (2002)
190. S.F. Sousa, P.A. Fernandes, M.J. Ramos, Protein-ligand docking: Current status and future challenges. *Proteins* **65**, 15–26 (2006)
191. B.L. Stoddard, D.E. Koshland Jr., Prediction of the structure of a receptor-protein complex using a binary docking method. *Nature* **358**(6389), 774–776 (1992)
192. Sugihara Homepage (2009), <http://home.mims.meiji.ac.jp/~sugihara/>.
193. P. Tao, L. Lai, Protein ligand docking based on empirical method for binding affinity estimation. *J. Comput. Aided Mol. Des.* **15**(5), 429–446 (2001)
194. W.R. Taylor, P. Jewsbury, J. Essex, A review of protein-small molecule docking methods. *J. Comput. Aided Mol. Des.* **16**(3), 151–166 (2002)
195. D. Tobi, I. Bahar, Optimal design of protein docking potentials: efficiency and limitations. *Proteins* **62**, 970–981 (2006)
196. A. Tsvchigrechko, I.A. Vakser, How common is the funnel-like energy landscape in protein-protein interactions? *Protein Sci.* **10**(8), 1572–1583 (2001)
197. J.-Y. Trosset, H.A. Scheraga, PRODOCK: software package for protein modeling and docking. *J. Comput. Chem.* **20**(4), 412–427 (1999)
198. J. Tsai, R. Taylor, C. Chothia, M. Gerstein, The packing density in proteins: Standard radii and volumes. *J. Mol. Biol.* **290**, 253–266 (1999)
199. UniProt Homepage (2011), <http://www.uniprot.org>
200. I.A. Vakser, O.G. Matar, C.F. Lam, A systematic study of low-resolution recognition in protein-protein complexes. *Proc. Natl. Acad. Sci. U. S. A.* **96**(15), 8477–8482 (1999)
201. H.F.G. Velec, H. Gohlke, G. Klebe, DrugScore^C SD knowledge-based scoring function derived from small molecule crystal data with superior recognition rate of near-native ligand poses and better affinity prediction. *J. Med. Chem.* **48**(20), 6296–6303 (2008)
202. C.A. Voigt, D.B. Gordon, S.L. Mayo, Trading accuracy for speed: A quantitative comparison of search algorithms in protein sequence design. *J. Mol. Biol.* **299**, 789–803 (2000)

203. R. Voorinthal, M.T. Kosters, G. Vegter, G. Vriend, W.G. Hol, A very fast program for visualizing protein surfaces, channels and cavities. *J. Mol. Graph.* **7**(4), 243–245 (1989)
204. Voronoi Diagram Research Center (2011), <http://voronoi.hanyang.ac.kr/>.
205. D.J. Wales, H.A. Scheraga, Global optimization of clusters, crystals, and biomolecules. *Science* **285**(5432), 1368–1372 (1999)
206. A. Wallqvist, D.G. Covell, Docking enzyme-inhibitor complexes using a preference-based free-energy surface. *Proteins* **25**, 403–419 (1996)
207. W.P. Walters, M.T. Stahl, M.A. Murcko, Virtual screening – an overview. *Drug Discov. Today* **3**(4), 160–178 (1998)
208. H. Wang, Grid-search molecular accessible surface algorithm for solving the protein docking problem. *J. Comput. Chem.* **12**(6), 746–750 (1991)
209. R. Wang, L. Liu, L. Lai, Y. Tang, SCORE: A new empirical method for estimating the binding affinity of a protein-ligand complex. *J. Mol. Model.* **4**(12), 379–394 (1998)
210. R. Wang, Y. Lu, S. Wang, Comparative evaluation of 11 scoring functions for molecular docking. *J. Med. Chem.* **46**(12), 2287–2303 (2003)
211. Z.R. Wasserman, C.N. Hodge, Fitting an inhibitor into the active site of thermolysin: a molecular dynamics case study. *Proteins* **24**(2), 227–237 (1996)
212. P.K. Weiner, P.A. Kollman, AMBER: Assisted model building with energy refinement. a general program for modeling molecules and their interactions. *J. Comput. Chem.* **2**(3), 287–303 (1981)
213. S.J. Weiner, P.A. Kollman, D.A. Case, U.C. Singh, C. Ghio, G. Alagona, S. Profeta Jr., P. Weiner, A new force field for molecular mechanical simulation of nucleic acids and proteins. *J. Am. Chem. Soc.* **106**, 764–784 (1984)
214. W. Welch, J. Ruppert, A.N. Jain, Hammerhead: fast, fully automated docking of flexible ligands to protein binding sites. *Chem. Biol.* **3**(6), 449–462 (1996)
215. D.R. Westhead, D.E. Clark, C.W. Murray, A comparison of heuristic search algorithms for molecular docking. *J. Comput. Aided Mol. Des.* **11**, 209–228 (1997)
216. S.J. Wodak, J. Janin, Computer analysis of protein-protein interaction. *J. Mol. Biol.* **124**, 323–342 (1978)
217. Y. Xia, E.S. Huang, M. Levitt, R. Samudrala, *Ab Initio* construction of protein tertiary structures using a hierarchical approach. *J. Mol. Biol.* **300**, 171–185 (2000)
218. D. Xu, C.-J. Tsai, R. Nussinov, Hydrogen bonds and salt bridges across protein-protein interfaces. *Protein Eng.* **10**(9), 999–1012 (1997)
219. Y. Zhu, Mixed-integer linear programming algorithm for a computational protein design problem. *Ind. Eng. Chem. Res.* **46**, 839–845 (2007)

Quadratic Assignment Problems

Rainer E. Burkard

Contents

1	Introduction	2743
2	Combinatorial Optimization Problems Formulated as QAPs	2745
2.1	The Traveling Salesman Problem	2745
2.2	The Linear Arrangement Problem	2745
2.3	Graph Partitioning and Maximum Clique Problem	2746
2.4	The Minimum-Weight Feedback Arc Set Problem	2746
2.5	Packing Problems in Graphs	2747
3	Formulations and Computational Complexity	2747
3.1	Quadratic Integer Programming Formulations	2748
3.2	Inner Product Formulation	2748
3.3	Trace Formulation	2750
3.4	Computational Complexity	2751
4	Linearizations	2751
4.1	Lawler's Linearization	2752
4.2	Kaufman and Broeckx Linearization	2752
4.3	Frieze and Yadegar Linearization	2753
4.4	Adams and Johnson Linearization	2754
4.5	Level-2 Reformulation–Linearization	2755
5	QAP Polytopes	2756
5.1	The QAP Polytope	2756
5.2	The Symmetric QAP Polytope	2758
6	Lower Bounds	2759
6.1	Admissible Transformations	2759
6.2	Gilmore–Lawler Type Lower Bounds	2761
6.3	Reduction Methods	2763
6.4	Bounds Based on Linear Programming Relaxations	2766
6.5	Lower Bounds Based on Eigenvalues	2767
6.6	Improving Bounds by Means of Decompositions	2773

R.E. Burkard

Institute of Optimization and Discrete Mathematics, Graz University of Technology, Graz, Austria
e-mail: burkard@tugraz.at

6.7	Bounds Based on Semidefinite Relaxations.....	2775
6.8	Bounds Based on Quadratic Programming and Conic Representations.....	2778
7	Exact Solution Methods.....	2779
7.1	Branch and Bound.....	2779
7.2	Benders' Decomposition Methods.....	2780
7.3	Branch-and-Cut Methods.....	2781
8	Heuristics.....	2782
8.1	Construction Methods.....	2783
8.2	Limited Enumeration Methods.....	2783
8.3	Local Search Methods.....	2783
8.4	Complexity Results on Local Search.....	2785
8.5	Greedy Randomized Adaptive Search Procedure.....	2786
8.6	Tabu Search.....	2787
8.7	Simulated Annealing.....	2788
8.8	Genetic Algorithms.....	2789
8.9	Ant Colony Optimization.....	2789
9	Available Computer Codes for the QAP.....	2791
10	Polynomially Solvable Cases.....	2791
11	Asymptotic Behavior.....	2795
12	Quadratic Bottleneck Assignment Problems.....	2800
13	Related Problems.....	2802
13.1	Cubic and Biquadratic Assignment Problems.....	2802
13.2	The Quadratic Semi-assignment Problem.....	2804
	Recommended Reading.....	2805

Abstract

This chapter introduces quadratic assignment problems (QAP) as models for finding an optimal assignment among two sets of interrelated objects. References to many applications of QAPs, like in location theory, are given. Moreover, several classical combinatorial optimization problems are identified as special cases of the QAP, like the traveling salesman problem, graph partitioning, max clique problem, minimum-weight feedback arc set problem, (FASP) and packing problems in graphs. These subproblems show that the QAP is \mathcal{NP} -hard.

Several different formulations of the QAP are presented in detail as being basic for different solution approaches. Special emphasis is laid to the different possibilities for describing QAPs as (mixed-)integer linear programs. Moreover, QAP polyhedra are discussed.

Crucial for exact solution approaches are lower bounding procedures. The paradigm of admissible transformations is introduced for describing several variants of Gilmore–Lawler type bounds. Moreover, bounds based on eigenvalues and bounds found by semidefinite programming are discussed in detail. Among exact solution methods, special emphasis is laid on branch-and-bound approaches. With respect to heuristics, not only simple construction heuristics and improvement heuristics are described, but also several metaheuristics like simulated annealing, tabu search, greedy randomized search, genetic algorithms, and ant systems are discussed. Links to available computer programs are delivered.

As QAPs are \mathcal{NP} -hard problems, there is a special interest in polynomially solvable special cases. A comprehensive survey of results in this direction is given. Moreover, QAPs show an interesting asymptotic behavior, as the ratio of best and worst solution value tends to one in probability. This phenomenon is thoroughly discussed and implications of this strange behavior are shown.

Finally, related problems are shortly treated. So, the quadratic bottleneck assignment problem is introduced and lower bounds, efficiently solvable special cases, as well as asymptotic results are reported. Moreover, cubic assignment problems, biquadratic assignment problems, and the quadratic semi-assignment problem are surveyed. An extensive bibliography provides the most important links to the literature in this field.

1 Introduction

The quadratic assignment problem (QAP) was introduced by Koopmans and Beckmann in 1957 as a mathematical model for the location of indivisible economical activities [146]. Consider the problem of allocating n facilities to n locations, with the cost being a function of the distance and flow between the facilities plus costs associated with placing a facility at a certain location. The objective is to assign each facility to a location such that the total cost is minimized. Specifically, let n be the number of facilities and locations and denote by N the set $N = \{1, 2, \dots, n\}$. We are given three $n \times n$ input matrices with real elements $F = (f_{ij})$, $D = (d_{kl})$, and $B = (b_{ik})$, where f_{ij} is the flow between the facility i and facility j , d_{kl} is the distance between the location k and location l , and b_{ik} is the cost of placing facility i at location k . The Koopmans–Beckmann version of the QAP can be formulated as follows:

$$\min_{\varphi \in \mathcal{S}_n} \sum_{i=1}^n \sum_{j=1}^n f_{ij} d_{\varphi(i)\varphi(j)} + \sum_{i=1}^n b_{i\varphi(i)}, \quad (1)$$

where \mathcal{S}_n is the set of all permutations $\varphi : N \rightarrow N$. Each individual product $f_{ij} d_{\varphi(i)\varphi(j)}$ is the cost of assigning facility i to location $\varphi(i)$ and facility j to location $\varphi(j)$. In the context of facility location, the matrices F and D are symmetric with zeros in the diagonal, and all matrices are nonnegative. An instance of a QAP with input matrices F , D , and B will be denoted by $\text{QAP}(F, D, B)$, while we will denote an instance by $\text{QAP}(F, D)$, if there is no linear term, that is, $B = 0$.

A more general version of the QAP was introduced by Lawler [152]. In this version, we are given a four-dimensional array $C = (c_{ijkl})$ of coefficients instead of the two matrices F and D and the problem can be stated as

$$\min_{\varphi \in \mathcal{S}_n} \sum_{i=1}^n \sum_{j=1}^n c_{ij\varphi(i)\varphi(j)}. \quad (2)$$

By adding a constant c to all cost coefficients, the objective function values change by the constant n^2c . Therefore, all cost coefficients can be assumed to be nonnegative. Clearly, a Koopmans–Beckmann problem QAP(F, D, B) can be formulated as a Lawler QAP by setting $c_{ijkl} := f_{ij}d_{kl}$ for all i, j, k, l with $i \neq j$ or $k \neq l$ and $c_{iikk} := f_{ii}d_{kk} + b_{ik}$, otherwise.

Although extensive research has been done for more than four decades, the QAP remains one of the hardest optimization problems, in contrast to its linear counterpart, the *linear assignment problem* (LAP). No exact algorithm can solve problems of size $n > 35$ in reasonable computational time nowadays. In fact, Sahni and Gonzalez [208] have shown that the QAP is NP-hard and that even finding an approximate solution within some constant factor from the optimal solution cannot be done in polynomial time unless $P = NP$. These results hold even for the Koopmans–Beckmann QAP with coefficient matrices fulfilling the triangle inequality (see Queyranne [194]). So far a polynomial-time approximation scheme has been found only for a very special case of the Koopmans–Beckmann QAP, the *dense linear arrangement problem*; see Arora et al. [16]. Complexity aspects of the QAP will be discussed in more detail in Sect. 3.4.

The QAP is widely known due to its many applications. In addition to facility layout problems, the QAP appears in applications such as chip design (cf. Hanan and Kurtzberg [128]), scheduling, process communications, turbine balancing, and many others. In the context of location theory, Dickey and Hopkins [83] discuss a *campus planning* problem with the objective to minimize the total walking distance between the buildings. A related hospital lay-out model was considered by Elshafei [92].

In the field of ergonomics, Burkard and Offermann [54] have shown that QAPs can be applied to the design of typewriter keyboards. The problem is to arrange the keys on a keyboard such as to minimize the time needed to write some text. Let $N = \{1, 2, \dots, n\}$ denote the set of symbols to be arranged. Moreover, let f_{ij} denote the frequency of the appearance of the pair of symbols i and j . The entries of the distance matrix $D = d_{kl}$ are the times needed to press the key in position l after pressing the key in position k . An assignment of symbols to keys is modeled as a permutation $\varphi \in S_n$. An optimal solution φ^* for the QAP minimizes the average time for writing a text. A similar application related to ergonomic design is the development of control boards in order to minimize eye fatigue by McCormick [162]. Other applications concern the arrangement of probes on a microarray chip [77], data analysis [131], coding theory [225], ranking of archeological data [147], ranking of a team in a relay race [129], scheduling parallel production lines [109], shunting of trams in a storage yard [224], and analyzing chemical reactions for organic compounds [220].

Literature

This survey is based on the previous article *The Quadratic Assignment Problem* by Burkard, Çela, Pardalos, and Pitsoulis in the first edition of the *Handbook of*

Combinatorial Optimization. A recent treatment on QAPs appeared in the monograph on assignment problems by Burkard et al. [46]. Quadratic assignment problems are also comprehensively treated in the monograph by Çela [60]. For further more recent surveys, see Anstreicher [12], Loiola et al. [160], the encyclopedia article of Pitsoulis and Pardalos [192], as well as Bayat and Sedghi [22] in the book *Facility Location* by Farahani and Hekmatfar.

2 Combinatorial Optimization Problems Formulated as QAPs

A number of well-known combinatorial optimization problems can be formulated as Koopmans–Beckmann QAPs with specific coefficient matrices. These problems allow an additional insight in structure and complexity of QAPs. QAP algorithms are in general, however, far inferior in comparison to specialized algorithms for these problems.

2.1 The Traveling Salesman Problem

In the *traveling salesman problem* (TSP), we are given n cities and the pairwise distances between them. The task is to find a tour which visits each city exactly once and has minimal length. Let the set of integers $N = \{1, 2, \dots, n\}$ represent the n cities and let the symmetric $n \times n$ matrix $D = (d_{ij})$ represent the distances between the cities, where d_{ij} is the distance between city i and city j ($d_{ii} = 0 \forall i = 1, 2, \dots, n$). The TSP can be formulated as

$$\begin{aligned} \min \quad & \sum_{i=1}^{n-1} d_{\varphi(i)\varphi(i+1)} + d_{\varphi(n)\varphi(1)} \\ \text{s.t. } & \varphi \in \mathcal{S}_n. \end{aligned} \tag{3}$$

The TSP can be formulated as a QAP with the given distance matrix D and a flow matrix F which is the adjacency matrix of a cycle on n vertices.

The TSP is a strongly NP-hard combinatorial optimization problem; see the monograph edited by Lawler et al. [153] as a comprehensive reference.

2.2 The Linear Arrangement Problem

In the *linear arrangement problem*, the vertices of a graph G have to be placed at the points $1, 2, \dots, n$ on a line so as to minimize the sum of pairwise distances on the line between vertices which are joined by an edge in G . The linear arrangement problem is an NP-hard problem; see Garey and Johnson [106]. It can be formulated

as a QAP where the distance matrix D is the (weighted) adjacency matrix of the given graph and the flow matrix $F = (f_{ij})$ given by $f_{ij} = |i - j|$, for all i, j .

2.3 Graph Partitioning and Maximum Clique Problem

The *graph partitioning problem* (GPP) and the *maximum clique problem* (MCP) are two well-studied NP-hard combinatorial optimization problems which are special cases of the QAP. In the GPP, we are given an (edge) weighted graph $G = (V, E)$ with n vertices and a number k which divides n . We want to partition the vertex set V into k sets V_1, \dots, V_k of equal cardinality such that the total weight of the edges between sets V_i and V_k , $i \neq k$, is minimized. This problem can be formulated as a QAP with distance matrix D equal to the weighted adjacency matrix of G and a flow matrix F . The flow matrix F is a block diagonal matrix with k blocks. Each block is the negative adjacency matrix of a complete subgraph with n/k vertices. For more informations on GPPs, the reader is referred to Lengauer [154].

In the MCP, we are again given a graph G with n vertices. We ask for the maximum number $k \leq n$ such that there exists a subset $V_1 \subseteq V$ with k vertices which induces a clique in G . This problem can be modeled as a QAP with distance matrix D equal to the adjacency matrix of G and flow matrix F given as adjacency matrix of a graph consisting of a clique of size k and $n - k$ isolated vertices, multiplied by -1 . There exists a clique of size k in G if and only if the optimal value of the corresponding QAP is $-k^2$. For a review on the MCP, the reader is referred to [190].

2.4 The Minimum-Weight Feedback Arc Set Problem

In the *minimum-weight FASP*, a weighted digraph $G = (V, E)$ with vertex set V and arc set E is given. The goal is to remove a set of arcs from E with minimum overall weight, such that all directed cycles, the so-called dicycles, in G are destroyed and an acyclic directed subgraph remains. Clearly, the minimum-weight FASP is equivalent to the problem of finding an acyclic subgraph of G with maximum weight. The unweighted version of the FASP (i.e., all edge weights are 0 or 1) is called *the acyclic subdigraph problem* and is treated extensively by Jünger [135].

An interesting application of the FASP is the so-called triangulation of input–output tables in economics which arises along the input–output analysis in order to forecast the development of industries; see Leontief [155]. For details, the reader is referred to Conrad [73] and Reinelt [196].

Since the vertices of an acyclic subdigraph can be labeled topologically, that is, such that in each arc the label of its head is larger than that of its tail, the FASP can be formulated as a QAP. The distance matrix of the QAP is the weighted adjacency matrix of G and the flow matrix $F = (f_{ij})$ is a *lower triangular matrix*, that is, $f_{ij} = -1$ if $i \leq j$ and $f_{ij} = 0$, otherwise.

The FASP is well known to be NP-hard (see [106, 142]).

2.5 Packing Problems in Graphs

Another well-known NP-hard problem which can be formulated as a QAP is the *graph packing problem*; see Bollobás [30]. In a graph packing problem we are given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with n vertices and edge sets E_1 and E_2 . A permutation φ of $\{1, 2, \dots, n\}$ is called a *packing of G_2 into G_1* , if $e_{ij} \in E_1$ implies $e_{\varphi(i)\varphi(j)} \notin E_2$, for all $1 \leq i, j \leq n$. In other words, a packing of G_2 into G_1 is an embedding of the vertices of G_2 into the vertices of G_1 such that no pair of edges coincides. The *graph packing problem* consists of finding a packing of G_2 into G_1 , if one exists, or proving that no packing exists. It can be formulated as a QAP with distance matrix equal to the adjacency matrix of G_2 and flow matrix equal to the adjacency matrix of G_1 . A packing of G_2 into G_1 exists if and only if the optimal value of this QAP is equal to 0. In this case the optimal solution of the QAP determines a packing.

3 Formulations and Computational Complexity

Many combinatorial optimization problems permit different but equivalent mathematical formulations, which stress different structural characteristics of the problem. Different problem statements are the basis for different solution approaches. Let us start with the observation that every permutation φ of the set $N = \{1, 2, \dots, n\}$ can be represented by an $n \times n$ matrix $X = (x_{ik})$, such that

$$x_{ik} = \begin{cases} 1 & \text{if } \varphi(i) = k, \\ 0 & \text{otherwise.} \end{cases}$$

Such a matrix X is called a *permutation matrix* and is characterized by the following *assignment constraints*:

$$\sum_{i=1}^n x_{ik} = 1, \quad k = 1, 2, \dots, n, \quad (4)$$

$$\sum_{k=1}^n x_{ik} = 1, \quad i = 1, 2, \dots, n, \quad (5)$$

$$x_{ik} \in \{0, 1\}, \quad i, k = 1, 2, \dots, n. \quad (6)$$

We denote the set of all permutation matrices by \mathbf{X}_n . A real $n \times n$ matrix $X = (x_{ik})$ is called a *doubly stochastic matrix*, if it fulfills the Eqs. (4) and (5) and $x_{ik} \geq 0$ for all $i, k = 1, 2, \dots, n$. The set of all doubly stochastic matrices forms the *assignment polytope* P_A . A famous result of Birkhoff shows the connection between doubly stochastic and permutation matrices.

Theorem 1 ([27]) *The vertices of the assignment polytope P_A correspond in a unique way to permutation matrices and vice versa.*

3.1 Quadratic Integer Programming Formulations

Using permutation matrices instead of permutations, the QAP (2) can be formulated as an integer program with a quadratic objective function (hence the name quadratic assignment problem by Koopmans and Beckmann [146]), namely,

$$\begin{aligned} \min \quad & \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n c_{ijkl} x_{ik} x_{jl} \\ \text{s.t. } & (x_{ik}) \in \mathbf{X}_n. \end{aligned} \quad (7)$$

Many authors have proposed methods for linearizing the quadratic objective function (7) by introducing additional variables. Some of these of linearizations are discussed in Sect. 4.

The coefficients c_{ijkl} in the objective function of (2) can be arranged in an $(n^2 \times n^2)$ matrix Q such that c_{ijkl} is on row $(i-1)n + j$ and column $(k-1)n + l$. Moreover, arrange the variables x_{ik} in a vector $x = (x_{11}, x_{12}, \dots, x_{1n}, x_{21}, \dots, x_{nn})^T = (x_1, \dots, x_{n^2})^T$. Thus the objective function of (2) becomes

$$x^T Q x. \quad (8)$$

Since $x^T Q x = x^T [1/2(Q + Q^T)]x$, matrix Q can be assumed to be symmetric. Adding a constant c to the diagonal elements of Q changes all objective function values by the same constant n^2c . If we add a sufficiently large positive constant c , all eigenvalues of Q become positive, that is, Q is positive definite and $x^T Q x$ becomes a convex function. Similarly, we can achieve that Q becomes negative definite and $x^T Q x$ becomes a concave function.

3.2 Inner Product Formulation

Inner products of matrices allow a more compact formulation of QAPs in Koopmans–Beckmann form. Given two real $n \times n$ matrices A, B , their *inner product* is defined by

$$\langle A, B \rangle := \sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{ij}.$$

Thus a linear assignment problem

$$\min_{\varphi} \sum_{i=1}^n c_{i\varphi(i)}$$

can be written as

$$\min_{X \in \mathbf{X}_n} \langle C, X \rangle.$$

Lawler [152] formulated the QAP as a linear assignment problem (LAP) of size n^2 with the additional constraint that only $(n^2 \times n^2)$ permutation matrices which are Kronecker products of $n \times n$ permutation matrices are feasible. The Kronecker product of the two $n \times n$ matrices A and B is defined as

$$A \otimes B := \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nn}B \end{pmatrix}.$$

Thus a QAP can be written as

$$\begin{aligned} \min \quad & \langle Q, Y \rangle \\ \text{s.t. } & Y = X \otimes X, \\ & X \in \mathbf{X}_n. \end{aligned} \tag{9}$$

Now let X_φ be the permutation matrix corresponding to permutation φ and let X_φ^T be the transposed matrix of X_φ . Given an $n \times n$ matrix A , the matrix products AX_φ^T and $X_\varphi A$ permute the columns and rows of A , respectively, according to the permutation φ . Therefore,

$$X_\varphi A X_\varphi^T = (a_{\varphi(i)\varphi(j)}).$$

Thus a Koopmans–Beckmann QAP can be formulated alternatively as

$$\begin{aligned} \min_{\varphi} \quad & \langle F, X_\varphi D X_\varphi^T \rangle + \langle B, X_\varphi \rangle \\ \text{s.t. } & X_\varphi \in \mathbf{X}_n. \end{aligned} \tag{10}$$

If in a Koopmans–Beckmann QAP one of the matrices F and D is symmetric, we can always assume that also the other matrix is symmetric. Let us assume that matrix F is symmetric, that is, $F = F^T$. From $\langle F, G \rangle = \langle F^T, G^T \rangle$ follows

$$\langle F, G \rangle = \frac{1}{2} (\langle F, G \rangle + \langle F^T, G^T \rangle) = \langle F, \frac{1}{2}(G + G^T) \rangle.$$

Setting $G = X_\varphi D X_\varphi^T$ and using $\frac{1}{2}(G + G^T) = X_\varphi \frac{1}{2}(D + D^T) X_\varphi^T$ show that also matrix D can be assumed to be symmetric.

3.3 Trace Formulation

The *trace* of an $n \times n$ matrix A is defined as the sum of its diagonal elements, that is,

$$\text{tr}A := \sum_{i=1}^n a_{ii}.$$

Well-known properties of the trace are $\text{tr}(AB) = \text{tr}(BA)$ and $\text{tr}A = \text{tr}A^T$. Obviously, the inner product $\langle A, B \rangle$ can be written as

$$\langle A, B \rangle = \text{tr}(AB^T).$$

Thus,

$$\langle F, X_\varphi DX_\varphi^T \rangle + \langle B, X_\varphi \rangle = \text{tr}(FX_\varphi D^T X_\varphi^T) + \text{tr}(BX_\varphi^T).$$

Therefore, a Koopmans–Beckmann QAP instance with input matrices F , D , and B can be formulated as

$$\begin{aligned} \min \quad & \text{tr}(FXD^T + B)X^T \\ \text{s.t. } & X \in \mathbf{X}_n. \end{aligned} \tag{11}$$

The trace formulation of the QAP appeared first in Edwards [90, 91] and was used by Finke et al. [98] to introduce lower bounds based on the eigenvalues of symmetric matrices F and D (see Sect. 7.1).

For any $n \times n$ matrix A , let $\text{vec}(A) \in \mathbb{R}^{n^2}$ be the vector formed by the columns of A . In the following, we write just x (as before) instead of $\text{vec}(X)$, if no confusion arises. A well-known identity from matrix analysis states that

$$\text{vec}(AXB) = (B^T \otimes A)\text{vec}(X). \tag{12}$$

Using

$$\text{tr}(AB^T) = \langle A, B \rangle = (\text{vec}B)^T(\text{vec}A),$$

the trace formulation of a QAP can be rewritten as

$$\text{tr}(FXD^T X^T) = x^T \text{vec}(FXD^T) = x^T(D \otimes F)x.$$

Therefore, a Koopmans–Beckmann problem can be formulated as

$$\begin{aligned} \min \quad & x^T(D \otimes F)x + \text{vec}(B)^T x, \\ \text{s.t. } & X \in \mathbf{X}_n. \end{aligned} \tag{13}$$

This formulation will be used in semidefinite relaxations of the QAP; see Sect. 6.7.

3.4 Computational Complexity

The QAP is a “very hard” problem from the theoretical point of view. Not only that the QAP cannot be solved efficiently but even it cannot be approximated efficiently within some constant approximation ratio. Furthermore, even finding local optima is a nontrivial task.

Let us recall that the TSP can be formulated as a $\text{QAP}(F, D)$ where D is the distance matrix between the n cities and F is the adjacency matrix of a cycle on n vertices (see Sect. 2.1). Since the TSP is strongly NP-hard, we get the same result for QAP. Now, let $Z(F, D, \varphi)$ denote the objective function value of a solution φ for a QAP with flow matrix F and distance matrix D . Given a real number $\epsilon > 0$, an algorithm \mathcal{A} for the QAP is said to be an ϵ -approximation algorithm if

$$\left| \frac{Z(F, D, \varphi_{\mathcal{A}}) - Z(F, D, \varphi_{\text{opt}})}{Z(F, D, \varphi_{\text{opt}})} \right| \leq \epsilon \quad (14)$$

holds for every instance $\text{QAP}(F, D)$, where $\varphi_{\mathcal{A}}$ is the solution of $\text{QAP}(F, D)$ computed by algorithm \mathcal{A} and φ_{opt} is an optimal solution of $\text{QAP}(F, D)$. Sahni and Gonzales showed by a reduction from the Hamiltonian cycle problem the following result.

Theorem 2 ([208]) *The quadratic assignment problem is strongly NP-hard.*

For an arbitrary $\epsilon > 0$, the existence of a polynomial-time ϵ -approximation algorithm for the QAP implies $\mathcal{P} = \mathcal{NP}$.

Queyranne [194] derives an even stronger result. He shows that, unless $P = NP$, $\text{QAP}(F, D)$ is not approximable in polynomial time within some finite approximation ratio, even if F is the distance matrix of some pointset on a line and D is a symmetric block diagonal matrix. Arkin et al. [15] considered the maximization version of a QAP where the entries of matrix D fulfill the triangle inequality. They designed an approximation algorithm of $O(n^3)$ time complexity which guarantees a solution value within $1/4$ from the optimum.

Complexity aspects of local search algorithms are dealt with in Sect. 8.4. Let us just mention here that no local criteria are known for deciding how good a locally optimal solution is as compared to a global one. From the complexity point of view, deciding whether a given local optimum is a globally optimal solution to a given instance of the QAP, is a hard problem; see Papadimitriou and Wolfe [185].

4 Linearizations

A first approach for solving QAPs consists in reformulating it as a linear (mixed-)integer program. The linearization of the quadratic objective function is usually achieved by introducing new variables and new linear (and binary)

constraints. Then existing methods for (mixed) linear integer programming (MILP) can be applied. The very large number of new variables and constraints, however, usually poses an obstacle for efficiently solving the resulting linear integer programs. But MILP formulations provide a means for computing lower bounds. In this context, the “tightness” of the continuous relaxation of the resulting linear integer program is a desirable property.

In this section, we present five linearizations of the QAP: Lawler’s linearization [152] which was the first, Kaufman and Broeckx’s linearization [143] which has the smallest number of variables and constraints, Frieze and Yadegar’s linearization [103], and the linearization of Adams and Johnson [4] which is a slight but relevant modification of the linearization proposed by Frieze and Yadegar and unifies most of the previous linearizations. Finally, we present the higher level reformulation technique by Adams et al. [3] which is important for improving lower bounds.

4.1 Lawler’s Linearization

Lawler [152] replaces the quadratic terms $x_{ik}x_{jl}$ in the objective function of (2) by n^4 variables

$$y_{ijkl} := x_{ik}x_{jl}, \quad i, j, k, l = 1, 2, \dots, n$$

and obtains the following 0–1 linear program (see [152] and the monograph by Burkard et al. [46]).

$$\begin{aligned} \min \quad & \sum_{i,j=1}^n \sum_{k,l=1}^n c_{ijkl} y_{ijkl} \\ \text{s.t.} \quad & (x_{ik}) \in \mathbf{X}_n, \\ & \sum_{i,j=1}^n \sum_{k,l=1}^n y_{ijkl} = n^2, \\ & x_{ik} + x_{jl} - 2y_{ijkl} \geq 0, \quad i, j, k, l = 1, 2, \dots, n, \\ & y_{ijkl} \in \{0, 1\}, \quad i, j, k, l = 1, 2, \dots, n. \end{aligned} \tag{15}$$

This 0–1 linear program has $n^4 + n^2$ binary variables and $n^4 + 2n^2 + 1$ constraints.

4.2 Kaufman and Broeckx Linearization

Recall that all cost coefficients c_{ijkl} can be assumed to be nonnegative. It is possible to rearrange the objective function in the following way:

$$\sum_{i,k=1}^n x_{ik} \sum_{j,l=1}^n c_{ijkl} x_{jl}. \quad (16)$$

Kaufman and Broeckx [143] define n^2 new real variables

$$w_{ik} := x_{ik} \sum_{j,l=1}^n c_{ijkl} x_{jl}, \quad i, k = 1, \dots, n \quad (17)$$

and plug them in the objective function (16) to obtain a linear objective function of the form

$$\sum_{i,k=1}^n w_{ik}.$$

Then they introduce n^2 constants $a_{ik} := \sum_{j,l=1}^n c_{ijkl}$ for $i, k = 1, \dots, n$ and show that the QAP (2) is equivalent to the following mixed 0–1 linear program:

$$\begin{aligned} \min \quad & \sum_{i,k=1}^n w_{ik} \\ \text{s.t.} \quad & (x_{ik}) \in \mathbf{X}_n, \\ & a_{ik} x_{ik} + \sum_{j,l=1}^n c_{ijkl} x_{jl} - w_{ik} \leq a_{ik}, \quad i, k = 1, \dots, n, \\ & w_{ik} \geq 0, \quad i, k = 1, 2, \dots, n. \end{aligned} \quad (18)$$

This formulation employs n^2 real variables, n^2 binary variables, and $n^2 + 2n$ constraints. The proof of equivalence of the QAP to the mixed integer linear program (18) can be found in [46, 143]. Similar linearizations were proposed by Balas and Mazzola [18] and Burkard and Bönniger [40]; see Burkard et al. [46]. All of them are related to general linearization strategy proposed by Glover [111].

Zhang et al. [228] observed that the bound obtained from the Kaufman–Broeckx linearization is dominated by the Gilmore–Lawler bound (see Sect. 6.2), and they proposed a way to strengthen the Kaufman–Broeckx model.

4.3 Frieze and Yadegar Linearization

Frieze and Yadegar [103] also replace the products $x_{ik} x_{jl}$ of the binary variables by continuous variables y_{ijkl} ($y_{ijkl} := x_{ik} x_{jl}$) and set up the following mixed integer linear programming formulation for the QAP (2):

$$\min \sum_{i,j=1}^n \sum_{k,l=1}^n c_{ijkl} y_{ijkl} \quad (19)$$

$$\text{s.t. } (x_{ik}) \in \mathbf{X}_n, \quad (20)$$

$$\sum_{i=1}^n y_{ijkl} = x_{jl}, \quad j, k, l = 1, \dots, n, \quad (21)$$

$$\sum_{j=1}^n y_{ijkl} = x_{jl}, \quad i, k, l = 1, 2, \dots, n, \quad (22)$$

$$\sum_{k=1}^n y_{ijkl} = x_{ik}, \quad i, j, l = 1, \dots, n, \quad (23)$$

$$\sum_{l=1}^n y_{ijkl} = x_{ik}, \quad i, j, k = 1, 2, \dots, n, \quad (24)$$

$$y_{ikik} = x_{ik}, \quad i, k = 1, 2, \dots, n, \quad (25)$$

$$0 \leq y_{ijkl} \leq 1, \quad i, j, k, l = 1, 2, \dots, n. \quad (26)$$

This mixed integer program has n^4 real variables, n^2 binary variables, and $n^4 + 4n^3 + n^2 + 2n$ constraints. For obtaining a lower bound Frieze and Yadegar, consider a Lagrangean relaxation of this mixed integer program by relaxing the constraints (23) and (26). The relaxed problem is solved approximately by applying subgradient optimization techniques. They show that the bound obtained by the Lagrangean relaxation is larger than all lower bounds derived from reduction techniques applied to the Gilmore–Lawler bound for the QAP (see Sect. 6.2). A result of Geoffrion [108] implies that the solution of the Lagrangean relaxation equals the solution of the continuous relaxation of the mixed integer program (19)–(26).

It is interesting to notice here that the gap between the optimal value of this continuous relaxation and the optimal value of the QAP can be very large. Dyer et al. [89] showed for QAPs whose coefficients c_{ijkl} are independent random variables uniformly distributed on $[0, 1]$ that the expected optimal value of the above mentioned linearization has a size of $O(n)$. On the other hand, the expected optimal value of such QAPs increases with high probability as $\Omega(n^2)$, as shown by Burkard and Fincke [50]. Consequences of this asymptotic behavior will be discussed in some detail in Sect. 11. No similar asymptotic result is known for the continuous relaxation of the linearization due to Adams and Johnson [4] which is presented in the following section.

4.4 Adams and Johnson Linearization

Adams and Johnson presented in [4] a new 0–1 linear integer programming formulation for the QAP, called *level-1 reformulation–linearization*, which resembles to a

certain extent the linearization of Frieze and Yadegar. It is based on the linearization technique for general 0–1 polynomial programs introduced by Adams and Sherali in [5, 6]. They represent again the product $x_{ik}x_{jl}$ by the variables y_{ijkl} and prove that the QAP is equivalent to the following mixed 0–1 linear program

$$\begin{aligned} \min \quad & \sum_{i,j=1}^n \sum_{k,l=1}^n c_{ijkl} y_{ijkl} \\ \text{s.t. } \quad & (x_{ik}) \in \mathbf{X}_n, \end{aligned} \tag{27}$$

$$\begin{aligned} \sum_{i=1}^n y_{ijkl} &= x_{jl}, \quad j, k, l = 1, \dots, n, \\ \sum_{j=1}^n y_{ijkl} &= x_{jl}, \quad i, k, l = 1, 2, \dots, n, \\ y_{ijkl} &= y_{jilk}, \quad i, j, k, l = 1, \dots, n, \\ y_{ijkl} &\geq 0, \quad i, j, k, l = 1, 2, \dots, n. \end{aligned} \tag{28}$$

The above formulation contains n^2 binary variables x_{ij} , n^4 continuous variables y_{ijkl} , and $n^4 + 2n^3 + 2n$ constraints excluding the nonnegativity constraints on the continuous variables. The constraint set of (27) describes a solution matrix Y which is the Kronecker product of two permutation matrices (i.e., $Y = X \otimes X$ where $X \in \mathcal{S}_n$). Hence, this formulation of the QAP is equivalent to (9). The theoretical strength of the linearization (27) lies in the fact that the constraints in the previous linearizations can be expressed as linear combinations of the constraints of (27); see [4, 134]. Many lower-bounding techniques are based on the Lagrangean dual of this relaxation. For more details on this topic, we refer to Sect. 6.4.

Adams and Johnson [4] noted already that a significant smaller formulation in terms of both the variables and constraints could be obtained, but the structure of the continuous relaxation above is favorable for solving QAPs approximately by means of the Lagrangean dual.

4.5 Level-2 Reformulation–Linearization

The level-2 reformulation of a QAP is obtained by multiplying every constraint in (28) with each variable x_{rs} [3]. Afterwards, the products $x_{ik}x_{jl}x_{rs}$ are linearized by introducing new variables z_{ijrkl} . Thus, the following constraints are added to (28):

$$\sum_{\substack{i=1 \\ i \neq j,r}}^n z_{ijrkl} = y_{jrls} \quad (j, k, l, r, s = 1, 2, \dots, n, k \neq l \neq s, j \neq r),$$

$$\sum_{\substack{k=1 \\ j \neq l, s}}^n z_{ijrkl} = y_{jrls} \quad (i, j, l, r, s = 1, 2, \dots, n, i \neq j \neq r, l \neq s), \quad (29)$$

$$z_{ijrkl} = z_{irjks} = z_{jirls} = z_{jrils} = z_{rjisl} = z_{rjislk}$$

$$(i, j, k, l, p, q = 1, 2, \dots, n, i < j < r, k \neq l \neq s),$$

$$z_{ijklp} \geq 0 \quad (i, j, k, l, p, q = 1, 2, \dots, n, i \neq k \neq p, j \neq l \neq q).$$

Adams et al. note that bounds from the continuous relaxation of (29) are at least as tight as those from (28). Indeed, numerical experiments show that considerably better bounds can be achieved by using a Lagrangean relaxation of (29).

The above technique can be extended to obtain level-3 reformulations; see Hahn et al. [126]. Level-3 reformulations provide better lower bounds but need a high computational effort. Nevertheless, Hahn et al. report favorable computational results when used in a branch-and-bound framework.

5 QAP Polytopes

All linearizations of Sect. 4 can be used as starting point for a definition of the QAP polytope. The QAP polytope is defined as the convex hull of all 0–1 vectors (x_{ik}, y_{ijkl}) , $1 \leq i, j, k, l \leq n$, which are feasible solutions of the MILP formulations given there. Jünger and Kaibel [136], however, used a graph theoretic approach for defining the QAP polytope. This has the advantage of leading to a better understanding of the relationship between the QAP polytope and related polytopes, like the traveling salesman polytope or the cut polytope (see [138]). In addition, the graph theoretic approach allows an easier use of some technical tools like projections and affine transformations.

5.1 The QAP Polytope

For each $n \in \mathbb{N}$, consider a graph $G_n = (V_n, E_n)$ with vertex set $V_n = \{(i, k) : 1 \leq i, k \leq n\}$ and edge set $E_n = \{((i, k), (j, l)) : i \neq j, k \neq l\}$. Clearly, maximal cliques in G_n have cardinality n and correspond to permutation matrices. Given an instance of the QAP (2), we view the cost coefficients c_{ikik} as vertex weights and c_{ijkl} as edge weights of the edge $((i, k), (j, l))$. Solving the above QAP instance is equivalent to finding a maximal clique with minimum total vertex and edge weight. For each clique C in G_n with n vertices, we denote its incidence vector by (x^C, y^C) , where $x^C \in \mathbb{R}^{n^2}$ and $y^C \in \mathbb{R}^{\frac{n^2(n-1)^2}{2}}$

$$x_{ik} = \begin{cases} 1 & \text{if } (i, k) \in C, \\ 0 & \text{otherwise} \end{cases} \quad y_{ijkl} = \begin{cases} 1 & \text{if } ((i, k), (j, l)) \in C, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, the QAP polytope denoted by QAP_n is obtained by

$$\text{QAP}_n := \text{conv}\{(x^C, y^C) : C \text{ is a clique with } n \text{ vertices in } G_n\}.$$

Barvinok [20], Padberg and Rijal [180], as well as Jünger and Kaibel [136] determined independently the dimension of QAP_n as

$$\dim(\text{QAP}_n) = 1 + (n - 1)^2 + n(n - 1)(n - 2)(n - 3)/2 \text{ for } n \geq 3$$

and showed that the inequalities $y_{ijkl} \geq 0, i \neq k, j \neq l$ are facet defining. (These are usually called *trivial facets* of QAP_n .) Moreover, Padberg and Rijal [180], as well as Jünger and Kaibel [136], showed independently that the *affine hull* of QAP_n , for $n \geq 3$, is described by the following equations:

$$\sum_{i=1}^n x_{ik} = 1, \quad 1 \leq k \leq n, \quad (30)$$

$$\sum_{k=1}^n x_{ik} = 1, \quad 1 \leq i \leq n, \quad (31)$$

$$-x_{jl} + \sum_{i=1}^{j-1} y_{ijkl} + \sum_{i=j+1}^n y_{ijkl} = 0 \quad 1 \leq i, j, k \leq n; i \neq j, \quad (32)$$

$$-x_{ik} + \sum_{l=1}^{k-1} y_{ijkl} + \sum_{l=k+1}^n y_{ijkl} = 0 \quad 1 \leq i, k, l \leq n; k \neq l. \quad (33)$$

The rank of this equation system is $2n(n - 1)^2 - (n - 1)(n - 2)$ for $n \geq 3$.

Padberg and Rijal [180] identified additionally two classes of valid inequalities for QAP_n , the *clique inequalities* and the *cut inequalities*. The clique inequalities are derived from cliques in the graph G_n , whereas the cut inequalities are derived from cuts in the graph G_n . The authors identify some conditions under which the cut inequalities are not facet defining. It is an open problem, however, to identify facet-defining inequalities within these classes. A larger class of valid inequalities, the so-called box inequalities, has been described by Kaibel [138]. Those inequalities are obtained by exploiting the relationship between the Boolean quadric polytope and the QAP polytope. A nice feature of the box inequalities is that it can be decided efficiently whether they are facet defining or not, and in the latter case some facet defining inequality which dominates the corresponding box inequality can be derived. A further class of valid inequalities for the QAP which partly correspond to facets of the QAP polytope was proposed by Blanchard et al. [28].

Both, the *traveling salesman polytope* and the *linear ordering polytope* are projections of the quadratic assignment polytope QAP_n . Moreover, QAP_n is a face of the *Boolean quadric polytope*; see [138].

5.2 The Symmetric QAP Polytope

Koopmans–Beckmann problems (1) with symmetric coefficient matrices F and D lead to the so-called symmetric *QAP* polytope SQAP_n . (Recall that the symmetry of already one of the matrices F and D leads to a symmetric problem; see Sect. 3.3). To define SQAP_n , we start from a hypergraph $H_n = (V_n, F_n)$, where V_n is the same set of vertices as in graph G_n and F_n is the set of hyperedges $\{(i, k), (j, l), (i, l), (j, k)\}$ for all $i \neq j, k \neq l$. A set $C \subset V_n$ is called a clique in H_n if C is a clique in G_n . Again, the incidence vector (x^C, y^C) of a clique C is introduced by

$$x_{ik} = \begin{cases} 1 & \text{if } (i, k) \in C \\ 0 & \text{otherwise} \end{cases} \quad y_{ijkl} = \begin{cases} 1 & \text{if } i < j, l \neq k, (i, k), (j, l) \in C \\ 0 & \text{otherwise.} \end{cases}$$

The vector x^C has n^2 components and the vector y^C has $\frac{n^2(n-1)^2}{4}$ components. The polytope SQAP_n is defined as convex hull of the vectors (x^C, y^C) , that is,

$$\text{SQAP}_n := \text{conv}\{(x^C, y^C) : C \text{ is a clique with } n \text{ vertices in } H_n\}.$$

Similar to the general case, the following structural results were derived:

Theorem 3 ([137, 180]) *Let $n \geq 3$.*

1. *The dimension of SQAP_n is equal to $(n-1)^2 + n^2(n-3)^2/4$.*
2. *The inequalities $y_{ijkl} \geq 0$ for $i < j, k < l$, and $x_{ik} \geq 0$ for $1 \leq i, k \leq n$, define facets of SQAP_n .*

Padberg and Rijal [180] as well as Jünger and Kaibel [137] derived a minimal linear description of the affine hull of SQAP_n which is similar to (30)–(33). Moreover, Jünger and Kaibel [137] introduced a class of facet-defining inequalities, the so-called row curtain inequalities

$$\sum_{k \in K} x_{ik} - \sum_{\substack{k,l \in K \\ k < l}} y_{ijkl} \geq 0 \text{ for } i < j \text{ and } K \subseteq \{1, 2, \dots, n\} \quad (34)$$

and the column curtain inequalities

$$\sum_{i \in I} x_{ik} - \sum_{\substack{i,j \in I \\ i < j}} y_{ijkl} \geq 0 \text{ for } k < l \text{ and } I \subseteq \{1, 2, \dots, n\}. \quad (35)$$

All curtain inequalities with $3 \leq |I|, |J| \leq n-3$ define facets of SQAP_n . The other curtain inequalities define facets which are contained in trivial facets of SQAP_n . The separation problem for these inequalities has been shown to be NP-hard.

6 Lower Bounds

Since QAPs are NP-hard optimization problems, there is a special need for lower bounds on the optimal objective function value for use in implicit enumeration methods and in heuristics. In general, lower bounds should be:

- *Tight*, that is, the gap between the bound and the optimal objective function value should be small.
- They should be much easier to compute than solving the original problem.
- Their computation should be possible in the case that already some assignments are fixed.

There is no clear performance ranking of the lower bounds that will be discussed below. In light of the asymptotic behavior of the QAP (see Sect. 11), it should be fair to assume that the tightness of the lower bound probably dominates all other of the above criteria. Namely, if there is a large number of feasible solutions close to the optimum, then a lower bound which is not tight will fail to eliminate a large number of subproblems in the enumeration process.

6.1 Admissible Transformations

Admissible transformations provide a general tool for deriving lower bounds of combinatorial optimization problems; see Burkard [39]. Consider a QAP

$$\min_{\varphi \in S_n} \sum_{i=1}^n \sum_{j=1}^n c_{ij\varphi(i)\varphi(j)}$$

and denote the objective function value of permutation φ by $Z(C, \varphi)$.

Definition 1 A transformation T of the cost array C in the cost array $T(C) = \bar{C}$ is called *admissible with index* $z(T)$, if for all permutations φ the equation

$$Z(C, \varphi) = z(T) + Z(\bar{C}, \varphi) \quad (36)$$

holds.

Admissible transformations can be iterated: performing an admissible transformation T with index $z(T)$ after an admissible transformation S with index $z(S)$ corresponds to the admissible transformation $T \circ S$ with index $z(T) + z(S)$. We get the following:

Proposition 1 *Let T be an admissible transformation of the cost array C into \bar{C} with $\bar{C} \geq 0$. Then $z(T)$ constitutes a lower bound on the optimal objective function value. If there is a permutation φ^* with $Z(\bar{C}, \varphi^*) = 0$, then $z(T)$ is the optimal value of the QAP.*

Proof Let φ be any permutation. Then we get

$$Z(C, \varphi) = z(T) + Z(\bar{C}, \varphi) \geq z(T) = z(T) + Z(\bar{C}, \varphi^*) = Z(C, \varphi^*).$$

□

In order to describe admissible transformations for QAPs, let us arrange the cost elements in an $(n^2 \times n^2)$ matrix C such that c_{ijkl} is on row $(i-1)n+j$ and column $(k-1)n+l$ (cf. Sect. 3.1). Matrix C can be written as a block matrix (C^{ik}) where every C^{ik} is the matrix (c_{ijkl}) with fixed indices i and k . The coefficients c_{iilk} with $k \neq l$ and the coefficients c_{ijkk} with $i \neq j$ can never occur in the objective function, since φ is a one-to-one mapping. Therefore, we can assume

$$c_{ijkl} = \infty \quad \text{for } (i = j \text{ and } k \neq l) \text{ or } (i \neq j \text{ and } k = l). \quad (37)$$

For $n = 3$, the cost matrix C has the form (using “*” for ∞)

$$\begin{aligned} C &= \begin{pmatrix} C^{11} & C^{12} & C^{13} \\ C^{21} & C^{22} & C^{23} \\ C^{31} & C^{32} & C^{33} \end{pmatrix} \\ &= \left(\begin{array}{ccc|ccc|ccc} c_{1111} & * & * & * & c_{1122} & * & * & * & * & c_{1133} \\ * & c_{1212} & c_{1213} & c_{1221} & * & c_{1223} & c_{1231} & c_{1232} & * \\ * & c_{1312} & c_{1313} & c_{1321} & * & c_{1323} & c_{1331} & c_{1332} & * \\ \hline * & c_{2112} & c_{2113} & c_{2121} & * & c_{2123} & c_{2131} & c_{2132} & * \\ c_{2211} & * & * & * & c_{2222} & * & * & * & * & c_{2233} \\ * & c_{2312} & c_{2313} & c_{2321} & * & c_{2323} & c_{2331} & c_{2332} & * \\ * & c_{3112} & c_{3113} & c_{3121} & * & c_{3123} & c_{3131} & c_{3132} & * \\ * & c_{3212} & c_{3213} & c_{3221} & * & c_{3223} & c_{3231} & c_{3232} & * \\ c_{3311} & * & * & * & c_{3322} & * & * & * & * & c_{3333} \end{array} \right). \end{aligned}$$

Moreover, due to $x_{ik}x_{jl} = 1$, we can symmetrize the problem and set:

$$c_{ijkl} := \begin{cases} c_{ijkl} + c_{klji} & \text{for } j > i, \\ 0 & \text{for } j < i. \end{cases} \quad (38)$$

Every submatrix C^{ik} has only one entry $c_{iikk} \neq \infty$ in row i and column k , the so-called leading element. Admissible transformations for the QAP were implicitly used by Hahn and Grant [124, 125]. They can be described in the following way; see Burkard [39].

Theorem 4 Let $C = (C^{ik})$ be the cost matrix of a general QAP (2). There are two types of admissible transformations:

- I. Let C^{ik} be a fixed submatrix of C . Adding arbitrary constants to the rows and columns of C^{ik} such that the constants sum up to 0 yields an admissible transformation T with index $z(T) = 0$.

II. Adding arbitrary constants to the rows and columns of C yields an admissible transformation T whose index $z(T)$ equals the negative sum of the constants.

The first part of [Theorem 4](#) tells that we can add arbitrary constants to the rows $j \neq i$ and columns $l \neq k$, provided that we subtract the same constants from the leading element c_{iikk} . Note that the constants can also be negative. The second part of [Theorem 4](#) tells that we can collect the leading elements c_{iikk} in an $n \times n$ matrix $L = (l_{ik})$ with $l_{ik} = c_{iikk}$. Any admissible transformation applied to the linear assignment problem with cost matrix L leads to an admissible transformation for the QAP (see [\[39, 46, 51\]](#)).

Admissible transformations are a main tool for deriving lower bounds for the QAP. In particular, the Gilmore–Lawler bound as well as reductions, reformulations, and redistributions can be interpreted as special applications of admissible transformations.

6.2 Gilmore–Lawler Type Lower Bounds

Based on the formulation of the general QAP as a linear assignment problem of dimension n^2 stated in formulation [\(9\)](#), Gilmore [\[110\]](#) and Lawler [\[152\]](#) derived lower bounds for the QAP by solving linear assignment problems.

Consider an instance of the Lawler QAP [\(2\)](#) with the $n^2 \times n^2$ coefficient matrix $C = (c_{ijkl})$. First, we solve n^2 linear assignment problems

$$\bar{l}_{ik} := \min_X \langle C^{ik}, X \rangle.$$

According to the first part of [Proposition 4](#), we can replace c_{iikk} in C by $c_{iikk} + \bar{l}_{ik}$ and the coefficients c_{ijkl} by the reduced cost coefficients in the optimal solution of the linear assignment problem $\min_X \langle C^{ik}, X \rangle$. According to the second part of [Proposition 4](#), we can collect the leading elements c_{iikk} in an $n \times n$ matrix $L = (l_{ik})$ with $l_{ik} = c_{iikk}$. Since the optimal value of the linear assignment problem with cost matrix L can be found by adding constants to the rows and columns of matrix L (and therefore of matrix C !), it constitutes a lower bound for the QAP, the so-called Gilmore–Lawler bound. Let X^* be an optimal solution of the linear assignment problem with cost matrix L and let $Y^* := X^* \otimes X^*$. According to [Proposition 1](#), matrix X^* is an optimal solution of the given QAP, if $\langle C, Y^* \rangle = 0$.

Since each linear assignment problem can be solved in $O(n^3)$ time, this lower bound for the Lawler QAP [\(2\)](#) of dimension n can be computed in $O(n^5)$ time.

In case of Koopmans–Beckmann problems [\(1\)](#), the computation can be sped up as $C = D \otimes F$. The computational effort for computing the lower bound can be reduced to $O(n^3)$. Namely, instead of solving n^2 linear assignment problems by the Hungarian method, the optimal value of the assignment problem $\min_X \langle C^{ik}, X \rangle$ can be obtained by using the following result:

Proposition 2 ([127]) Let two n -dimensional real vectors $a = (a_i)$, $b = (b_i)$ with $0 \leq a_1 \leq a_2 \leq \dots \leq a_n$ and $b_1 \geq b_2 \geq \dots \geq b_n \geq 0$ be given. Then the following inequalities hold for any permutation φ of $1, 2, \dots, n$:

$$\sum_{i=1}^n a_i b_i \leq \sum_{i=1}^n a_i b_{\varphi(i)} \leq \sum_{i=1}^n a_i b_{n-i+1}.$$

Given an arbitrary nonnegative vector $a \in \mathbb{R}^n$, let ψ be a permutation which sorts a nondecreasingly and π a permutation which sorts a nonincreasingly. We denote

$$\langle a, b \rangle^- := \sum_{i=1}^n a_{\psi(i)} b_{\pi(i)} \quad \text{and} \quad \langle a, b \rangle^+ := \sum_{i=1}^n a_{\psi(i)} b_{\psi(i)}. \quad (39)$$

Consider now an instance QAP(F, D, B) of the Koopmans–Beckmann QAP (1). It can be written as a general QAP by setting

$$c_{ijkl} := \begin{cases} f_{ij}d_{kl}, & \text{for } i \neq j, k \neq l, \\ f_{ii}d_{kk} + b_{ik}, & \text{for } i = j, k = l. \end{cases}$$

Each submatrix $C^{(ik)}$ of the $n^2 \times n^2$ matrix C is given by $C^{(ik)} = (f_{ij}d_{kl})$ with fixed values i and k . The optimal objective function value of the linear assignment problem $\min_X \langle C^{(ik)}, X \rangle$ can be found by applying Proposition 2 to the $(n-1)$ -dimensional vectors $\hat{f}_{(i,.)}$ and $\hat{d}_{(k,.)} \in \mathbb{R}^{n-1}$ which are obtained from the i th and the k th row of F and D by deleting the i th and the k th element, respectively. We get

$$\min_X \langle C^{(ik)}, X \rangle = f_{ii}d_{kk} + \langle \hat{f}_{(i,.)}, \hat{d}_{(k,.)} \rangle^-. \quad (40)$$

The Gilmore–Lawler bound is obtained by solving a linear assignment problem with cost matrix $L = (l_{ik})$ where l_{ik} are given by

$$l_{ik} = b_{ik} + f_{ii}d_{kk} + \langle \hat{f}_{(i,.)}, \hat{d}_{(k,.)} \rangle^-. \quad (41)$$

The appropriate sorting of the rows and columns of F and D can be done in $O(n^2 \log n)$ time. Then the computation of all l_{ij} takes $O(n^3)$ time and the same amount of time is needed to solve the last linear assignment problem.

Li et al. [157] have shown that the decision problem whether the Gilmore–Lawler bound equals to the optimal objective function value is NP-complete.

Similar bounds have been proposed by Christofides and Gerrard [67]. The basic idea relies again on decomposing the given QAP into a number of subproblems which can be solved efficiently. First solve each subproblem, then build a matrix with the optimal values of the subproblems, and solve a linear assignment problem with that matrix as cost matrix to obtain a lower bound for the given QAP.

Christofides et al. decompose the Koopmans–Beckmann QAP(F, D) based on isomorphic subgraphs of graphs whose weighted adjacency matrices are F and D . The Gilmore–Lawler bound is obtained as a special case, if these subgraphs are star graphs. In general the Gilmore–Lawler bound outperforms the bounds obtained by employing other subgraphs, like single edges or double stars (see also [107]).

6.3 Reduction Methods

The Gilmore–Lawler bound is simple to compute, but it deteriorates fast as n increases. The quality of this lower bound can be improved if the given problem is transformed such that the contribution of the quadratic term in the objective function is decreased by moving costs to the linear term. This is the aim of the so-called reduction methods. A reduction corresponds to an admissible transformation of type I which increases the leading element c_{iikk} and reduces the other elements in C^{ik} but keeps them nonnegative. The reduction approach was introduced by Conrad [73]. Reductions have been investigated by many researchers (see [36, 91, 103, 205]).

In the case of the Koopmans–Beckman QAP, the reduction scheme is

$$f_{ij} = \bar{f}_{ij} + \lambda_j, \quad i \neq j, \quad (42)$$

$$d_{kl} = \bar{d}_{kl} + \mu_l, \quad k \neq l. \quad (43)$$

Frieze and Yadegar [103] have shown that a simultaneous reduction of rows and columns in F and D does not improve the lower bound. Therefore, it is unnecessary to consider a column reduction in matrix F and matrix D . Several rules have been suggested for the choice of λ and μ , for example,

$$\lambda_j = \min\{f_{ij} \mid 1 \leq i \leq n, i \neq j\} \quad (44)$$

$$\mu_l = \min\{d_{kl} \mid 1 \leq k \leq n, k \neq l\}. \quad (45)$$

Li et al. [157] choose λ_j and μ_l such as to minimize the variance of all off-diagonal elements in the j th column of F and the l th column of D . Let $m(F_j)$ be the arithmetic mean of the off-diagonal entries in j th column of matrix F :

$$m(F_j) = \frac{1}{n-1} \sum_{\substack{i=1 \\ i \neq j}}^n f_{ij}. \quad (46)$$

The value $m(D_l)$ is defined in an analogous way. The choice proposed in [157] is:

$$\lambda_j = m(A_n) - m(A_k), \quad (47)$$

$$\mu_l = m(B_n) - m(B_l). \quad (48)$$

Li et al. developed also another reduction scheme where the matrices F and D are replaced by $F + \Delta F$ and $D + \Delta D$, respectively. ΔF and ΔD are chosen such as to minimize the variance of the off-diagonal elements in the flow matrix F and distance matrix D . These bounds perform well on QAPs whose input matrices have high variances, but their performance reduces to that of the Gilmore–Lawler bound when the variance of the input matrices is small. For details the reader is referred to the original paper [157].

Similar reduction schemes can be applied to the general Lawler QAP (2).

As mentioned in Sect. 4.3, Frieze and Yadegar derived lower bounds for the QAP based on a Lagrangean relaxation of the mixed integer linear programming formulation (19)–(26). They include the constraints (21) and (22) in the objective function (19) by Lagrangean multipliers e and g :

$$\begin{aligned} & \sum_{ijkl} c_{ijkl} y_{ijkl} + \sum_{jkl} e_{jkl} (x_{jl} - \sum_i y_{ijkl}) + \sum_{ikl} g_{ikl} (x_{jl} - \sum_j y_{ijkl}) \\ &= \sum_{ijkl} (c_{ijkl} - e_{jkl} - g_{ikl}) y_{ijkl} + \sum_{kl} (\sum_j e_{jkl} + \sum_i g_{ikl}) x_{jl} \end{aligned}$$

and obtain the Lagrangean problem

$$\begin{aligned} \mathcal{L}(e, g) = \\ \min \quad & \{\sum_{ijkl} (c_{ijkl} - e_{jkl} - g_{ikl}) y_{ijkl} + \sum_{ij} (\sum_k e_{kij} + \sum_l g_{lij}) x_{ij}\} \\ \text{s.t.} \quad & \text{constraints (20), (23), ..., (26).} \end{aligned}$$

Frieze and Yadegar [103] showed that $\max_{e,g} \mathcal{L}(e, g)$ constitutes a lower bound for the QAP which is larger (i.e., better) than all Gilmore–Lawler bounds obtained after applying reduction methods. They propose subgradient algorithms to approximately solve $\max_{e,g} \mathcal{L}(e, g)$. The lower bounds obtained in such a way seem to be sharper than Gilmore–Lawler bounds obtained after applying reductions.

Carraresi and Malucelli [59] as well as Assad and Xu [17] suggested to apply a sequence of admissible transformations for computing a lower bound. Carraresi and Malucelli proposed to modify the coefficients of a QAP (2) iteratively by

$$\begin{aligned} c'_{ijkl} &= c_{ijkl} + \tau_{ijkl} - \alpha_{ijk} - \beta_{ikl} + \theta_{ik}, \quad 1 \leq i, j, k, l \leq n, i \neq j, k \neq l \\ c'_{iikk} &= c_{iikk} + \sum_{j=1}^n \alpha_{ijk} + \sum_{l=1}^n \beta_{ikl} - (n-1)\theta_{ik}, \quad 1 \leq i, k \leq n. \end{aligned}$$

This is obviously an admissible transformation of Type I. Carraresi and Malucelli choose in particular the following update formulae:

$$\tau_{ijkl}^{(t+1)} = c_{ijkl}^{(t)} - c_{jilk}^{(t)}, \quad (49)$$

$$\alpha_{ijk}^{(t+1)} = u_{ijk}^{(t)}, \quad (50)$$

$$\beta_{ikl}^{(t+1)} = v_{ikl}^{(t)}, \quad (51)$$

$$\theta_{ik}^{(t+1)} = \frac{1}{n-1} (c_{ik}^{(t)} + u_i^{(t)} + v_k^{(t)}), \quad (52)$$

for all $1 \leq i, j, k, l \leq n$; $i \neq j, k \neq l$. Here t is a counting index for the reformulations. $u_{ijk}^{(t)}$, $1 \leq j \leq n$, and $v_{ikl}^{(t)}$, $1 \leq l \leq n$, are the optimal values of the dual variables of the linear assignment problem with cost matrix C^{ik} , for $1 \leq i, k \leq n$. Let $l_{ik}^{(t)}$ be the optimal values of these LAPs, $1 \leq i, k \leq n$. Then $u_i^{(t)}$, $1 \leq i \leq n$, and $v_k^{(t)}$, $1 \leq k \leq n$, are optimal values of the dual variables for the linear assignment problem with cost matrix $L^t = (l_{ik}^t)$ (i.e., the final linear assignment problem solved to compute the Gilmore–Lawler bound of the t th reformulation).

The bound produced with these settings is often denoted by CMB. The reformulation schemes generally produce bounds of good quality. These bounding techniques, however, are quite time-consuming as n^2+1 linear assignment problems have to be solved per iteration. The computation of CMB (as well as the computation of the bounds obtained by applying the reformulation schemes proposed in [17, 58]) involves $O(n^5)$ elementary operations per iteration.

It has been shown in [58] that in the case $c_{ijkl} = c_{jilk}$, for all $1 \leq i, j, k, l \leq n$, the general reformulation scheme cannot produce lower bounds which are better than the optimal value of the continuous relaxation of the mixed integer programming formulation of Frieze and Yadegar.

Hahn and Grant [124, 125] suggested to use the following two operations for the bounding process:

- (R1) Add a constant to all entries of some row (column) of some submatrix $C^{(ij)}$ and either subtract the same constant from another row (column) of the same submatrix or subtract it from the leader in that submatrix.
- (R2) Add a constant to all entries of some row (column) of the $n^2 \times n^2$ matrix (c_{ijkl}) .

Clearly, operations of class R1 are admissible transformations of type I which just redistribute the entries of the submatrices C^{ik} . Operations of class R2 add a constant to the objective function, and hence, they maintain the order of permutations with respect to the corresponding values of the objective function. The goal is to transform C by applying operations of the classes R1 and R2 so as to decrease the objective function by some amount, say R , and to preserve the nonnegativity of entries of the transformed array C' . Then, R is a lower bound for the optimal solution of the given QAP. If, moreover, $\langle C', Y \rangle = 0$, where Y is the Kronecker product $X_\varphi \otimes X_\varphi$ for some permutation matrix X_φ , then R is the optimal value of the original QAP and permutation φ is an optimal solution. The whole process is a repeated computation of Gilmore–Lawler bounds on iteratively transformed problem data. The time complexity of each iteration is basically $O(n^5)$.

A deeper investigation of this bounding procedure reveals that it is an iterative approach in which the dual of some LP relaxation of the original problem is solved and reformulated iteratively (see [140]). Computational results of Hahn et al. [125] show that it is promising to involve the Hahn–Grant bound in branch-and-bound approaches.

6.4 Bounds Based on Linear Programming Relaxations

In Sect. 4 several mixed integer linear programming (MILP) formulations have been proposed for the QAP. Clearly, the optimal solution of the continuous relaxation of a MILP formulation is a lower bound for the optimal value of the corresponding QAP. Moreover, each feasible solution of the dual of this relaxation is also a lower bound. The identification of appropriate continuous relaxations of MILP formulations and the development of solution methods for solving these relaxations or their duals have been important research aspects.

In the context of lower bound computation, two MILP formulations of the QAP play a special role: the formulation of Frieze and Yadegar [103] described in Sect. 4.3 and that of Adams and Johnson [4] described in Sect. 4.4. As already mentioned, Frieze and Yadegar consider a Lagrangean relaxation of their MILP formulation and develop subgradient algorithms to approximately solve the latter. The resulting bounds perform better than the Gilmore–Lawler bound.

Adams and Johnson [4] consider a Lagrangean relaxation $AJ(\alpha)$ of (27) obtained by adding the so-called complementary constraints (28) to the objective function via Lagrangean multipliers α_{ijkl} . Let $\theta(\alpha)$ denote the optimal value of $AJ(\alpha)$. Then $\max_{\alpha} \theta(\alpha)$ equals the optimal value of the continuous relaxation of (27). Adams and Johnson show that for each fixed set of the multipliers α the problem, $AJ(\alpha)$ can be solved efficiently by solving $n^2 + 1$ linear assignment problems. Moreover, they develop an iterative dual ascent procedure to approximately solve the above maximization problem. In each iteration problem $AJ(\alpha)$ is solved to optimality and the optimal value $\theta(\alpha)$ is computed. Clearly, $\theta(\alpha)$ is a lower bound for the considered QAP. Afterwards, the multipliers α_{ijkl} are updated by using the information contained in the dual variables of the linear assignment problems solved during the previous iteration. The algorithm stops after a prespecified number of iterations. Two updating rules for the multipliers are suggested, one of them leading to a nondecreasing sequence of lower bounds $\theta(\alpha)$. In both cases, the time complexity of this bounding procedure is dominated by the solution of $n^2 + 1$ linear assignment problems. The obtained bound is called Adams–Johnson bound (AJB).

The strength of AJB relies on the fact that it generalizes and unifies all Gilmore–Lawler-like bounds (see Sect. 6.2) but not the Hahn–Grant bound (HGB). Adams et al. have shown that $\theta(0)$ equals to the Gilmore–Lawler bound. Reductions as well as the bounds of Carraresi and Malucelli [58] and Assad and Xu [17] equal $\theta(\alpha)$ for special settings of the Lagrangean multipliers α_{ijkl} . Numerical experiments with instances from QAPLIB show that AJB generally outperforms the above-mentioned bounds. However, according to the numerical results reported in [4, 124],

the Hahn–Grant bound (HGB) outperforms AJB in terms of quality while having higher computation time requirements.

The theoretical relationship between AJB and HGB has been investigated by Karisch et al. [140]. Both, AJB and HGB, can be obtained as feasible solutions of the dual of the continuous relaxation of the MILP formulation (27) proposed by Adams and Johnson. Karisch et al. propose an iterative algorithm to approximately solve this dual and show that AJB, HGB, and all other Gilmore–Lawler-like bounds can be obtained by applying this algorithm with specific settings for the control parameters. Moreover, the same authors identify a setting of parameters which seems to produce a bound which is competitive with HGB in terms of quality and provides a better time/quality trade-off. This bound denoted by KCCEB seems to be especially suitable for use within branch-and-bound algorithms (see [140] for more details).

Adams and Johnson point out that the continuous relaxation of (27) is highly degenerated, and degeneracy poses a problem for primal approaches. In order to overcome this difficulty, Resende et al. [201] solve this LP by an interior-point approach. The bounds obtained in this way have a very good quality but require rather high computation times. Moreover, memory requirements restrict this approach to problems with $n \leq 30$.

6.5 Lower Bounds Based on Eigenvalues

Bounds based on the eigenvalues of the input matrices F and D of symmetric Koopmans–Beckmann problems were introduced by Finke et al. [98]. The starting point is the trace formulation (11) of the Koopmans–Beckmann QAP. When designed and implemented carefully, these techniques produce bounds of good quality in comparison with Gilmore–Lawler-like bounds or, more generally, with bounds based on linear relaxations. However, these bounds are quite expensive in terms of computation time requirements and therefore they are not appropriate for use within branch-and-bound algorithms. Moreover, these bounds deteriorate quickly when lower levels of the branch-and-bound tree are searched, as shown by Karisch et al. [68].

Upon the introduction of the method in [98], many improvements and generalizations have appeared [120–123, 197, 200]. There is a resemblance with the Gilmore–Lawler-based lower bounds in the sense that, based upon a general eigenvalue bound, reduction techniques are applied to the quadratic terms of the objective function in order to improve its quality. In this case, however, the reduction techniques yield a significant improvement which is not really the case with the GLB.

Let F and D be real symmetric matrices. Hence all their eigenvalues are real. We denote the eigenvalues of F by $\lambda_1, \dots, \lambda_n$ and the eigenvalues of D by μ_1, \dots, μ_n . For any permutation matrix X_φ , the matrix $D_\varphi^T = X_\varphi D^T X_\varphi^T$ has the same eigenvalues as D . The matrices F and D admit diagonalizations of the form $F = P\Lambda P^T$ and $D_\varphi^T = QM Q^T$ with orthogonal matrices P and Q and diagonal matrices $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ and $M = \text{diag}(\mu_1, \dots, \mu_n)$. Let p_1, p_2, \dots, p_n

denote the columns of P and q_1, q_2, \dots, q_n the columns of Q . Then

$$F = \sum_{i=1}^n \lambda_i p_i p_i^T \quad \text{and} \quad D_\varphi^T = \sum_{k=1}^n \mu_k q_k q_k^T.$$

Therefore

$$\langle F, D_\varphi \rangle = \text{tr}(FD_\varphi^T) = \text{tr}\left(\sum_{i=1}^n \sum_{k=1}^n \lambda_i \mu_k p_i p_i^T q_k q_k^T\right) = \sum_{i=1}^n \sum_{k=1}^n \lambda_i \mu_k \langle p_i, q_k \rangle^2.$$

Thus, we get with $\lambda = (\lambda_1, \dots, \lambda_n)^T$, $\mu = (\mu_1, \dots, \mu_n)^T$, and matrix $S_\varphi = (\langle p_i, q_k \rangle^2)$

$$\text{tr}(FXDX^T) = \lambda^T S_\varphi \mu. \quad (53)$$

Matrix S_φ is a doubly stochastic matrix. According to Birkhoff's [Theorem 1](#), it can be written as convex combination of permutation matrices. This yields immediately the following bounds on the optimal objective function value of QAP(F, D):

Theorem 5 ([98]) *Let F and D be symmetric $n \times n$ matrices with real entries and eigenvalues $\lambda = (\lambda_1, \dots, \lambda_n)^T$ and $\mu = (\mu_1, \dots, \mu_n)^T$, respectively. Then*

$$\langle \lambda, \mu \rangle^- \leq \text{tr}(FXDX^T) \leq \langle \lambda, \mu \rangle^+. \quad (54)$$

In case of a symmetric Koopmans–Beckmann problem with a linear term $\langle B, X \rangle$, we obtain the lower bound

$$\text{EVB} := \langle \lambda, \mu \rangle^- + \min_{X \in \mathbf{X}_n} \text{tr}(BX^T).$$

The second term is the optimal value of a linear assignment problem. EVB is not a strong bound. It often takes a negative value, even for QAP instances with nonnegative coefficient matrices.

The smaller the interval $[\langle \lambda, \mu \rangle^-, \langle \lambda, \mu \rangle^+]$ is, the closer is $\langle \lambda, \mu \rangle^-$ to $\text{tr}(FXDX^T)$. Thus one can try to transform the given QAP so as to decrease the length of that interval.

Reduction Methods and Bound EV1

One possibility to decrease the length of the interval $[\langle \lambda, \mu \rangle^-, \langle \lambda, \mu \rangle^+]$, and hence to improve EVB, is to decompose the matrices F and D such that some amount will be transferred to the linear term, and the eigenvalues of the matrices resulting in the quadratic term are as uniform in value as possible. The *spread* $sp(F)$ of matrix F is defined by

$$sp(F) := \max \{ |\lambda_i - \lambda_j| : i, j = 1, \dots, n \}.$$

Mirsky [172] showed the following upper bound on the spread:

$$sp(F) \leq m(F) = \left[2 \sum_{i=1}^n \sum_{j=1}^n f_{ij}^2 - \frac{2}{n} (\text{tr} F)^2 \right]^{1/2}. \quad (55)$$

Minimizing $m(F)$ leads to a system of linear equations from which the coefficients of the reduction

$$f_{ij} = \bar{f}_{ij} + e_i + e_j + r_{ij} \quad (56)$$

can be computed explicitly by

$$\begin{aligned} z &= \frac{1}{2(n-1)} \left(\sum_{i=1}^n \sum_{j=1}^n f_{ij} - \sum_{i=1}^n f_{ii} \right), \\ e_i &= \frac{1}{n-2} \left(\sum_{j=1}^n f_{ij} - f_{ii} - z \right), \\ r_{ij} &= \begin{cases} f_{ii} - 2e_i & \text{for } i = j; \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The reduced matrix \bar{F} is again symmetric and has zeroes in the diagonal. Moreover, every row and column sum of \bar{F} equals 0. Matrix D can be reduced in an analogous way:

$$d_{kl} = \bar{d}_{kl} + g_k + g_l + s_{kl}, \quad (57)$$

where the values z , g_k , and s_{kl} can be computed as above. By replacing F and D in (11), we obtain

$$\text{tr}(FXD + B)X^T = \text{tr}(\bar{F}X\bar{D} + \bar{B})X^T,$$

where

$$\bar{b}_{ik} = b_{ik} + f_{ii}d_{kk} + 2e_i \sum_{\substack{l=1 \\ l \neq k}}^n d_{kl}.$$

Let $\bar{\lambda} = (\bar{\lambda}_1, \dots, \bar{\lambda}_n)^T$ and $\bar{\mu} = (\bar{\mu}_1, \dots, \bar{\mu}_n)^T$ be the eigenvalues of matrices \bar{F} and \bar{D} , respectively. By applying EVB to the QAP with the transformed coefficient matrices, we obtain the new eigenvalue bound EVB1

$$\text{EVB1} := \langle \bar{\lambda}, \bar{\mu} \rangle^- + \min_{X \in \mathbf{X}_n} \text{tr} \bar{B} X^T.$$

If we restrict ourselves to purely quadratic, symmetric QAPs, that is, $f_{ii} = d_{kk} = 0$, for all i and k , and $B = 0$, matrix \bar{B} in the above decomposition becomes $\bar{B} = cw^T$, where $c = 2(e_1, \dots, e_n)^T$ and $w = (\sum_l d_{1l}, \dots, \sum_l d_{nl})^T$. Therefore, $\min_{X \in \mathbf{X}_n} \text{tr}(\bar{B}X^T) = \langle c, w \rangle^-$, and

$$\text{EVB1} = \langle \bar{\lambda}, \bar{\mu} \rangle^- + \langle c, w \rangle^- \leq \min_{X \in \mathbf{X}_n} \text{tr}(\bar{F} X \bar{D} + \bar{B}) X^T.$$

Rendl [197] suggested a further improvement of this bound by ranking scalar products of the form $\langle c, Xw \rangle$ for certain permutation matrices X . Moreover, Rendl [197] defines a ratio ρ called the *degree of linearity* based on the ranges of the quadratic and linear terms that compose the lower bound

$$\rho := \frac{\langle \bar{\lambda}, \bar{\mu} \rangle^+ - \langle \bar{\lambda}, \bar{\mu} \rangle^-}{\langle c, w \rangle^+ - \langle c, w \rangle^-}.$$

The influence of the linear term on the lower bound is inversely proportional to the value of ρ . A *small* value of ρ suggests that the ranking procedure mentioned above would be beneficial for the improvement of EVB1. For large values of ρ , we can expect that the quadratic term dominates the linear term in the objective function. In this case Finke et al. [98] suggest the following improvement of EVB1. Consider **Theorem 5** applied to the reduced matrices \bar{F} and \bar{D} and denote the elements of matrix S_φ by s_{ik} , $s_{ik} = \langle p_i, q_k \rangle^2$. It is easy to see that $l_{ik} \leq s_{ik} \leq u_{ik}$, where

$$u_{ik} = \max\{(\langle p_i, q_k \rangle^-)^2, (\langle p_i, q_k \rangle^+)^2\},$$

and

$$l_{ij} = \begin{cases} 0, & \text{if } \langle p_i, q_k \rangle^- \cdot \langle p_i, q_k \rangle^+ < 0, \\ \min\{(\langle p_i, q_k \rangle^-)^2, (\langle p_i, q_k \rangle^+)^2\}, & \text{otherwise.} \end{cases}$$

Recalling the fact that s_{ik} are elements of a doubly stochastic matrix, we can then form the *capacitated transportation problem* (CTP)

$$\begin{aligned} \text{CTP}^* = \min & \sum_{i=1}^n \sum_{k=1}^n \bar{\lambda}_i \bar{\mu}_k s_{ik} \\ \text{s.t.} & \sum_{i=1}^n s_{ik} = 1, \quad k = 1, \dots, n, \\ & \sum_{k=1}^n s_{ik} = 1, \quad i = 1, \dots, n, \\ & l_{ik} \leq s_{ik} \leq u_{ik}. \end{aligned}$$

Then, a new lower bound would be

$$\text{EVB2} = \text{CTP}^* + \langle c, w \rangle^-.$$

Other Eigenvalue-Related Bounds

Rendl and Wolkowicz [200] derive a new lower bound similar to EVB2. Notice that the decomposition scheme in (56) and (57) is uniquely determined by the $4n$ -dimensional vector $d := (e^T, g^T, r^T, s^T) \in \mathbb{R}^{4n}$, where $r = (r_{11}, \dots, r_{nn})^T$ and $s = (s_{11}, \dots, s_{nn})^T$. EVB1 is then a function of d . Maximizing this function with respect to d will result in a lower bound with the best possible decomposition with respect to both the linear and the quadratic term. Maximizing EVB1 as a function of d leads to a nonlinear, nonsmooth, nonconcave maximization problem which is hard to solve to optimality. Rendl et al. propose a steepest ascent algorithm to approximately solve this problem (see [200]). The new bound, denoted EVB3, produces the best lower bounds for a number of QAP instances from QAPLIB, with the expense, however, of high computational time requirements.

A more general approach to eigenvalue-based lower bounding techniques was employed by Hadley et al. [121]. Consider the following sets of $n \times n$ matrices, where I is the $n \times n$ identity matrix and $u := (1, \dots, 1)^T$ is the n -dimensional vector of all ones:

$$\begin{aligned}\mathcal{O} &:= \{X : X^T X = I\}, && \text{set of orthogonal matrices.} \\ \mathcal{E} &:= \{X : Xu = X^T u = u\}, && \text{set of matrices with row} \\ &&& \text{and column sums equal to one.} \\ \mathcal{N} &:= \{X : X \geq 0\}, && \text{set of nonnegative matrices.}\end{aligned}\tag{58}$$

It is a well-known result that $\mathbf{X}_n = \mathcal{O} \cap \mathcal{E} \cap \mathcal{N}$, while the set Ω of *doubly stochastic matrices* is given as $\Omega = \mathcal{E} \cap \mathcal{N}$. Moreover, by Birkhoff's theorem [27], we know that Ω is a convex polyhedron with vertex set \mathbf{X}_n , that is, $\Omega = \text{conv}\{X : X \in \mathbf{X}_n\}$. The above characterization of \mathbf{X}_n implies that we get a relaxation of the QAP, if we delete one or two of the matrix sets \mathcal{O} , \mathcal{E} , and \mathcal{N} in the intersection $\mathbf{X}_n = \mathcal{O} \cap \mathcal{E} \cap \mathcal{N}$. Obviously, the relaxation, and therefore the lower bound, will be tighter if only one of the matrix sets is excluded. In relation to [Theorem 5](#), Rendl and Wolkowicz [200] have shown that

$$\begin{aligned}\min_{X \in \mathcal{O}} \text{tr}(FXDX^T) &= \text{tr}(F\Lambda_F \Lambda_D^T D \Lambda_D \Lambda_F^T) = \langle \lambda, \mu \rangle^-, \\ \max_{X \in \mathcal{O}} \text{tr}(FXDX^T) &= \text{tr}(F\Lambda_F \Lambda_D^T D \Lambda_D \Lambda_F^T) = \langle \lambda, \mu \rangle^+,\end{aligned}$$

where Λ_F and Λ_D are matrices whose columns consist of the eigenvectors of F and D , respectively, in the order specified by their minimal (maximal) inner product. In other words, the lower bound on the quadratic part of the QAP as obtained in EVB is derived by relaxing the feasible set to the set of orthogonal matrices.

All eigenvalue bounds discussed above relax the set of permutation matrices to \mathcal{O} . A tighter relaxation was proposed in [120, 122], where the set of permutation matrices was relaxed to $\mathcal{O} \cap \mathcal{E}$. The authors incorporate \mathcal{E} in the objective function by exploiting the fact that the vector of ones u is both a left and right eigenvector

with eigenvalue 1, for any $X \in \mathbf{X}_n$. More specifically, define

$$P := [u/\|u\| : V], \quad \text{where } V^T u = 0, \quad V^T V = I_{n-1}.$$

Then, V is an orthonormal basis for $\{u\}^\perp$, while $Q := VV^T$ is the orthogonal projection on $\{u\}^\perp$. The following characterization of the permutation matrices is given in [122].

Lemma 1 ([120,122]) *Let X be a real $n \times n$ matrix and Y be a real $(n-1) \times (n-1)$ matrix. If*

$$X = P \begin{bmatrix} 1 & 0 \\ 0 & Y \end{bmatrix} P^T, \quad (59)$$

then

$$X \in \mathcal{E}, \quad X \in \mathcal{N} \Leftrightarrow VYV^T \geq -uu^T/\|u\|^2, \quad \text{and} \quad X \in \mathcal{O} \Leftrightarrow Y \in \mathcal{O}_{n-1}.$$

Conversely, if $X \in \mathcal{E}$, there exists a Y such that (59) holds.

Note that the above characterization of permutation matrices preserves the orthogonality and the trace structure of the problem. By substituting $X = -uu^T/\|u\|^2 + VYV^T$ in the trace formulation of the QAP (11) as suggested by (59), we obtain an equivalent *projected problem* (PQAP) of dimension $n-1$ with variable matrix Y . The new lower bound, often called *elimination bound* and denoted by ELI, is obtained by dropping the requirement $VYV^T \geq -uu^T/\|u\|^2$ and simply requiring $Y \in \mathcal{O}_{n-1}$. In this way we derive a lower bound for the quadratic part of the PQAP. The linear part can be solved exactly as a linear assignment problem. Relations between projected eigenvalue bounds for the QAP and bounds found by semidefinite relaxations (see Sect. 6.7) are discussed by Anstreicher [11].

Concluding this section, notice that there is a possibility to apply eigenvalue bounds also to nonsymmetric QAPs, that is, QAPs where both coefficient matrices F and D are nonsymmetric. Hadley [120] and Hadley et al. [123] involve Hermitian matrices instead of orthogonal matrices. Recall that a matrix H is *Hermitian* if H equals to the transposed, conjugate complex matrix H^* . Every Hermitian matrix has real eigenvalues. In the following, let “ i ” denote $\sqrt{-1}$. For any real $n \times n$ matrix A , define

$$A_+ = \frac{1}{2}(A + A^T), \quad A_- = \frac{1}{2}(A - A^T), \quad (60)$$

$$\tilde{A}_+ = \frac{1}{2}(A_+ + iA_-), \quad \tilde{A}_- = \frac{1}{2}(A_+ - iA_-). \quad (61)$$

Note that A_+ is symmetric, A_- is skew symmetric, and \tilde{A}_+ and \tilde{A}_- are Hermitian, that is, $\tilde{A}_+ = \tilde{A}_+^*$ and $\tilde{A}_- = \tilde{A}_-^*$. \tilde{A}_+ is the positive Hermitian part of A and \tilde{A}_- is the negative Hermitian part of A . In particular, \tilde{A}_+ and \tilde{A}_- have only real eigenvalues. Hadley et al. [122] have shown the following:

Proposition 3 *Let F , D , and X be any real $n \times n$ matrices. Then*

$$\text{tr}(FXD^T X^T) = \text{tr}(\tilde{F}_+ X \tilde{D}_+^* X^T). \quad (62)$$

An immediate consequence of this relation are the following eigenvalue bounds for nonsymmetric Koopmans–Beckmann problems:

Proposition 4 *Let a Koopmans–Beckmann problem with real matrices F and D be given. Let $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)^T$ and $\mu = (\mu_1, \mu_2, \dots, \mu_n)^T$ be the vector of the eigenvalues of the Hermitian matrices \tilde{F}_+ and \tilde{D}_+ , respectively. Then the following inequality holds for any permutation φ .*

$$\langle \lambda, \mu \rangle^- \leq \sum_{i=1}^n \sum_{j=1}^n f_{ij} d_{\varphi(i)\varphi(j)} \leq \langle \lambda, \mu \rangle^+. \quad (63)$$

6.6 Improving Bounds by Means of Decompositions

The idea of applying so-called decompositions to improve lower bounds for specially structured QAPs was initially proposed by Chakrapani and Skorin-Kapov [64] and then further elaborated by Karisch and Rendl [141]. The applicability of this approach seems to be restricted to so-called grid QAPs (or *rectilinear QAPs*). This procedure yields a good trade-off between computation time and bound quality.

A grid QAP is a Koopmans–Beckmann QAP with flow matrix F and distance matrix $D = (d_{kl})$ being the distance matrix of a uniform rectangular grid with unit distance 1. If $d_{kl} = d_{km} + d_{ml}$, we say that m is on a shortest path connecting k and l . In this case the triple $u = (k, m, l)$ is called a *shortest path triple*. A shortest path triple $v = (k, m, l)$ for which $d_{km} = d_{ml} = 1$ is called a *shortest triangle*.

We associate a matrix $R_u = (r_{ij}^{(u)})$ to each shortest path triple $u = (k, m, l)$ and a matrix $T_v = (t_{ij}^{(v)})$ to each shortest triangle $v = (k', m', l')$, where R_u and T_v are defined by

$$r_{kl}^{(u)} = r_{lk}^{(u)} = r_{ml}^{(u)} = r_{bm}^{(u)} = 1, r_{km}^{(u)} = r_{mk}^{(u)} = -1,$$

$$t_{k'm'}^{(v)} = t_{l'm'}^{(v)} = t_{m'l'}^{(v)} = t_{l'k'}^{(v)} = t_{k'm'}^{(v)} = t_{m'k'}^{(v)} = 1,$$

$$r_{ij}^{(u)} = 0, \text{ and } t_{ij}^{(v)} = 0 \text{ if } \{i, j\} \not\subseteq \{k, l, m\}.$$

The set of all shortest path triples is denoted by \mathcal{R} and the set of all shortest triangles is denoted by \mathcal{T} .

The key observation is that, for each $R_u \in \mathcal{R}$ and for each $T_v \in \mathcal{T}$, the identity permutation is an optimal solution of $\text{QAP}(R_u, D)$ and $\text{QAP}(T_v, D)$. The optimal values for these QAPs are 0 and 8, respectively, and these simple QAPs can be used to improve the quality of lower bounds for an arbitrary grid QAP. Let us decompose the flow matrix F as

$$F = \sum_{u \in \mathcal{R}} \alpha_u R_u + \sum_{v \in \mathcal{T}} \beta_v T_v + F_r, \quad (64)$$

where F_r is the *residual* matrix given as

$$F_r := F - \sum_{u \in \mathcal{R}} \alpha_u R_u - \sum_{v \in \mathcal{T}} \beta_v T_v.$$

For every choice of the parameters $\alpha_u \geq 0$, $u \in \mathcal{R}$, and $\beta_v \geq 0$, $v \in \mathcal{T}$, and for any permutation φ , we have

$$Z(F, D, \varphi) = \sum_{u \in \mathcal{R}} \alpha_u Z(R_u, D, \varphi) + \sum_{v \in \mathcal{T}} \beta_v Z(T_v, D, \varphi) + Z(F_r, D, \varphi). \quad (65)$$

This implies

$$\begin{aligned} \min_{\varphi} Z(F, D, \varphi) &\geq 8 \sum_{v \in \mathcal{T}} \beta_v + \min_{\varphi} Z(F_r, D, \varphi) \\ &\geq 8 \sum_{v \in \mathcal{T}} \beta_v + \text{LB}(F_r, D), \end{aligned}$$

where $\text{LB}(F_r, D)$ is any lower bound for the QAP with flow matrix F_r and distance matrix D . Clearly, the expression on the right-hand side of (65) is a lower bound for the original QAP. This lower bound, which depends on the vectors $\alpha = (\alpha_u)$, $\beta = (\beta_v)$, is denoted by $h(\alpha, \beta)$. Hence, $h(0, 0)$ equals $\text{LB}(F, D)$, and therefore,

$$\max_{\alpha \geq 0, \beta \geq 0} h(\alpha, \beta) \geq \text{LB}(F, D).$$

Thus $\max_{\alpha \geq 0, \beta \geq 0} h(\alpha, \beta)$ improves the bound $\text{LB}(\text{QAP}(F, D))$.

Chakrapani et al. [64] improve the Gilmore–Lawler bound (GLB), and the elimination bound (ELI), by using only the matrices R_u , $u \in \mathcal{R}$, for the decomposition. Karisch et al. [141] use the decomposition scheme (64) to improve the elimination bound (ELI) (as introduced by Hadley et al. [122]).

Palubeckis [181] used a decomposition into triangles to generate QAPs with known objective function values. Such QAPs are important for testing heuristics. Further improved test generators are suggested in Palubeckis [182–184], where simple graphs other than triangles are used in the decomposition. Cyganski et al. [74]

observed that QAP instances generated by Palubeckis' test example generator are “easy” in the sense that their optimal value can be computed in polynomial time by solving a linear program (see also Cela [60]). Notice, however, that nothing is known about the computational complexity of QAP instances generated by Palubeckis' generator. We believe that *finding an optimal solution* for these QAPs is NP-hard, although the corresponding decision problem is polynomially solvable.

6.7 Bounds Based on Semidefinite Relaxations

Semidefinite programming (SDP) is a generalization of linear programming where the variables are taken from the Euclidean space of matrices with the trace operator acting as an inner product. The nonnegativity constraints are replaced by semidefiniteness constraints and the linear constraints are formulated in terms of linear operators on the space of matrices. Semidefinite programming has been proven to be a useful tool in discrete optimization; see, for example, Goemans and Williamson [116] and Lovász and Schrijver [161].

Semidefinite programming relaxations for the QAP were considered by Karisch [139], Zhao [226], and Zhao et al. [229]; Rendl and Sotirov [199]; Roupin [207], Peng et al. [191]; and others. The SDP relaxations considered in these papers are solved by interior-point methods or cutting-plane methods. The obtained solutions provide lower bounds for the QAP. Anstreicher and Brixius [13] developed a powerful bounding technique which is based on convexity arguments for quadratic programs. In terms of quality, the bounds obtained by semidefinite programming and convex programming are competitive with the best existing lower bounds for the QAP. We refer to [139, 229] for a detailed description of SDP approaches to the QAP and illustrate the idea by describing just two semidefinite programming relaxations for the QAP.

According to (13), a Koopmans–Beckmann problem can be formulated as

$$\begin{aligned} \min \quad & x^T(D \otimes F)x \\ \text{s.t.} \quad & X \in \mathbf{X}_n, \end{aligned} \tag{66}$$

where $x = \text{vec}(X)$. Since the objective function is equivalent to

$$\text{tr}((D \otimes F)xx^T),$$

we get the equivalent formulation

$$\begin{aligned} \min \quad & \text{tr}((D \otimes F)Y) \\ \text{s.t.} \quad & Y = xx^T \\ & X \in \mathbf{X}_n. \end{aligned} \tag{67}$$

When we want to linearize this problem, we have to choose Y in the convex hull P of the $(n^2 \times n^2)$ matrices xx^T where $x = \text{vec}(X)$, X being a permutation matrix:

$$P = \text{conv}\{xx^T : x = \text{vec}(X), X \in \mathbf{X}_n\}.$$

It can be shown that the matrix $Y - yy^T$ is positive semidefinite for $Y \in P$ and $y = \text{diag}(Y)$. This leads immediately to a semidefinite relaxation of QAP(A, B). Zhao et al. [229] showed that the semidefiniteness constraint

$$Y - yy^T \succeq 0$$

can be strengthened by imposing the orthogonality constraints

$$XX^T = X^TX = I,$$

on X , where I is the $n \times n$ identity matrix. These constraints are quadratic in X but linear in Y and use $O(n^4)$ variables and constraints. One can exploit that all row and column sums of a permutation matrix X are equal to 1. This leads to a bound similar to the projection bound shown in Lemma 1. Further improvements are possible by adding the symmetry relations

$$x_{ik}x_{jl} = x_{jl}x_{ik} \text{ for } i, j, k, l = 1, \dots, n$$

and the pattern constraints

$$\begin{aligned} x_{ik}x_{il} &= 0 \text{ for } i, k, l = 1, \dots, n, k \neq l, \\ x_{ik}x_{jk} &= 0 \text{ for } i, j, k = 1, \dots, n, i \neq j. \end{aligned}$$

Interior-point methods are not well suited for solving the above semidefinite program with nonnegativity constraints $y_{ijkl} \geq 0$ and pattern constraints. Therefore, Rendl and Sotirov [199] developed a *bundle method* based on subgradient optimization for handling these additional constraints and obtained good results. Other approaches for solving the semidefinite relaxation of the QAP are a lift-and-project method, suggested by Burer and Vandenbussche [35], as well as a cutting-plane algorithm of Faye and Roupin [96]. Recently, de Klerk and Sotirov [78] describe how symmetries in the problem data of QAPs can be exploited within semidefinite programming relaxations.

In order to get a low-dimensional semidefinite relaxation, Ding and Wolkowicz [84] replace the constraint $Y = xx^T$ by $Y \succeq XX^T$ which is equivalent to

$$\begin{pmatrix} I & X^T \\ X & Y \end{pmatrix} \succeq 0.$$

Moreover, they remove D from the objective function by using the constraint $R = XD$. This results in the following significantly smaller semidefinite programming relaxation for QAP(F, D, B) which has only $O(n^2)$ variables and constraints

$$\begin{aligned} \min \quad & \text{tr}FY + \text{tr}BX^T \\ \text{s.t.} \quad & R = XD \\ & \begin{pmatrix} I & X^T & R^T \\ X & I & Y \\ R & Y & Z \end{pmatrix} \succeq 0 \\ & X \in \mathbf{X}_n \\ & Y, Z \text{ symmetric } (n \times n) \text{ matrices} \\ & R \text{ } (n \times n) \text{ matrix.} \end{aligned} \tag{68}$$

Since X is a permutation matrix, we get the additional constraints

$$\begin{aligned} \text{diag}(Y) &= X\text{diag}(D) \\ \text{diag}(Z) &= X\text{diag}(D^2) \\ Ye &= XDe \\ Ze &= XD^2e \end{aligned} \tag{69}$$

where $e = (1, 1, \dots, 1)^T$. The bound based on relaxation (68) and (69) is provably better than the projected eigenvalue bound of Hadley et al. [122] (based on Lemma 1) and competitive with the quadratic programming bound by Anstreicher and Brixius [13] (see below).

By taking advantage of the special structure of distance matrices stemming from Hamming distances or from the l_1 -norm, Mittelmann and Peng [174] arrive at semidefinite programming relaxations which have a smaller size compared to standard SDP relaxations. QAPs with such distance matrices occur in location theory and in communication systems.

In a later paper, Peng et al. [191] generalize the approach of [174]. A matrix pair (D^+, D^-) is called a *positive semidefinite splitting* (PSD splitting) of matrix D , if

$$D = D^+ - D^-, \quad D^+, D^- \succeq 0.$$

If a PSD splitting of matrix D is available, then the QAP can be relaxed to

$$\min \quad \text{tr}F(Y^+ - Y^-) + \text{tr}BX^T \tag{70}$$

$$\text{s.t.} \quad \text{constraints on } Y^+, Y^-, \text{ and } X. \tag{71}$$

Here, $Y^+ = XD^+X^\top$ and $Y^- = XD^-X^\top$ is a PSD splitting of XDX^\top . Peng et al. introduce two PSD splitting schemes, one based on a singular value decomposition of matrix D and the other based on the Laplace operator of a graph. Suppose D is the adjacency matrix of a complete, connected graph G . Let $\delta_i = \sum_{j=1}^n d_{ij}$ and $\Delta = \text{diag}(\delta_1, \dots, \delta_n)$. The *Laplace matrix* $L(G)$ of graph G is defined by $L(G) = \Delta - D$. This leads to the decomposition $D^+ = \Delta$ and $D^- = L(G)$. Numerical experiments on a large number of QAP show that these bounds are comparable to the strongest bounds known for the QAP. In particular, Wu et al. [225] use the decomposition approach based on the singular value decomposition for computing strong lower bounds for the index assignment problem (cf. [227]) in coding theory.

6.8 Bounds Based on Quadratic Programming and Conic Representations

Anstreicher and Brixius [13] developed another powerful approach which leads to a convex quadratic program. It is based on the identity

$$0 = \text{tr}(S(I - XX^\top)) = \text{tr}(S) - \text{tr}((I \otimes S)xx^\top)$$

which holds for any orthogonal matrix X and any symmetric matrix S . This identity yields, for any orthogonal matrix X and any symmetric matrices S and T ,

$$\begin{aligned} \text{tr}(FXD^\top X^\top) &= \text{tr}(S) + \text{tr}(T) + \text{tr}((D \otimes F - I \otimes S - T \otimes I)xx^\top) \\ &= \text{tr}(FXD^\top X^\top) = \text{tr}(S) + \text{tr}(T) + x^\top Qx \end{aligned} \quad (72)$$

with $Q = (D \otimes F - I \otimes S - T \otimes I)$. If S and T are fixed, the right-hand side of (72) becomes a convex, quadratic program in x .

Using diagonalizations of the symmetric flow matrix F and distance matrix D , Anstreicher and Brixius [13] showed the following result:

Proposition 5 *Let F and D be symmetric $(n \times n)$ matrices and let X be an orthogonal matrix. Then*

$$\begin{aligned} \min_X \text{tr}(FXD^\top X^\top) &= \max \{ \text{tr}(S) + \text{tr}(T) : D \otimes F - I \otimes S - T \otimes I \succeq 0; \\ &\quad S, T \text{ symmetric} \}. \end{aligned} \quad (73)$$

Anstreicher and Brixius used the optimal dual solutions $s = (s_1, s_2, \dots, s_n)^\top$ and $t = (t_1, t_2, \dots, t_n)^\top$ of (73) which correspond to the diagonal entries of the optimal symmetric matrices S and T in the program on the right-hand side of (73) and obtained

$$\sum_{i=1}^n s_i + \sum_{i=1}^n t_i + \min \{ x^\top Q : X \text{ doubly stochastic} \} \quad (74)$$

as new bound for the optimal objective function value of $\text{QAP}(F, D)$. Anstreicher et al. [14] applied this bound in a parallel branch-and-bound algorithm for the QAP which showed an excellent performance: several up to the time unsolved QAP instances were solved to optimality.

Povh and Rendl [193] investigate various conic representations of the QAP. They show that the QAP can be formulated as a linear program over the cone of completely positive matrices of order $O(n^2)$. A copositive program itself is currently computationally intractable, but the paper opens new possibilities for solving QAPs approximately.

7 Exact Solution Methods

An exact algorithm for a combinatorial optimization problem provides the global optimal solution to the problem. In this section we briefly discuss several exact algorithms that have been used for solving the QAP, like branch-and-bound, Benders' decomposition, and branch-and-cut algorithms. But also methods from global optimization have successfully been applied to the QAP; see Enkhbat and Javzandulam [93].

7.1 Branch and Bound

Branch-and-bound algorithms have been applied successfully to many hard combinatorial optimization problems, and they appear to be the most efficient exact algorithms for solving the QAP.

The basic ingredients of branch-and-bound algorithms are the *bounds*, the *branching rule*, and the *selection rule*. For a long time, all branch-and-bound algorithms used primarily variations of the Gilmore–Lawler bound, as it is simple to employ and other bounds are just too expensive in terms of computation time to be used in branch-and-bound algorithms. But meanwhile also other bounding strategies showed their strength: Pardalos et al. [189] employed the variance reduction bound with success within a branch-and-bound algorithm. Anstreicher et al. [14] used successfully the convex quadratic programming bound (74) in a parallel branch-and-bound algorithm. Moreover, Adams et al. [3] obtained very good results with lower bounds based on the level-2 reformulation–linearization technique (see Sect. 4.5).

Among branching strategies, the *single assignment branching* introduced by Gilmore [110] and Lawler [152] is widely used for the QAP. It assigns a facility to a location in each branching step, that is, each problem is divided into subproblems by fixing the location of one of the facilities which are not assigned yet. Several rules have been proposed for choosing the facility-location pair in the next level of the search tree. The appropriate rule usually depends on the bounding technique. If the Gilmore–Lawler bound is employed, the new facility-location pair is frequently determined by the reduced costs of the last linear assignment problem solved [23, 38].

The pair assignment algorithms, suggested by Gavett and Plyter [107], Land [150], and Nugent et al. [179], assign a pair of facilities to a pair of locations at a branching step. In relative positioning algorithms (see Mirchandani and Obata [171]), the fixed assignments within each subproblem are determined in terms of distances between facilities, that is, their relative positions. Numerical results show that pair assignment and relative positioning algorithms are outperformed by single-assignment algorithms.

Roucairol [206] developed the so-called polytomic or k -partite branching rule. The search tree produced by this algorithm is not binary as in most of the other approaches. The branching rule is based on the solution φ of the last linear assignment problem solved to compute the lower bound at the current node of the search tree. Let $\mathbf{X}_n^{(i)}$ be the set of permutation matrices with $x_{i\varphi(i)} = 1$. Further, $\bar{\mathbf{X}}_n^{(i)}$ is the set of permutation matrices with $x_{i\varphi(i)} = 0$. The current node is branched into $n + 1$ new nodes which correspond to the feasible solutions of $\mathbf{X}_n^{(1)}, \mathbf{X}_n^{(1)} \cap \bar{\mathbf{X}}_n^{(2)}, \dots, \mathbf{X}_n^{(1)} \cap \mathbf{X}_n^{(2)} \cap \dots \cap \mathbf{X}_n^{(n-1)} \cap \bar{\mathbf{X}}_n^{(n)}$ and $\mathbf{X}_n^{(1)} \cap \mathbf{X}_n^{(2)} \cap \dots \cap \mathbf{X}_n^{(n)}$.

Another polytomic branching strategy used in successful recent algorithms is due to Mautor and Roucairol [166]. In this strategy either one facility is assigned to all the available locations (*row branching*) or all available facilities are assigned to one location (*column branching*).

Another issue in the implementation of branch-and-bound algorithms concerns the so-called selection rule which determines the choice of the subproblem to be branched next. Several strategies, ranging from problem-independent depth or breadth-first search to instance-dependent criteria related to the maximization of lower bounds or reduced costs, have been tested by different authors. There seems to be no clear winner among the tested strategies.

The best results on solving large-size problems have been achieved with parallel and massively parallel implementations; see Pardalos and Crouse [186]; Bruengger et al. [33]; and Clausen and Perregaard [69]. In 2002, Anstreicher et al. [14] built up a network of over 2500 CPUs around the globe. They solved the test example nug30 in QAPLIB [52], a QAP with $n = 30$, in 7 CPU years. In 2007, Adams et al. [3] solved this problem with the level-2 reformulation–linearization technique in 2.5 CPU years. (The total CPU times were normalized to times on a single HP9000 C3000 workstation).

7.2 Benders' Decomposition Methods

Several authors, like Bazaraa and Sherali [24, 25], Balas and Mazzola [18, 19], as well as Kaufman and Broeckx [143], used Benders' decomposition for solving the mixed integer linear programs arising from a linearization of the QAP. The MILP formulation is decomposed into a master problem and a subproblem, called *slave problem*. The master problem contains the original assignment variables and constraints. For a fixed assignment, the slave problem is usually a linear program and hence, solvable in polynomial time. The master problem is a linear program

formulated in terms of the original assignment variables and of the dual variables of the slave problem. It is solvable in polynomial time for fixed values of those dual variables. The algorithms work typically as follows. First, a heuristic is applied to generate a starting assignment. Then the slave problem is solved for fixed values of the assignment variables implied by that assignment, and optimal values of the primal and dual variables of the slave problem are computed. If the dual solution of the slave problem satisfies all constraints of the master problem, we have an optimal solution for the original MILP formulation of the QAP. Otherwise, at least one of the constraints of the master problem is violated. In this case, the master problem is solved with fixed values for the dual variables of the slave problem and the obtained solution is given as input to the slave problem. This procedure is repeated until the solution of the slave problem fulfills all constraints of the master problem.

Clearly, any solution of the master problem obtained by fixing the dual variables of the slave problem to feasible values provides a lower bound for the considered QAP. On the other side, the objective function value of the QAP corresponding to any feasible setting of the assignment variables is an upper bound. The algorithm terminates when the lower and the upper bounds coincide.

Generally, the time needed for the convergence of the upper and the lower bounds to a common value is rather large. Hence these methods can only be applied to QAPs with small n . Heuristics derived from cutting plane approaches, however, produce good suboptimal solutions in early stages of the search; see, for example, Burkard and Bönniger [40] and Bazaraa and Sherali [25]. Miranda et al. [170] used Benders' decomposition for QAP instances arising from electronic component placement problems.

7.3 Branch-and-Cut Methods

Branch-and-cut methods are based on a polyhedral description of the underlying problem, in our case, on the QAP polytope (see Sect. 5). First, a linear relaxation of the QAP is solved by linear programming. If the optimal solution of this problem is feasible for the QAP, this solution is also optimal for the QAP. Otherwise, one tries to deduce from the polyhedral structure of the problem a valid inequality, that is, an inequality that is fulfilled by all feasible solutions of the QAP but violated by the current solution. Often, such a valid inequality can be derived from facet-defining inequalities of the QAP polytope. Valid inequalities which are violated by the current solution are added to the linear program. They cut off the current solution from the feasible region. Now the linear program is solved again. As the complete linear description of the QAP polytope is not known, it might happen that no violated valid inequality can be found. In this case the algorithm performs a branching step. The branching steps produce a *search tree* like in branch-and-bound algorithms. Each node of this tree is processed by performing cuts and finally it is branched, if necessary. Upper bounds, selection and branching rules in branch-and-cut algorithms play a similar role as in branch-and-bound algorithms.

Efforts to design branch-and-cut algorithms for the QAP have been made by Padberg and Rijal [180] who tested their algorithm on sparse problems and by Erdoğan and Tansel [95] who designed a branch-and-cut algorithm by means of a flow-based linearization technique. Kaibel [138] computed lower bounds for QAP instances from QAPLIB by a branch-and-cut approach using box inequalities (see Sect. 5.1). A large family of valid inequalities for the QAP was proposed by Blanchard et al. [28].

Fischetti et al. [99] report very favorable results of a branch-and-cut method applied to QAPs with Hamming distances. They use shifted variables as introduced by Zhang et al. [228] and improve the cuts by lifting procedures.

8 Heuristics

Although substantial improvements have been done in the development of exact algorithms for the QAP, problems of dimension $n > 20$ are still not practical to solve by exact methods because of very high computer time requirements. This makes the development of heuristics indispensable. So far there exists no performance guarantee for any of the algorithms developed for the QAP. Many heuristics, however, provide good-quality solutions in a reasonable time. This is a feature of QAPs which can be explained by the behavior of random QAPs; see Sect. 11. Much research has been devoted to the development of powerful heuristics. In the sequel we discuss shortly the following different types of heuristic algorithms:

- Construction methods
- Limited enumeration methods
- Local search methods
- Greedy randomized adaptive search procedures (GRASP)
- Tabu search (TS)
- Simulated annealing (SA)
- Genetic algorithms (GA)
- Ant colony optimization (ACO)

One possibility to evaluate the performance of heuristics and to compare different heuristics is given by QAP instances with known optimal solution. QAP instances with known optimal solution should resemble general QAPs, so they should be hard for heuristics. Two classes of QAPs with known optimal solution have been proposed so far: the generators of Palubeckis [181–184] see (Sect. 6.6) and the generators proposed by Li and Pardalos [156].

Li and Pardalos start with an instance $\text{QAP}(F, D)$ where D is a constant matrix. This is a trivial QAP, as any permutation is optimal. Now the matrices F and D are iteratively transformed such that the identical permutation id remains an optimal solution. This can be achieved by transformations which keep the property $f_{ij} \geq f_{i'j'}$ if and only if $d_{ij} \leq d'_{i'j'}$, for all i, j, i', j' , at the end of each iteration.

8.1 Construction Methods

Construction methods for the QAP are greedy-type approaches which iteratively assign some facility which has not yet been assigned to some free location. A *partial permutation* contains those facilities which are already assigned and their corresponding locations. Mathematically, a partial permutation can be described as a set of pairs $(i, \varphi(i)) | i \in M \subseteq N$ where all indices $\varphi(i)$ are different. The partial permutation which does not contain any pair is called *empty*. Starting with the empty permutation, a heuristic `heur` chooses iteratively an unassigned facility i and a free location k and adds the pair (i, k) to the current partial permutation until all facilities are assigned.

Gilmore [110] proposed to fix index k in the heuristic `heur` by using the Gilmore–Lawler bound for a QAP on currently unassigned facilities and locations. For a refinement of this argument, see Burkard [38]. In both cases, the time complexity of this heuristic is $O(n^4)$.

Another good construction method which starts with an arbitrary prefixed ordering of the facilities has been proposed by Müller-Merbach [178].

8.2 Limited Enumeration Methods

Limited enumeration methods rely on the observation that enumeration methods (e.g., branch-and-bound algorithms) find often good solutions in early stages of the search and employ then a lot of time to marginally improve that solution or prove its optimality. This behavior of enumeration methods suggests a way to save time in the case that we are interested in a good but not necessarily optimal solution: impose some limit to the enumeration process. This limit could be a time limit or a limit on the number of iterations the algorithm may perform.

Another strategy which serves the same goal is to manipulate the lower bound. This can be done by increasing the lower bound if no improvement in the solution is achieved during a large number of iterations and would yield deeper cuts in the search tree to speed up the process. Clearly, such an approach may cut off the optimal solution. Graves and Whinston [118] use probabilistic bounds (see also Burkard et al. [46]) in the branch-and-bound tree. These are stronger but may also cut off an optimal solution.

When using Benders' decomposition (cf. Sect. 7.2), there is the possibility to consider only a subset of the constraints in the master problem. A heuristic based on this idea was designed by Burkard and Bönniger [40]; see also the monograph [46].

8.3 Local Search Methods

Given a permutation φ_0 of the set $N = \{1, 2, \dots, n\}$, a neighborhood $N(\varphi_0)$ of φ_0 consists of all permutations which are in some sense *close* to φ_0 . A *locally optimal*

solution of the QAP is a permutation $\bar{\varphi}$ such that

$$Z(C, \bar{\varphi}) := \sum_{i=1}^n \sum_{j=1}^n c_{ij\bar{\varphi}(i)\bar{\varphi}(j)} = \min_{\varphi \in N(\varphi_0)} \sum_{i=1}^n \sum_{j=1}^n c_{ij\varphi(i)\varphi(j)}.$$

An algorithm which produces a locally optimal solution is called a *local search* algorithm. Basic ingredients of local search methods are the definition of *neighborhoods* and the *update strategy* for the solutions. Frequently, a local search procedure starts with an initial feasible solution and iteratively tries to improve the current solution by moving to a neighbored solution. This iterative step is repeated until no further improvement can be found. The CRAFT method introduced by Buffa et al. [34] is one of the first examples of an iterative improvement algorithm for the QAP. A comprehensive discussion of theoretical and practical aspects of local search in combinatorial optimization can be found in Aarts and Lenstra [2].

Frequently used neighborhoods for QAPs are the *pair-exchange* (or *2-opt*) neighborhood \mathcal{N}_2 and the *cyclic triple-exchange* neighborhood. Given a permutation φ_0 , the permutations $\varphi \in \mathcal{N}_2$ have the form

$$\varphi(j) = \begin{cases} \varphi_0(k), & \text{for } j = i, i \neq k, \\ \varphi_0(i), & \text{for } j = k, \\ \varphi_0(j), & \text{for } j \neq i, k. \end{cases}$$

They are obtained by swapping the locations of just two facilities. Thus the pair-exchange neighborhood of a permutation $\varphi_0 \in \mathcal{S}_n$ consists of all permutations $\varphi \in \mathcal{S}_n$ obtained from φ_0 by applying some transposition (i, j) to it, that is, $\mathcal{N}_2(\varphi_0) = \{\varphi \circ (i, j) : 1 \leq i, j \leq n, i \neq j\}$. Similarly, triple-exchange neighborhoods are obtained by a cyclic exchange of three locations for some triple of facilities.

A pair-exchange neighborhood has the size $\binom{n}{2}$. Computing the objective function values for all neighbors of a given permutation φ_0 takes $O(n^3)$ time as it takes $O(n)$ time to compute the difference of the objective function values of φ_0 and a permutation φ in the neighborhood of φ_0 . If the neighborhood of φ_0 is already scanned, then its pair-exchange neighborhood can be searched in $O(n^2)$; see Frieze et al. [104].

A Kernighan and Lin-type neighborhood structure for the Koopmans–Beckmann QAP was introduced by Murthy et al. [177]. Consider a permutation $\varphi_0 \in \mathcal{S}_n$. A *swap* (or pair exchange) of φ_0 interchanges the facilities assigned to two locations i and j . A *greedy swap* in permutation φ_0 is a swap which minimizes the difference $Z(F, D, \varphi) - Z(F, D, \varphi_0)$ with respect to all swaps in φ_0 . Let $\varphi_0, \varphi_1, \dots, \varphi_l$ be a set of permutations in \mathcal{S}_n , each of them obtained by a greedy swap of the preceding one. Such a sequence is called *monotone* if for each pair of permutations φ_k, φ_l in the sequence, $\{i_k, j_k\} \cap \{i_l, j_l\} = \emptyset$, where $\varphi_k (\pi_l)$ is obtained by applying transposition $(i_k, j_k) ((i_l, j_l))$ to the preceding permutation in the sequence. The *neighborhood* of φ_0 consists of all permutations which occur in the (unique) maximal monotone sequence obtained by greedy swaps starting with

permutation φ_0 . This neighborhood structure for the QAP is denoted as \mathcal{N}_{KL} . It is not difficult to see that given a QAP(F, D) of size n and a permutation $\varphi \in S_n$, the cardinality of $\mathcal{N}_{KL}(\pi)$ does not exceed $[n/2] + 1$.

A similar but wider neighborhood was proposed by Li et al. [158]. For $k > 3$ the size of k -exchange neighborhoods becomes so large that it is practically impossible to determine explicitly a best solution of the QAP in such a neighborhood. Several implicit enumeration techniques have been developed to overcome this difficulty like the *improvement graph* by Ahuja et al. [7] and the variable neighborhood search by Mladenović and Hansen [175].

Another important ingredient of improvement methods is the *update rule*. In the case of the *first improvement rule* the current solution is updated as soon as the first improving neighbor solution is found. The *best improvement rule* scans first the whole neighborhood and chooses the *best* improving neighbor solution (if such a solution exists at all). *Heider's rule* starts by scanning the neighborhood of the initial solution in a prespecified *cyclic* order. The current solution is updated as soon as an improving neighbor solution is found. The scanning of the neighborhood of the new solution starts there where the scanning of the previous one was interrupted (in the prespecified cyclic order).

In order to get better results, local search algorithms are performed several times starting with different initial solutions.

8.4 Complexity Results on Local Search

A pair (P, \mathcal{N}) , where P is a (combinatorial) optimization problem P and \mathcal{N} is an associated neighborhood structure, defines a *local search problem* which consists in finding a locally optimal solution of P with respect to the neighborhood structure \mathcal{N} . A *polynomial-time local search problem*, shortly *PLS problem*, (cf. Johnson et al. [133]), is a local search problem for which local optimality can be checked in polynomial time. In analogy to decision problems, there exist complete problems in the class of PLS problems. The *PLS-complete* problems are—in the usual sense of complexity—the most difficult among the PLS problems.

In connection to QAPs, the complexity with respect to two neighborhood structures has been investigated, namely, the pair-exchange neighborhood and the Kernighan–Lin-type neighborhood. In both cases the QAP turns out to be PLS-complete.

Schäffer and Yannakakis [209] have proven that the graph partitioning problem (GPP) with a 2-opt neighborhood structure is PLS-complete. A similar PLS-reduction as in Pardalos et al. [187] implies that the local search problem (QAP, \mathcal{N}_2), where \mathcal{N}_2 is the pair-exchange neighborhood, is PLS-complete. This implies that the time complexity of a general local search algorithm for the QAP involving the pair-exchange neighborhood is also exponential in the worst case.

Pardalos et al. [187] have shown that the PLS-complete *GPP* with the neighborhood structure defined by Kernighan and Lin [144] is PLS-reducible to (QAP, \mathcal{N}_{KL}). This implies the following result:

Theorem 6 ([187]) *The local search problem (QAP, \mathcal{N}_{KL}) , where \mathcal{N}_{KL} is the Kernighan–Lin-type neighborhood structure for the QAP, is PLS-complete.*

The PLS-completeness of (QAP, \mathcal{N}_{KL}) implies that, in the worst case, a general local search algorithm involving the Kernighan–Lin-type neighborhood finds a local minimum only after a time which is exponential on the problem size. Numerical results, however, show that such local search algorithms perform quite well when applied to QAP test instances, as reported in [177].

A neighborhood $N(\varphi)$ of a permutation φ of the set $N = \{1, 2, \dots, n\}$ is called *exponential*, if the size $|N(\varphi)|$ is exponential in n . For the traveling salesman problem, several neighborhoods of exponential size are known which can be searched in polynomial time. In the case of QAPs, this is not possible. Deineko and Woeginger [82] showed for several exponential neighborhoods that it is NP-hard to minimize the objective function of a Koopmans–Beckmann QAP over the permutations in such neighborhoods. They conjecture the following:

Conjecture 1 ([82]) *Under the assumption $P \neq NP$, every neighborhood of the QAP which can be searched in polynomial time has a polynomial size.*

8.5 Greedy Randomized Adaptive Search Procedure

The greedy randomized adaptive search procedure (GRASP) was introduced by Feo and Resende [97] and has been applied successfully to different hard combinatorial optimization problems, among them to the QAP [158, 188]. The reader is referred to [97] for a survey and tutorial on GRASP. GRASP is a combination of greedy elements with random search in a two-phase heuristic. It consists of a construction phase and a local improvement phase. In the construction phase good feasible solutions are constructed, whereas in the local improvement phase the neighborhood of these solution is searched for possible improvements.

For the QAP the construction of a first good solution is based on the so-called restricted candidate list (RCL). At the beginning arbitrary, two facilities i and j are assigned to all possible locations. The corresponding cost values

$$C_{ij}^\varphi = f_{ii}d_{\varphi(i)\varphi(i)} + f_{jj}d_{\varphi(j)\varphi(j)} + f_{ij}d_{\varphi(i)\varphi(j)} + f_{ji}d_{\varphi(j)\varphi(i)}$$

are sorted and the k smallest values are included in the initial RCL. In the second step of the construction method, an element of RCL and an index $p \neq i, j$ are randomly chosen. From now on, the indices i and j are fixed. Now the cost increment is evaluated, when facility p is assigned to any location different from $\varphi(i)$ and $\varphi(j)$. The corresponding costs are again sorted and RCL is replaced to contain now the k partial permutations for the facilities i, j , and p with smallest cost. This procedure is continued, until a location for all facilities has been fixed.

Finally, the second phase of the algorithm completes a GRASP iteration by applying an improvement method starting from the solution constructed in the first phase and employing the 2-exchange neighborhood (see also [Sect. 8.3](#)).

8.6 Tabu Search

Tabu search is a local search method introduced by Glover [[112](#), [113](#)] which uses a technique to leave neighborhoods of local optima. The basic idea of tabu search is to “remember” which solutions have already been visited in the course of the algorithm and which might be promising for a further search in their neighborhoods. Thus, the search is driven by the memory and the neighborhood investigation of the current solution. For a comprehensive introduction to tabu search algorithms, the reader is referred to the book by Glover and Laguna [[115](#)].

The main ingredients of tabu search are the *neighborhood structure*, the *moves*, the *tabu list*, and the *aspiration criterion*. A *move* applied to a feasible solution φ generates a neighbor φ' of it. In the case of QAPs, the pair-exchange neighborhood is used and the moves are usually swaps. A *tabu list* is a list of forbidden or *tabu* moves, that is, moves which are not allowed to be applied to the current solution. The tabu status of the moves changes along with the search and the tabu list is updated during the search. When a tabu move fulfills an *aspiration criterion*, its tabu status is cancelled.

A generic tabu search procedure starts with an initial feasible solution φ_0 and selects a *best-quality* solution among (a part of) the neighbors of φ_0 obtained by non-tabu moves. Note that this neighboring solution does not necessarily improve the value of the objective function. Then the current solution is updated, that is, it is substituted by the selected solution. Obviously, this procedure can *cycle*, that is, visit some solution more than once. To avoid this phenomenon, a tabu criterion is introduced in order to identify moves which are expected to lead to cycles. Such moves are declared to be tabu and are added to the tabu list. As forbidding certain moves, however, could prohibit visiting “interesting” solutions, an aspiration criterion distinguishes the potentially interesting moves among the forbidden ones. The search stops when a stop criterion (running time limit, limited number of iterations) is fulfilled.

There is a lot of freedom in the implementation of tabu search algorithms. The performance of tabu search algorithms depends very much on the implementation chosen for its basic ingredients, for example, the tabu list (length and maintenance), the aspiration criterion, and the tabu criteria. There is no general agreement about the best implementation of any of those.

Different tabu search implementations for the QAP have been proposed by Skorin-Kapov [[210](#), [211](#)] who uses a tabu search with a fixed size for the tabu list, by Taillard [[216](#)] where the size of the tabu list is randomly chosen between a maximum and a minimum value (the so-called robust tabu search), and by Battiti and Tecchiolli [[21](#)]. Their *reactive tabu search* involves a mechanism for properly adopting the size of the tabu list: the algorithm notices when a cycle occurs, that is, when a certain solution is revisited, and increases the tabu list size according to the

length of the detected cycle. The numerical results show that generally the reactive tabu search outperforms other tabu search algorithms for the QAP (see [21]).

Tabu search algorithms allow a natural parallel implementation by dividing the burden of the search in the neighborhood among several processors. The first parallel tabu search implementation for the QAP was proposed by Taillard [216]. This approach was later improved by Chakrapani and Skorin-Kapov [63]. In a recent paper, James et al. [132] implement several multi-start strategies and possibilities to diversify the search. They report on a large number of computational experiments and comparisons with other heuristics.

8.7 Simulated Annealing

Simulated annealing is a local search approach which exploits the analogy between combinatorial optimization problems and problems from statistical mechanics. Kirkpatrick et al. [145] and, independently, Černý [62] were the first authors who recognized this analogy and showed how to apply the Metropolis algorithm [168] which simulates the behavior of a physical many-particle system as a heuristic to the TSP.

The analogy between a combinatorial optimization problem and a many-particle physical system basically relies on two facts:

- Feasible solutions of the combinatorial optimization problem correspond to states of the physical system.
- The objective function values corresponds to the energy of the states of the physical system.

In *condensed matter physics*, *annealing* is known as a cooling process which produces low-energy *thermal equilibrium* states of a solid in a heat bath. The aim is to reach the so-called ground state which is characterized by a minimum of energy.

In 1984, Burkard and Rendl [55] showed that a simulated cooling process yields a general heuristic for any combinatorial optimization problem, as soon as a neighborhood structure has been introduced in the set of its feasible solutions. In particular, these authors applied simulated annealing to the QAP. Other simulated annealing (SA) algorithms for the QAP have been proposed by Wilhelm and Ward [223] and Connolly [72]. All these algorithms employ the pair-exchange neighborhood. They differ in the way the *cooling process* or the *thermal equilibrium* is implemented. The numerical experiments show that the performance of SA algorithms strongly depends on the values of the control parameters and particularly on the choice of the cooling schedule.

Simulated annealing (SA) can be modeled mathematically by an inhomogeneous ergodic Markov chain. Under natural conditions on the involved neighborhood structure and on not very restrictive conditions on the slowness of the cooling process, it can be shown that SA asymptotically converges to an optimal solution of the considered problem. The investigation of the speed of this convergence remains an (apparently difficult) open problem. For a detailed discussion on the convergence and other theoretical aspects of simulated annealing, the reader is referred to the books by Aarts and Korst [1] and Laarhoven and Aarts [149].

8.8 Genetic Algorithms

The basic idea of *genetic algorithms* (GA) is to adapt the evolutionary mechanisms acting in the selection process in nature to combinatorial optimization problems. The first genetic algorithm for optimization problems was proposed by Holland [130] in 1975.

A genetic algorithm starts with an initial set of feasible solutions (generated randomly or by using some heuristic) called the *initial population*. The elements of a population are usually termed “individuals.” The algorithm *selects* several pairs of individuals or *parents* from the current population and uses so-called cross-over rules to produce a new feasible solution, called *child*, from each pair of parents. Afterwards, “bad” solutions, that is, solutions yielding too high objective function values, are thrown out of the current population. This process is repeated until a *stop criterion*, for example, a time limit, a limit on the number of iterations, or some measure of convergence, is fulfilled. In course of the algorithm, *mutations* or *immigrations* are periodically applied to the current population to improve the overall quality by *modifying* some of the individuals or *replacing* them by better ones, respectively. Often, local optimization tools are periodically used within GAs resulting in so-called hybrid algorithms. The search is diversified by means of so-called tournaments. A tournament consists of applying several runs of a GA starting from different initial populations and stopping them before they converge. A “better” population is derived as a union of the final populations of these different runs, and then a new run of the GA is started over this population. For a good coverage of theoretical and practical issues on genetic algorithms, the reader is referred to Davis [76] and Goldberg [117].

A number of authors have proposed genetic algorithms for the QAP. The quality of standard algorithms like that by Tate and Smith [217] is rather modest even for QAPs of small or moderate size. Hybrid approaches, however, like combinations of GA techniques with tabu search like the methods by Fleurent and Ferland [100], Misevicius [173], Drezner [87, 88], and Rodríguez et al. [204] yield much better results. Also another hybrid algorithm, the so-called greedy genetic algorithm proposed by Ahuja et al. [8], produces very good results on large-scale QAPs from QAPLIB. It generates the initial population by the GRASP heuristic (see Sect. 8.5). Three crossover strategies are suggested, among them a *path relinking strategy*; see Glover [114]. Finally, Lim et al. [159] suggested a hybrid genetic algorithm using the k -exchange neighborhood.

8.9 Ant Colony Optimization

Ant colony optimization (ACO) is a heuristic for combinatorial optimization which imitates the behavior of ants in search for food. This approach was introduced by Dorigo [86] and Colomi et al. [70] and has already produced good results for the TSP and the QAP [71, 105].

Initially, the ants search for food in the vicinity of their nest in a random way. As soon as an ant finds a source of food, it takes some food from the source and carries

it back to the nest. During this trip back, the ant leaves a trail of a substance called *pheromone* on the ground. The pheromone trail serves to guide the future search of ants towards the already found food source. The intensity of the pheromone on the trail is proportional to the richness of the food source.

To translate this search to a combinatorial optimization problem, we note the following analogies:

- The area searched by the ants resembles the set of feasible solutions of a combinatorial optimization problem.
- The amount of food at food sources resembles the value of the objective function.
- The pheromone trail resembles a component of adaptive memory.

In the case of the QAP, the pheromone trail which is the key element of ACO is implemented by a matrix $T = (\tau_{ij})$. τ_{ij} is a measure for the desirability of locating facility i at location j in the solutions generated by the ants (the algorithm). To illustrate the idea, we briefly describe the algorithm of Gambardella et al. [105].

The algorithm is iterative and constructs a fixed number, say m , of solutions in each iteration. (This number is a control parameter and is also thought of as number of ants.) In the first iteration, these solutions are generated randomly, whereas in the subsequent iteration they are updated by exploiting the information contained in the pheromone trail matrix T . Initially, the pheromone trail matrix is a constant matrix; the constant is inversely proportional to the best value of the objective function found so far. This is in compliance with the behavior of ants whose search directions are initially chosen at random. Let us denote the best solution found so far by φ^* and its corresponding objective function value by $Z(\varphi^*)$. In the further iterations, the entries $\tau_{i\varphi^*(i)}$ of T are increased by the same value which is proportional to $Z(\varphi^*)$. The update of the m solutions in each iteration is done first by means of the pheromone trail matrix and then by applying some improvement method. In both cases the update consists of swapping the locations for a sequence of facility pairs. First, the current solution is updated by swapping the locations of pairs of facilities chosen so as to maximize the (normalized) sum of the corresponding pheromone trail entries. Then, the solution obtained after this update is improved by applying some improvement methods; see Sect. 8.3. As soon as an improvement of the best-known solution is detected, an intensification component “forces the ants” to further explore the part of the solution space where the improvement was found. If after a large number of iterations there is no improvement of the best-known solution, a diversification – which is basically a new random start – is performed.

Tseng and Liang [218] combine ACO with a genetic algorithm. They propose to start the ACO with solutions found by a hybrid genetic algorithm. Another population-based ant algorithm has been described by Ramkumar et al. [195]. A parallel ant colony optimization procedure for the QAP has been proposed by Tsutsui [219].

Numerical results presented in [71, 105] show that ant systems are competitive heuristics especially for real-life instances of the QAP with a few very good clustered solutions. For randomly generated instances which have many good solutions distributed somehow uniformly in the search space, AS are outperformed by other heuristics, for example, genetic algorithms or tabu search approaches.

9 Available Computer Codes for the QAP

Burkard et al. [52] compiled a library of QAP instances (QAPLIB) which is widely used to test bounds, exact algorithms, and heuristics for the QAP. The instances collected in QAPLIB stem from different authors and range from instances arising in real-life applications to instances generated randomly only for test purposes. Among them are the often-tested test instances due to Nugent et al. [179]. QAPLIB is currently maintained by Peter Hahn at the University of Pennsylvania at <http://www.seas.upenn.edu/qplib/>. There is a mirror site at the University of Waterloo at <http://qplib.uwaterloo.ca/>. New benchmark instances for the QAP have been published by Stützle and Fernandes [214].

A number of codes can be downloaded from QAPLIB, namely:

- `qapbb.f` A FORTRAN code for the branch-and-bound algorithm described by Burkard and Derigs [47]. Currently the code is dimensioned to handle problems with $n \leq 30$.
- `qapglb.f` A FORTRAN code from [47] to compute the Gilmore–Lawler bound (see Sect. 6.2) for problems with $n \leq 256$.
- `qapeli.f` A FORTRAN code for the elimination bound developed by Karisch and Rendl [141] for symmetric QAPs with $n \leq 256$ which fulfill the triangle inequality.
- GRASP The user is redirected to the homepage of Mauricio Resende, where FORTRAN codes for the GRASP heuristic (see Sect. 8.5) for dense and sparse QAPs can be found.
- Ro-TS A PASCAL implementation of the robust tabu search (see Sect. 8.6) by Taillard [216]. Dimensioned for $n \leq 150$.
- `qapsim.f` A FORTRAN code for the simulated annealing algorithm (cf. Sect. 8.7) by Burkard and Rendl [55]. Dimensioned for $n \leq 256$.
- SA A C-implementation due to Taillard for the simulated annealing algorithm of Connally [72].
- FANT A C++ code by Taillard for the ACO approach (see Sect. 8.9). Applicable to QAPs with $n \leq 51$.

10 Polynomially Solvable Cases

Since the QAP is NP-hard, restricted versions which can be solved in polynomial time are an interesting aspect of the problem. A basic question arising with respect to polynomially solvable versions is the identification of those versions and the investigation of the border line between hard and easy versions of the problem. There are two ways to approach this topic: first, certain structural conditions imposed on the coefficient matrices of the QAP lead to polynomially solvable QAPs. Second, certain graph problems formulated as QAPs lead to polynomially solvable versions of QAPs. These two approaches yield two groups of restricted QAPs which

are briefly reviewed in this section. For a detailed information on this topic, the reader is referred to the monographs of Burkard et al. [46] and Çela [60].

If in a Koopmans–Beckmann problem QAP(F, D), matrix F is symmetric and matrix D is skew symmetric, then all solutions φ yield the same objective function value 0. A QAP with the property that all feasible solutions yield the same value is called *constant QAP*. Another example for a constant QAP is given by QAP(F, D) with a sum matrix F and a circulant matrix D . A *sum matrix* $A = (a_{ij})$ is generated by two vectors $u = (u_1, \dots, u_n)$ and $v = (v_1, \dots, v_n)$ such that $a_{ij} = u_i + v_j$. An $n \times n$ matrix $D = (d_{ij})$ is called a *circulant matrix* if there exist numbers $c_{-n+1}, \dots, c_{-1}, c_0, c_1, \dots, c_{n-1}$ with $c_i = c_{n-i}$, for $0 \leq i \leq n - 1$, such that $d_{ij} = c_{j-i}$, for all i, j .

A Koopmans–Beckmann problem QAP(F, D) where F is a sum matrix and either F or D is symmetric or skew symmetric can be reduced to a special linear assignment problem, whose optimal solution can be found in $O(n \log n)$ time by applying [Proposition 2](#). This time requirement, however, is majorized by the time $O(n^2)$ needed to compute the coefficients of the linear assignment problem.

A product matrix $A = (a_{ij})$ is generated by two vectors $u = (u_1, \dots, u_n)$ and $v = (v_1, \dots, v_n)$ such that $a_{ij} = u_i \cdot v_j$. A QAP(F, D) where F and D are symmetric product matrices generated by vectors u and v whose components have all the same sign can be solved in $O(n \log n)$ time. A matrix $A = (a_{ij})$ is a *chessboard matrix* if the entries are given by $a_{ij} = (-1)^{i+j}$. The Koopmans–Beckmann problem where F and D are chessboard matrices can be solved in polynomial time. The problem, however, where F is a symmetric product matrix and D is a chessboard matrix is NP-hard.

Product matrices play a role in the so-called Wiener QAP; see Çela et al. [61]. The *Wiener index* of a connected, undirected graph G is the sum of all shortest distances between all pairs of vertices of G . This index was introduced by the chemist Harold Wiener [222] as it characterizes structural properties of saturated hydrocarbons. For a survey on theoretical and practical aspects of the Wiener index, see Dobrynin et al. [85]. Çela et al. showed that finding a tree with prescribed degree sequence and maximum Wiener index can be accomplished in $O(n^2)$ time by maximizing the objective function of a special QAP(F, D), where F is a product matrix and D is the distance matrix of points on a line. Here, n denotes the vertices in the degree sequence with a degree > 1 .

A matrix $A = (a_{ij})$ is called *small* if it is generated by two vectors $u = (u_1, \dots, u_n)$ and $v = (v_1, \dots, v_n)$ such that $a_{ij} = \min(u_i, v_j)$. It is called *large* if $a_{ij} = \max(u_i, v_j)$. A Koopmans–Beckmann QAP where F is a symmetric small or large matrix and D is a chessboard matrix can be solved in $O(n^2)$ time.

Sum, product, and small and large matrices are special cases of so-called Monge matrices. A matrix $A = (a_{ij})$ is a *Monge matrix* if the following inequalities are fulfilled for each 4-tuples of indices i, j, k, l , $i < k, j < l$:

$$a_{ij} + a_{kl} \leq a_{il} + a_{kj}, \quad (\text{Monge inequalities}).$$

A matrix $A = (a_{ij})$ is an *anti-Monge matrix* if the following inequalities are fulfilled for each 4-tuples of indices $i, j, k, l, i < k, j < l$:

$$a_{ij} + a_{kl} \geq a_{il} + a_{kj}, \quad (\text{anti-Monge inequalities}).$$

Another class of matrices similar to the Monge matrices are the *Kalmanson matrices*. A matrix $A = (a_{ij})$ is a Kalmanson matrix if it is symmetric and if its elements satisfy the following inequalities for all indices $i, j, k, l, i < j < k < l$:

$$a_{ij} + a_{kl} \leq a_{ik} + a_{jl}, \quad a_{il} + a_{jk} \leq a_{ik} + a_{jl}.$$

For more information on Monge, anti-Monge, and Kalmanson matrices and their properties, the reader is referred to the survey article of Burkard et al. [53].

Whereas the TSP with a Monge matrix as distance matrix is polynomially solvable, it turns out that the QAP where both coefficient matrices are Monge or anti-Monge is NP-hard. The complexity of a QAP with one coefficient matrix being Monge and the other one being anti-Monge is still open; see Burkard et al. [42]; Burkard et al. [46]; and Cela [60]. An interesting case is a QAP(F, D), where the $n \times n$ matrix F is a Monge matrix and D is a chessboard matrix. If $n = 2m$ is even, then the fixed permutation $\varphi = (1, m+1, 2, m+2, \dots, n)$ is an optimal solution. In the case of an odd n , the problem may have several optimal solutions and it is conjectured to be NP-hard.

Other restricted versions of the QAP involve matrices with a specific diagonal structure. An $n \times n$ matrix $A = (a_{ij})$ is called a *Toeplitz matrix* if there exist numbers $c_{-n+1}, \dots, c_{-1}, c_0, c_1, \dots, c_{n-1}$ such that $a_{ij} = c_{j-i}$, for all i, j . A circulant matrix is a special Toeplitz matrix where the generating numbers c_i fulfill the conditions $c_i = c_{n-i}$. A Toeplitz matrix has constant entries along lines parallel to the diagonal, whereas a circulant is given by its first row and the entries of the i th row resemble the first row shifted by $i - 1$ places to the right.

A well-studied problem is the so-called anti-Monge–Toeplitz QAP where the rows and columns of the anti-Monge matrix are nondecreasing; see Burkard et al. [45]. It has been shown that this problem is NP-hard and contains as a special case the so-called turbine problem introduced by Bolotnikov [31] and Stoyan et al. [213]. Laporte and Mercure [151] observed that the turbine problem can be formulated as a quadratic assignment problem; see also Mosevich [176]. In the turbine problem, we are given a number of blades to be welded in regular spacing around the cylinder of the turbine. Due to inaccuracies in the manufacturing process, the weights of the blades differ slightly and consequently the gravity center of the system does not lie on the rotation axis of the cylinder, leading to instabilities. In an effort to make the system as stable as possible, it is desirable to locate the blades so as to minimize the distance between the center of gravity and the rotation axis. The mathematical formulation of this problem leads to an NP-hard anti-Monge–Toeplitz QAP. (For more details and for a proof of NP-hardness, see Burkard et al. [45].) The *maximization* version of this problem, however, is polynomially solvable. Further polynomially solvable special cases of the anti-Monge–Toeplitz

QAP arise if additional constraints, for example, *benevolence* or *k-benevolence*, are imposed on the Toeplitz matrix. These conditions are expressed in terms of properties on the coefficients $c_{-1}, c_0, c_1, \dots, c_{n-1}$; see Burkard et al. [45]. In this case the cyclic permutation

$$\langle 1, 3, 5, \dots, n, \dots, 6, 4, 2 \rangle \quad (75)$$

becomes the optimal solution. Demidenko et al. [80] specify other conditions on the matrices F and D such that permutation (75) becomes an optimal solution of $\text{QAP}(F, D)$. Further results in this direction are due to Demidenko and Dolgui [79].

The polynomially solvable QAPs where F is an anti-Monge (Monge) matrix and D is a Toeplitz (or circulant) matrix with special properties as mentioned above have a permutation with a certain well-defined structure as optimal solution. The techniques used to prove this fact and to identify the optimal permutation is called *reduction to extremal rays*. This technique exploits two facts: first, the involved matrix classes form cones, and second, the objective function of the QAP is linear with respect to each of the coefficient matrices. These two facts allow us to restrict the investigations to instances of the QAP with 0–1 coefficient matrices which are extremal rays of the above-mentioned cones. Such instances can then be handled by elementary means (exchange arguments, bounding techniques) more easily than the general given QAP.

The Koopmans–Beckmann QAP where the coefficient matrix F is a Kalmanson matrix and matrix D is a symmetric decreasing circulant matrix has been investigated by Deineko and Woeginger [81]. A symmetric $n \times n$ circulant matrix $D = (d_{ij})$ is called *decreasing* if $d_{11} \leq d_{12} \leq \dots \leq d_{1\lceil \frac{n+1}{2} \rceil}$. Deineko and Woeginger investigated the cones generated by these two matrix classes. By reduction to extremal rays, they showed that the identical permutation is an optimal solution.

A matrix $A = (a_{ij})$ is called *left-higher graded* if both its rows and columns are nondecreasing, that is, $a_{ij} \leq a_{i,j+1}$ and $a_{ij} \leq a_{i+1,j}$ for all i, j . A matrix is called *right-lower graded* if both its rows and columns are nonincreasing, that is, $a_{ij} \geq a_{i,j+1}$ and $a_{ij} \geq a_{i+1,j}$ for all i, j . The following proposition generalizes an earlier result by Krushevski [148]:

Proposition 6 *Let F be a left-higher graded matrix and let D be a right-lower graded matrix. Then the identical permutation solves $\text{QAP}(F, D)$.*

The complexity of $\text{QAP}(F, D)$ where F is left-higher graded and D is right-higher graded is not known; see Burkard et al. [43].

Further polynomially solvable cases of $\text{QAP}(F, D)$ arise, when F and D are viewed as weighted adjacency matrices of graphs. For example, Erdogan and Tansel [94] showed the following: let F be the adjacency matrix of a graph whose vertices have only the degrees 0, 1, and 2 and let D be the weighted adjacency matrix of a grid with a rows and b columns such that $ab = n$. In this case the optimal solution of $\text{QAP}(F, D)$ can be found in polynomial time. Moreover, polynomially

solvable versions of the linear arrangement problem, the minimum FASP, or packing problems in graphs lead to polynomially solvable cases of the QAP. The coefficient matrices of these QAPs are the (weighted) adjacency matrices of the underlying graphs, and the special structure of these matrices is imposed by properties of these graphs.

Finally, graph isomorphism problems lead to polynomially solvable QAPs. Christofides and Gerrard [67] showed the following: let T_1 and T_2 be two isomorphic trees with n vertices and arbitrary weights on the edges. Moreover, let F and D be the symmetric weighted adjacency matrices of T_1 and T_2 , respectively. Let $\mathcal{I}(T_1, T_2)$ denote the set of all permutations φ which are isomorphisms between T_1 and T_2 . Then the quadratic assignment problem

$$\min_{\varphi \in \mathcal{I}(T_1, T_2)} \sum_{i=1}^n \sum_{k=1}^n a_{ik} b_{\varphi(i)\varphi(k)} \quad (76)$$

can be solved by a polynomial time dynamic programming algorithm. On the other hand, the cheapest embedding of a tree with n vertices in a weighted complete graph K_n is \mathcal{NP} -hard. Rendl [198] generalized the result on two isomorphic trees to certain vertex series parallel digraphs.

11 Asymptotic Behavior

Quadratic assignment problems show an interesting asymptotic behavior: under certain probabilistic conditions on the coefficient matrices of QAPs, the ratio between its “best” and “worst” objective function values approaches 1, as the size of the problem tends to infinity. This asymptotic behavior suggests that any heuristic finds almost always an almost optimal solution when applied to sufficiently large QAP instances. In other words, the QAP becomes in some sense trivial as the size of the problem tends to infinity. Burkard and Fincke [50] identify a common combinatorial property on a number of combinatorial optimization problems which, under natural probabilistic conditions on the problem data, behave as described above. This property seems to govern also the specific asymptotic behavior of QAPs.

In an early work, Burkard and Fincke [49] investigate the relative difference between the worst and the best value of the objective function for Koopmans–Beckmann QAPs. They first consider the case where the coefficient matrix D is the matrix of pairwise distances of points chosen independently and uniformly from the unit square in the plane. Then the general case where entries of the flow and distance matrices F and D are independent random variables taken from a uniform distribution on $[0, 1]$ is considered. In both cases it is shown that the relative difference mentioned above approaches 0 with probability tending to 1 as the size of the problem tends to infinity.

Later Burkard and Fincke [50] consider the ratio between the objective function values corresponding to an optimal (or best) and a worst solution of a generic combinatorial optimization problem described below.

Consider a sequence (P_n) , $n \in \mathbb{N}$, of combinatorial optimization (minimization) problems with sum objective function. Let \mathcal{E}_n and \mathcal{F}_n be the ground set and the set of feasible solutions of problem P_n , respectively. Moreover, let $c_n: \mathcal{E}_n \rightarrow \mathbb{R}^+$ and $f: \mathcal{F}_n \rightarrow \mathbb{R}^+$ be the nonnegative cost function and the objective function for problem P_n , respectively. For $n \in \mathbb{N}$, let

$$BV(n) = \min_{X \in \mathcal{F}_n} f(X),$$

$$WV(n) = \max_{X \in \mathcal{F}_n} f(X).$$

In [50] it is shown that the behavior of the ratio $WV(n)/BV(n)$) is strongly related to the ratio $\ln |\mathcal{F}_n|/|X_n|$ between the cardinality of the set of feasible solutions \mathcal{F}_n and the cardinality of an arbitrary feasible solution X_n of P_n , under the assumption that all feasible solutions have the same cardinality.

Theorem 7 ([50]) *Let (P_n) be a sequence of combinatorial minimization problems with sum objective function as described above. Assume that the following conditions are fulfilled:*

- (BF1) *All feasible solutions $X \in \mathcal{F}_n$ have the same cardinality $|X|_n$.*
- (BF2) *For all n the costs $c_n(x)$, $x \in X$, $X \in \mathcal{F}_n$ are independent random variables identically distributed on $[0, 1]$ with finite expected value $\mu = E(c_n(x))$ and finite variance $\sigma^2 = \text{Var}(c_n(x))$.*
- (BF3) *$|\mathcal{F}_n|$ and $|X|_n$ tend to infinity as n tends to infinity and moreover*

$$\lim_{n \rightarrow \infty} \lambda_0 |X|_n - \ln |\mathcal{F}_n| \rightarrow +\infty$$

where λ_0 is defined by $\lambda_0 := (\epsilon_0 \sigma / (\epsilon_0 + 2\sigma^2))^2$ and ϵ_0 fulfills

$$0 < \epsilon_0 < \sigma^2 \quad \text{and} \quad 0 < \frac{\mu + \epsilon_0}{\mu - \epsilon_0} \leq 1 + \epsilon, \quad (77)$$

for a given $\epsilon > 0$.

Then, as $n \rightarrow \infty$

$$P \left\{ \frac{WV(n)}{BV(n)} < 1 + \epsilon \right\} \geq 1 - 2|\mathcal{F}_n| \exp(-|X|_n \lambda_0) \rightarrow 1.$$

The combinatorial condition represented by the limit in (BF3) says that the cardinality of the feasible solutions is large enough with respect to the cardinality of the set of feasible solutions. Namely, the result of the theorem is true if the following equality holds:

$$\lim_{n \rightarrow \infty} \frac{\ln |\mathcal{F}_n|}{|X|_n} = 0.$$

The other conditions of [Theorem 7](#) are natural probabilistic requirements on the coefficients of the problem. [Theorem 7](#) states that for each $\epsilon > 0$, the ratio between the best and the worst value of the objective function lies in $(1 - \epsilon, 1 + \epsilon)$, with probability tending to 1, as the “size” of the problem approaches infinity. Thus, we have convergence with probability. Under one additional natural (combinatorial) assumption (condition (S3) of the theorem below), Szpankowski strengthens this result and improves the range of the convergence to almost surely. In the almost sure convergence, the probability that the above-mentioned ratio tends to 1 is equal to 1.

Theorem 8 ([215]) Let P_n be a sequence of combinatorial minimization problems with sum objective function as above. Assume that the following conditions are fulfilled:

- (S1) All feasible solutions $X \in \mathcal{F}_n$ have the same cardinality $|X|_n$.
- (S2) For all n the costs $c_n(x)$, $x \in X$, $X \in \mathcal{F}_n$ are random variables identically and independently distributed on $[0, 1]$. The expected value $\mu = E(c_n(x))$, the variance, and the third moment of the common distribution are finite.
- (S3) For increasing values of n , the worst objective function values $WV(n)$ are nondecreasing.
- (S4) $|\mathcal{F}_n|$ and $|X|_n$ tend to infinity as n tends to infinity and moreover $\ln |\mathcal{F}_n| = o(|X|_n)$.

Then, the following equalities hold almost surely:

$$BV(n) = |X|_n \mu - o(|X|_n)$$

$$WV(n) = |X|_n \mu + o(|X|_n).$$

[Theorems 7](#) and [8](#) can be applied to the QAP. The reason is that the QAP fulfills the combinatorial condition (S4) in [Theorem 8](#) [and, therefore, also condition (BF3) in [Theorem 7](#)]. Thus, we immediately get the following corollary:

Corollary 1 Consider a sequence of problems QAP($F^{(n)}, D^{(n)}$) for $n \in \mathbb{N}$, with $n \times n$ coefficient matrices $F^{(n)} = (f_{ij}^{(n)})$ and $D = (d_{ij}^{(n)})$. Assume that $f_{ij}^{(n)}$ and $d_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \leq i, j \leq n$, are independently distributed random variables on $[0, M]$, where M is a positive constant. Moreover, assume that entries $f_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \leq i, j \leq n$, have the same distribution, and entries $d_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \leq i, j \leq n$, have also the same distribution (which does not necessarily coincide with that of $f_{ij}^{(n)}$). Furthermore, assume that these variables have finite expected values, variances, and third moments.

Let $\pi_{\text{opt}}^{(n)}$ and $\pi_{\text{wor}}^{(n)}$ denote an optimal and a worst solution of QAP($F^{(n)}, D^{(n)}$), respectively. Then the following equality holds almost surely:

$$\lim_{n \rightarrow \infty} \frac{Z(F^{(n)}, D^{(n)}, \pi_{\text{opt}}^{(n)})}{Z(F^{(n)}, D^{(n)}, \pi_{\text{wor}}^{(n)})} = 1.$$

The above result suggests that the value of the objective function of the problem $\text{QAP}(F^{(n)}, D^{(n)})$ (corresponding to an arbitrary feasible solution) gets somehow close to its expected value $n^2 \mu(F)\mu(D)$, as the size of the problem increases, where $\mu(F)$ and $\mu(D)$ are the expected values of $f_{ij}^{(n)}$ and $d_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \leq i, j \leq n$, respectively. Frenk et al. [101] and Rhee [202, 203] provide different analytical evaluations for this “getting close,” by imposing different probabilistic conditions on the data. The following theorem states two important results proved in [101] and [203]:

Theorem 9 ([101, 203]) Consider the sequence of $\text{QAP}(F^{(n)}, D^{(n)})$, $n \in \mathbb{N}$, as in Corollary 1. Assume that the following conditions are fulfilled:

- (C1) $f_{ij}^{(n)}, f_{ij}^{(n)}, n \in \mathbb{N}, 1 \leq i, j \leq n$ are random variables independently distributed on $[0, M]$.
- (C2) $f_{ij}^{(n)}, n \in \mathbb{N}, 1 \leq i, j \leq n$ have the same distribution on $[0, M]$. $d_{ij}^{(n)}, n \in \mathbb{N}, 1 \leq i, j \leq n$ have also the same distribution on $[0, M]$.

Let $\mu(F), \mu(D)$ be the expected values of the variables $f_{ij}^{(n)}$ and $d_{ij}^{(n)}$, respectively. Then, there exists a constant K_1 (which does not depend on n), such that the following inequality holds almost surely, for $\pi \in \mathcal{S}_n$:

$$\limsup_{n \rightarrow \infty} \frac{\sqrt{n}}{\sqrt{\log n}} \left| \frac{Z(F^{(n)}, D^{(n)}, \pi)}{n^2 \mu(F)\mu(D)} - 1 \right| \leq K_1.$$

Moreover, let Y be a random variable defined by

$$Y = Z(F^{(n)}, D^{(n)}, \pi_{\text{opt}}^{(n)}) - n^2 \mu(F)\mu(D),$$

where $\pi_{\text{opt}}^{(n)}$ is an optimal solution of $\text{QAP}(F^{(n)}, D^{(n)})$. Then there exists another constant K_2 , also independent of the size of the problem, such that

$$\begin{aligned} \frac{1}{K_2} n^{3/2} (\log n)^{1/2} &\leq E(Y) \leq K_2 n^{3/2} (\log n)^{1/2} \\ P \{|Y - E(Y)| \geq t\} &\leq 2 \exp \left(\frac{-t^2}{4n^2 \|F\|_\infty^2 \|D\|_\infty^2} \right) \end{aligned}$$

for each $t \geq 0$, where $E(Y)$ denotes the expected value of variable Y and $\|F\|_\infty$ ($\|D\|_\infty$) is the so-called row sum norm of matrix F (D) defined by $\|F\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |f_{ij}|$.

These results on the asymptotic behavior of the QAP have been exploited by Dyer et al. [89] to analyze the performance of branch-and-bound algorithms for QAPs with coefficients generated randomly as described above. In this case the gap between the optimal value of the continuous relaxation due to Frieze–Yadegar and the optimal value of the QAP grows like $O(n)$ with probability tending to 1 as n tends to infinity. This result leads to the following theorem:

Theorem 10 ([89]) *Consider any branch-and-bound algorithm for solving a QAP with randomly generated coefficients as in Corollary 1 that uses a single assignment branching and employs a bound obtained by solving the continuous relaxation of the linearization (19)–(26). The number of branched nodes explored is at least $n^{(1-o(1))n/4}$ with probability tending to 1 as the size n of the QAP tends to infinity.*

It is interesting to note that the asymptotic results on generic combinatorial optimization problems can also be derived by means of statistical mechanics. Recall that also *simulated annealing*, Sect. 8.7, is a technique from statistical mechanics. Bonomi and Lutton [32] applied the framework of statistical mechanics to the QAP: the feasible solutions of a combinatorial optimization problem correspond to the states of a physical system, the objective function models the energy of the state. Albrecher et al. [10] define a *partition function* by

$$Q(\tau) = \sum_{X \in \mathcal{F}} \exp(-f(X)\tau), \quad (78)$$

where τ mimics the inverse temperature $1/T$. They assign the probability

$$\mathbb{P}(X) = \frac{\exp(-f(X)\tau)}{Q(\tau)} \quad (79)$$

to each feasible solution $X \in \mathcal{F}$. Let $E(f(X), \tau)$ denote the expected value of the objective function $f(X)$ for fixed τ . It can easily be shown that

$$\mathbb{E}(f(X), \tau) = -\frac{d}{d\tau}(\log Q(\tau)). \quad (80)$$

A careful investigation of $\log Q(\tau)/|X|$ finally yields

$$\mathbb{P}\left(\lim_{n \rightarrow \infty} \frac{BV(n)}{|X|_n} = \mu\right) = 1 \quad (81)$$

where μ is the expectation value of the cost coefficients of the generic problem.

12 Quadratic Bottleneck Assignment Problems

The *quadratic bottleneck assignment problem (QBAP)* asks for a permutation φ which minimizes the *maximum* of the cost terms $f_{ij}d_{\varphi(i)\varphi(j)}$. Given an $n \times n$ flow matrix F and an $n \times n$ distance matrix D , we want to find a permutation $\varphi \in S_n$ which minimizes the objective function

$$\max\{f_{ij}d_{\varphi(i)\varphi(j)} : 1 \leq i, j \leq n\}.$$

A more general QBAP analogous to the QAP in (2) is obtained if the coefficients of the problem are of the form c_{ijkl} , $1 \leq i, j, k, l \leq n$:

$$\min_{\varphi \in S_n} \max_{1 \leq i, j \leq n} c_{ij\varphi(i)\varphi(j)}.$$

The QBAP has many applications. Basically, all QAP applications give rise to QBAPs because often it makes sense to minimize the largest cost instead of the overall cost incurred by some decision. A well-studied problem in graph theory which can be modeled as a QBAP is the *bandwidth problem*. In the bandwidth problem, we are given an undirected graph $G = (V, E)$ with vertex set $V = \{1, 2, \dots, n\}$ and edges $(i, j) \in E$. A labeling φ of the vertices has to be found such that

$$\max_{(i,j) \in E} |\varphi(i) - \varphi(j)|$$

is minimized. It is easy to see that this problem can be modeled as a special QBAP with the flow matrix equal to the adjacency matrix of G and distance matrix $D = (d_{kl}) = (|k - l|)$.

The QBAP is NP-hard since it contains the bottleneck TSP as a special case. In order to solve QBAP by branch-and-bound methods (or more generally, implicit enumeration methods), lower bounds can be derived in a similar way as in the case of QAPs. In practice, however, QBAPs are often easier to solve than QAPs, since threshold techniques can be applied which reduce the problem to the decision problem whether or not there exists a permutation φ such that for a given constant K $c_{ij\varphi(i)\varphi(j)} \leq K$ holds for all $1 \leq i, j \leq n$.

For computing lower bounds, one can proceed as follows: let matrix C be written as a block matrix (C^{ik}) where every C^{ik} is the matrix (c_{ijkl}) with fixed indices i and k . As in the sum case we can assume

$$c_{ijkl} = \infty \quad \text{for } (i = j \text{ and } k \neq l) \text{ or } (i \neq j \text{ and } k = l). \quad (82)$$

and

$$c_{ijkl} := \begin{cases} \max(c_{ijkl}, c_{klji}) & \text{for } j > i, \\ 0 & \text{for } j < i. \end{cases} \quad (83)$$

For every i, k denote the optimal solution value of a linear bottleneck assignment problem with cost matrix C^{ik} by l_{ik} . A lower bound for the optimum objective

function value of the QBAP is obtained by the optimal objective function value of a linear bottleneck assignment problem with cost matrix $L = (l_{ik})$.

In case of a QBAP in Koopmans–Beckmann form, one can proceed like in the Gilmore–Lawler bound. In particular a modification of Hardy et al. [127] inequality holds also for bottleneck problems (cf. [Proposition 2](#)).

Proposition 7 *Let two n -dimensional real vectors $a = (a_i)$, $b = (b_i)$ with $0 \leq a_1 \leq a_2 \leq \dots \leq a_n$ and $b_1 \geq b_2 \geq \dots \geq b_n \geq 0$ be given. Then the following inequalities hold for any permutation φ of $1, 2, \dots, n$:*

$$\max_{1 \leq i \leq n} a_i b_i \leq \max_{1 \leq i \leq n} a_i b_{\varphi(i)} \leq \max_{1 \leq i \leq n} a_i b_{n-i+1}.$$

This proposition yields that a linear bottleneck assignment problem with cost coefficients $c_{ik} = a_i b_k$ can be solved by ordering the vectors $a = (a_i)$ and $b = (b_k)$.

The lower bounds can be used in branch-and-bound algorithms; see Burkard [37].

Like in the sum case, there exist a number of special cases of the QBAP which can be solved in polynomial time; see Burkard and Rissner [56]. In particular, a result like [Proposition 6](#) holds also in the bottleneck case. Moreover, a QBAP with two symmetric, nonnegative product matrices can be solved in $O(n \log n)$ time.

The Monge condition of sum problems can be replaced by the bottleneck Monge condition; see Burkard et al. [53].

$$\max(a_{ij}, a_{kl}) \leq \max(a_{il}, a_{kj}).$$

A careful study of matrices endowed with the bottleneck Monge structure (so-called bottleneck Monge matrices) shows, for example, that a QBAP of Koopmans–Beckmann form where F is a bottleneck Monge matrix and D is a chessboard matrix with even $n = 2m$ is solved by the permutation

$$\varphi(i) = \begin{cases} 2i - 1 & 1 \leq i \leq m \\ 2(i - m) & m + 1 \leq i \leq n. \end{cases}$$

For further polynomial special cases, see [56].

QBAP show a similar asymptotic behavior as QAPs. Burkard and Fincke [48] proved the following: if the coefficients of a QBAP are independent random variables taken from a uniform distribution on $[0, 1]$, then the relative difference between the worst and the best value of the objective function approaches 0 with probability tending to 0 as the size of the problem approaches infinity. Albrecher [9] sharpened the convergence rate by showing the following result. Let

$$BV(n) = \min_{\varphi} \max_{1 \leq i, j \leq n} c_{ij\varphi(i)\varphi(j)},$$

$$WV(n) = \max_{\varphi} \max_{1 \leq i, j \leq n} c_{ij\varphi(i)\varphi(j)}.$$

If the cost coefficients are independent and identically distributed random variables in $[0, 1]$, then

$$\frac{WV(n) - BV(n)}{BV(n)} \leq \sqrt{\frac{2 \log n}{n}} \left(1 - \frac{1}{2 \log n} - \frac{1}{8(\log n)^2} \right) \text{ almost surely.}$$

The asymptotic behavior of generic combinatorial optimization problems (cf. Sect. 11) with a bottleneck objective function was studied by Burkard and Fincke [50] and Szpankowski [215].

The QBAP and the QAP are special cases of a more general quadratic assignment problem which is called *the algebraic QAP* (in analogy to the algebraic linear assignment problem (LAP) introduced by Burkard et al. [51]). If $(H, *, \preceq)$ is a totally ordered commutative semigroup with composition $*$ and order relation \preceq , the algebraic QAP with cost coefficients $c_{ijkl} \in H$ is formulated as

$$\min_{\varphi \in S_n} c_{11\varphi(1)\varphi(1)} * c_{12\varphi(1)\varphi(2)} * \dots * c_{1n\varphi(1)\varphi(n)} * \dots * c_{nn\varphi(n)\varphi(n)}.$$

The study of the bottleneck QAP and more generally the algebraic QAP was the starting point for the investigation of a number of algebraic combinatorial optimization problems with coefficients taken from linearly ordered semimodules, for example, linear assignment and transportation problems and flow problems. The reader is referred to Burkard and Zimmermann [57] for a detailed discussion on this topic.

13 Related Problems

In this section several of generalizations and problems related to the QAP are presented.

13.1 Cubic and Biquadratic Assignment Problems

One possibility to generalize the QAP is to consider objective functions of higher degree. In this way *cubic*, *biquadratic*, and generally N -adic assignment problems (see, e.g., [152]) are obtained. The cubic assignment problem, for example, has n^6 cost coefficients c_{ijklmp} with $i, j, k, l, m, p = 1, \dots, n$ and can be stated as follows:

$$\begin{aligned} \min & \sum_{i,j=1}^n \sum_{k,l=1}^n \sum_{m,p=1}^n c_{ijklmp} x_{ij} x_{kl} x_{mp} \\ \text{s.t } & (x_{ij}) \in \mathbf{X}_n. \end{aligned}$$

As noted by Lawler [152], we can construct an $n^3 \times n^3$ matrix S containing the cost coefficients, such that the cubic assignment problem is equivalent to the linear assignment problem

$$\begin{aligned} \min \quad & \langle S, Y \rangle \\ \text{s.t.} \quad & Y = X \otimes X \otimes X, \\ & X \in \mathbf{X}_n. \end{aligned}$$

In an analogous way, the LAP can be extended to any N -adic assignment problem, by considering the solution matrix Y to be the Kronecker $N - th$ power of a permutation matrix in \mathbf{X}_n .

The *biquadratic assignment problem* denoted BiQAP is essentially a quartic assignment problem with cost coefficients formed by the products of two four-dimensional arrays. More specifically, consider two $n^4 \times n^4$ arrays $F = (f_{ijkl})$ and $D = (d_{mpst})$. Then the BiQAP has the form

$$\begin{aligned} \min \quad & \sum_{i,j=1}^n \sum_{k,l=1}^n \sum_{m,p=1}^n \sum_{s,t=1}^n f_{ijkl} d_{mpst} x_{im} x_{jp} x_{ks} x_{lt} \\ \text{s.t.} \quad & \sum_{i=1}^n x_{ij} = 1, \quad j = 1, 2, \dots, n, \\ & \sum_{j=1}^n x_{ij} = 1, \quad i = 1, 2, \dots, n, \\ & x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \dots, n. \end{aligned}$$

Equivalently, the BiQAP can be stated as

$$\min_{\varphi \in \mathcal{S}_n} \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n f_{ijkl} d_{\varphi(i)\varphi(j)\varphi(k)\varphi(l)},$$

where \mathcal{S}_n denotes the set of all permutations of $N = \{1, 2, \dots, n\}$. All different formulations for the QAP can be extended to the BiQAP, as well as most of the linearizations that have appeared for the QAP.

An application of the BiQAP arises in very-large-scale integrated (VLSI) circuit design. The majority of VLSI circuits are sequential circuits, and their design process consists of two steps: first, translate the circuit specifications into a state transition table by modeling the system using finite state machines, and second, try to find an encoding of the states such that the actual implementation is of minimum size. A detailed description of the mathematical modeling of the VLSI problem to a BiQAP is given by Burkard et al. [44].

Burkard et al. [44] compute lower bounds for the BiQAP derived from lower bounds of the QAP. The computational results showed that these bounds are weak and deteriorate as the dimension of the problem increases. This observation suggests that branch-and-bound methods will only be effective on very small instances. For larger instances, efficient heuristics, which find good-quality approximate solutions, are needed. Several heuristics for the BiQAP have been developed by Burkard and Çela [41], in particular deterministic improvement methods and variants of simulated annealing and tabu search algorithms. Computational experiments on test problems of size up to $n = 32$, with known optimal solutions (a test problem generator is presented in [44]), suggest that one version of simulated annealing is best among those tested. The GRASP heuristic has also been applied to the BiQAP by Mavridou et al. [167] and produced the optimal solution for all the test problems generated in [44].

Quartic assignment problems show the same asymptotic behavior as QAPs. Burkard et al. [44] proved for quartic assignment problems under natural probabilistic assumptions that

$$\lim_{n \rightarrow \infty} \mathbb{P} \left\{ \frac{WV(n)}{BV(n)} < 1 + \varepsilon \right\} = 1.$$

13.2 The Quadratic Semi-assignment Problem

The quadratic semi-assignment problem (SQAP) assigns n “objects” to m “locations” with $n > m$ so as to minimize the overall distance covered by the flow of materials (or people) moving between different objects. The objective function is the same as that of the QAP, the only difference concerns the feasible solutions which are not any more one-to-one mappings (permutations) between the set of objects and locations, but arbitrary functions mapping the set of objects to the set of locations. Given two coefficient matrices, the flow matrix $F = (f_{ij})$ and the distance matrix $D = (d_{ij})$, the SQAP can be formulated in the following way:

$$\min \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^n \sum_{l=1}^m f_{ij} d_{kl} x_{ik} x_{jl} + \sum_{i,j=1}^n b_{ij} x_{ij} \quad (84)$$

$$\text{s.t. } \sum_{j=1}^m x_{ij} = 1, \quad i = 1, 2, \dots, n, \quad (85)$$

$$x_{ij} \in \{0, 1\}, \quad i, j = 1, 2, \dots, n. \quad (86)$$

The SQAP unifies some interesting combinatorial optimization problems like *clustering* and *m-coloring*. In a clustering problem, we are given n objects and a dissimilarity matrix $F = (f_{ij})$. The goal is to find a partition of these objects into m classes so as to minimize the sum of dissimilarities of objects belonging to the

same class. Obviously, this problem is an SQAP with coefficient matrices F and D , where D is an $m \times m$ identity matrix. In the m -coloring problem, we are given a graph with n vertices and we want to check whether its vertices can be colored by m different colors such that each two vertices which are joined by an edge receive different colors. This problem can be modeled as an SQAP with F equal to the adjacency matrix of the given graph and D the $m \times m$ identity matrix. The m -coloring has an answer “yes” if and only if the above SQAP has the optimal value equal to 0. The first application of the SQAP concerned the supply of space stations; see Freeman et al. [102]. Other applications of the SQAP include distributed computing [212], schedule synchronization in transit networks [75, 164], and scheduling [66].

The SQAP was originally introduced by Greenberg [119]. As pointed out by Malucelli [163], this problem is NP-hard. Milis and Magirou [169] propose a Lagrangean relaxation algorithm for this problem and show that similarly as for the QAP, it is very hard to provide optimal solutions even for SQAPs of small size. Lower bounds for the SQAP have been provided by Malucelli and Pretolani [165], as well as by Billionet and Elloumi [26]. Polynomially solvable special cases have been discussed by Bokhari [29], Chhajed and Lowe [65], and Malucelli [163]. A survey of heuristics for SQAP is provided by Voß [221].

Recommended Reading

1. E.H.L. Aarts, J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing* (Wiley, Chichester, 1989)
2. E. Aarts, J.K. Lenstra (eds.), *Local Search in Combinatorial Optimization* (Wiley, Chichester, 1997)
3. W.P. Adams, M. Guignard, P.M. Hahn, W.L. Hightower, A level-2 reformulation-linearization technique bound for the quadratic assignment problem. *Eur. J. Oper. Res.* **180**, 983–996 (2007)
4. W.P. Adams, T.A. Johnson, Improved linear programming-based lower bounds for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, ed. by P.M. Pardalos, H. Wolkowicz. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 16 (AMS, Providence, 1994), pp. 43–75
5. W.P. Adams, H.D. Sherali, A tight linearization and an algorithm for zero-one quadratic programming problems. *Manag. Sci.* **32**, 1274–1290 (1986)
6. W.P. Adams, H.D. Sherali, Linearization strategies for a class of zero-one mixed integer programming problems. *Oper. Res.* **38**, 217–226 (1990)
7. R.K. Ahuja, K.C. Jha, J.B. Orlin, D. Sharma, Very large-scale neighborhood search for the quadratic assignment problem. *INFORMS J. Comput.* **19**, 646–657 (2007)
8. R.K. Ahuja, J.B. Orlin, A. Tivari, A greedy genetic algorithm for the quadratic assignment problem. Working paper 3826-95, Sloan School of Management, MIT, 1995
9. H. Albrecher, A note on the asymptotic behaviour of bottleneck problems. *Oper. Res. Lett.* **33**, 183–186 (2005)
10. H. Albrecher, R.E. Burkard, E. Çela, An asymptotical study of combinatorial optimization problems by means of statistical mechanics. *J. Comput. Appl. Math.* **186**, 148–162 (2006)
11. K.M. Anstreicher, Eigenvalue bounds versus semidefinite relaxations for the quadratic assignment problem. *SIAM J. Optim.* **11**, 254–265 (2000)
12. K.M. Anstreicher, Recent advances in the solution of quadratic assignment problems. *Math. Program.* **97**, 27–42 (2003)

13. K.M. Anstreicher, N.W. Brixius, A new bound for the quadratic assignment problem based on convex quadratic programming. *Math. Program.* **89**, 341–357 (2001)
14. K.M. Anstreicher, N.W. Brixius, J.P. Goux, J. Linderoth, Solving large quadratic assignment problems on computational grids. *Math. Program.* **91**, 563–588 (2002)
15. E.M. Arkin, R. Hassin, M. Sviridenko, Approximating the maximum quadratic assignment problem. *Inform. Process. Lett.* **77**, 13–16 (2001)
16. S. Arora, A. Frieze, H. Kaplan, A new rounding procedure for the assignment problem with applications to dense graph arrangement problems, in *Proceedings of the 37-th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1996, pp. 21–30
17. A.A. Assad, W. Xu, On lower bounds for a class of quadratic 0–1 programs. *Oper. Res. Lett.* **4**, 175–180 (1985)
18. E. Balas, J.B. Mazzola, Nonlinear programming: I. Linearization techniques. *Math. Program.* **30**, 1–21 (1984)
19. E. Balas, J.B. Mazzola, Nonlinear programming: II. Dominance relations and algorithms. *Math. Program.* **30**, 22–45 (1984)
20. A.I. Barvinok, Computational complexity of orbits in representations of symmetric groups. *Adv. Soviet Math.* **9**, 161–182 (1992)
21. R. Battiti, G. Tecchiori, The reactive tabu search. *ORSA J. Comput.* **6**, 126–140 (1994)
22. M. Bayat, M. Sedghi, Quadratic assignment problems, in *Facility Location*, ed. by R.Z. Farahani, M. Hekmatfar (Springer, Berlin, 2010), pp. 111–143
23. M.S. Bazaraa, O. Kirca, Branch and bound based heuristics for solving the quadratic assignment problem. *Naval Res. Logist. Q.* **30**, 287–304 (1983)
24. M.S. Bazaraa, H.D. Sherali, Benders’ partitioning scheme applied to a new formulation of the quadratic assignment problem. *Naval Res. Logist. Q.* **27**, 29–41 (1980)
25. M.S. Bazaraa, H.D. Sherali, On the use of exact and heuristic cutting plane methods for the quadratic assignment problem. *J. Oper. Res. Soc.* **33**, 991–1003 (1982)
26. A. Billionet, S. Elloumi, Best reduction of the quadratic semi-assignment problem. *Discrete Appl. Math.* **109**, 197–213 (2001)
27. G. Birkhoff, Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumán Rev. Ser. A*, **5**, 147–151 (1946)
28. A. Blanchard, S. Elloumi, A. Faye, M. Wicker, Un algorithme de génération de coupes pour le problème de l’affectation quadratique. *INFOR* **41**, 35–49 (2003)
29. S.H. Bokhari, A shortest tree algorithm for optimal assignments across space and time in a distributed processor system. *IEEE Trans. Softw. Eng.* **7**, 583–589 (1981)
30. B. Bollobás, *Extremal Graph Theory* (Academic, London, 1978)
31. A.A. Bolotnikov, On the best balance of the disk with masses on its periphery. *Problemi Mashinostroenia* **6**, 68–74 (1978) (in Russian)
32. E. Bonomi, J. Lutton, The asymptotic behavior of quadratic sum assignment problems: A statistical mechanics approach. *Eur. J. Oper. Res.* **26**, 295–300 (1986)
33. A. Bruegger, J. Clausen, A. Marzetta, M. Perregaard, Joining forces in solving large-scale quadratic assignment problems in parallel, in *Proceedings of the 11-th IEEE International Parallel Processing Symposium (IPPS)*, Geneva, Switzerland, 1997, pp. 418–427
34. E.S. Buffa, G.C. Armour, T.E. Vollmann, Allocating facilities with CRAFT. *Harv. Bus. Rev.* **42**, 136–158 (1962)
35. S. Burer, D. Vandebrussche, Solving lift-and-project relaxations of binary integer programs. *SIAM J. Optim.* **16**, 726–750 (2006)
36. R.E. Burkard, Die Störungsmethode zur Lösung quadratischer Zuordnungsprobleme. *Oper. Res. Verfahren* **16**, 84–108 (1973)
37. R.E. Burkard, Quadratische Bottleneckprobleme. *Oper. Res. Verfahren* **18**, 26–41 (1974)
38. R.E. Burkard, Locations with spatial interactions: the quadratic assignment problem, in *Discrete Location Theory*, ed. by P.B. Mirchandani, R.L. Francis (Wiley, 1991)
39. R.E. Burkard, Admissible transformations and assignment problems. *Vietnam J. Math.* **35**, 373–386 (2007)

40. R.E. Burkard, T. Bönniger, A heuristic for quadratic boolean programs with applications to quadratic assignment problems. *Eur. J. Oper. Res.* **13**, 374–386 (1983)
41. R.E. Burkard, E. Çela, Heuristics for biquadratic assignment problems and their computational comparison. *Eur. J. Oper. Res.* **83**, 283–300 (1995)
42. R.E. Burkard, E. Çela, V.M. Demidenko, N.N. Metelski, G.J. Woeginger, Perspectives of easy and hard cases of the quadratic assignment problems. SFB Report 104, Institute of Mathematics, Technical University Graz, Austria, 1997
43. R.E. Burkard, E. Çela, V.M. Demidenko, N.N. Metelski, G.J. Woeginger, A unified approach to simple special cases of extremal permutation. *Optimization* **44**, 123–138 (1998)
44. R.E. Burkard, E. Çela, B. Klinz, On the biquadratic assignment problem, in *Quadratic Assignment and Related Problems*, ed. by P.M. Pardalos, H. Wolkowicz. DIMACS Series on Discrete Mathematics and Theoretical Computer Science, vol. 16 (AMS, Providence, 1994), pp. 117–146
45. R.E. Burkard, E. Çela, G. Rote, G.J. Woeginger, The quadratic assignment problem with an Anti-Monge and a Toeplitz matrix: easy and hard cases. *Math. Program. B* **82**, 125–158 (1998)
46. R.E. Burkard, M. Dell'Amico, S. Martello, *Assignment Problems* (SIAM, Philadelphia, 2009). ISBN 978-0-898716-63-4, revised reprint 2012, ISBN 978-1-611972-22-1
47. R.E. Burkard, U. Derigs, *Assignment and Matching Problems: Solution Methods with Fortran Programs*. Lecture Notes in Economics and Mathematical Systems, vol. 184 (Springer, Berlin, 1980)
48. R.E. Burkard, U. Fincke, On random quadratic bottleneck assignment problems. *Math. Program.* **23**, 227–232 (1982)
49. R.E. Burkard, U. Fincke, The asymptotic probabilistic behavior of the quadratic sum assignment problem. *Z. Oper. Res.* **27**, 73–81 (1983)
50. R.E. Burkard, U. Fincke, Probabilistic asymptotic properties of some combinatorial optimization problems. *Discrete Appl. Math.* **12**, 21–29 (1985)
51. R.E. Burkard, W. Hahn, U. Zimmermann, An algebraic approach to assignment problems. *Math. Program.* **12**, 318–327 (1977)
52. R.E. Burkard, S.E. Karisch, F. Rendl, QAPLIB—a quadratic assignment problem library. *J. Global Optim.* **10**, 391–403 (1997). An on-line version is available via World Wide Web at the following URL: <http://www.seas.upenn.edu/qplib/>
53. R.E. Burkard, B. Klinz, R. Rudolf, Perspectives of Monge properties in optimization. *Discrete Appl. Math.* **70**, 95–161 (1996)
54. R.E. Burkard, J. Offermann, Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. *Z. Oper. Res.* **21**, B121–B132 (1977) (in German)
55. R.E. Burkard, F. Rendl, A thermodynamically motivated simulation procedure for combinatorial optimization problems. *Eur. J. Oper. Res.* **17**, 169–174 (1984)
56. R.E. Burkard, R. Rissner, Polynomially solvable special cases of the quadratic bottleneck assignment problem. *J. Combin. Optim.* **22**, 845–856 (2011)
57. R.E. Burkard, U. Zimmermann, Combinatorial optimization in linearly ordered semimodules: a survey, in *Modern Applied Mathematics*, ed. by B. Korte (North Holland, Amsterdam, 1982), pp. 392–436
58. P. Carraresi, F. Malucelli, A new lower bound for the quadratic assignment problem. *Oper. Res.* **40**(Suppl. 1), S22–S27 (1992)
59. P. Carraresi, F. Malucelli, A reformulation scheme and new lower bounds for the QAP, in *Quadratic Assignment and Related Problems*, ed. by P. Pardalos, H. Wolkowicz. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (AMS, Providence, RI, 1994), 147–160
60. E. Çela, *The Quadratic Assignment Problem: Theory and Algorithms* (Kluwer Academic, Dordrecht, 1998)
61. E. Çela, N.S. Schmuck, S. Wimer, G.J. Woeginger, The Wiener maximum quadratic assignment problem. *J. Discrete Optim.* **8**, 411–416 (2011)

62. V. Černý, Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *J. Optim. Theory Appl.* **45**, 41–51 (1985)
63. J. Chakrapani, J. Skorin-Kapov, Massively parallel tabu search for the quadratic assignment problem. *Ann. Oper. Res.* **41**, 327–342 (1993)
64. J. Chakrapani, J. Skorin-Kapov, A constructive method to improve lower bounds for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, ed. by P. Pardalos, H. Wolkowicz. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (AMS, Providence, 1994), pp. 161–171
65. D. Chhajed, T.J. Lowe, m -Median and m -center problems with mutual communication: solvable special cases. *Oper. Res.* **40**, S56–S66 (1992)
66. P. Chrétienne, A polynomial algorithm to optimally schedule tasks on a virtual distributed system under tree-like precedence constraints. *Eur. J. Oper. Res.* **43**, 225–230 (1989)
67. N. Christofides, M. Gerrard, A graph theoretic analysis of bounds for the quadratic assignment problem, in *Studies on Graphs and Discrete Programming*, ed. by P. Hansen (North Holland, Amsterdam, 1981), pp. 61–68
68. J. Clausen, S.E. Karisch, M. Perregaard, F. Rendl, On the applicability of lower bounds for solving rectilinear quadratic assignment problems in parallel. *Comput. Optim. Appl.* **10**, 127–147 (1998)
69. J. Clausen, M. Perregaard, Solving large quadratic assignment problems in parallel. *Comput. Optim. Appl.* **8**, 111–127 (1997)
70. A. Colomni, M. Dorigo, V. Maniezzo, The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B* **26**, 29–41 (1996)
71. A. Colomni, V. Maniezzo, The ant system applied to the quadratic assignment problem. *IEEE Trans. Knowl. Data En.* **11**, 769–778 (1999)
72. D.T. Connolly, An improved annealing scheme for the QAP. *Eur. J. Oper. Res.* **46**, 93–100 (1990)
73. K. Conrad, *Das Quadratische Zuweisungsproblem und zwei seiner Spezialfälle* (Mohr-Siebeck, Tübingen, 1971)
74. D. Cyganski, R.F. Vaz, V.G. Virball, Quadratic assignment problems with the Palubeckis' algorithm are degenerate. *IEEE Trans. Circ. Syst. I* **41**, 481–484 (1994)
75. J.R. Daduna, S. Voß, Practical experiences in schedule synchronization, in *Computer-Aided Transit Schedule*, ed. by J.R. Daduna, I. Branco, J.M. Pinto Paix ao. Lecture Notes in Economics and Mathematical Systems, vol. 430 (Springer, 1995), pp. 39–55
76. L. Davis, *Genetic Algorithms and Simulated Annealing* (Pitman, London, 1987)
77. S.A. de Carvalho Jr., S. Rahmann, Microarray layout as a quadratic assignment problem, in *Proceedings of the German Conference in Bioinformatics (GCB)*, ed. by D.H. Huson, O. Kohlbacher, A. Lupus, K. Nieselt, A. Zell. Lecture Notes in Computer Science, vol. P-38 (Springer, Berlin, 2006), pp. 11–20
78. E. de Klerk, R. Sotirov, Exploiting group symmetry in semidefinite programming relaxations of the quadratic assignment problem. *Math. Program.* **122**, 225–246 (2010)
79. V.M. Demidenko, A. Dolgui, Efficiently solvable cases of the quadratic assignment problem with generalized monotonic and incomplete Anti-Monge matrices. *Cybern. Syst. Anal.* **43**, 112–125 (2007)
80. V.M. Demidenko, G. Finke, V.S. Gordon, Well solvable cases of the quadratic assignment problem with monotone and bimonotone matrices. *J. Math. Model. Algorithms* **5**, 167–187 (2006)
81. V.G. Deineko, G.J. Woeginger, A solvable case of the quadratic assignment problem. *Oper. Res. Lett.* **22**, 13–17 (1998)
82. V.G. Deineko, G.J. Woeginger, A study of exponential neighborhoods for the travelling salesman problem and the quadratic assignment problem. *Math. Program. Ser. A* **87**, 519–542 (2000)
83. J.W. Dickey, J.W. Hopkins, Campus building arrangement using TOPAZ. *Transp. Res.* **6**, 59–68 (1972)

84. Y. Ding, H. Wolkowicz, A low dimensional semidefinite relaxation for the quadratic assignment problem. *Math. Oper. Res.* **34**, 1008–1022 (2009)
85. A.A. Dobrynin, R. Entringer, I. Gutman, Wiener index of trees: theory and applications. *Acta Appl. Math.* **66**, 211–249 (2001)
86. M. Dorigo, Optimization, learning, and natural algorithms. Ph.D. Thesis, Dipartimento die Elettronica e Informazione, Politecnico di Milano, Milano, Italy, 1992 (in Italian)
87. Z. Drezner, A new genetic algorithm for the quadratic assignment problem. *INFORMS J. Comput.* **15**, 320–330 (2003)
88. Z. Drezner, Compound genetic algorithms for the quadratic assignment problem. *Oper. Res. Lett.* **33**, 475–480 (2005)
89. M.E. Dyer, A.M. Frieze, C.J.H. McDiarmid, On linear programs with random costs. *Math. Program.* **35**, 3–16 (1986)
90. C.S. Edwards, The derivation of a greedy approximator for the Koopmans-Beckmann quadratic assignment problem, in *Proceedings of the 77-th Combinatorial Programming Conference (CP77)*, Liverpool, 1977, pp. 55–86
91. C.S. Edwards, A branch and bound algorithm for the Koopmans-Beckmann quadratic assignment problem. *Math. Program. Study* **13**, 35–52 (1980)
92. A.N. Elshafei, Hospital layout as a quadratic assignment problem. *Oper. Res. Q.* **28**, 167–179 (1977)
93. R. Enkhbat, T. Javzandulam, A global search algorithm for the quadratic assignment problem. *J. Mongolian Math. Soc.* **4**, 16–28 (2000)
94. G. Erdogan, B. Tansel, A note on a polynomial time solvable case of the quadratic assignment problem. *J. Discrete Optim.* **3**, 382–384 (2006)
95. G. Erdogan, B. Tansel, A branch and cut algorithm for quadratic assignment problems based on linearizations. *Comput. Oper. Res.* **34**, 1085–1106 (2007)
96. A. Faye, F. Roupin, A cutting plane algorithm based upon a semidefinite relaxation for the quadratic assignment problem, in *Algorithms—ESA 2005*, ed. by G.S. Brodal, S. Leonardi. Lecture Notes in Computer Science, vol. 3669 (Springer, Heidelberg, 2005), pp. 850–861
97. T.A. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures. *J. Global Optim.* **6**, 109–133 (1995)
98. G. Finke, R.E. Burkard, F. Rendl, Quadratic assignment problems. *Ann. Discrete Math.* **31**, 61–82 (1987)
99. M. Fischetti, M. Monaci, D. Salvagnin, Three ideas for the quadratic assignment problem. *Oper. Res.* **60**, 954–964 (2012)
100. C. Fleurent, J. Ferland, Genetic hybrids for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, ed. by P. Pardalos, H. Wolkowicz. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (AMS, Providence, 1994), pp. 173–187
101. J.B.G. Frenk, M. van Houweninge, A.H.G. Rinnooy Kan, Asymptotic properties of the quadratic assignment problem. *Math. Oper. Res.* **10**, 100–116 (1985)
102. R.L. Freeman, D.C. Gogerty, G.W. Graves, R.B.S. Brooks, A mathematical model of supply for space operations. *Oper. Res.* **14**, 1–15 (1966)
103. A.M. Frieze, J. Yadegar, On the quadratic assignment problem. *Discrete Appl. Math.* **5**, 89–98 (1983)
104. A.M. Frieze, J. Yadegar, S. El-Horbaty, D. Parkinson, Algorithms for assignment problems on an array processor. *Parallel Comput.* **11**, 151–162 (1989)
105. L.M. Gambardella, E.D. Taillard, M. Dorigo, Ant colonies for the QAP. *J. Oper. Res. Soc.* **50**, 167–176 (1999)
106. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman and Company, New York, 1979)
107. J.W. Gavett, N.V. Plyter, The optimal assignment of facilities to locations by branch and bound. *Oper. Res.* **14**, 210–232 (1966)
108. A.M. Geoffrion, Lagrangean relaxation and its uses in integer programming. *Math. Program. Study* **2**, 82–114 (1974)

109. A.M. Geoffrion, G.W. Graves, Scheduling parallel production lines with changeover costs: practical applications of a quadratic assignment/LP approach. *Oper. Res.* **24**, 595–610 (1976)
110. P.C. Gilmore, Optimal and suboptimal algorithms for the quadratic assignment problem. *SIAM J. Appl. Math.* **10**, 305–313 (1962)
111. F. Glover, Improved linear integer programming formulations of nonlinear integer problems. *Manag. Sci.* **22**, 455–460 (1975)
112. F. Glover, Tabu search—Part I. *ORSA J. Comput.* **1**, 190–206 (1989)
113. F. Glover, Tabu search—Part II. *ORSA J. Comput.* **2**, 4–32 (1989)
114. F. Glover, Scatter search and path relinking, in *New Ideas in Optimization*, ed. by D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, K.V. Price (McGraw Hill, Maidenhead, 1999), pp. 297–316
115. F. Glover, M. Laguna, *Tabu Search*. (Kluwer Academic Publ., Norwell, 1997)
116. M.X. Goemans, D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM* **42**, 1115–1145 (1995)
117. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, Wokingham, 1989)
118. G.W. Graves, A.B. Whinston, An algorithm for the quadratic assignment problem. *Manag. Sci.* **17**, 453–471 (1970)
119. H. Greenberg, A quadratic assignment problem without column constraints. *Naval Res. Logist. Q.* **16**, 417–422 (1969)
120. S.W. Hadley, Continuous optimization approaches for the quadratic assignment problem. PhD thesis, University of Waterloo, Ontario, 1989
121. S.W. Hadley, F. Rendl, H. Wolkowicz, Bounds for the quadratic assignment problem using continuous optimization techniques, in *Proceedings of the 1-st Integer Programming and Combinatorial Optimization Conference (IPCO)*, University of Waterloo Press, Waterloo, 1990, pp. 237–248
122. S.W. Hadley, F. Rendl, H. Wolkowicz, A new lower bound via projection for the quadratic assignment problem. *Math. Oper. Res.* **17**, 727–739 (1992)
123. S.W. Hadley, F. Rendl, H. Wolkowicz, Nonsymmetric quadratic assignment problems and the Hoffman-Wielandt inequality. *Linear Algebra Appl.* **58**, 109–124 (1992)
124. P.M. Hahn, T. Grant, Lower bounds for the quadratic assignment problem based upon a dual formulation. *Oper. Res.* **46**, 912–922 (1998)
125. P.M. Hahn, T. Grant, N. Hall, Solution of the quadratic assignment problem using the Hungarian method. *Eur. J. Oper. Res.* **108**, 629–640 (1998)
126. P.M. Hahn, Y.-R. Zhu, M. Guignard, W.L. Hightower, M.J. Saltzman, A level-3 reformulation-linearization technique-based bound for the quadratic assignment problem. *INFORMS J. Comput.* **24**, 202–209 (2012)
127. G.G. Hardy, J.E. Littlewood, G. Pólya, *Inequalities* (Cambridge University Press, New York, 1952)
128. M. Hanan, J.M. Kurtzberg, A review of the placement and quadratic assignment problems. *SIAM Rev.* **14**, 324–342 (1972)
129. D.R. Heffley, Assigning runners to a relay team, in *Optimal Strategies in Sports*, ed. by S.P. Ladany, R.E. Machol (North-Holland, Amsterdam, 1977), pp. 169–171
130. J.H. Holland, *Adaptation in Natural and Artificial Systems* (University of Michigan Press, Ann Arbor, 1975)
131. L.J. Hubert, *Assignment Methods in Combinatorial Data Analysis* (Marcel Dekker, New York, 1987)
132. T. James, C. Rego, F. Glover, Multistart tabu search and diversification strategies for the quadratic assignment problem. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **39**, 579–596 (2009)
133. D.S. Johnson, C.H. Papadimitriou, M. Yannakakis, How easy is local search, *J. Comput. Syst. Sci.* **37**, 79–100 (1988)
134. T.A. Johnson, New linear programming-based solution procedures for the quadratic assignment problem. Ph.D. Thesis, Clemson University, SC, 1992

135. M. Jünger, *Polyhedral Combinatorics and the Acyclic Subdigraph Problem* (Heldermann Verlag, Berlin, 1985)
136. M. Jünger, V. Kaibel, A basic study of the QAP polytope. Technical Report 96.215, Institut für Informatik, Universität zu Köln, Germany, 1996
137. M. Jünger, V. Kaibel, On the SQAP polytope. SIAM J. Optim. **11**, 444–463 (2000)
138. V. Kaibel, Polyhedral combinatorics of the quadratic assignment problem. Ph.D. Thesis, Universität zu Köln, Germany, 1997
139. S.E. Karisch, Nonlinear approaches for quadratic assignment and graph partition problems. Ph.D. Thesis, Graz University of Technology, Austria, 1995
140. S.E. Karisch, E. Çela, J. Clausen, T. Espersen, A dual framework for lower bounds of the quadratic assignment problem based on linearization. Computing **63**, 351–403 (1999)
141. S.E. Karisch, F. Rendl, Lower bounds for the quadratic assignment problem via triangle decompositions. Math. Program. **71**, 137–151 (1995)
142. R.M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, ed. by R.E. Miller, J.W. Thatcher (Plenum, New York, 1972), pp. 85–103
143. L. Kaufman, F. Broeckx, An algorithm for the quadratic assignment problem using Benders' decomposition. Eur. J. Oper. Res. **2**, 204–211 (1978)
144. B. Kernighan, S. Lin, An efficient heuristic procedure for partitioning graphs. Bell Syst. J. **49**, 291–307 (1972)
145. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing. Science **220**, 671–680 (1983)
146. T.C. Koopmans, M.J. Beckmann, Assignment problems and the location of economic activities. *Econometrica* **25**, 53–76 (1957)
147. J. Krarup, P.M. Pruzan, Computer-aided layout design. Math. Program. Study **9**, 75–94 (1978)
148. A.V. Krushevski, The linear programming problem on a permutation group, in *Proceedings of the Seminar on Methods of Mathematical Modeling and Theory of Electrical Circuits*, vol. 3, Institute of Cybernetics of the Academy of Sciences of Ukraine, Kiev, 1964, pp. 364–371 (Russian)
149. P.J.M. van Laarhoven, E.H.L. Aarts, *Simulated Annealing: Theory and Applications* (D. Reidel Publishing Company, Dordrecht, 1988)
150. A.M. Land, A problem of assignment with interrelated costs. Oper. Res. Q. **14**, 185–198 (1963)
151. G. Laporte, H. Mercure, Balancing hydraulic turbine runners: a quadratic assignment problem. Eur. J. Oper. Res. **35**, 378–382 (1988)
152. E.L. Lawler, The quadratic assignment problem. Manag. Sci. **9**, 586–599 (1963)
153. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (eds.), *The Traveling Salesman Problem* (Wiley, Chichester, 1985)
154. T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout* (Wiley, Chichester, 1990)
155. W. Leontief, *Input-Output Economics* (Oxford University Press, New York, 1966)
156. Y. Li, P.M. Pardalos, Generating quadratic assignment test problems with known optimal permutations. Comput. Optim. Appl. **1**, 163–184 (1992)
157. Y. Li, P.M. Pardalos, K.G. Ramakrishnan, M.G.C. Resende, Lower bounds for the quadratic assignment problem. Ann. Oper. Res. **50**, 387–410 (1994)
158. Y. Li, P.M. Pardalos, M.G.C. Resende, A greedy randomized adaptive search procedure for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, ed. by P. Pardalos, H. Wolkowicz. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (AMS, Providence, 1994), pp. 237–261
159. M.H. Lim, Y. Yuan, S. Omatsu, Extensive testing of a hybrid genetic algorithm for solving quadratic assignment problems. Comput. Optim. Appl. **23**, 47–64 (2002)
160. E.M. Loiola, N.M. Maia de Abreu, P.O. Boaventura-Netto, P.M. Hahn, T. Querido, A survey for the quadratic assignment problem. Eur. J. Oper. Res. **176**, 657–690 (2007)
161. L. Lovász, A. Schrijver, Cones of matrices and set functions and 0–1 optimization. SIAM J. Optim. **1**, 166–190 (1991)

162. E.J. McCormick, *Human Factors Engineering* (McGraw-Hill, New York, 1970)
163. F. Malucelli, Quadratic assignment problems: solution methods and applications. Ph.D. Thesis, Dipartimento di Informatica, Università di Pisa, Italy, 1993
164. F. Malucelli, A polynomially solvable class of quadratic semi-assignment problems. Eur. J. Oper. Res. **91**, 619–622 (1996)
165. F. Malucelli, D. Pretolani, Lower bounds for the quadratic semi-assignment problem. Eur. J. Oper. Res. **83**, 365–375 (1995)
166. T. Mautor, C. Roucairol, A new exact algorithm for the solution of quadratic assignment problems. Discrete Appl. Math. **55**, 281–293 (1994)
167. T. Mavridou, P.M. Pardalos, L.S. Pitsoulis, M.G.C. Resende, A GRASP for the biquadratic assignment problem. Eur. J. Oper. Res. **105**, 613–621 (1998)
168. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equations of state calculations by fast computing machines. J. Chem. Phys. **21**, 1087–1092 (1953)
169. I.Z. Milis, V.F. Magirou, A Lagrangean relaxation algorithm for sparse quadratic assignment problems. Oper. Res. Lett. **17**, 69–76 (1995)
170. G. Miranda, H.P.L. Luna, G.R. Mateus, R.P.M. Ferreira, A performance guarantee heuristic for electronic components placement problems including thermal effects. Comput. Oper. Res. **32**, 2937–2957 (2005)
171. P.B. Mirchandani, T. Obata, Locational decisions with interactions between facilities: the quadratic assignment problem a review. Working Paper Ps-79-1, Rensselaer Polytechnic Institute, Troy, New York, May 1979
172. L. Mirsky, The spread of a matrix. Mathematika **3**, 127–130 (1956)
173. A. Misevicius, An improved hybrid optimization algorithm for the quadratic assignment problem. Math. Model. Anal. **9**, 149–168 (2004)
174. H.D. Mittelmann, J. Peng, Estimating bounds for quadratic assignment problems with Hamming and Manhattan distance matrices based on semidefinite programming. SIAM J. Optim. **20**, 3408–3426 (2010)
175. N. Mladenović, P. Hansen, Variable neighborhood search. Comput. Oper. Res. **24**, 1097–1100 (1997)
176. J. Mosevich, Balancing hydraulic turbine runners—a discrete combinatorial optimization problem. Eur. J. Oper. Res. **26**, 202–204 (1986)
177. K.A. Murthy, P. Pardalos, Y. Li, A local search algorithm for the quadratic assignment problem. Informatica **3**, 524–538 (1992)
178. H. Müller-Merbach, *Optimale Reihenfolgen* (Springer, Berlin, 1970), pp. 158–171
179. C.E. Nugent, T.E. Vollmann, J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations. J. Oper. Res. **16**, 150–173 (1969)
180. M.W. Padberg, M.P. Rijal, *Location, Scheduling, Design and Integer Programming* (Kluwer Academic, Boston, 1996)
181. G.S. Palubeckis, A generator of test quadratic assignment problems with known optimal solution. USSR Comput. Math. Math. Phys. **28**, 97–98 (1988) [Translated from Zh. Vychisl. Mat. Fiz. **28**, 1740–1743 (1988)]
182. G.S. Palubeckis, The use of special graphs for obtaining lower bounds in the geometric quadratic assignment problem. Informatica **8**, 377–400 (1997)
183. G.S. Palubeckis, Generating hard test instances with known optimal solution for the rectilinear quadratic assignment problem. J. Global Optim. **15**, 127–156 (1999)
184. G.S. Palubeckis, An algorithm for construction of test cases for the quadratic assignment problem. Informatica **11**, 281–296 (2000)
185. C.H. Papadimitriou, D. Wolfe, The complexity of facets resolved, in *Proceedings of the 25-th Annual IEEE Symposium on the Foundations of Computer Science (FOCS)*, Portland, USA, 1985, pp. 74–78
186. P. Pardalos, J. Crouse, A parallel algorithm for the quadratic assignment problem, in *Proceedings of the Supercomputing Conference 1989*, ACM, 1989, pp. 351–360
187. P. Pardalos, F. Rendl, H. Wolkowicz, The quadratic assignment problem: a survey and recent developments, in *Quadratic Assignment and Related Problems*, ed. by P. Pardalos,

- H. Wolkowicz. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (AMS, Providence, 1994), pp. 1–42
188. P.M. Pardalos, L.S. Pitsoulis, M.G.C. Resende, A parallel GRASP implementation for solving the quadratic assignment problem, in *Parallel Algorithms for Irregular Problems: State of the Art*, ed. by A. Ferreira, J.D.P. Rolim (Kluwer Academic, Norwell, 1995), pp. 115–133
189. P.M. Pardalos, K.G. Ramakrishnan, M.G.C. Resende, Y. Li, Implementation of a variable reduction based lower bound in a branch and bound algorithm for the quadratic assignment problem. *SIAM J. Optim.* **7**, 280–294 (1997)
190. P.M. Pardalos, J. Xue, The maximum clique problem. *J. Global Optim.* **4**, 301–328 (1994)
191. J. Peng, H.D. Mittelmann, X. Li, A new relaxation framework for quadratic assignment problems based on matrix splitting. *Math. Program. Comput.* **2**, 59–77 (2010)
192. L. Pitsoulis, P.M. Pardalos, Quadratic assignment problem, in *Encyclopedia of Optimization*, 2nd edn., ed. by Ch.A. Floudas, P.M. Pardalos (Springer, Berlin, 2009), pp. 3119–3149
193. J. Povh, F. Rendl, Copositive and semidefinite relaxations of the quadratic assignment problem. *J. Discrete Optim.* **6**, 231–241 (2009)
194. M. Queyranne, Performance ratio of heuristics for triangle inequality quadratic assignment problems. *Oper. Res. Lett.* **4**, 231–234 (1986)
195. A.S. Ramkumar, S.G. Ponnambalam, N. Jawahar, A population-based hybrid ant system for quadratic assignment formulations in facility layout design. *Int. J. Adv. Manuf. Technol.* **44**, 548–558 (2008)
196. G. Reinelt, *The Linear Ordering Problem: Algorithms and Applications* (Heldermann Verlag, Berlin, 1985)
197. F. Rendl, Ranking scalar products to improve bounds for the quadratic assignment problem. *Eur. J. Oper. Res.* **20**, 363–372 (1985)
198. F. Rendl, Quadratic assignment problems on series-parallel digraphs. *Z. Oper. Res.* **30**, 161–173 (1986)
199. F. Rendl, R. Sotirov, Bounds for the quadratic assignment problem using the bundle method. *Math. Program. B* **109**, 505–524 (2007)
200. F. Rendl, H. Wolkowicz, Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem. *Math. Program.* **53**, 63–78 (1992)
201. M.G.C. Resende, K.G. Ramakrishnan, Z. Drezner, Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. *Oper. Res.* **43**, 781–791 (1995)
202. W.T. Rhee, A note on asymptotic properties of the quadratic assignment problem. *Oper. Res. Lett.* **7**, 197–200 (1988)
203. W.T. Rhee, Stochastic analysis of the quadratic assignment problem. *Math. Oper. Res.* **16**, 223–239 (1991)
204. J.M. Rodríguez, F.C. MacPhee, D.J. Bonham, V.C. Bhavsar, Solving the quadratic assignment and dynamic plant layout problems using a new hybrid meta-heuristic approach. *Int. J. High Perform. Comput. Netw.* **4**, 286–294 (2006)
205. C. Roucairol, A reduction method for quadratic assignment problems. *Oper. Res. Verfahren* **32**, 183–187 (1979)
206. C. Roucairol, A parallel branch and bound algorithm for the quadratic assignment problem. *Discrete Appl. Math.* **18**, 221–225 (1987)
207. F. Roupin, Semidefinite relaxations of the quadratic assignment problem in a Lagrangian framework. *Int. J. Math. Oper. Res.* **1**, 144–162 (2009)
208. S. Sahni, T. Gonzalez, P-complete approximation problems. *J. Assoc. Comput. Mach.* **23**, 555–565 (1976)
209. A. Schäffer, M. Yannakakis, Simple local search problems that are hard to solve. *SIAM J. Comput.* **20**, 56–87 (1991)
210. J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem. *ORSA J. Comput.* **2**, 33–45 (1990)
211. J. Skorin-Kapov, Extensions of tabu search adaptation to the quadratic assignment problem. *Comput. Oper. Res.* **21**, 855–865 (1994)

212. H.S. Stone, Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Trans. Softw. Eng.* **3**, 85–93 (1977)
213. Y.G. Stoyan, V.Z. Sokolovskii, S.V. Yakovlev, A method for balancing discretely distributed masses under rotation. *Energomashinostroenia* **2**, 4–5 (1982) (in Russian)
214. Th. Stützle, S. Fernandes, New benchmark instances for the QAP and the experimental analysis of algorithms, in *EvoCOP 2004*, ed. by J. Gottlieb, G.R. Raidl. Lecture Notes in Computer Science, vol. 3004 (Springer, Berlin, 2004), pp. 199–209
215. W. Szpankowski, Combinatorial optimization problems for which almost every algorithm is asymptotically optimal! *Optimization* **33**, 359–367 (1995)
216. E. Taillard, Robust tabu search for the quadratic assignment problem. *Parallel Comput.* **17**, 443–455 (1991)
217. D.M. Tate and A.E. Smith, A genetic approach to the quadratic assignment problem. *Comput. Oper. Res.* **22**, 73–83 (1995)
218. L.-Y. Tseng, S.-C. Liang, A hybrid metaheuristic for the quadratic assignment problem. *Comput. Optim. Appl.* **34**, 85–113 (2006)
219. S. Tsutsui, Parallel ant colony optimization for the quadratic assignment problems with symmetric multi processing, in *ANTS 2008*, ed. by M. Dorigo et al. Lecture Notes in Computer Science, vol. 5217 (Springer, Berlin, 2008), pp. 363–370
220. I. Ugi, J. Bauer, J. Friedrich, J. Gasteiger, C. Jochum, W. Schubert, Neue Anwendungsbereiche für Computer in der Chemie. *Angewandte Chemie* **91** (1979), 99–111
221. S. Voß, Heuristics for nonlinear assignment problems, in *Nonlinear Assignment Problems*, ed. by M. Desrochers, J.M. Rousseau. Lecture Notes in Economics and Mathematical Systems, vol. 386 (Springer, Berlin, 2000), pp. 137–152
222. H. Wiener, Structural determination of paraffin boiling points. *J. Am. Chem. Soc.* **69**, 17–20 (1947)
223. M.R. Wilhelm, T.L. Ward, Solving quadratic assignment problems by simulated annealing. *IEEE Trans.* **19**, 107–119 (1987)
224. T. Winter, U. Zimmermann, Dispatch of trams in storage yards. *Ann. Oper. Res.* **96**, 287–315 (2000)
225. X. Wu, H.D. Mittelmann, X. Wang, J. Wang, On computation of performance bounds of optimal index assignment, in *Data Compression Conference (DCC) 2010*, IEEE, pp. 189–198. doi:10.1109/DCC.2010.24
226. Q. Zhao, Semidefinite programming for assignment and partitioning problems. Ph.D. Thesis, University of Waterloo, Ontario, 1996
227. S.B. Zahir Azami, P. Duhamel, O. Rioul, Joint source-channel coding: panorama of methods, in Proceedings of the CNES Workshop on Data Compression, Toulouse (France), November 1996, pp. 1232–1254, <http://www.comelec.enst.fr/~rioul/research/dvips/csccp.ps>
228. H. Zhang, C. Beltran-Royo, L. Ma, Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers. *Ann. Oper. Res. publ.* (2012), DOI 10.1007/s10479-012-1079-4, <http://www.optimization-online.org>
229. Q. Zhao, S.E. Karisch, F. Rendl, H. Wolkowicz, Semidefinite relaxations for the quadratic assignment problem. *J. Combin. Optim.* **2**, 71–109 (1998)

Reactive Business Intelligence: Combining the Power of Optimization with Machine Learning

Roberto Battiti and Mauro Brunato

Contents

1	Introduction	2816
2	Clustering.....	2819
2.1	Hard and Soft Clustering.....	2820
2.2	Visualization by Linear Transformations.....	2822
2.3	Visualizing Graphs and Networks by Nonlinear Maps.....	2823
3	Supervised Learning.....	2824
3.1	Learning from Labeled Examples: Minimization and Generalization.....	2825
3.2	Nearest Neighbors Methods.....	2826
3.3	Linear Regression.....	2826
3.4	Multilayer Perceptrons.....	2828
3.5	Statistical Learning Theory and SVMs.....	2829
4	Semi-Supervised Learning.....	2832
4.1	Graph-Based Algorithms.....	2834
4.2	Learning the Metric.....	2835
5	Text and Web Mining.....	2836
5.1	From Documents to Vectors: The Vector-Space Model.....	2837
6	Collaborative Filtering and Recommendation.....	2838
7	Multi-Objective Optimization and Pareto Optimality.....	2840
8	Conclusion.....	2843
	Cross-References.....	2844
	Recommended Reading.....	2844

Abstract

Learning and Intelligent Optimization (LION) and Reactive business intelligence (RBI) consider techniques based on data analysis, model building, and feedback by the users for the realization of effective computer-supported tools for business analytics and decision making.

R. Battiti (✉) • M. Brunato

LION Lab, Università di Trento, and Lionsolver Inc., Trento, Italy
e-mail: battiti@reactive-search.com; brunato@reactive-search.com

The systematic usage of online machine learning methods is instrumental to raise the level of automation, therefore permitting a wider adoption of real-time self-tuning systems. This development liberates the decision maker from the burden of routine operations and from the rigidity of many current BI tools.

Optimization, and Learning and Intelligent OptimizatioN (LION) methods play a crucial role for solving the various complex problems arising in nontrivial business intelligence and analytics applications and for identifying models through machine learning schemes.

This chapter positions the LION and RBI fields in the context of different related approaches and summarizes some paradigmatic applications.

1 Introduction

Business intelligence (BI) denotes an automated (computer-supported) process to systematically transform raw data originated in business activities into models and insight, which can be used to decide, act, and improve business processes and results.

The term was coined in [54] as “the ability to apprehend the interrelationships of presented facts in such a way as to guide action towards a desired goal” and later publicized under different and related keywords, starting from *decision support systems* (DSS) in the 1960s. DSS purpose was to use computers and algorithms to assist with decision making and planning. From DSS, data warehouses (integrated databases providing collection, storage, and distributions of business data), executive information systems, OLAP (online analytical processing), and business intelligence came into focus beginning in the late 1980s with the growing adoption of computers. The usage of “business intelligence”—according to H. Dresner an umbrella term to describe “concepts and methods to improve business decision making by using fact-based support systems”—became widespread the late 1990s. Related terms are *data mining* (extracting patterns from large data sets, usually part of a wider business intelligence effort), *business analytics* (the application of computer technology, optimization, and statistics to solve problems in business and industry), and *KDD* (knowledge discovery in databases [33]).

The term *reactive*, originated from the reactive search optimization framework [5, 8, 12], is related to designing optimization schemes with an automated and rapid capability to adapt by responding to stimuli, to changes in the external environment. Analogies are in complex biological systems, in which *reactivity* as response to current and past inputs is crucial in living organisms [34]. Reactive systems “live” in order to react. The automation is obtained through machine learning schemes acting in real-time, in an online manner, while the system is running. *Reactive business intelligence* (RBI) advocates a holistic approach that integrates data mining, modeling, and interactive visualization into an end-to-end discovery and continuous innovation process powered by human and automated learning [11].

The RBI methodology stands on the shoulders of the tradition of experimental science going back at least to Galileo (measurements, models, experimental validation, refinement of models) but departs from the tradition in its use of online machine learning in the process of continuous adaptation of the models. The traditional laws of nature, like the law of gravitation, are immutable, while the laws of business need to account for unexpected changes in the competition, in the users' preferences, in the available information, and in the technology. In hard science like physics no self-tuning is needed, and the cyclic process of building explanatory models converges when the model reproduces the experimental results, at least to a sufficient degree, while in business the iterated modeling and organization of activities never ends.

One of the main criticisms to traditional BI systems is that they tend to be static and difficult to change. Usually, a huge investment is required and changes are extremely costly. Managers using new information for decisions usually need to interact with technical people to realize new software, a long and costly iteration. Reactive BI starts by the assumption that change is *inherent* in the business activities and prepares for that, with growing levels of self-adaptation. Applications of neural networks and machine learning for selected business applications starting from the late eighties go in this direction [76,79], as well the autonomic management of computer systems and telecommunication networks advocated in the nineties [49,69].

The reactive trend in business intelligence is in essence related to reaching higher levels of automation, by transferring the capability to execute changes in response to business stimuli—traditionally requiring human intervention—directly to the BI system, starting from the simplest changes and progressively attacking higher and higher levels and the decision and implementation hierarchy. The human decision maker (DM) is always in charge of the final decisions, but the preparatory work to mine data and prepare potential solutions according to the DM preferences allows seamless decision processes.

Note that *reactive* denotes a property of the software system, not of the system designer, the decision maker, or the final user of the system. In fact, the system designer must act in a very proactive (anticipatory) manner to endow the software with the capability to be reactive. The main reason why a software system needs to be reactive is that businesses are confronted with a rapidly changing context and, in some cases, with dramatic events requiring very fast reactive automated responses.

The role of *optimization* in RBI is crucial for different reasons. The first and most obvious one is to solve the various complex problems arising in nontrivial business intelligence applications. Most problems are NP-hard so that optimization heuristics, often based on stochastic local search [46], are the only practical way to deliver *satisficing* solutions. A second reason is related to the intensive use of machine learning in RBI: most machine learning schemes employ optimization to identify effective and simple models. A third nontrivial reason is related to a *separation of concerns* between problem definition and problem solution. By delegating the problem solution to optimization schemes, the decision maker is free to concentrate his energies on defining the objectives in a clear, understandable, and measurable form. Furthermore, different formulations of the objectives can be

experimented with, and to increase the automation, systems can be developed to progressively *learn the problem definition* from the user feedback during interactive problem-solving sessions.

The issues of evolving problem definitions and finding approximate solutions are often neglected by traditional computer science approaches to business intelligence. Some relevant scientific results are summarized in the following paragraphs.

Starting at least from the pioneering work of Herbert Simon [67], it is recognized that human decision processes are severely constrained and that models must depart from abstract mathematical formulations of traditional economics, based on an abstract *homo economicus*, rational and narrowly self-interested in his exact optimization of clearly defined objectives. The term *bounded rationality* is used to designate rational choice that takes into account the cognitive limitations of both knowledge and cognitive capacity. Bounded rationality is concerned with the ways in which the actual decision-making process influences decisions. In particular, the modeling (problem definition) and solution processes are never ending in nontrivial situations and characterized by changes caused by the acquisition of new information, modifications of expectations, and longer time devoted to searching. More recently, interactive multi-objective optimization [37], discussed in Sect. 7, acknowledges that tentative solutions are built through an iterative process alternating phases of problem definitions, presentations of candidate solutions, evaluations by the decision maker, and modifications of his preferences. The relevance of the “test and learn” approach to business, based on systematically assembled data and analysis, and of high-performance business processes as differentiation point is discussed in [27]. Opportunities for the optimization research community to contribute to data mining are summarized in [19]. A review of optimization use for data mining focussed onto customer relationship management (CRM) is presented in [60], including feature selection, active learning, DM model selection and optimization, classification using mathematical programming, clustering, and rule and constraint discovery. Limitations of current BI technologies are discussed in [28], which advocates supporting tools for systematically capturing business requirements to be translated into optimized designs, in particular automated or semi-automated ways to help in dealing with the complexity of business data integration and processes. Reactive components capable of monitoring operational processes to allow DMs to tune their actions according to the company strategy and real-time data are proposed in [39]. Real-time business intelligence taking actions in response to analysis results is proposed in [3], while autonomic self-optimization for e-commerce is used in [2]. The integration of business intelligence and knowledge management is reviewed in [26]. Emerging trends in business analytics including integration with action and measurements in an incremental and cyclic manner, real-time BI and business performance management, and knowledge discovery in databases are considered in [50, 78].

A related and growing area is that of “Learning and Intelligent OptimizatioN” (LION); see [16] and the other proceedings of the LION conference. The large variety of heuristic algorithms for hard optimization problems raises issues related to selecting the most appropriate method, in many cases through an expensive

algorithm configuration and parameter tuning process, and subject to a steep learning curve. LION aims at designing and engineering ways of “learning” about the performance of different techniques and ways of using past experience about the algorithm behavior to improve performance in the future. Intelligent learning schemes for mining the knowledge obtained from different runs or during a single run can improve the algorithm development and design process and simplify the applications of high-performance optimization methods. Combinations of algorithms can further improve the robustness and performance of the individual components, provided that sufficient knowledge of the relationship between problem instance characteristics and algorithm performance is obtained. Most of the LION techniques are relevant for many business intelligence applications.

The following part of this chapter will mention the main research results and directions related to using optimization techniques for relevant RBI applications. A mixture of combinatorial and continuous optimization techniques, depending on the specific problem, is the characteristic of RBI, and it reflects the complexity of the real world of business. Completeness in this huge area is beyond the limited scope of this chapter. A more detailed presentation of some topics, in particular related to combining interactive visualization with problem solving, is present in the authors’ recent book [11] and in the recent paper [20].

The following description is organized around some paradigmatic problems and issues encountered in business applications, with references to some representative research papers.

2 Clustering

A first motivation to consider computer-assisted analysis and decision methods is related to the growing amount of data generated daily in modern business environments. The human decision maker represents a bandwidth bottleneck unless the raw data is reduced to a quantity which he can handle. The role of models is indeed that of describing in a summarized—and understandable—way a large number of related entities. *Clustering* is an extremely used and effective technique to reduce the number of items under consideration by grouping similar items together. Actually, giving names (labels) to different cases in a human understandable ways is an application of clustering, although in some cases names are not required. Clustering means identifying structure in the data which is often hidden and non-obvious, collecting similar or related entities together, often identifying important dimensions capturing most of the data diversity. In many cases, a prototype for each cluster (e.g., a cluster centroid) is sufficient to describe an entire cluster and to reason about it, in this manner reducing the information overload. Relevant business applications of clustering are in marketing, when huge number of customers can be reduced to a selected number of classes to be handled by specific marketing campaigns, in social network analysis and identification of communities (e.g., in the web), in collaborative recommendation, in the grouping of search results, and in the analysis of shopping items.

Clustering techniques are usually separated between bottom-up (agglomerative) and top-down techniques. There are two different contexts in clustering, depending on the source of information guiding the process. In some cases one starts from an *internal representation* of each entity (typically an M -dimensional vector \mathbf{x}_d assigned to entity d) and derives mutual dissimilarities or mutual similarities from the internal representation. In this case one can derive *prototypes* (*or centroids*) for each cluster, for example, by averaging the characteristics of the contained entities (the vectors). In other cases only an *external representation* of dissimilarities is available, and the resulting model is an undirected and weighted graph of entities connected by edges. This short summary concentrates on hard and top-down clustering, where “hard” means that each entity belongs to one and only one cluster.

The effectiveness of a clustering method depends on the *similarity metric* between items \mathbf{x} and \mathbf{y} , $\delta(\mathbf{x}, \mathbf{y})$, which needs to be strongly problem dependent. If an internal representation is present, a metric can be derived by the usual Euclidean distance:

$$\delta_E(\mathbf{x}, \mathbf{y}) = \|\mathbf{y} - \mathbf{x}\|$$

In the general case, a positive definite matrix \mathbf{M} can be determined to transform the original metric:

$$d_{ij} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)}$$

An example is the Mahalanobis distance [55].

2.1 Hard and Soft Clustering

The *hard clustering problem* consists of partitioning the entities D into k disjoint subsets $C = \{C_1, \dots, C_k\}$ to reach the following objectives:

- Minimization of the average intra-cluster dissimilarities:

$$\min \sum_{d_1, d_2 \in C_i} \delta(\mathbf{x}_{d_1}, \mathbf{x}_{d_2}). \quad (1)$$

If an internal representation is available, the cluster centroids \mathbf{p}_i can be derived as the average of the internal representation vectors over the members of the i -th cluster $\mathbf{p}_i = (1/|C_i|) \sum_{d \in C_i} \mathbf{x}_d$.

In these cases the intra-cluster distances can be measured with respect to the cluster centroid \mathbf{p}_i , obtaining the related but different minimization problems:

$$\min \sum_{d \in C_i} \delta(\mathbf{x}_d, \mathbf{p}_i). \quad (2)$$

- Maximization of inter-cluster distance. One wants the different clusters to be clearly separated.

As one can imagine, the various objectives are not always compatible, clustering is indeed a *multi-objective optimization* task. It is left to the final user to weigh the importance of achieving clusters of very similar entities *versus* achieving clusters which are well separated, also depending on the chosen number of clusters.

Divisive algorithms are among the simplest clustering algorithms. They begin with the whole set and proceed to divide it into successively smaller clusters. One simple method is to subdivide the data set into k subsets.

If one wants to have groups of entities represented by a single vector, obtaining a more *compact* way to express them, a suitable approach is to select the prototypes which minimize the average *quantization error*, the error incurred when the entities are substituted with their prototypes:

$$\text{Quantization Error} = \sum_d \|\mathbf{x}_d - \mathbf{p}_{c(d)}\|^2, \quad (3)$$

where $c(d)$ is the cluster associated with data d .

In statistics and machine learning, *k-means clustering* is a method of cluster analysis which aims at partitioning the observations into k clusters represented by *centroids* (prototypes for cluster c , denoted as \mathbf{p}_c), so that each observation belongs to the cluster with the *nearest* centroid. The iterative method to determine the prototypes in *k-means* consists of the following steps:

1. Randomly generate k clusters and determine the cluster centroids \mathbf{p}_c or directly generate k random points as cluster centroids.
2. Repeat the following steps until some convergence criterion is met: usually when the last assignment has not changed, or a maximum number of iterations has been executed.
 - a. Assign each point \mathbf{x} to the nearest cluster centroid, the one minimizing $\delta(\mathbf{x}, \mathbf{p}_c)$.
 - b. Recompute the new cluster centroid by averaging the points assigned in the previous step:

$$\mathbf{p}_c \leftarrow \frac{\sum_{\text{entities in cluster } c} \mathbf{x}}{\text{number of entities in cluster } c}.$$

K-means clustering can be seen as a skeletal version of the *expectation maximization* algorithm, a method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables. EM is an iterative method which alternates between performing an expectation (E) step, which computes the expectation of the log-likelihood evaluated using the current estimate for the latent variables, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step [57]. Scaling of EM to large databases is considered, for example, in [18].

Hard clustering and k -means are related to some classic combinatorial optimization problems. Hyper-graph partitioning for finding association rules is proposed in [42]: a weighted hyper-graph represents the relationships between items and partitioning identifies highly related items. Graph partitioning is used for clustering in [17] for web documents clusterization. RSO methods for graph and hyper-graph partitioning are considered in [10]. To mention some relevant applications, He et al. [44] considers spectral graph partitioning methods based on the normalized cut measure [66] to identify topics (communities) in the web. Efficient clustering techniques for spatial data mining (with spatial attributes related to maps, satellite images, etc.) including randomized search techniques are considered in [59]. An extended survey of clustering techniques applied to data mining, including relevant BI applications like CRM is presented in [13], and includes an updated discussion of general algorithmic issues like assessment of results, choice of number of clusters, data preparation, proximity measures, and handling of outliers.

Clustering is example of *unsupervised learning*, one searches for structure in the data without the need of a teacher labeling a set of example data. In fact, in clustering, giving names (labels) to the clusters is not part of the process and must be dealt with in a second step, if needed. The advantage of most clustering techniques is that they are simple to implement and easy to explain in a manner similar to *case-based reasoning* (CBR): a human explanation for a cluster is the prototype (centroid) associated to the cluster. In marketing, this represents the typical characteristics of a customer in a specific group.

More sophisticated clustering techniques can be obtained as a side effect of projecting data to a lower-dimensional space (usually two-dimensional) with constraints intended to enforce the similarity of points mapped to close low-dimensional positions. For example, a *self-organizing map* (SOM) is a type of artificial neural network that is trained by using unsupervised learning to produce a two-dimensional representation of the training samples, called a map. This model was introduced as an artificial neural network by Teuvo Kohonen and is also called a *Kohonen map* [51]. The extension to discrete symbol strings is presented in [52].

2.2 Visualization by Linear Transformations

In *exploratory data analysis* of entities, it is often useful to map them to two dimensions, so that they can be analyzed by our eyes. The mapping has to preserve as much as possible the relevant information present in the original data, describing *similarities and diversities* between entities. A discussion of visualization for knowledge discovery and exploration is present in [25], with particular emphasis on hierarchical clustering. The integration of constraint-based and multidimensional mining into a unified interactive and exploratory framework is present in [43].

To establish a suitable notation, let the n entities be characterized by some mutual dissimilarities δ_{ij} . Some of the dissimilarities may be unknown. An appropriate model is an undirected graph, in which each entity is represented by a node, and a connection with weight δ_{ij} is present between two nodes, if and only if a distance d_{ij} is defined for the corresponding entities. The set of edges in the graph is denoted by E .

A way to distinguish the different contexts has to do with the *level of supervision* in the visualization, that is, the type of hints given to the process. The type of supervision ranges from a purely *unsupervised* approach (only coordinates are given) to a *supervised* approach (relationships or dissimilarities are fully specified) and to mixed approaches combining unsupervised exploration in the vector space of coordinated and labeling methods.

Methods derived from linear algebra include traditional techniques like principal component analysis (PCA), see [62] for an application to vendor selection, and Fisher linear discrimination, a critical assessment is present in [31]. The following part concentrates on the methods based on more general *nonlinear* mappings which are more challenging and requiring advanced optimization methods.

2.3 Visualizing Graphs and Networks by Nonlinear Maps

An appropriate model for this situation is an *undirected weighted graph* $G(V, E)$, given by a set of vertices (or nodes) V , and edges $E \subset V \times V$, in which each entity is represented by a node, and a connection (i, j) labeled d_{ij} is present between two nodes if and only if a dissimilarity is defined for the corresponding entities.

The problem is the following: given a set of (positive) dissimilarities d_{ij} between items, find the two-dimensional or three-dimensional coordinates p_i for all items that provide a convenient placement of such items on the plane. The simplest approach is *stress minimization*. A straightforward error measure can be defined to quantify by how much the distances in the plane are different with respect to the original dissimilarities. For simplicity, consider a two-dimensional visualization.

Let $\delta_{ij} = \|p_i - p_j\|$ be the distance between the coordinates of items i and j on the plane. A natural “global mapping error” can be defined as the sum of the individual errors, squared:

$$\sum_{(i,j) \in E} (d_{ij} - \delta_{ij})^2.$$

Additional flexibility can be obtained by adding an arbitrary weight w_{ij} representing the impact that an individual error has on the overall stress:

$$\text{Global Mapping Error} = \text{Stress} = \sum_{(i,j) \in E} w_{ij} (d_{ij} - \delta_{ij})^2. \quad (4)$$

For example, if $w_{ij} = 1/d_{ij}^2$, one considers the *relative errors* $(\delta_{ij} - d_{ij})/d_{ij}$ instead of the absolute errors.

Obtaining low error values means that many distances tend to be rather close to the original ones. In other words, the problem is now to *minimize* the global mapping error measure by changing the point positions p_i . There is a physical model related to minimizing the above *global mapping error*, where points are connected by springs of strength related to the mutual similarity, and this explains the widely used term *stress* to denote the function to be minimized. This model leads to what is called the *force-directed approach* for drawing graphs. Methods based on this approach consist of two main components. The first is the *model that quantifies the quality of a drawing* (or of the two-dimensional map if you prefer a more technical term). The second is an *optimization method* for computing a drawing that is locally optimal with respect to this model. The resulting final layout brings the system to equilibrium, where the total force on each vertex is zero or, equivalently, the potential energy is locally minimal with respect to the vertex positions.

Introducing additional terms in the global mapping error function is in some cases crucial to obtain smoother and less entangled drawings. For example, one can complete the distance matrix by the *shortest path* calculation. All distances between nodes which are not directly connected are set equal to the *shortest path* between such nodes.

More complex functions to be minimized for graph layout consider *additional aesthetic criteria*, like minimizing the number of edge crossings, or guaranteeing a certain minim angle between edges connected to a node (small angles make readability difficult), or allowing curved edges. A complete enumeration is out of the scope of this introductory chapter. In all cases, after defining in quantitative terms a suitable compromise between the desirable aesthetic criteria, one has to search for an effective minimization algorithm, in most cases looking for an approximate but fast solution. For example, graph layout is considered for visualizing online activity in e-commerce in [30] and for visualizing semantic meta-data and ontologies in information management systems in [58].

3 Supervised Learning

A separate class of business intelligence models is the so-called *predictive models*. The objective is to predict an output value (e.g., a classification). Predictive models can be used in *what-if* scenario analysis, to reason about the effects of different possible decisions. In many cases the model is built by starting from labeled data. In this case one talks about *supervised learning*: a system is trained by a supervisor (teacher) giving *labeled examples*. Each example is a vector of input parameters x called *features* with an associated output label y . Features are the individual measurable properties of the phenomena being observed.

In *classification* (recognition of the class of a specific object described by features x), the output is a suitable code for the class. To take care of indecision, a

real-valued output ranging between zero and one can be interpreted as the posterior probability for a given class, given the input values. In *regression*, the output is from the beginning of a real number, and the objective is to model the relationship between a dependent variable (the output y) and one or more independent variables (the input features x).

3.1 Learning from Labeled Examples: Minimization and Generalization

By using the examples, a supervised learning method builds an association $y = \hat{f}(x)$ between input x and output y . The association is selected within a flexible model $\hat{f}(x; w)$, where the flexibility is given by some tunable parameters—or weights— w , and the best choice for the value w^* of the parameter depends on the examples analyzed.

The learned model has to work correctly on the examples in the *training set*, and this requires that the error between the correct answer (given by the example label) and the outcome predicted by the model is minimized. Supervised learning therefore becomes *minimization of a specific error function*, depending on parameters w . If the function is differentiable, a simple but effective approach is the traditional *gradient descent*; this is in fact the popular technique in neural networks known as learning by *back-propagation* of the error [79]. If the weights are quantized (actually real numbers are always discretized when represented as floating point numbers in a computer) and a suitable representation like Gray coding is used, combinatorial techniques like reactive search optimization method can be used for the training process [9].

Now, minimization of an error function is a first critical component, but not the only one. If the complexity (the flexibility, the number of tunable parameters) of the model is too large, learning the examples with zero errors becomes trivial, but predicting outputs for new data may fail brutally. This is related to the *bias-variance* dilemma and requires care in model selection or minimization of a weighted combination of model error plus model complexity. The preference of simple models to avoid overcomplicated models has also been called *Occam's razor*, referring to “shaving away” unnecessary complications in theories.

A summary of the main techniques used in supervised learning, with a focus on optimization techniques, follows. To fix the notation, a training set of ℓ tuples is considered, where each tuple is of the form (x_i, y_i) , $i = 1, \dots, \ell$, $x_i \in R^d$ being a vector of input parameter values, y_i being the measured outcome to be learned by the algorithm.

It is useful to distinguish between two families of methods for supervised classification. *Generative methods* try to model the process by which the measured data x are generated by the different classes y . In other words, one learns a class-conditional density $p(x|y)$. Then, given a fresh measurements, an assignment can be made by maximizing the posterior probability of a class given a measurement,

obtained by Bayes theorem:

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)} \quad (5)$$

where $p(x|y)$ is known as the likelihood of the data and $p(y)$ is the prior probability, which reflect the probability of the outcome before any measure is performed.

Discriminative algorithms do not attempt at modeling the data generation process, they just aim at directly estimating $p(y|x)$, a problem which is in some cases simpler than the two-step process implied by generative methods. Multilayer perceptron neural networks, as well as support vector machines (SVM) are examples of discriminative methods.

3.2 Nearest Neighbors Methods

A basic form of learning, a.k.a. *instance-based learning* or *case-based* or *memory-based*, works as follows. The examples are stored and no action is taken until a new input pattern demands an output value. These systems are called *lazy learners*: they do nothing but storing the examples until the teacher interrogates them. When a new input pattern arrives, the memory is searched for examples which are *near* the new pattern, and the output is obtained by retrieving the stored outputs of the close patterns.

A simple version is that the output for the new input is simply that of the *closest* example in memory. A more robust and flexible technique considers a set of k nearest neighbors instead of one, called *k -nearest-neighbors* (KNN). In the *weighted k -nearest-neighbors* technique (WKNN), the weights determining the linear combination to obtain the predicted output as a function of the nearest neighbors output depend on the distance. The WKNN algorithm is simple to implement, and it often achieves low estimation errors. Its main drawbacks are the massive amounts of memory required and the computational cost of the testing phase. A way to reduce the amount of memory required considers clustering the examples and then storing only the prototypes (centroids) of the identified clusters.

Kernel methods and locally weighted regression can be seen as flexible and smooth generalizations of the nearest-neighbors idea; instead of applying a brute exclusion of the distant points, all points contribute to the output but with a significance (“weight”) related to their distance from the query point.

3.3 Linear Regression

A linear dependence of the output from the input features is a widely used model. The hypothesis of a linear dependence of the outcomes on the input parameters can

be expressed as

$$y_i = w^T \cdot x_i + \varepsilon_i,$$

where $w = (w_1, \dots, w_d)$ is a (column) vector of *weights* to be determined and ε_i is the error, which is normally assumed to have Gaussian distribution. The weight vector w must be found so that the linear function

$$\hat{f}(x) = w^T \cdot x \quad (6)$$

approximates as closely as possible our experimental data. This goal is achieved by finding the vector w^* that minimizes the sum of the squared errors:

$$\text{error}(w) = \sum_{i=1}^{\ell} (w^T \cdot x_i - y_i)^2. \quad (7)$$

By equating the gradient to zero, one obtains the following optimal value for w :

$$w^* = (X^T X)^{-1} X^T y \quad (8)$$

where $y = (y_1, \dots, y_\ell)$ and X is the matrix whose rows are the x_i vectors. The matrix $(X^T X)^{-1} X^T$ is called *pseudo-inverse*.

3.3.1 Dealing with Nonlinear Dependencies

A function in the form $f(x) = w^T x$ is too restrictive to be useful in most cases. To admit nonlinear dependencies while remaining in the (easier) context of linear least-squares approximations, the linear model can be applied to *nonlinear features* calculated from the raw input data instead of the original input.

It is possible to define a set of functions

$$\phi_1, \dots, \phi_n : R^d \rightarrow R^n$$

that map the parameters space into some more complex space, in order to apply the linear regression to the vector $\phi(x) = (\phi_1(x), \dots, \phi_n(x))$ rather than to x directly.

More precisely, a dependence will be given by a scalar product between a vector of weights w and a vector of features $\phi(x)$, as follows:

$$\hat{f}(x) = w^T \cdot \phi(x).$$

The output is a weighted sum of the features. Let $x'_i = \phi(x_i)$, $i = 1, \dots, \ell$, be the transformations of the training input tuples x_i . If X' is the matrix whose rows are the x'_i vectors, then the optimal weights with respect to the least-squares approximation are computed as above:

$$w^* = (X'^T X')^{-1} X'^T y. \quad (9)$$

3.3.2 Preventing Numerical Instabilities

When the number of examples is large, Eq. (9) is the solution of a linear system in the overdetermined case (more linear equations than variables). In particular, matrix $X^T X$ must be nonsingular, and this can only happen if the training set points x_1, \dots, x_ℓ do not lie in a proper subspace of \mathbb{R}^d . In many cases, even though $X^T X$ is invertible, the distribution of the training points is not generic enough to make it *stable*. *Stability* here means that small perturbations of the sample points lead to small changes in the results. The standard mathematical tool to ensure numerical stability consists of the addition of a *regularization* term to the (least squares) error function to be minimized:

$$\text{error}(w; \lambda) = \sum_{i=1}^{\ell} (w^T \cdot x_i - y_i)^2 + \lambda w^T \cdot w.$$

The minimization with respect to w leads to the following:

$$w^* = (\lambda I + X^T X)^{-1} X^T y.$$

The insertion of a small diagonal term makes the inversion more robust. The theory justifying the approach is based on *Tichonov regularization*, which is the most commonly used method for curing *ill-posed* problems [71].

3.3.3 Linear Regression for Classification

Moving to *classification* problems, let the outcome variable be two-valued (e.g., ± 1). In this case, linear functions can be used as discriminants; the idea is to have a hyperplane defined by a vector w separating the two classes. The goal of the training procedure becomes that of finding the best coefficient vector w so that the classification procedure:

$$y = \begin{cases} +1 & \text{if } w^T \cdot x \geq 0 \\ -1 & \text{otherwise} \end{cases} \quad (10)$$

performs the best classification. Depending on the problem, the criterion for deciding the “best” classification changes.

3.4 Multilayer Perceptrons

A multilayer perceptron (MLP) neural network, capable on nonlinear models, is composed of a large number of highly interconnected units (*neurons*) organized in layers with a feed-forward information flow. Many training techniques have been

proposed for determining the weights. An example is the *one-step-secant* (OSS) method with fast line searches using second-order derivative information described in [4]. The time complexity of the learning phase is usually high, although it varies greatly according to the heuristic adopted for calculating the weights (usually iterative) and to the halting condition. A large number of passes is required, each involving computation of the outcome for every training set point, followed by modification of weights. On the other hand, calculating the outcome of an MLP neural network is a straightforward procedure.

3.5 Statistical Learning Theory and SVMs

As mentioned before, minimizing the error on a set of examples is not the only objective of a statistically sound learning algorithm, also the modeling architecture has to be considered. Statistical Learning Theory [74] provides mathematical tools for *deriving unknown functional dependencies* on the basis of observations. A shift of paradigm occurred in statistics starting from the sixties: previously, following Fisher's research in the 1920–1930s, in order to derive a functional dependency from observations one had to know the detailed form of the desired dependency and to determine only the values of a finite number of *parameters* from the experimental data. The new paradigm does not require the detailed knowledge and proves that some general properties of the set of functions to which the unknown dependency belongs are sufficient to estimate the dependency from the data. *Nonparametric techniques* is a term used for these flexible models, which can be used even if one does not know a detailed form of the input–output function.

A brief summary of the main methodological points of Statistical Learning Theory is useful to motivate the use of SVMs as a learning mechanism. Let $P(x, y)$ be the unknown probability distribution from which the examples are drawn. The learning task is to learn the mapping $x_i \rightarrow y_i$ by determining the values of the parameters of a function $f(x, w)$. The function $f(x, w)$ is called *hypothesis*, the set $\{f(x, w), w \in \mathcal{W}\}$ is called the *hypothesis space* and denoted by \mathcal{H} , and \mathcal{W} is the set of abstract parameters. A choice of the parameter $w \in \mathcal{W}$, based on the labeled examples, determines a “trained machine.”

The *expected test error* or *expected risk* of a trained machine for the classification case is

$$R(w) = \int \|y - f(x, w)\| dP(x, y), \quad (11)$$

while the *empirical risk* $R_{\text{emp}}(w)$ is the mean error rate measured on the training set:

$$R_{\text{emp}}(w) = \frac{1}{\ell} \sum_{i=1}^{\ell} \|y_i - f(x_i, w)\|. \quad (12)$$

The classical learning method is based on the *empirical risk minimization* (ERM) inductive principle: one approximates the function $f(x, w^*)$ which minimizes the risk in (11) with the function $f(x, \hat{w})$ which minimizes the empirical risk in (12).

The rationale for the ERM principle is that, if R_{emp} converges to R in probability (as guaranteed by the law of large numbers), the minimum of R_{emp} may converge to the minimum of R . If this does not hold, the ERM principle is said to be *not consistent*.

As shown by Vapnik and Chervonenkis [75], consistency holds if and only if convergence in probability of R_{emp} to R is *uniform*, meaning that as the training set increases the probability that $R_{\text{emp}}(w)$ approximates $R(w)$ uniformly tends to 1 on the whole \mathcal{W} . Necessary and sufficient conditions for the consistency of the ERM principle is the finiteness of the *Vapnik-Chervonenkis dimension* (VC-dimension) of the hypothesis space \mathcal{H} .

The VC-dimension of the hypothesis space \mathcal{H} is, loosely speaking, the largest number of examples that can be separated into two classes in all possible ways by the set of functions $f(x, w)$. The VC-dimension h measures the complexity and descriptive power of the hypothesis space and is often proportional to the number of free parameters of the model $f(x, w)$. The choice of an appropriate value of the VC-dimension h is crucial to get good generalization performance, especially when the number of data points is limited.

The method of *structural risk minimization* (SRM) has been proposed by Vapnik based on the above bound, as an attempt to overcome the problem of choosing an appropriate value of h . For the SRM principle, one starts from a nested structure of hypothesis spaces

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_n \subset \cdots \quad (13)$$

with the property that the VC-dimension $h(n)$ of the set \mathcal{H}_n is such that $h(n) \leq h(n+1)$. As the subset index n increases, the minima of the empirical risk decrease, but the term responsible for the confidence interval increases. The SRM principle chooses the subset \mathcal{H}_n for which minimizing the empirical risk yields the best bound on the actual risk. Disregarding logarithmic factors, the following problem must be solved:

$$\min_{\mathcal{H}_n} \left(R_{\text{emp}}(w) + \sqrt{\frac{h(n)}{\ell}} \right). \quad (14)$$

The SVM algorithm described in the following is based on the SRM principle, by minimizing a bound on the VC-dimension and the number of training errors at the same time.

The mathematical derivation of SVMs is summarized first for the case of a linearly separable problem. Assume that the labeled examples are linearly separable, meaning that there exist a pair (w, b) such that

$$\begin{aligned} w \cdot x + b &\geq 1 \quad \forall x \in \text{Class}_1 \\ w \cdot x + b &\leq -1 \quad \forall x \in \text{Class}_2. \end{aligned}$$

The hypothesis space contains the functions

$$f_{w,b} = \text{sign}(w \cdot x + b).$$

Because scaling the parameters (w, b) by a constant value does not change the decision surface, the following constraint is used to identify a unique pair:

$$\min_{i=1,\dots,\ell} |w \cdot x_i + b| = 1.$$

A structure on the hypothesis space can be introduced by limiting the norm of the vector w . It has been demonstrated by Vapnik that if all examples lie in an n -dimensional sphere with radius R , then the set of functions $f_{w,b} = \text{sign}(w \cdot x + b)$ with the bound $\|w\| \leq A$ has a VC-dimension h that satisfies

$$h \leq \min\{\lceil R^2 A^2 \rceil, n\} + 1.$$

The problem can be formulated as

$$\begin{aligned} & \text{Minimize}_{w,b} && \frac{1}{2} \|w\|^2 \\ & \text{subject to} && y_i(w \cdot x_i + b) \geq 1 \quad i = 1, \dots, \ell. \end{aligned}$$

The problem can be solved by using standard quadratic programming (QP) optimization tools. The dual quadratic program, after introducing a vector $\Lambda = (\lambda_1, \dots, \lambda_\ell)$ of nonnegative Lagrange multipliers corresponding to the constraints is as follows:

$$\begin{aligned} & \text{Maximize}_{\Lambda} && \Lambda \cdot \mathbf{1} - \frac{1}{2} \Lambda \cdot D \cdot \Lambda \\ & \text{subject to} && \begin{cases} \Lambda \cdot y = 0 \\ \Lambda \geq 0 \end{cases} \end{aligned} \tag{15}$$

where y is the vector containing the example classification and D is a symmetric $\ell \times \ell$ matrix with elements $D_{ij} = y_i y_j x_i \cdot x_j$.

If the hypothesis set is unchanged but the examples are not linearly separable, one introduces a penalty proportional to the constraint violation ξ_i (collected in vector Ξ), solving the following problem:

$$\begin{aligned} & \text{Minimize}_{w,b,\Xi} && \frac{1}{2} \|w\|^2 + C \left(\sum_{i=1}^{\ell} \xi_i \right)^k \\ & \text{subject to} && \begin{cases} y_i(w \cdot x_i + b) \geq 1 - \xi_i & i = 1, \dots, \ell \\ \xi_i \geq 0 & i = 1, \dots, \ell \\ \|w\|^2 \leq c_r; \end{cases} \end{aligned} \tag{16}$$

where the parameters C and k determine the cost caused by constraint violation, while c_r limits the norm of the coefficient vector. In fact, the first term to be

minimized is related to the VC-dimension, while the second is related to the empirical risk. (See the above described SRM principle). Often, k is set to 1. Extending the above techniques to nonlinear classifiers is based on mapping the input data x into a higher-dimensional vector of *features* $\varphi(x)$ and using *linear* classification in the transformed space, called the *feature space*. Large-scale business applications require scalable SVM implementations [48]. Text classification is considered in [47]. An example of SVM usage for real-time business intelligence is [77].

Obtaining models which are self-explanatory is of crucial importance in business, when the user demands a verbal or probabilistic explanation of the causes for reaching a specific result. The result can be a classification, an illness or the effects of pharmaceuticals in the medical field, the success of a marketing campaign, etc. *Learning causal Bayesian networks* from business data is a possibility. A Bayesian network (BN), or belief network, is a probabilistic graphical model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). A theoretical negative result about the possibility to learning BN structure is demonstrated in [23]. Algorithms for learning Bayesian networks from data have two components: a scoring metric and a search procedure. The scoring metric computes a score measuring the goodness of fit of the structure to the data. The search procedure tries to identify network structures with high scores. This chapter [23] shows that the search problem of identifying a Bayesian network among those where each node has at most K parents that has a relative posterior probability greater than a given constant is NP-complete, when the BDe metric is used, computing the relative posterior probability of a network structure given data. The decision theoretic approach in [45] uses the information of observational data to learn a completed partially directed acyclic graph (a DAG) using a structure learning technique and try to discover the directions of the remaining edges by means of experiment. Their adaptive algorithm assumes that experiments are performed during learning. In general, computational algorithms for learning Bayesian network (BN) structures from experimental data are still an open challenge. For example, greedy search for optimal structure identification is used in [24]. Monte Carlo techniques to generate sample of the BN structure are considered in [32].

4 Semi-Supervised Learning

In many cases labeling examples is costly. Fortunately, one can use some information from the *unlabeled* data to improve classification, or one can use some supervised data to aid unsupervised learning and clustering.

Semi-supervised learning (SSL) uses both supervised and unsupervised data to improve performance. This brief summary and its notation derive from the introductory chapter of [22] and from [80].

The standard forms of supervision are labels associated with some examples. In this case the training set X is divided into a labeled portion $X_L = \{x_1, \dots, x_l\}$, for

which labels $Y_L = \{y_1, \dots, y_l\}$ and an unlabeled portion $X_U = \{x_{l+1}, \dots, x_{l+u}\}$ are given. Other forms of supervision can be related to constraints or *hints* given to the system [1]. For example, the hints can take the form “the output function must be growing as a function of one input coordinate,” while constraints can be formulated as “these two points must be in the same class” (*must-link*) or “these two points cannot be in the same class” (*cannot-link*).

A first idea, originated in the sixties [65] related to SSL, is the so-called self-learning or *self-labeling* method where a wrapper algorithm repeatedly uses a supervised learning method. Initially, learning is executed on the labeled examples. Then, some additional unlabeled examples are labeled by using the current trained system, and learning is repeated by adding the newly labeled examples. The effect of the wrapper depends on the supervised method enclosed, and the assumptions to make it work effectively are unclear.

A context related to SSL was introduced by Vapnik as *transductive learning*. Inductive learning wants to derive a prediction function valid for arbitrary inputs, while transductive learning just aims at predicting only a fixed set of test points, by using all available information. Usually, transductive learning is based on a *labeled graph representation of the data*, labeled nodes are classified training examples, and edges represent similarity/dissimilarity relationships or constraints. A combinatorial optimization on the labels to maximize an overall consistency measure is then performed. Because of the graph representation with edges to represent similarity a graph *mincut* problem is obtained under specific assumptions [15]. An extended literature survey is presented in [82].

In general, SSL looks promising if the unsupervised information about the density $p(x)$ is useful in deriving $p(y|x)$. By analogy with the supervised learning smoothness assumption, the *semi-supervised smoothness assumption* states that if two input points x_1 and x_2 in a *high-density region* are close, the corresponding outputs y_1 and y_2 should also be close. By transitivity, if two points are linked by a path in a high-density region (they belong to the same cluster), they should be close. If points are close but in a low-density area, the requirement that outputs are similar is less stringent.

Some SSL techniques are based on encouraging the separation between classes (the decision boundaries) to pass through low-density areas, away from most data examples.

An immediate algorithm is obtained by adopting a margin-maximization algorithm like SVM and maximizing the margin for both labeled and unlabeled examples, this is called *transductive SVM* (TSVM). To the function to be minimized, one adds a term like

$$\lambda_2 \sum_{\text{unlabeled data } i} (1 - |f(x_i)|), \quad (17)$$

$f(x_i)$ being the classification function which has to be greater than 1 for one class, less than -1 for the other class. The penalty introduced in the function is of λ_2 when $f(x_i) = 0$, and it linearly becomes equal to zero when $f(x_i)$ becomes 1 or, in the other direction, when $f(x_i)$ becomes -1 (the penalty has a triangular form

centered around zero). In other words, a penalty is incurred if an unlabeled data point falls in the “gray” boundary region where $|f(x_i)| \leq 1$: therefore unlabeled data tend to guide the linear boundary away from the dense regions. The corresponding problem is not convex, and therefore robust heuristic optimization schemes have to be adopted, for example, deterministic annealing strategies starting from an easy problem and gradually transforming it into the TSVM optimization function [68] or the continuation approach of [21] following a similar paradigm of first optimizing an “ironed” version of the function and then gradually introducing finer and finer details.

4.1 Graph-Based Algorithms

The graph-based methods for semi-supervised learning are based on representing the problem as a graph, where the nodes correspond to the examples and edges are labeled with the pairwise similarity w_{ij} of two nodes i and j .

As usual, one can think in terms of similarities or in terms of dissimilarities/distances. If the distance is calculated from an initial estimate by obtaining the *minimum-path distances* between two points, this can be seen as an approximation of the geodesic *distance of two points along the manifold* of the data points.

Matrix \mathbf{W} is introduced to represent similarities: $\mathbf{W}_{ij} = w_{ij}$ if the edge is present, zero otherwise, and the diagonal degree matrix \mathbf{D} , so that $\mathbf{D}_{ii} = \sum_j w_{ij}$.

The basic method to encourage *smoothness along light edges* (smoothness when connected nodes are similar) is related to defining and using the *graph Laplacian* operator. The normalized \mathcal{L} and un-normalized combinatorial graph Laplacian operator \mathcal{L} are defined as

$$\mathcal{L} = \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2} \quad (18)$$

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \quad (19)$$

The graph Laplacian is related to the more traditional Laplace operator used for continuous functions $f(x_1, \dots, x_n)$:

$$\nabla \phi = \sum_{i=1}^n \frac{\partial^2 f}{\partial x_i^2}. \quad (20)$$

In fact, the Laplacian matrix of a lattice, when applied to the values of f at the vertices, corresponds to the finite differences approximation of the continuous operator on a regular grid of points. The Laplacian matrix of a graph can be seen as a generalization of the lattice definition.

A semi-supervised learning using Gaussian fields and harmonic function is proposed in [81]. Classification algorithms for Gaussian fields can be seen as a form of nearest-neighbor approach, where the nearest labeled examples are computed by a random walk on the graph. The method’s equations are related to electrical

networks and to spectral graph theory. The problem is represented as a graph, with some nodes labeled with $y \in \{0, 1\}$ (a binary labeling is considered for simplicity). Weighted edges represent similarities: w_{ij} is large for similar cases. For example $w_{ij} = \exp\{-\|\mathbf{x}_i - \mathbf{x}_j\|_A^2\}$ for a suitable metric. The strategy is to first compute a “smooth” real-valued function f for all nodes and then assign labels based on f . The “smoothness” desire of having similar values between similar points is expressed by formulating the problem as one of minimizing the quadratic energy function:

$$E(f) = \frac{1}{2} \sum_{i,j} w_{ij} (f(i) - f(j))^2. \quad (21)$$

The minimum energy function is *harmonic*: it satisfies $Lf = 0$ on unlabeled points and is equal to the label value on the labeled ones. L is the graph Laplacian defined above, and the harmonic property means that the value of f at an unlabeled point is equal to the weighted average of f at neighboring points:

$$f(j) = \frac{\sum_i w_{ij} f(i)}{\sum_i w_{ij}}. \quad (22)$$

In matrix notation: $f = \mathbf{P}f$, where $\mathbf{P} = \mathbf{D}^{-1} \mathbf{W}$. This is consistent with the intuitive notion of smoothness with respect to the similarity relationships. A simple rule to assign label is to label a node i with 1 if $f(i) > 1/2$, 0 otherwise. Ways to incorporate class prior knowledge (desirable proportions of the two classes) by modifying the threshold to label the nodes, as well as possible ways to learn a weight matrix \mathbf{W} from labeled and unlabeled data are described in [81].

4.2 Learning the Metric

Some semi-supervised algorithms proceed in two steps: first *a new metric or representation is identified* by performing an unsupervised step on all data (ignoring the existence of labels), then a pure supervised learning phase is executed by using the newly identified metric or representation. In many cases, when dealing with an optimization problem defined over more than one variable, a sequential method which first minimizes over the first variable, then over the second (leaving the first variable untouched), etc., gives in general a solution which can be improved if all variables are considered at the same time. This holds also for SSL. For example, the work in [14] shows how to combine constraints and metric learning into semi-supervised clustering.

Constraint-based clustering approaches start from pairwise *must-* or *cannot-link* constraints (requests that two points fall or do not fall in the same cluster) and insert into the objective function to be minimized a penalty for violating constraints. For example, the Euclidean k -means algorithm partitions the points into k sets so that the function:

$$\sum_i \|\mathbf{x}_i - \boldsymbol{\mu}_{l_i}\|^2$$

is locally minimized. In it, the vector $\boldsymbol{\mu}_{l_i}$ is the winning centroid associated with point \mathbf{x}_i , the one minimizing the distance.

If two sets of must-link pairs \mathcal{M} and cannot-link pairs \mathcal{C} are available, one can encourage a placement of the centroids in order to satisfy the constraints by adding a penalty w_{ij} for a single violation of a constraint in \mathcal{M} and a penalty \bar{w}_{ij} for a single violation of a constraint in \mathcal{C} , obtaining the following function to be minimized (*pairwise constrained k-means*):

$$E_{\text{pkmeans}} = \sum_i \|\mathbf{x}_i - \boldsymbol{\mu}_{l_i}\|^2 + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M} \text{ and } l_i \neq l_j} w_{ij} + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C} \text{ and } l_i = l_j} \bar{w}_{ij}. \quad (23)$$

Pairwise constraints can also be used for *metric learning*. If the metric is parameterized with a symmetric positive-definite matrix \mathbf{A} as follows:

$$\|\mathbf{x}_i - \mathbf{x}_j\|_{\mathbf{A}} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{A} (\mathbf{x}_i - \mathbf{x}_j)},$$

the problem amounts to determining appropriate values of the matrix coefficients. If the matrix is diagonal, the problem becomes that of *weighing* the different features.

After the metric is modified, one can use a traditional clustering algorithm like *k-means*. Asking for a single metric for the entire space can be inappropriate and a different metric \mathbf{A}_h can be used for each *k-means* cluster h . The MPCK-MEANS algorithm in [14] uses the method of expectation maximization and alternates between cluster assignment in the E-step and centroid estimation and metric learning in the M-step. Constraints are used during cluster initialization and when assigning points to clusters. The distance metric is adapted by reestimating \mathbf{A}_h during each iteration based on the current cluster assignment and constraint violations. An interesting paper dealing with metric learning for text documents is [53].

5 Text and Web Mining

When the data consist of a collection of nonnumerical elements, for example, documents, many of the techniques used to analyze numerical data are still available, but they need to be adapted by suitably preprocessing the documents and fine tuning the methods. *Information visualization* denotes the visual representation of large-scale collections of nonnumerical information, such as files and lines of software, library and bibliographic databases, and social networks.

Information retrieval deals mostly with searching for documents and for information within documents, and *web mining* is related to adapting methods to the context of the World Wide Web. The web is an *unstructured* (or, at most, *semi-structured*) collection of data mostly in form of human readable texts and images, connected by hyperlinks.

5.1 From Documents to Vectors: The Vector-Space Model

To use standard techniques designed for vector spaces to search, cluster, and classify documents, each document is mapped onto a vector (the *vector-space model*). After preprocessing, the document is reduced to a *bag of words*, actually a bag of tokens, and the most straightforward way to obtain a vector is to (a) fix a set of terms (tokens), (b) have a separate axis to represent each term (token), and (c) set the value of the vector along the axis t to zero if the document does not contain the token t , to a number greater than zero if the document contains the token one or more times.

If $n(d, t)$ is the number of times that document d contains the term t , the *term frequency* $\text{TF}(d, t)$ of term t in document d is defined as a figure that increases monotonically with the relative frequency of t in d . Some possible definitions are the following:

$$\begin{aligned}\text{TF}(d, t) &= \frac{n(d, t)}{\sum_{\tau} n(d, \tau)} \\ \text{TF}_{\text{SMART}}(d, t) &= \begin{cases} 0 & \text{if } n(d, t) = 0 \\ 1 + \log(1 + \log n(d, t)) & \text{otherwise.} \end{cases}\end{aligned}$$

The TF_{SMART} formula is meant to avoid an exaggerated value along a dimension if a term is present too many times. This was a frequent case in the initial years of the web, when simple search engines were just counting term occurrences. Actually, often the most interesting terms are the ones which do not appear in many documents (rare terms), and an *inverse document frequency* can be defined as a figure that monotonically *decreases* as the overall frequency of a term in the whole document corpus increases:

$$\text{IDF}(t) = \log \frac{1 + |D|}{|D_t|},$$

where D_t is the set of documents containing term t and the logarithm is used to avoid an exaggerated multiplier for very rare terms. After discounting the importance of weak terms appearing in too many documents, a specific document d in TFIDF-space (term-frequency inverse-document-frequency) is represented by vector

$$d = (d_t)_{t \in \text{terms}} \in \mathbb{R}^{\text{terms}},$$

where component d_t is

$$d_t = \text{TF}(d, t) \text{ IDF}(t).$$

A query q is a sequence of terms; therefore, it admits a representation $q = (q_t)$ in the same space as documents. Given the query q and the document d , one can now measure their proximity, by considering vector-space similarity measures.

The analysis of *social networks* of employers, customers, and business units is another interesting method in business intelligence. In particular, ranking nodes in

the network, assigning a measure of importance, is critical in many applications. After a seminal paper by Marchiori [56] highlighting the importance of *hyper-information* (information in the hyperlinks), Larry Page and Sergey Brin developed the PageRank algorithm, which follows the same basic social networks principles, by substituting “recommendations” and “citations” with hyperlinks [61]. They define a “measure of prestige” such that the prestige of a page is related to how many pages of prestige link to it. Note that this is a *recursive definition*.

After defining a matrix which derives from the graph of links between documents, a stationary solution is obtained by calculating the eigenvector of the matrix corresponding to the largest eigenvalue. Of special importance are iterative techniques which can be applied to huge networks and matrices.

6 Collaborative Filtering and Recommendation

A possible application of business intelligence techniques related to having a network of consumers is called *collaborative recommendation*. Word of mouth has always been a powerful and effective technique to spread information and opinions from person to person, in a viral manner. It is effective because human beings naturally tend to speak with people similar to them, who share their habits, opinions, and way of life. By choosing to interact with a selected and small number of similar people, persons effectively *filter* the data. It is then up to them to integrate and weigh the information that they receive, to reach their final decisions. A similar process can be simulated through data mining and modeling methods.

An interesting application is in the marketing sector: data collected about users and products, either bought or at least evaluated, can be used to estimate how a customer would evaluate a product he did not see before.

Collaborative filtering and recommendation is a method of predicting the interests of a person by collecting taste information from many other collaborating people.

A simple method to predict an unknown rating r_{ui} considers the ratings of other users on the same item i and the *similarity* between user u and other users. A generic unknown rating r_{ui} is calculated by the following equation:

$$r_{ui} = \frac{\sum_{\text{known } r_{ki}} \text{similarity}(u, k) \cdot r_{ki}}{\sum_{\text{known } r_{ki}} \text{similarity}(u, k)}. \quad (24)$$

The vote is predicted with a weighted average of the other users’ votes, with weights given by similarities. The *sparsity* of the raw user-item evaluation matrix can be a problem. Each users evaluates only a very small subset of items and most entries are unknown. By *compressing and summarizing the user characteristics* into

a much smaller vector, one hopes to reach better generalization results and possibly a better understanding of the model, as explained by the Occam's razor principle.

A possible way to determine the interest or the vote of a user for an item is to associate a small vector of characteristics with each user and with each item and then deriving votes by observing the similarity between the user and item characteristic vectors. Vector $q_i \in \mathfrak{N}^f$ will be defined to contain the characteristics (factors) of item i , and vector $p_u \in \mathfrak{N}^f$ to denote the extent of interest that user u has in each item factor. One would like to obtain the rating of a user u for an item i by a simple scalar product of the corresponding vectors:

$$\hat{r} = q_i^T p_u. \quad (25)$$

Among automated ways to build effective factors to predict ratings, the traditional singular value decomposition (SVD) can be used to find informative q_i 's and p_u 's. By using SVD, one decomposes a matrix R containing *all* ratings as $R = U\Sigma M^T$, where the rows of matrix U and M are the set of p_u and q_i , scaled by the diagonal matrix Σ . By considering the size of the diagonal values in Σ , one can then reduce the dimension of the vectors and keep only the most relevant components. Unfortunately, in most cases one does *not* have the value of all cells in the rating matrix. More flexible and robust learning algorithms based on optimization can be adopted to find effective approximations of the factor vectors q_i and p_u . As usual, examples of expressed votes guide the learning process: to learn the factor vectors q_i and p_u , one aims at minimizing the *regularized squared error (RSE)* on the set of *known* ratings:

$$\text{RSE} = \frac{1}{|K|} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2). \quad (26)$$

Here, K is the set of the (u, i) pairs for which r_{ui} is known (the training set). The first term in the summation is the squared error between the model $q_i^T p_u$ and the known result r_{ui} . To facilitate *generalization* (prediction of novel ratings), it is useful to penalize the magnitudes of the factor vectors in a manner proportional to a constant λ . This term is called a *regularizing term*. When example ratings abound, most of the contribution to RSE derives from errors in reconstructing the expressed votes. On the other hand, when ratings are scarce, the regularizing term becomes crucial and it acts to discourage very large vectors with potentially dramatic (and wrong) effects on the predictions. The problem is now one of minimizing a continuous function of the free parameters p_u and q_i .

Other examples of discrete structures related to identification problems and learning are related to industrial and business processes. The task is known as *process mining*. Process mining techniques [72,73] extract process information from event logs. For example, the audit trails of a workflow management system or the transaction logs of an enterprise resource planning (ERP) system can be used to discover models describing processes, organizations, and products. Petri nets [64]

are a widely used modeling tool. Relevant problems in process mining are NP-hard; the work in [38] shows that the computational problem of finding a minimum finite-state acceptor compatible with given data is NP-hard. Suitable heuristic optimization schemes are used, for example, in [40, 41].

7 Multi-Objective Optimization and Pareto Optimality

Consider now one of the most expensive and crucial issue in business intelligence: that of building a suitable model. If the tool is optimization, it must work on a suitable function to be optimized, which is hard, costly, and time-consuming to be defined. Consultants are aware that most of their time is spend on working with the customer to define the model, not to solve it. A traditional formulation which recognizes that a detailed function can be missing is as a *multi-objective optimization problem* (MOOP), which assumes partial knowledge in the form of a series of desirable objectives, but lack of their detailed combination into a comprehensive *utility function*. A MOOP can be stated as

$$\begin{aligned} & \text{minimize } \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \dots, f_m(\mathbf{x})\} \\ & \text{subject to} \quad \mathbf{x} \in \Omega, \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of n decision variables and $\Omega \subset \mathbb{R}^n$ is the *feasible region* and is typically specified as a set of constraints on the decision variables.

The vector $\mathbf{f} : \Omega \rightarrow \mathbb{R}^m$ is made of m objective functions which need to be jointly minimized. Objective vectors are images of decision vectors and can be written as $\mathbf{z} = \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \dots, f_m(\mathbf{x})\}$. The above problem is ill-posed whenever objective functions are conflicting, a situation which typically occurs in real world applications. In these cases, an objective vector is considered optimal if none of its components can be improved without worsening at least one of the others. An objective vector \mathbf{z} is said to *dominate* \mathbf{z}' , denoted as $\mathbf{z} \prec \mathbf{z}'$, if $z_k \leq z'_k$ for all k and there exist at least one h such that $z_h < z'_h$. A point $\hat{\mathbf{x}}$ is *Pareto optimal* if there is no other $\mathbf{x} \in \Omega$ such that $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\hat{\mathbf{x}})$. The *Pareto frontier* (or Pareto front) consists of all Pareto-optimal points and is particularly useful in decision making: by restricting attention to the set of choices that are Pareto-efficient, a designer can make tradeoffs within this set, rather than considering the full range of every parameter. A critical review of MOOP in data mining is considered in [36]. No single criterion exists to judge models, but different measures, like precision, recall, simplicity, and unexpectedness (surprisingness) [35]. A rule like “IF (patient is pregnant) THEN (gender is female)” is accurate and simple but not very surprising. An explicit MOOP formulation empowers the decision maker with the freedom to guide the final solution towards his detailed goals in a systematic and explicit manner.

As mentioned, asking a user to quantify his utility function (e.g., by choosing weights of a linear combination of the different objectives) *a priori*, before seeing

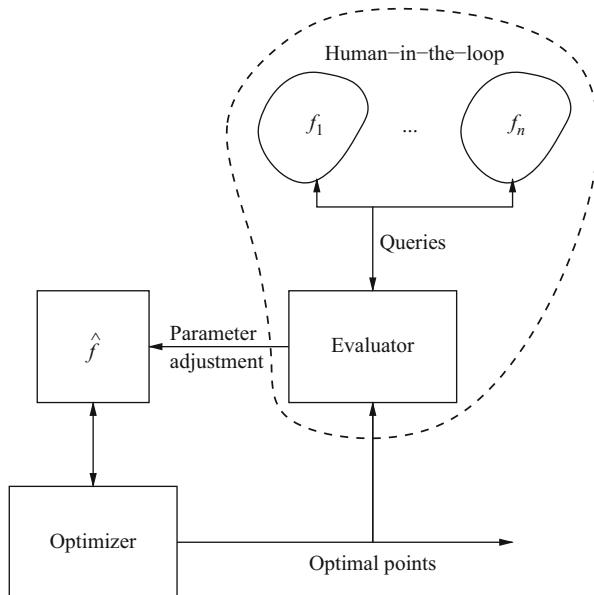


Fig. 1 Brain–computer optimization: learning the problem definition from the final user in the case of interactive multi-objective optimization (adapted from [6])

the actual optimization results, is in some cases extremely difficult. If the final user is cooperating with an optimization expert, misunderstanding between the persons may arise. This problem can become dramatic when the number of the conflicting objectives in MOOP increases. As a consequence, the final user may be dissatisfied with the solutions presented by the optimization expert. Different drawbacks are present in *a posteriori* approaches, which present to the final user a representative set of solutions on the entire Pareto front so that the user can pick his most preferred solution.

Although providing explicit weights and mathematical formulas can be difficult for the user, for sure he can *evaluate the returned solutions*. In most cases, the strategy to overcome this situation consists of a joint interactive work between the final user and the optimization expert to change the definition of the problem itself. The optimization tool will then be re-executed over the new version of the problem. This process may be iterated an arbitrary number of times, as shown in Fig. 1.

As mentioned, as a rule of thumb, most of the problem-solving effort in the real world is spent on *defining the problem*, on specifying in a computable manner the function to be optimized. After this modeling work is completed, optimization becomes in certain cases a commodity. The implication for researchers and developers is that much more effort should be devoted to designing supporting techniques and tools to help the final user, often without expertise in mathematics and in optimization, to define and refine the function to be optimized which corresponds to his real objectives.

Reactive search optimization is dedicated to online learning techniques to support the quest for a solution by self-adapting a local search method in a manner depending on the previous history of the search. The learning signals consist of data about the structural characteristics of the instance collected while the algorithm is running, for example, data about sizes of basins of attraction, entrapment of trajectories, and repetitions of previously visited configurations. The algorithm learns by interacting from a previously unknown environment given by an existing (and fixed) problem definition.

It can be argued that there is a second interesting online learning loop, where learning signals originate from a final user and are intended to *modify and refine the problem definition* itself. This context is potentially very wide, depending on the amount of knowledge about the problem given a priori, on the allowed modifications, on the form of the questions posed to the user.

An example of interactive multi-objective optimization with RSO is described in [6]. The methods considered in our approach share with the work in [29, 70, 83] the interaction with the final user realized via pairwise comparison of solutions but tackles a broader class of problems with nonlinear preference functions. This case is of interest because many (maybe most) decision problems are *nonlinear*, to reflect our preference for reasonable compromise solutions. The reader interested in state of the art results for learning arbitrary (*nonlinear*) models may consider reading the recent technical publication [7].

The focus here is on the simpler and classical linear case [6], and the goal consists of *learning the non-dominated solution preferred by the final user*. Assume that the user provides the different objectives in the MOOP problem, but he cannot quantify the *weights* of the different objectives before seeing the actual optimization results. The system aims at learning the weights vector $w = (w_1, w_2, \dots, w_m)$ optimizing the linear combination g :

$$g(x, w) = w_1 f_1(x) + w_2 f_2(x) + \dots + w_m f_m(x).$$

In a more compact form:

$$g(x_1, x_2, \dots, x_n, w) = f(x)^T w$$

where $f = (f_1, f_2, \dots, f_m)$. Without loss of generality, assume that g must be minimized.

The feedback from the decision maker at each iteration is obtained by presenting two solutions and asking him to indicate his favorite solution among them, if any. The preferences stated by the final user are translated into *constraints* that the weights must satisfy. This guarantees that the obtained utility function is consistent with the user's judgments. If $a = (a_1, a_2, \dots, a_n)$ and $b = (b_1, b_2, \dots, b_n)$ are the two solutions provided by the system, the preference $a \prec b$ of the final user for the solution **a** w.r.t. the solution **b** is represented by the following constraint:

$$g(a, w) < g(b, w).$$

The problem of learning the user preference then becomes that of finding a solution w for the set of constraints on the weights generated by the user's feedback.

All the weights are initialized with random values in the interval $(0,1)$ and then normalized so that their sum is 1. At each iteration, two non-dominated solutions a and b are compared by the final user. Both solutions are obtained by minimizing a linear combination of the objective functions of the input problem:

$$\min_x g(x, w).$$

In particular, the first solution \mathbf{a} is obtained by using the current weights vector w^{curr} found by applying the *middlemost weights* technique [63], by solving the following linear programming problem:

$$\begin{aligned} \max_w \quad & \gamma \\ \text{subject to} \quad & \begin{cases} g(a, w) \leq g(b, w) - \gamma & \forall a \prec b \\ w_i \geq \gamma & \forall i = 1, \dots, m \\ \gamma \geq 0. \end{cases} \end{aligned}$$

The meaning of the above is to search for a weight which is *consistent* but also *far from the boundaries* of the consistent region. The bigger the positive γ parameter, the safer the inequalities. Even if limited noise is added (e.g., caused by physical quantities with measurement errors) and the g values are slightly changed, there is still a *safety margin* before the direction of the inequality is changed.

The second solution \mathbf{b} can be obtained by using the weights vector w^{pert} , generated by perturbing w^{curr} , and by ensuring that the two generated solutions are sufficiently dissimilar. Issues related to considering possible infeasible sets of linear constraints (which can be generated by a confused decision maker or because a linear approximation is a too rough) are considered in [6]. The more complex nonlinear case is solved with machine learning techniques based on the SVM method in [7].

8 Conclusion

Understanding and optimizing business processes through a sound data-based *experimental science* approach is becoming critical to ensure competitiveness. The growing amount of data produced and the growing complexity of the interrelationships demand computer-assisted means for monitoring and making decisions in real time. This fact has to be translated into *reactive* business intelligence (RBI) systems,

with internal mechanisms based on machine learning and intelligent optimization (LION) to change processes and decisions based on new data and to identify critical situations, anomalies, and opportunities for improvement, without the need for a human intervention in the daily operation.

For challenging business intelligent applications, the *reactive* capability to modify the optimization process while the system is running can help both to *define* exactly what one wants to accomplish and to actually *compute* one or more solutions. Reactive BI objective is to make it easier to create dynamic and flexible systems, though LION intelligent optimization methods, without the direct intervention of technical people and programmers for every required change. Furthermore, by hiding most of the data mining and modeling complexity from the human decision maker he is free to concentrate on the most relevant and top level decisions.

Representative combinatorial and continuous optimization techniques were reviewed, with examples of their use in selected and representative business contexts.

Among the open issues for future research are scalability to massive data sets and larger network of relationships, implementations supported by massively parallel and distributed computing systems, simplicity of use for the decision maker and comprehensibility of the extracted models, strategic use of interaction with the decision maker (which need to be limited) to build more accurate models of utility functions, and seamless integration of combinatorial optimization tools (usually needed to handle discrete data like networks, categories, processes) with continuous optimization tools intended for numerical data.

Cross-References

- [Algorithms and Metaheuristics for Combinatorial Matrices](#)
 - [Job Shop Scheduling with Petri Nets](#)
 - [Neural Network Models in Combinatorial Optimization](#)
 - [Resource Allocation Problems](#)
-

Recommended Reading

1. Y.S. Abu-Mostafa, Learning from hints in neural networks. *J. Complex.* **6**(2), 192–198 (1990)
2. S. Aiber, D. Gilat, A. Landau, N. Razinkov, A. Sela, S. Wasserkrug, Autonomic self-optimization according to business objectives, in Proceedings International Conference on Autonomic Computing, IEEE Computer Society, 2004, pp. 206–213
3. B. Azvine, Z. Cui, D.D. Nauck, B. Majeed, Real time business intelligence for the adaptive enterprise, in The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on E-Commerce Technology, 2006, p. 29
4. R. Battiti, First-and second-order methods for learning: between steepest descent and newton's method. *Neural Comput.* **4**, 141–166 (1992)
5. R. Battiti, M. Brunato, Reactive search optimization: learning while optimizing, in *Handbook of Metaheuristics*, 2010, pp. 543–571

6. R. Battiti, P. Campigotto, Reactive search optimization: learning while optimizing. An experiment in interactive multi-objective optimization, in *Proceedings of MIC 2009, VIII Metaheuristic International Conference* ed. by S. Voss, M. Caserta, Lecture Notes in Computer Science (Springer, New York, 2010)
7. R. Battiti, A. Passerini, Brain-computer evolutionary multiobjective optimization (BC-EMO): A genetic algorithm adapting to the decision maker. *IEEE Trans. Evol. Comput.* **14**(15), 671–687 (2010)
8. R. Battiti, G. Tecchiori, The reactive tabu search. *ORSA J. Comput.* **6**(2), 126–140 (1994)
9. R. Battiti, G. Tecchiori, Training neural nets with the reactive tabu search. *IEEE Trans. Neural Network* **6**(5), 1185–1200 (1995)
10. R. Battiti, A.A. Bertossi, Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Trans. Comput.* **48**(4), 361–385 (1999)
11. R. Battiti, M. Brunato, *Reactive Business Intelligence. From Data to Models to Insight*. Reactive Search Srl, Via della Stazione 27, I-38123 Trento - Italy, February 2011. ISBN: 978-88-905795-0-9
12. R. Battiti, M. Brunato, F. Mascia, *Reactive Search and Intelligent Optimization*. Operations Research/Computer Science Interfaces (Springer, New York, 2008). ISBN: 978-0-387-09623-0
13. P. Berkhin, Survey of clustering data mining techniques. *Grouping Multidimensional Data: Recent Advances in Clustering*, 2006, pp. 25–71
14. M. Bilenko, S. Basu, R.J. Mooney, Integrating constraints and metric learning in semi-supervised clustering, in *Proceedings of the Twenty-First International Conference on Machine Learning*, ACM, 2004, p. 11
15. A. Blum, S. Chawla, Learning from labeled and unlabeled data using graph mincuts, in *Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, 2001, pp. 19–26
16. C. Blum, R. Battiti, in *Learning and Intelligent Optimization: 4th International Conference*, LION 4, Venice, Italy, January 2010. Selected Papers, vol. 6073 (Springer, New York, 2010)
17. D. Boley, M. Gini, R. Gross, E.H.S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, J. Moore, Partitioning-based clustering for web document categorization. *Decis. Support Syst.* **27**(3), 329–341 (1999)
18. P.S. Bradley, U. Fayyad, C. Reina, Scaling EM clustering to large databases. Technical Report MSR-TR-98-35, Microsoft Research, 1998
19. P.S. Bradley, U.M. Fayyad, O.L. Mangasarian, Mathematical programming for data mining: Formulations and challenges. *INFORMS J. Comput.* **11**, 217–238 (1999)
20. M. Brunato, R. Battiti, Grapheur: a software architecture for reactive and interactive optimization, in *Learning and Intelligent Optimization* (2010), pp. 232–246
21. O. Chapelle, M. Chi, A. Zien, A continuation method for semi-supervised SVMs, in *Proceedings of the 23rd International Conference on Machine Learning*, ACM, 2006, pp. 185–192
22. O. Chapelle, B. Schölkopf, A. Zien, *Semi-Supervised Learning* (MIT, Cambridge, MA, 2006)
23. D.M. Chickering, Learning Bayesian networks is NP-complete. *Learn. Data Artif. Intell. Stat.* v **112**, 121–130 (1996)
24. D.M. Chickering, Optimal structure identification with greedy search. *J. Mach. Learn. Res.* **3**, 507–554 (2003)
25. W. Chung, H. Chen, J.F. Nunamaker Jr, A visual framework for knowledge discovery on the Web: An empirical study of business intelligence exploration. *J. Manag. Inform. Syst.* **21**(4), 57–84 (2005)
26. W.F. Cody, J.T. Kreulen, V. Krishna, W.S. Spangler, The integration of business intelligence and knowledge management. *IBM Syst. J.* **41**(4), 697–713 (2002)
27. T.H. Davenport, J.G. Harris, *Competing on Analytics*. (Harvard Business School Publishing, Boston, 2007)
28. U. Dayal, M. Castellanos, A. Simitsis, K. Wilkinson, Data integration flows for business intelligence, in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, ACM, 2009, pp. 1–11

29. R.F. Dell, M.H. Karwan, An interactive MCDM weight space reduction method utilizing a Tchebycheff utility function. *Nav. Res. Logist.* **37**(2), 403–418 (1990)
30. S.G. Eick, Visualizing online activity. *Comm. ACM* **44**(8), 45–50 (2001)
31. R.A. Eisenbeis, Pitfalls in the application of discriminant analysis in business, finance, and economics. *J. Finance* **32**(3), 875–900 (1977)
32. B. Ellis, W.H. Wong, Learning causal Bayesian network structures from experimental data. *J. Am. Stat. Assoc.* **103**(482), 778–789 (2008)
33. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases. *AI Mag.* **17**(3), 37 (1996)
34. J. Fisher, D. Harel, T.A. Henzinger, Biology as reactivity. *Comm. ACM* **54**(10), 72–82 (2011)
35. A. Freitas, On objective measures of rule surprisingness. *Principles of Data Mining and Knowledge Discovery*. 1998, pp. 1–9
36. A.A. Freitas, A critical review of multi-objective optimization in data mining: a position paper. *ACM SIGKDD Exploration Newslett.* **6**(2), 77–86 (2004)
37. A.M. Geoffrion, J.S. Dyer, A. Feinberg, An interactive approach for multi-criterion optimization, with an application to the operation of an academic department. *Management Science* **357**–368 (1972)
38. E.M. Gold, Complexity of automaton identification from given data. *Information and Control* **37**(3), 302–320 (1978)
39. M. Golfarelli, S. Rizzi, I. Cellai, Beyond data warehousing: what's next in business intelligence?, in *Proceedings of the 7th ACM international Workshop on Data Warehousing and OLAP*, ACM, 2004, pp. 1–6
40. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, M.C. Shan, Business process intelligence. *Comput. Ind.* **53**(3), 321–343 (2004)
41. D. Grigori, F. Casati, U. Dayal, M.C. Shan, Improving business process quality through exception understanding, prediction, and prevention, in *Proceedings of the International Conference on Very Large Data Bases*, Citeseer, 2001, pp. 159–168
42. E.H. Han, G. Karypis, V. Kumar, B. Mobasher, *Clustering Based on Association Rule Hypergraphs*. University of Minnesota, Department of Computer Science, 1997
43. J. Han, L.V.S. Lakshmanan, R.T. Ng, Constraint-based, multidimensional data mining. *Computer* **32**(8), 46–50 (1999)
44. X. He, C.H.Q. Ding, H. Zha, H.D. Simon, Automatic topic identification using webpage clustering, in *ICDM 2001, Proceedings IEEE International Conference on Data Mining, 2001.*, IEEE Computer Society, 2001, pp. 195–202
45. D. Heckerman, D. Geiger, D.M. Chickering, Learning Bayesian networks: The combination of knowledge and statistical data. *Mach. Learn.* **20**(3), 197–243 (1995)
46. H.H. Hoos, T. Stuetzle, *Stochastic Local Search: Foundations and Applications* (Morgan Kaufmann, 2005)
47. T. Joachims, Text categorization with support vector machines: Learning with many relevant features. *Machine Learning: ECML-98*, pp. 137–142, 1998
48. T. Joachims, Making large scale SVM learning practical. Technical report, Universität Dortmund, 1999
49. J.O. Kephart, D.M. Chess, The vision of autonomic computing. *Computer* **36**(1), 41–50 (2003)
50. R. Kohavi, N.J. Rothleider, E. Simoudis, Emerging trends in business analytics. *Comm. ACM* **45**(8), 45–48 (2002)
51. T. Kohonen, Self-organized formation of topologically correct feature maps. *Biol. Cybern.* **43**, 59–69 (1982)
52. T. Kohonen, P. Somervuo, Self-organizing maps of symbol strings. *Neurocomputing* **21**(1–3), 19–30 (1998)
53. G. Lebanon, Metric learning for text documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006, pp. 497–508
54. H.P. Luhn, A business intelligence system. *IBM J. Res. Dev.* **2**(4), 314–319 (1958)

55. P.C. Mahalanobis, On the generalized distance in statistics, in *Proceedings of the National Institute of Science*, Calcutta, vol. 12, 1936, p. 49
56. M. Marchiori, The quest for correct information on the web: Hyper search engines. *Comput. Network ISDN Syst.* **29**(8–13), 1225–1235 (1997)
57. T.K. Moon, The expectation-maximization algorithm. *IEEE Signal Process. Mag.* **13**(6), 47–60 (1996)
58. P. Mutton, J. Golbeck, Visualization of semantic metadata and ontologies, in *Proceedings of the Seventh International Conference on Information Visualization*, IEEE Computer Society, 2003, pp. 300–305
59. R.T. Ng, J. Han, Efficient and effective clustering methods for spatial data mining, in *Proceedings of the International Conference on Very Large Data Bases*, Citeseer, 1994, pp. 144–155
60. B. Padmanabhan, A. Tuzhilin, On the use of optimization for data mining: Theoretical interactions and eCRM opportunities. *Manag. Sci.* 1327–1343 (2003)
61. L. Page, S. Brin, R. Motwani, T. Winograd, The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998
62. A. Petroni, M. Braglia, Vendor selection using principal component analysis. *J. Supply Chain Manag.* **36**(2), 63–69 (2000)
63. S.P. Phelps, M. Köksalan, An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. *Manag. Sci.* **49**(12), 1726–1738 (2003)
64. W. Reisig, G. Rozenberg, *Lectures on Petri Nets: Advances in Petri Nets* (Springer, New York, 1998)
65. H. Scudder III, Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. Inform. Theor.* **11**(3), 363–371 (1965)
66. J. Shi, J. Malik, Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(8), 888–905 (2000)
67. H.A. Simon, A behavioral model of rational choice. *Q. J. Econ.* **69**(1), 99 (1955)
68. V. Sindhwani, S.S. Keerthi, O. Chapelle, Deterministic annealing for semi-supervised kernel machines, in *Proceedings of the 23rd International Conference on Machine Learning*, ACM, 2006, p. 848
69. R. Sterritt, D. Bustard, Towards an autonomic computing environment, in *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, IEEE, 2003, pp. 694–698
70. R.E. Steuer, E. Choo, An interactive weighted Tchebycheff procedure for multiple objective programming. *Math. Program.* **26**(1), 326–344 (1983)
71. A.N. Tichonov, On the regularization of ill-posed problems (Russian). *Dokl. Akad. Nauk SSSR* **153**, 49–52 (1963)
72. W.M.P. Van Der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes* (Springer, New York, 2011)
73. W.M.P. van der Aalst, A. Weijters, Process mining: a research agenda. *Comput. Ind.* **53**(3), 231–244 (2004)
74. V.N. Vapnik, *The Nature of Statistical Learning Theory* (Springer, New York, 1995)
75. V.N. Vapnik, A.Ja. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities. *Theor. Probab. Appl.* **16**, 264–280 (1971)
76. A. Vellido, P.J.G. Lisboa, J. Vaughan, Neural networks in business: a survey of applications (1992–1998). *Expert Syst. Appl.* **17**(1), 51 (1999)
77. J. Wang, X. Wu, C. Zhang, Support vector machines based on K-means clustering for real-time business intelligence systems. *Int. J. Bus. Intell. Data Min.* **1**(1), 54–64 (2005)
78. H.J. Watson, B.H. Wixom, The current state of business intelligence. *Computer* **40**(9), 96–99 (2007)
79. B. Widrow, D.E. Rumelhart, M.A. Lehr, Neural networks: applications in industry, business and science. *Comm. ACM* **37**(3), 93–105 (1994)
80. X. Zhu, Semi-supervised learning literature survey, in *Computer Science* (University of Wisconsin-Madison, 2006)

81. X. Zhu, Z. Ghahramani, J. Lafferty, Semi-supervised learning using Gaussian fields and harmonic functions, in *Machine Learning International Conference*, vol. 20, 2003, p. 912
82. X. Zhu, Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005
83. S. Zionts, J. Wallenius, An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions. *Management Science* **29**(5), 519–529 (1983)

Reformulation–Linearization Techniques for Discrete Optimization Problems

Hanif D. Sherali and Warren P. Adams

Contents

1	Introduction	2850
2	RLT Hierarchy for Mixed-Integer Zero-One Problems	2854
3	Properties of the RLT Relaxation	2859
4	Generating Valid Inequalities and Facets Using RLT	2861
5	Multilinear Programs and Equality Constraints with Applications to the Quadratic Assignment Problem	2862
6	Exploiting Special Structures to Generate Tighter RLT Representations	2867
7	Applications of RLT for Some Particular Special Structures	2873
7.1	Generalized Upper Bounding (GUB) or Multiple Choice Constraints	2873
7.2	Variable Upper Bounding Constraints	2877
7.3	Sparse Constraints	2879
8	Using Conditional Logic to Strengthen RLT Constraints with Application to the Traveling Salesman Problem	2880
8.1	Application to the Traveling Salesman Problem	2884
9	Semidefinite Cuts and Extensions to Continuous Global Optimization	2886
10	Conclusion	2887
	Recommended Reading	2892

H.D. Sherali (✉)

Grado Department of Industrial and Systems Engineering (0118), Virginia Polytechnic Institute and State University, Blacksburg, VA, USA
e-mail: hanifs@vt.edu

W.P. Adams

Department of Mathematical Sciences, Clemson University, Clemson, SC, USA
e-mail: wadams@clemson.edu

Abstract

This chapter describes the reformulation – linearization technique (RLT) for solving mixed-integer 0-1 and general mixed-discrete optimization problems. The use of RLT for directly lifting mixed-integer programs (MIPs) into higher-dimensional representations as well as for deriving strong valid inequalities in the space of the original variables is discussed. Methods for exploiting inherent special structures for particular applications as well as for enhancing RLT-based relaxations in general through conditional logic deductions and semidefinite cuts are also presented. Finally, certain persistency results and extensions to solving general mixed-discrete convex programs, as well as continuous, nonconvex factorable programming problems, along with implementation guidelines are outlined.

1 Introduction

Discrete and continuous nonconvex programming problems arise in a host of practical applications in the context of production, location – allocation, distribution, economics and game theory, process design, and engineering design situations. Several recent advances have been made in the development of branch-and-cut algorithms for discrete optimization problems and in polyhedral outer-approximation methods for continuous nonconvex programming problems. At the heart of these approaches is a sequence of linear programming problems that drive the solution process. The success of such algorithms is strongly related to the strength or tightness of the linear programming representations employed.

This chapter addresses the generation of equivalent representations having tight linear programming (LP) relaxations via an automatic reformulation technique for solving discrete mixed-integer linear (and polynomial) programming problems. The particular approach of this type that is the focus of the present chapter is called the *reformulation – linearization technique (RLT)*, a procedure that can be used to generate tight linear (or sometimes convex) programming representations. Such equivalent, lifted representations facilitate the design of effective exact as well as heuristic solution procedures. The tight relaxations produced can often be used to derive good quality solutions (in polynomial time) to problems of practical sizes that arise in the aforementioned applications.

The RLT procedure is capable of generating representations of increasing degrees of strength, although with an accompanied increase in problem size. Coupled with recent advances in linear programming (LP) technology, this permits the incorporation of tighter RLT-based representations within the context of exact or heuristic methods to enhance problem solvability. Also, the RLT methodology can be extended to solve continuous nonconvex factorable programs as well – the interested reader is referred to Sherali and Tuncbilek [82–84] and Sherali and Wang [86] for an introduction to this subject (see Sherali and Adams [68], for an extensive treatise on this topic).

The motivation for constructing good models, that is, models that have tight underlying linear programming representations, rather than simply mathematically correct models, has led to some crucial and critical research on the model formulation process as in Balas [13], Jeroslow and Lowe [35, 36], Johnson [37], Meyer [49], Sherali and Adams [64, 65], Williams [103], and Wolsey [105]. Much work has also been done in converting classes of separable or polynomial nonlinear integer programming problems into equivalent linear integer programs and, for generating tight, valid inequalities for such problems, as in Adams et al. [7], Adams and Sherali [3–5], Balas and Mazzola [14, 15], Glover [32], and Sherali and Adams [64, 65]. Sometimes, a special variable redefinition technique may be applicable depending on the problem structure as in Martin [48]. In this approach, a linear transformation is defined on the variables to yield an equivalent formulation that tightens the continuous relaxation by constructing a partial convex hull of a specially structured subset of constraints. A more generally applicable technique is to augment the formulation through the addition of valid or implied inequalities that typically provide some partial characterization for the convex hull of feasible solutions. Some cutting plane generation schemes in this vein include the ones described in Nemhauser and Wolsey [51], Padberg [56], Van Roy and Wolsey [100], and Wolsey [104, 106]. Automatic reformulation procedures utilizing such constraint generation schemes within a branch-and-bound or branch-and-cut framework are presented in Crowder et al. [24], Hoffman and Padberg [34], Johnson et al. [38], Nemhauser et al. [52], Oley and Sjouquist [54], Padberg and Rinaldi [57], and Van Roy and Wolsey [101]. Besides these studies, ample evidence is available in the literature on the efficacy of providing tight linear programming relaxations for pure and mixed zero-one programming problems as in Adams and Sherali [3–5], Crowder and Padberg [25], Geoffrion and McBryde [31], and Magnanti and Wong [47], among many others.

This chapter details the reformulation – linearization technique (RLT) of Sherali and Adams [64, 65] along with its several enhancements and extensions. The RLT procedure is an automatic reformulation technique that can be used to derive tight LP representations as well as to generate strong valid inequalities. Consider a mixed-integer zero-one linear programming problem whose feasible region X is defined in terms of some inequalities and equalities in binary variables $x = (x_1, \dots, x_n)$ and a set of bounded continuous variables $y = (y_1, \dots, y_m)$. Given a value of $d \in \{1, \dots, n\}$, the fundamental RLT procedure constructs various polynomial factors of degree d comprised of the products of some d binary variables or their complements. These factors are then used to multiply each of the constraints defining X (including the variable bounding restrictions), to create a (nonlinear) polynomial mixed-integer zero-one programming problem. Using the relationship $x_j^2 = x_j$ for each binary variable x_j , $j = 1, \dots, n$; substituting a variable w_j and v_{jk} , respectively, in place of each nonlinear term of the type $\prod_{j \in J} x_j$ and $y_k \prod_{j \in J} x_j$; and relaxing integrality, the nonlinear polynomial problem is re-linearized into a higher-dimensional polyhedral set X_d defined in terms of the original variables (x, y) and the new variables (w, v) . For X_d to be equivalent to X , it is only necessary to enforce x to be binary-valued, and the remaining variables

may be treated as continuous, since the binariness on the x -variables is shown to automatically enforce the required product relationships on the w - and v -variables. Denoting the projection of X_d onto the space of the original (x, y) variables as X_{Pd} , it can be shown that the following holds true as d varies from 1 to n :

$$X_{P0} \supseteq X_{P1} \supseteq X_{P2} \supseteq \dots \supseteq X_{Pn} = \text{conv}(X),$$

where X_{P0} is the ordinary linear programming relaxation and $\text{conv}(X)$ represents the convex hull of X . The hierarchy of higher-dimensional representations produced in this manner markedly strengthens the usual continuous relaxation, as evidenced by the fact not only that the convex hull representation is obtained at the highest level but that in computational studies on many classes of problems, even the first-level representation helps design algorithms that significantly dominate existing procedures in the literature (see Sherali and Adams [67, 68]). The theoretical implications of this hierarchy are noteworthy; the resulting representations subsume and unify many published linearization methods for nonlinear 0-1 programs, and the algebraic representation available at level n promotes new methods for identifying and characterizing facets and valid linear inequalities in the original variable space as well as for providing information that directly bridges the gap between discrete and continuous sets. Indeed, since the level- n formulation characterizes the convex hull, all valid inequalities in the original variable space must be obtainable via a suitable projection; thus such a projection operation serves as an all-encompassing tool for generating valid inequalities.

Sherali and Adams [65] and Sherali [62] also demonstrate the relationship between their hierarchy of relaxations and that generated through disjunctive programming techniques. Balas [13] has shown how a hierarchy spanning the spectrum from the linear programming relaxation to the convex hull of feasible solutions can be generated for linear mixed-integer zero-one programming problems by inductively representing the feasible region at each stage as a conjunction of disjunctions and then taking its hull relaxation. This hull relaxation amounts to constructing the intersection of the individual convex hulls of the different disjunctive sets and hence yields a relaxation. Sherali and Adams show that their hierarchy produces a different, stronger set of intermediate relaxations that lead to the underlying convex hull representation. To view their relaxations as Balas' hull relaxations requires manipulating the representation at any stage d to write it as a conjunction of a *non-standard* set of disjunctions. Moreover, by the nature of the RLT approach, Sherali and Adams also show (see Sect. 5) how a hierarchy of *linear* relaxations leading to the convex hull representation can be readily constructed for mixed-integer zero-one *polynomial* programming problems that have no cross-product terms among the continuous variables.

In connection with the final comment above, Boros et al. [21] have also independently developed a similar hierarchy for the special case of the unconstrained, quadratic pseudo-Boolean programming problem. For this case, they construct a standard linear programming relaxation that coincides with the RLT relaxation at level $d = 1$ and then show in an *existential* fashion how a hierarchy of

relaxations, indexed by $d = 1, \dots, n$, can be generated that leads to the convex hull representation at level n . At any level d , this is done by including constraints corresponding to the extreme directions of the cone of nonnegative quadratic pseudo-Boolean functions that involve at most d of the n -variables. Each such relaxation can be viewed as the projection of one of the *explicitly* stated higher-order RLT relaxations onto the variable space of the first level for this special case. Moreover, the RLT approach further accommodates the consideration of general pseudo-Boolean polynomials, constrained problems, as well as general mixed-discrete situations.

Lovasz and Shrijver [46] have also independently proposed a similar hierarchy of relaxations for linear, *pure* 0-1 programming problems, which essentially amounts to deriving X_1 from X_0 , finding the projection X_{P1} , and then repeating this step by replacing X_0 with X_{P1} . Continuing in this fashion, they show that in n steps, $\text{conv}(X)$ is obtained. However, from a practical viewpoint, while the relaxations X_1, X_2, \dots, X_n of Sherali and Adams are explicitly available and directly implementable, the projections required by Lovasz and Shrijver for level-2 and higher-order relaxations are computationally burdensome, necessitating the potentially exponential task of vertex enumeration. Moreover, extensions to mixed-integer or to nonlinear zero-one problems are not evident using this development.

Another hierarchy along the same lines has been proposed by Balas et al. [19]. In this hierarchy, the set X_1 is generated as in Sherali and Adams but using factors involving only one of the binary variables, say, x_1 . Projecting the resulting formulation onto the space of the original variables produces the convex hull of feasible solutions to the original LP relaxation with the added restriction that x_1 is binary-valued. This follows from Sherali and Adams [65] since it is equivalent to treating only x_1 as binary-valued and the remaining variables as continuous, and then generating the convex hull representation. Using the fact that x_1 is now binary-valued at all vertices, this process is then repeated using another binary variable, say, x_2 , in order to determine the convex hull of feasible vertices at which both x_1 and x_2 are binary-valued. Continuing with the remaining binary variables x_3, \dots, x_n in this fashion produces the convex hull representation at the final stage. Based on this construct, which amounts to a specialized application of Sherali and Adams' hierarchy, Balas et al. describe and test a so-called *lift-and-project* cutting plane algorithm. Encouraging computational results are reported, despite the fact that this specialization produces weaker relaxations than the original Sherali–Adams' relaxations. Since this cutting plane generation scheme is based simply on the first level of Sherali and Adams' hierarchy in which binariness is enforced on only a single variable at a time, the prospect of enhancing computational performance by considering multiple binary variables at a time appears to be promising. Balas and Perregaard [16, 17] provide further discussion and connections related to such lift-and-project cuts.

The remainder of this chapter is organized as follows. Section 2 presents the basic RLT procedure of Sherali and Adams [64, 65]. Its various properties are summarized in Sect. 3, and ideas for characterizing and generating facets of the convex hull of feasible solutions (or strong valid inequalities) are presented in Sect. 4.

Extensions to multilinear programs and specializations for equality constrained problems are addressed in Sect. 5 and are illustrated using the popular quadratic assignment problem. Section 6 deals with a different RLT process proposed by Sherali, Adams, and Driscoll [90] that can be used to exploit special structures inherent in the problem, and Sect. 7 provides illustrations for some such structures, including generalized/variable upper bounding, covering, partitioning, and packing constraints, and even problem sparsity. In any RLT approach, a significant additional tightening can be obtained by using conditional logic while generating the RLT constraints. This is discussed in Sect. 8 and is illustrated on the extensively studied traveling salesman problem. Section 9 discusses classes of semidefinite cuts for enhancing RLT relaxations, along with extensions to continuous global optimization. Section 10 concludes the chapter by briefly discussing some special persistency results, extensions to general integer and mixed-discrete linear and convex programs, and various computational guidelines. To lighten the reading, proofs are omitted in this exposition, and the reader is referred to the original cited papers for such details.

2 RLT Hierarchy for Mixed-Integer Zero-One Problems

Consider a linear mixed-integer 0-1 programming problem whose (nonempty) feasible region is given as follows:

$$X = \left\{ (x, y) : \sum_{j=1}^n \alpha_{rj} x_j + \sum_{k=1}^m \gamma_{rk} y_k \geq \beta_r, \text{ for } r = 1, \dots, R, \right. \\ \left. 0 \leq x \leq e_n, \text{ } x \text{ integer}, 0 \leq y \leq e_m \right\}, \quad (1)$$

where e_n and e_m are, respectively, column vectors of n and m entries of 1 and where the continuous variables y_k are assumed to be bounded and appropriately scaled to lie in the interval $[0, 1]$ for $k = 1, \dots, m$. (Upper bounds on the continuous variables are imposed here only for convenience in exposition, as discussed further in the sequel.) Note that any equality constraints present in the formulation can be accommodated in a similar manner as are the inequalities in the following derivation. However, as shown later, it is worthwhile noting that the equality constraints can be treated in a special manner which, in fact, might sometimes encourage the writing of the R inequalities in (1) as equalities by using slack variables.

For the region described in (1), the basic construction process of the *reformulation – linearization technique* (RLT), given any level $d \in \{0, \dots, n\}$, proceeds as follows. In essence, this technique first converts the constraint set into a polynomial mixed-integer zero-one set of restrictions by multiplying the constraints with some suitable d -degree polynomial factors involving the n binary variables and their complements and subsequently linearizes the resulting problem through appropriate variable transformations. For each level d , this produces a higher-dimensional representation of the feasible region (1) in terms of the original

variables x and y and some new RLT variables (w, v) that are defined to linearize the problem. Relaxing integrality, the projection, or *shadow*, of this higher-dimensional polyhedral set onto the original variable space produces a tighter envelope for the convex hull of feasible solutions to (1) than does its ordinary linear programming (LP) or continuous relaxation. In fact, as d varies from zero to n , a hierarchy of such relaxations results, each nested within the previous one, spanning the spectrum from the ordinary linear programming relaxation to the convex hull of feasible solutions. As mentioned earlier, the first-level relaxation has itself proven to be sufficiently tight to benefit solution algorithms for several classes of problems.

More specifically, the RLT process for constructing a relaxation X_d of the region X defined in (1) corresponding to any level $d \in \{0, 1, \dots, n\}$, proceeds as follows. For $d = 0$, the relaxation X_0 is simply the LP relaxation obtained by deleting the integrality restrictions on the x -variables. In order to construct the relaxation for any level $d \in \{1, \dots, n\}$, consider the *bound factors* $x_j \geq 0$ and $(1 - x_j) \geq 0$ for $j = 1, \dots, n$ and compose *bound-factor products of degree (or order) d* by selecting some d distinct variables from the set x_1, \dots, x_n and by using either the bound factor x_j or $(1 - x_j)$ for each selected variable in a product of these terms. Mathematically, for any $d \in \{1, \dots, n\}$ and for each possible selection of d distinct variables, these (nonnegative polynomial) *bound-factor products of degree (or order) d* are given by

$$F_d(J_1, J_2) = \left[\prod_{j \in J_1} x_j \right] \left[\prod_{j \in J_2} (1 - x_j) \right] \text{ for each } J_1, J_2 \subseteq N \equiv \{1, \dots, n\} \quad (2)$$

such that $J_1 \cap J_2 = \emptyset$ and $|J_1 \cup J_2| = d$.

Any (J_1, J_2) satisfying the conditions in (2) is said to be of *order d*. For example, for $n = 3$ and $d = 2$, these factors are $x_1x_2, x_1x_3, x_2x_3, x_1(1 - x_2), x_1(1 - x_3), x_2(1 - x_3), (1 - x_1)x_2, (1 - x_1)x_3, (1 - x_2)x_3, (1 - x_1)(1 - x_2), (1 - x_1)(1 - x_3)$, and $(1 - x_2)(1 - x_3)$. In general, there are $\binom{n}{d} 2^d$ such factors. For convenience, the single factor of degree 0 is defined to be $F_0(\emptyset, \emptyset) \equiv 1$, and accordingly products over null sets are assumed to be one. Using these factors, a relaxation X_d of X , for any given $d \in \{0, \dots, n\}$, is constructed using the following two steps that comprise the reformulation – linearization technique (RLT):

Step 1 (Reformulation Phase): Multiply each of the inequalities in (1), including $0 \leq x \leq e_n$ and $0 \leq y \leq e_m$ by each of the factors $F_d(J_1, J_2)$ of degree d as defined in (2). Upon using the identity $x_j^2 \equiv x_j$ (and so $x_j(1 - x_j) = 0$) for each binary variable $x_j, j = 1, \dots, n$, this gives the following set of additional, implied, nonlinear constraints:

$$\begin{aligned} & \left[\sum_{j \in J_1} \alpha_{rj} - \beta_r \right] F_d(J_1, J_2) + \sum_{j \in N - (J_1 \cup J_2)} \alpha_{rj} F_{d+1}(J_1 + j, J_2) \\ & + \sum_{k=1}^m \gamma_{rk} y_k F_d(J_1, J_2) \geq 0 \end{aligned}$$

for $r = 1, \dots, R$, and for each (J_1, J_2) of order d ; (3)

$F_D(J_1, J_2) \geq 0$ for each (J_1, J_2) of order $D \equiv \min\{d + 1, n\}$; (4)

$F_d(J_1, J_2) \geq y_k F_d(J_1, J_2) \geq 0$

for $k = 1, \dots, m$ and for each (J_1, J_2) of order d . (5)

Step 2 (Linearization Phase): Viewing the constraints in (3)–(5) in expanded form as a sum of monomials, linearize them by substituting the following variables for the corresponding nonlinear terms for each $J \subseteq N$:

$$w_J \equiv \prod_{j \in J} x_j \text{ and } v_{Jk} \equiv y_k \prod_{j \in J} x_j, \text{ for } k = 1, \dots, m, \quad (6)$$

where, for convenience, it is assumed that

$$w_j \equiv x_j \text{ for } j = 1, \dots, n, w_\emptyset \equiv 1, \text{ and } v_{\emptyset k} \equiv y_k \text{ for } k = 1, \dots, m. \quad (7)$$

Denoting by $f_d(J_1, J_2)$ and $f_d^k(J_1, J_2)$ the respective linearized forms of the polynomial expressions $F_d(J_1, J_2)$ and $y_k F_d(J_1, J_2)$ under such a substitution, the following polyhedral set X_d is obtained, whose projection onto the (x, y) space is claimed to yield a relaxation for X :

$$\begin{aligned} X_d = \{(x, y, w, v): & \\ & \left[\sum_{j \in J_1} \alpha_{rj} - \beta_r \right] f_d(J_1, J_2) + \sum_{j \in N - (J_1 \cup J_2)} \alpha_{rj} f_{d+1}(J_1 + j, J_2) \\ & + \sum_{k=1}^m \gamma_{rk} f_d^k(J_1, J_2) \geq 0 \\ & \text{for } r = 1, \dots, R, \text{ and for each } (J_1, J_2) \text{ of order } d; \end{aligned} \quad (8)$$

$$f_D(J_1, J_2) \geq 0 \text{ for each } (J_1, J_2) \text{ of order } D \equiv \min\{d + 1, n\}; \quad (9)$$

$$\begin{aligned} & f_d(J_1, J_2) \geq f_d^k(J_1, J_2) \geq 0 \\ & \text{for } k = 1, \dots, m, \text{ and for each } (J_1, J_2) \text{ of order } d \}. \end{aligned} \quad (10)$$

Denote the projection of X_d onto the space of the original variables (x, y) by

$$X_{Pd} = \{(x, y) : (x, y, w, v) \in X_d\} \text{ for } d = 0, 1, \dots, n. \quad (11)$$

Then, Sherali and Adams [65] show that

$$X_{P0} \equiv X_0 \supseteq X_{P1} \supseteq X_{P2} \supseteq \dots \supseteq X_{Pn} = conv(X), \quad (12)$$

where $X_{P0} \equiv X_0$ (for $d = 0$) denotes the ordinary LP relaxation and $\text{conv}(X)$ denotes the convex hull of X .

Notice that the nonlinear product constraints generated by this RLT process are implied by the original constraints, and were it not for the fact that Step 1 explicitly imposes $x_j^2 = x_j$ (or $x_j(1 - x_j) = 0$) for each binary variable, this process would have simply produced a relaxation that is equivalent to the ordinary LP relaxation. Hence, the key to the tightening lies in the recognition of the binary nature of x_j in replacing x_j^2 by x_j for each $j = 1, \dots, n$.

Example 1 Consider the following mixed-integer 0-1 constraint region:

$$X = \{(x, y) : x + y \leq 2, \quad -x + y \leq 1, \quad 2x - 2y \leq 1, \quad x \text{ binary and } y \geq 0\}. \quad (13)$$

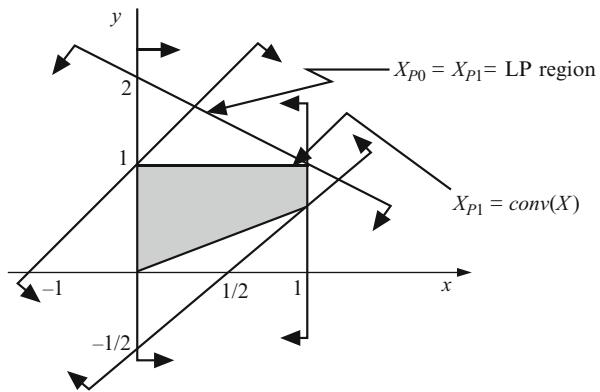
The corresponding LP relaxation of X is depicted in Fig. 1. Note that no explicit upper bound on the continuous variable y is included in (13), although the other defining constraints of X imply the boundedness of y . The RLT process can be applied directly to X as above, without creating any explicit upper bound on y . Notice that $n = 1$ for this instance, and so by the foregoing discussion, the relaxation X_1 at level $d = 1$ should produce the convex hull representation. This is verified below. For $d = 1$, the bound-factor products of order 1 are simply x and $(1 - x)$. Multiplying the constraints of X by x and by $(1 - x)$ and using $x^2 = x$ along with the linearizing substitution $v = xy$ as given by (6) produces the following constraints in the higher-dimensional space of the variables (x, y, v) , representing the set X_1 :

$$\begin{aligned} x + y \leq 2 * x &\Rightarrow -x \leq -v \\ *(1 - x) &\Rightarrow 2x + y \leq 2 + v \\ -x + y \leq 1 * x &\Rightarrow -2x \leq -v \\ *(1 - x) &\Rightarrow x + y \leq 1 + v \\ 2x - 2y \leq 1 * x &\Rightarrow x \leq 2v \\ *(1 - x) &\Rightarrow x - 2y \leq 1 - 2v \\ y \geq 0 * x &\Rightarrow v \geq 0 \\ *(1 - x) &\Rightarrow -y \leq -v \end{aligned}$$

along with $0 \leq x \leq 1$. To examine the projection X_{P1} of X_1 onto the space of the (x, y) variables, consider a restatement of X_1 as follows:

$$\begin{aligned} X_1 = \{(x, y, v) : v &\geq 2x + y - 2, \quad v \geq x + y - 1, \quad v \geq x/2, \quad v \geq 0, \\ v \leq x, \quad v \leq 2x, \quad v \leq (1-x+2y)/2, \quad v \leq y\}. \end{aligned}$$

Fig. 1 Illustration of RLT relaxations for Example 1



Then, using Fourier–Motzkin elimination, the projection X_{P1} is given by

$$\begin{aligned} X_{P1} &= \{(x, y) : \max\{2x + y - 2, x + y - 1, x/2, 0\} \\ &\leq \min\{x, 2x, (1 - x + 2y)/2, y\}\}. \end{aligned}$$

Writing the corresponding equivalent linear inequalities and dropping redundant constraints, this yields

$$X_{P1} \equiv X_{Pn} = \{(x, y) : x \leq 2y, 0 \leq x \leq 1, y \leq 1\},$$

which describes $\text{conv}(X)$ as seen in the shaded region of Fig. 1.

Example 2 Consider the set

$$X = \{(x, y) : \alpha_1 x_1 + \alpha_2 x_2 + \gamma_1 y_1 + \gamma_2 y_2 \geq \beta, 0 \leq x \leq e_2, x \text{ integer}, 0 \leq y \leq e_2\}.$$

Hence, $n = m = 2$. Consider $d = 2$, so that $D = \min\{n + 1, d\} = 2$ as well. The various sets (J_1, J_2) of order 2 and the corresponding factors are given below:

(J_1, J_2)	$\{\{1, 2\}, \emptyset\}$	$\{\{1\}, \{2\}\}$	$\{\{2\}, \{1\}\}$	$(\emptyset, \{1, 2\})$
$F_2(J_1, J_2)$	$x_1 x_2$	$x_1(1 - x_2)$	$x_2(1 - x_1)$	$(1 - x_1)(1 - x_2)$
$f_2(J_1, J_2)$	w_{12}	$x_1 - w_{12}$	$x_2 - w_{12}$	$1 - (x_1 + x_2) + w_{12}$
$f_2^k(J_1, J_2)$	v_{12k}	$v_{1k} - v_{12k}$	$v_{2k} - v_{12k}$	$y_k - (v_{1k} + v_{2k}) + v_{12k}$

Hence, this yields the following constraints (14)–(16) corresponding to (8)–(10), respectively, where v_{jk} has been written as $v_{J,k}$ for clarity:

$$X_2 = \{(x, y, w, v) :$$

$$\left. \begin{aligned} & (\alpha_1 + \alpha_2 - \beta)w_{12} + \gamma_1 v_{12,1} + \gamma_2 v_{12,2} \geq 0, \\ & (\alpha_1 - \beta)[x_1 - w_{12}] + \gamma_1(v_{1,1} - v_{12,1}) + \gamma_1(v_{1,2} - v_{12,2}) \geq 0, \\ & (\alpha_2 - \beta)[x_2 - w_{12}] + \gamma_1(v_{2,1} - v_{12,1}) + \gamma_2(v_{2,2} - v_{12,2}) \geq 0, \\ & \beta[-1 + (x_1 + x_2) - w_{12}] + \gamma_1[y_1 - (v_{1,1} + v_{2,1}) + v_{12,1}] \\ & \quad + \gamma_2[y_2 - (v_{1,2} + v_{2,2}) + v_{12,2}] \geq 0, \end{aligned} \right\} \quad (14)$$

$$w_{12} \geq 0, x_1 - w_{12} \geq 0, x_2 - w_{12} \geq 0, 1 - (x_1 + x_2) + w_{12} \geq 0, \quad (15)$$

$$\left. \begin{aligned} & w_{12} \geq v_{12,k} \geq 0, (x_1 - w_{12}) \geq (v_{1,k} - v_{12,k}) \geq 0, \\ & (x_2 - w_{12}) \geq (v_{2,k} - v_{12,k}) \geq 0, \\ & \text{and } [1 - (x_1 + x_2) + w_{12}] \geq [y_k - (v_{1,k} + v_{2,k}) + v_{12,k}] \geq 0, \end{aligned} \right\} \quad (16)$$

for $k = 1, 2\}$.

Some comments and illustrations are in order at this point. First, note that inner-products of other defining constraints could have been used in constructing the relaxation at level d , so long as the degree of the intermediate polynomial program generated at the reformulation step remains the same and no nonlinearities are created with respect to the y -variables. Hence, linearity would be preserved upon using the substitution (6) and (7). While the additional inequalities thus generated would possibly yield tighter relaxations for levels $1, \dots, n-1$, by the convex hull assertion in (12), these constraints would be implied by those defining X_n . Hence, the RLT hierarchy can be directly extended to include such additional constraints, which have been omitted for simplicity. Nonetheless, these types of additional restrictions can be advantageously included in a computational scheme employing the sets X_d for $d < n$.

Second, as alluded to earlier, note that for the case $d = 0$, using the fact that $f_0(\emptyset, \emptyset) \equiv 1$; that $f_0^k(\emptyset, \emptyset) \equiv y_k$ for $k = 1, \dots, m$; and that $f_1(j, \emptyset) = x_j$ and $f_1(\emptyset, j) = (1 - x_j)$ for $j = 1, \dots, n$, it follows that X_0 given by (8)–(10) is precisely the continuous (LP) relaxation of X in which the integrality restrictions on the x -variables are dropped. Finally, for $d = n$, note that the inequalities (9) are implied by (10) and can therefore be omitted from the representation X_n .

3 Properties of the RLT Relaxation

Equation (12) asserts the main RLT result that for $d = 0, 1, \dots, n$, the (conceptual) projections of the sets X_d represent a sequence of nested, valid relaxations leading up to the convex hull representation. Below are some salient properties of the RLT relaxations that lead to this result.

Property 1 Constraints (8)–(10) for any level's relaxation can be surrogated in a particular fashion using nonnegative multipliers to produce the constraints of the lower level relaxations. Hence, the hierarchy among these relaxations is established

via this property, whereby lower-level constraints are directly implied by higher-level constraints.

Property 2 Consider the RLT constraints (10) that are generated by multiplying the bound factors of order d with the constraints $0 \leq y \leq e_m$. If \hat{x} is any binary vector for which $(\hat{x}, \hat{y}, \hat{w}, \hat{v})$ is feasible to these constraints, then $0 \leq \hat{y} \leq e_m$, and, moreover, the identities (6) hold true. Hence, these constraints serve to ensure the nonlinear relationship (6) within the linear set X_d for any solution having binary values for x . In fact, for any binary vector \hat{x} , the solution $(\hat{x}, \hat{y}, \hat{w}, \hat{v}) \in X_d$ if and only if $(\hat{x}, \hat{y}) \in X$ and (6) holds true.

Property 3 The highest level relaxation is equivalent, via a nonsingular affine transformation, to the polytope generated by taking the convex hull of all feasible solutions to X . Hence, this result along with *Property 1* establishes (12). In fact, $(\hat{x}, \hat{y}, \hat{w}, \hat{v})$ is a vertex of X_n if and only if \hat{x} is binary-valued, (6) holds true, and \hat{y} is an extreme point of the set $\{y : (\hat{x}, y) \in X\}$.

A few remarks are pertinent at this point.

Remark 1 Observe that the upper bounds on the y -variables are not needed in X of (1) for (12) to hold true and that the foregoing analysis follows by simply eliminating any RLT product constraints that correspond to nonexistent bounding restrictions. Given feasibility of the underlying mixed-integer program, it holds that x is binary-valued at an optimum to any linear program $\max\{cx + dy : (x, y) \in X_{Pn}\}$ for which there exists an optimal solution, that is, for which this LP is not unbounded. Hence, by the foregoing properties, (12) holds true in this case as well.

Remark 2 A direct consequence of the convex hull result stated in (12) is that the convex hull representation over subsets of the x -variables can be computed in an identical fashion. In particular, suppose that the first $p \leq n$ variables are treated as binary in the set X of (1), with the remaining $(n - p)$ x -variables relaxed to be continuous. Then it directly follows that the p th level representation of this relaxed problem produces $\text{conv}\{X_0 \cap \{(x, y) : x_j \text{ is binary, } \forall j = 1, \dots, p\}\}$. The special case in which $p = 1$ precisely recovers the main step in the convexification argument of Balas et al. [19] mentioned in Sect. 1.

Remark 3 Finally, consider a situation in which there exist certain constraints from the first set of inequalities in (1) that involve only the x -variables. In this case, such constraints can be multiplied with the factors y_k and $(1 - y_k)$ for $k = 1, \dots, m$, and then the resulting constraints can be linearized using (6) and (7), similar to the other constraints (3)–(5). By *Property 2*, these additional constraints are implied when x is binary-valued in any feasible solution, but they might serve to tighten the continuous relaxations at the intermediate levels. Naturally, these constraints are implied by the other constraints defining the highest level relaxation X_n . In a likewise fashion, inequalities defining X that involve only the x -variables can be used as constraint

factors in the same spirit as the bound factors $x_j \geq 0$ and $(1 - x_j) \geq 0$ to generate products of inequalities up to a given order level for further tightening intermediate level relaxations.

4 Generating Valid Inequalities and Facets Using RLT

The discussion thus far has focused on presenting a hierarchy of relaxations leading to the convex hull representation for zero-one mixed-integer programming problems. A key advantage of this development is that the RLT produces an algebraically explicit convex hull representation at the highest level. Whereas it might not be computationally feasible to actually generate and solve the linear program based on this convex hull representation because of its potentially exponential size, there are other ways to exploit this information or facility to advantage.

One obvious strategy is to identify a subset of variables along with either original or implied constraints involving these variables for which such a convex hull representation (or some higher-order relaxation) is manageable sized. Another approach is to exploit any special structures in order to derive a simplified algebraic form of the convex hull representation for which the polyhedral cone that identifies all defining facets itself possesses a special structure. Exploiting this structure, it might then be possible to generate specific extreme directions of this cone and hence identify certain (classes of) facets that could be used to strengthen the original problem formulation. Perhaps, this algebraic simplification could yield more compact, specially structured, representations of each level in the hierarchy, hence enabling an actual implementation of a full or partial application of RLT at some intermediate level for composing tight relaxations.

Each of these ideas has been explored in different particular contexts. For example, for the Boolean quadric polytope, Sherali et al. [88] derived a characterization for all its defining facets via the extreme directions of a specially structured polyhedral cone. Accordingly, they demonstrated how various known classes of facets, including the clique, cut, and generalized cut inequality facets, emerge via the enumeration of different types of extreme directions. Moreover, by examining a particular lower-dimensional projected restriction of this polyhedral cone, they discovered a new class of facets. (This class can also be recovered via a combination of two types of facets for the related cut cone.) This type of an analysis was also used by Sherali and Lee [76] for generalized upper bounding (GUB) constrained knapsack polytopes, for which new classes of facets were characterized through a polynomial-time sequential-simultaneous lifting procedure. Known lower bounds on the coefficients of lifted facets derived from minimal covers associated with the ordinary knapsack polytope were also tightened using this framework. For the set partitioning polytope, Sherali and Lee [77] exhibited that a number of published valid inequalities along with constraint tightening procedures are automatically captured within the first- and second-level RLT relaxations themselves. A variety of partial applications of the RLT scheme were also described for deleting fractional linear programming solutions while tightening

the relaxation in the vicinity of such solutions, and various simplified forms of RLT relaxations were derived by exploiting their special structures. The polyhedral structure of many other specially structured combinatorial optimization problems can be studied using similar constructs, and tight relaxations and strong valid inequalities can be generated for such problems. As shown by Sherali [61], even convex envelopes of some special multilinear nonconvex functions can be generated using this approach.

Finally, the following strategy can be used to derive strong valid inequalities in the space of the original variables through the intermediate higher-dimensional RLT relaxations X_d , for $d \geq 1$. Note that by linear programming duality, the set of inequalities that define the projection X_{Pd} onto the (x, y) space of the higher-dimensional set X_d is characterized via the set of surrogates of the constraints defining X_d that zero out the coefficients of the new variables (w, v) in the lifted space. Whereas the enumeration of all such projected inequalities might be prohibitive, it is possible to formulate linear programming separation problems in this fashion to derive specific facets of X_{Pd} that delete a given infeasible continuous solution. The lift-and-project cutting plane algorithm of Balas et al. [19] precisely does this for the special case of $d = 1$, using RLT bound-factor products involving only a single binary variable that turns out to be fractional at an optimum to the current LP relaxation. Stronger level 1 or higher relaxations can potentially lead to improved cuts within this framework.

5 Multilinear Programs and Equality Constraints with Applications to the Quadratic Assignment Problem

This section presents two important extensions of the RLT procedure described in Sect. 2. The first of these concerns multilinear mixed-integer zero-one *polynomial* programming problems in which the continuous variables $0 \leq y \leq e_m$ appear linearly in the constraints and the objective function. This is discussed next.

Extension 1 *Multilinear mixed-integer zero-one polynomial programming problems.*

Consider the set

$$X = \{(x, y) : \sum_{t \in T_{r0}} \alpha_{rt} p(J_{1t}, J_{2t}) + \sum_{k=1}^m y_k \sum_{t \in T_{rk}} \gamma_{rkt} p(J_{1t}, J_{2t}) \geq \beta_r, \quad (17)$$

$$\forall r = 1, \dots, R, \quad 0 \leq x \leq e_n \text{ and integer, } 0 \leq y \leq e_m\},$$

where $p(J_{1t}, J_{2t}) \equiv [\prod_{j \in J_{1t}} x_j][\prod_{j \in J_{2t}} (1 - x_j)]$ are polynomial terms for the various sets (J_{1t}, J_{2t}) , for all t indexed by the sets T_{r0} and T_{rk} in (17). For $d = 0, 1, \dots, n$, a polyhedral relaxation X_d for X can be constructed as before by using the bound factors $F_d(J_1, J_2)$ to multiply the first set of constraints, where the sets (J_1, J_2) are of order d . However, denoting δ_1 as the maximum degree of the polynomial terms in x not involving the y -variables and δ_2 as the maximum

degree of the polynomial terms in x that are associated with products involving y -variables, in lieu of (4), the RLT process now uses $F_{D_1}(J_1, J_2) \geq 0$ for each (J_1, J_2) of order $D_1 \equiv \min\{d + \delta_1, n\}$, and, in lieu of (5), it employs the constraints $F_{D_2}(J_1, J_2) \geq y_k F_{D_2}(J_1, J_2) \geq 0$ for $k = 1, \dots, m$ and for each (J_1, J_2) of order $D_2 \equiv \min\{d + \delta_2, n\}$. Of course, if $D_2 \geq D_1$, then the former restrictions are unnecessary, as they are implied by the latter. Note that the computation of δ_1 and δ_2 in an optimization context must consider the terms in the objective function as well and that $\delta_1 = 1$ and $\delta_2 = 0$ for the linear case.

Now, linearizing the resulting constraints under the substitution (6) and (7) produces the desired set X_d . Because of [Property 2](#), which again holds true here, when the integrality on the x -variables is enforced, each such set X_d is equivalent to the set X . Moreover, similar to [Property 1](#), each constraint from the first set of inequalities in X_d for any $d < n$ is obtainable by surrogating two appropriate constraints from X_{d+1} . In fact, the following similar hierarchy of relaxations results:

$$\text{conv}(X) = X_{Pn} \subseteq X_{P(n-1)} \subseteq \dots \subseteq X_{P1} \subseteq X_0.$$

Extension 2 Construction of Relaxations Using Equality Constraint Representations.

Note that given any set X of the form (1), by adding slack variables to the first R inequalities, determining upper bounds on these slacks as the sum of the positive constraint coefficients minus the right-hand-side constant, and accordingly scaling these slack variables onto the unit interval, the set X can be equivalently written as follows:

$$\begin{aligned} X = \{(x, y) : & \sum_{j=1}^n \alpha_{rj} x_j + \sum_{k=1}^m \gamma_{rk} y_k + \gamma_{r(m+r)} y_{m+r} = \beta_r \text{ for } r = 1, \dots, R, \\ & 0 \leq x \leq e_n, x \text{ integer}, 0 \leq y \leq e_{m+R}\}. \end{aligned} \tag{18}$$

Now, for any $d \in \{1, \dots, n\}$, observe that the bound factor $F_d(J_1, J_2)$ for any (J_1, J_2) of order d is a linear combination of the factors $F_p(J, \emptyset)$ for $J \subseteq N$, $p \equiv |J| = 0, 1, \dots, d$. Hence, the constraint derived by multiplying an *equality* from (18) by $F_d(J_1, J_2)$ and then linearizing it via (6) and (7) is obtainable via an appropriate surrogate (with mixed-sign multipliers) of the constraints derived similarly, but using the factors $F_p(J, \emptyset)$ for $J \subseteq N$, $p \equiv |J| = 0, 1, \dots, d$. Hence, these latter factors produce constraints that can generate the other constraints, and so X_d defined by (8)–(10) corresponding to X as in (18) is *equivalent* to the following, where $f_p(J, \emptyset) \equiv w_J$ and $f_p^k(J, \emptyset) \equiv v_{Jk}$ for all $p = 0, 1, \dots, d+1$, as in (6) and (7):

$$X_d = \{(x, y, w, v) : \text{constraints of types (9) and (10) hold, and for } r = 1, \dots, R,$$

$$\left[\sum_{j \in J} \alpha_{rj} - \beta_r \right] w_J + \sum_{j \in N-J} \alpha_{rj} w_{J+j} + \sum_{k=1}^m \gamma_{rk} v_{Jk} + \gamma_{r(m+r)} v_{J(m+r)} = 0$$

$$\text{for all } J \subseteq N \text{ with } |J| = 0, 1, \dots, d\}. \quad (19)$$

Note that the savings in the number of constraints in (19) over that in (8)–(10) corresponding to the set X as in (18) is given by

$$R \left[2^d \binom{n}{d} - \sum_{i=0}^d \binom{n}{i} \right].$$

Also, observe that for $J = \emptyset$, the equalities in (19) are precisely the original equalities defining X in (18). Of course, because (19) is equivalent to the set of the types (8)–(10) that would have been derived using the factors $F_d(J_1, J_2)$ of degree d , all the foregoing results continue to hold true for (19).

Whereas the RLT approach developed for the inequality constrained case in Sect. 2 was convenient for theoretical proof purposes, note that from a computational viewpoint, when $d < n$, the representation in (19) has fewer type (8) complicating constraints and variables (when slacks for (8) are included) than does (8)–(10) as given by the above savings expression but has $R \times 2^d \binom{n}{d}$ additional constraints of the type (10), counting the nonnegativity restrictions on the slacks in (8), as compared with the inequality constrained case. Hence, depending on the structure, either form of the representation of these relaxations may be employed, as convenient.

To summarize, in general, when constructing the level d relaxation X_d in the presence of (perhaps original) equality constraints, each equality constraint needs to be multiplied by the factors $F_p(J, \emptyset)$ for $J \subseteq N$, $p \equiv |J| = 0, 1, \dots, d$ in the RLT reformulation phase. Naturally, factors $F_p(J, \emptyset)$ that are known to be zeros (and so we explicitly set $w_J \equiv 0$), that is, any such factor for which we know that there does not exist a feasible solution that has $x_j = 1$, $\forall j \in J$, need not be used in constructing these product constraints. The RLT linearization phase is then applied as before.

Example 3 To illustrate both the foregoing extensions, consider the celebrated quadratic assignment problem defined as follows:

$$\mathbf{QAP : Minimize} \sum_{i=1}^m \sum_{j=1}^m \sum_{k>i} \sum_{\ell \neq j} c_{ijkl} x_{ij} x_{k\ell} \quad (20)$$

$$\text{subject to } \sum_{i=1}^m x_{ij} = 1, \quad \forall j = 1, \dots, m \quad (21)$$

$$\sum_{j=1}^m x_{ij} = 1, \quad \forall i = 1, \dots, m \quad (22)$$

$$x \geq 0 \text{ and integer.} \quad (23)$$

This problem has equality restrictions, so additional slack variables as in (18) are not needed. Furthermore, there exist no continuous variables y , and the restrictions $x \leq e_n$ found in (1) and (18) are implied by the remaining constraints.

First Level Relaxation: To construct the relaxation at the first level, in addition to the constraints (21) and (22) and the nonnegativity restrictions on x in (23), the lifted representation includes the product constraints obtained by multiplying each constraint in (21) and (22) by each $x_{k\ell} = F_1(\{x_{k\ell}\}, \emptyset)$, and then the rest of the RLT procedure is applied as usual, using the substitution $w_{ijk\ell} = x_{ij} x_{k\ell}$, $\forall i, j, k, \ell$. Note that for any $j \in \{1, \dots, m\}$, multiplying (21) by x_{kj} for each $k \in \{1, \dots, m\}$ produces upon using $x_{kj}^2 = x_{kj}$ that

$$\sum_{i \neq k} w_{ijkj} = 0, \quad \forall k = 1, \dots, m. \quad (24)$$

Hence, since $w \geq 0$, this implies that $w_{ijkj} \equiv 0$, $\forall i \neq k$. Similarly, from (22), $w_{ijil} = 0$, $\forall j \neq l$. Furthermore, noting that $w_{ijkl} \equiv w_{klij}$, it is only necessary to define w_{ijkl} , $\forall i < k, j \neq l$. (For convenience, however, the exposition below defines w_{ijkl} , $\forall i \neq k, j \neq l$, and explicitly imposes $w_{ijkl} \equiv w_{klij}$.) Additionally, and as alluded to above, since the continuous relaxation only needs to impose $x_{ij} \geq 0$ in (23) because $x_{ij} \leq 1$ is implied by the constraints (21) and (22), the reformulation phase only multiplies the factors $x_{k\ell} \geq 0$ and $(1 - x_{k\ell}) \geq 0$ with each constraint $x_{ij} \geq 0$. This produces the restrictions $w_{ijkl} \leq x_{ij}$ and $w_{ijkl} \leq x_{k\ell}$, $\forall i, j, k, \ell$, along with $w \geq 0$. However, the former variable upper bounding constraints are implied by the constraints (28)–(30) below. Hence, the first-level relaxation (that yields a lower bound on QAP) is given as follows:

$$\text{Minimize } \sum_i \sum_j \sum_{k>i} \sum_{\ell \neq j} c_{ijkl} w_{ijkl} \quad (25)$$

$$\text{subject to } \sum_i x_{ij} = 1, \quad \forall j \quad (26)$$

$$\sum_j x_{ij} = 1, \quad \forall i \quad (27)$$

$$\sum_{i \neq k} w_{ijkl} = x_{k\ell}, \quad \forall j, k, \ell \neq j \quad (28)$$

$$\sum_{j \neq \ell} w_{ijkl} = x_{k\ell}, \quad \forall i, \ell, k \neq i \quad (29)$$

$$x \geq 0, \quad w \geq 0, \quad w_{ijkl} \equiv w_{klij}, \quad \forall i < k, j \neq \ell. \quad (30)$$

Second Level Relaxation: Following the same procedure as above and eliminating null variables as well as null and redundant constraints produces the same

relaxation as in (25)–(30) with additional constraints (and variables) corresponding to multiplying each equality constraint in (21) by each factor $x_{k\ell}x_{pq}$ for $k < p$, $\ell \neq q \neq j$, and by multiplying each constraint in (22) by each factor $x_{k\ell}x_{pq}$ for $k < p$, $k \neq i \neq p$, $\ell \neq q$. This generates the additional constraints

$$\sum_{i \neq k, p} w_{ijk\ell pq} = w_{k\ell pq}, \quad \forall j, k, \ell, p, q, \text{ where } k < p, \text{ and } \ell \neq q \neq j, \quad (31)$$

$$\sum_{j \neq \ell, q} w_{ijk\ell pq} = w_{k\ell pq}, \quad \forall j, k, \ell, p, q, \text{ where } k < p, k \neq i \neq p, \text{ and } \ell \neq q, \quad (32)$$

along with $w_{ij,k\ell,pq} \geq 0$, noting that this is the same variable for each permutation of

$$ij, k\ell, pq \text{ for all distinct } i, k, p \text{ and } j, \ell, q. \quad (33)$$

Such relaxations have been computationally tested by Ramakrishnan et al. [59], Ramachandran and Pekny [58], Adams and Johnson [2], and Adams et al. [10] with promising results. Extensions of such specializations for general set partitioning problems are discussed by Sherali and Lee [77], and concise representations having the strengths of the different level RLT relaxations for set partitioning structures are explored in Adams et al. [9]. In fact, Adams and Johnson [2] show that the lower bound produced by the first-level relaxation itself subsumes a multitude of known lower bounding techniques in the literature, including a host of matrix reduction strategies. These authors designed a heuristic dual ascent procedure for the level-1 relaxation, to quickly solve the large linear program by exploiting special structures. In an effort to make this algorithm generally applicable, no special exploitation of flow and/or distance symmetries was considered. This work was extended in Adams et al. [10] where a similar type of dual ascent algorithm was applied to the level-2 representation. This enabled the solution of the largest $m = 30$ test cases from the standard test bed of Nugent et al. [53] to optimality. Only one other paper by Anstreicher et al. [12] was successful in verifying an optimal solution, and this paper reported enumerating approximately 11.9 billion nodes in the binary search tree. The RLT enumerated 543,000 nodes and required less than 40% of the CPU time. As far as the strength of the RLT relaxations is concerned, on a set of standard test problems of sizes $m = 8$ –20, the lower bounds from the level-1 RLT produced by the dual ascent procedure in Adams and Johnson [2] uniformly dominated 12 other competing lower bounding schemes except for one problem of size 20, where the RLT procedure yielded a lower bound of 2145 (later improved to 2,179 in Adams et al. [10]), while an eigenvalue-based procedure produced a lower bound of 2,229, the optimum value being 2,570 for this problem. The bounds obtained from the dual ascent applied to the level-2 RLT representation formulated in Adams et al. [10] improved this bound to 2,508. Also, Resende et al. [60] have been able to solve the first-level RLT relaxation exactly for problems of size up to 30 using an interior point method that employs a preconditioned conjugate gradient technique to solve the system of equations for computing the

search directions. (For the problem of size 20, the exact solution value of the lower bounding RLT relaxation turned out to be 2,182, compared to a value of 2,179 using the aforementioned dual ascent scheme.) Sherali and Brown [70] have also applied RLT to the problem of assigning aircraft to gates at an airport, with the objective of minimizing passenger walking distances. The problem is modeled as a variant of the quadratic assignment problem with partial assignment and set packing constraints. The quadratic problem is then equivalently linearized by applying the first level of the RLT. In addition to simply linearizing the problem, the application of this technique generates additional constraints that provide a tighter linear programming representation. Since even the first-level relaxation can get quite large, several alternative relaxations are investigated that either delete or aggregate classes of RLT constraints. All these relaxations are embedded in a heuristic that solves a sequence of such relaxations, automatically selecting at each stage the tightest relaxation that can be solved with an acceptable estimated effort, and based on the solution obtained, it fixes a suitable subset of variables to 0-1 values. This process is repeated until a feasible solution is constructed. The procedure was computationally tested using realistic data obtained from *USAir* for problems having up to 7 gates and 36 flights. For all the test problems ranging from 4 gates and 36 flights to 7 gates and 14 flights, for which the size of the first-level relaxation was manageable (having 14,494 and 4,084 constraints, respectively, for these two problem sizes), this first-level RLT relaxation itself always produced an optimal 0-1 solution.

As a final note, Adams and Sherali [3] had earlier developed a technique for solving general 0-1 quadratic programming problems using a relaxation that turns out to be precisely the level-1 RLT relaxation discussed above. This relaxation was shown to theoretically dominate other existing linearizations and was demonstrated to computationally produce far tighter lower bounds, where the test cases involved quadratic set covering problems having up to 70 variables and 40 constraints. For example, for this largest size problem, where the optimum objective value was 1,312, the first-level RLT relaxation produced an initial lower bound of 1,289 at the root node and enumerated 14 nodes to solve the problem in 79 cpu seconds on an IBM 3081 Series D24 Group K computer. When the same algorithmic strategies were used on a relaxation that did not include the special RLT constraints, the initial lower bound obtained was only 398, and the algorithm enumerated 2,130 nodes, consuming 197 cpu seconds.

6 Exploiting Special Structures to Generate Tighter RLT Representations

The basic RLT procedure discussed in Sect. 2 generates a hierarchy of relaxations that spans the spectrum from the continuous LP relaxation to the convex hull of feasible solutions for linear mixed-integer 0-1 programming problems. The key construct here is to compose a set of multiplication factors based on the bounding constraints $0 \leq x \leq e_n$ for the binary variables x and to use these factors to generate implied nonlinear product constraints, then tighten these constraints using the fact

that $x_j^2 = x_j$, $\forall j = 1, \dots, n$, and subsequently linearize the resulting polynomial problem through a variable substitution process. This process then yields a family of tighter representations of the problem in higher-dimensional spaces.

It should seem intuitive that given a set S of constraints involving the x -variables that imply the bounding restrictions $0 \leq x \leq e_n$, it might be possible to generate a similar hierarchy by utilizing multiplicative factors that are composed from the constraints defining S itself. Indeed, as exemplified in this section, such generalized so-called *S-factors* can not only be used to construct a hierarchy of relaxations leading to the convex hull representation, but they also provide an opportunity to exploit inherent special structures. An artful application of this strategy can produce relaxations that are more compact than the ones available through the traditional RLT process of Sect. 2 while at the same time being tighter as well as affording the opportunity to construct the convex hull representation at lower levels in the hierarchy in certain cases.

Consider the feasible region of a mixed-integer 0-1 programming problem stated in the following form:

$$X = \{(x, y) \in R^n \times R^m : Ax + Dy \geq b, x \in S, x \text{ binary}, y \geq 0\}, \quad (34)$$

where

$$S \equiv \{x : g_i x - g_{0i} \geq 0 \text{ for } i = 1, \dots, p\}. \quad (35)$$

Here, the constraints defining the set S have been specially composed to generate useful, tight relaxations as revealed in the sequel. For now, in theory, all that is required of the set S is that it implies the bounds $0 \leq x \leq e_n$ on the x -variables where, as before, e_n is a vector of n -ones. Specifically, assume that for each $t = 1, \dots, n$,

$$\min\{x_t : x \in S\} = 0 \text{ and } \max\{x_t : x \in S\} = 1. \quad (36)$$

Note that $\min\{x_t\} > 0$ implies that $x_t \equiv 1$ and $\max\{x_t\} < 1$ implies that $x_t \equiv 0$, and if both these conditions hold for any t , then the problem is infeasible. Therefore, without loss of generality, assume that the equalities of (36) hold for all $t = 1, \dots, n$. Furthermore, note that (36) ensures that $p \geq n + 1$.

Now, define the sets P and \bar{P} as follows, where \bar{P} is a multi-set that duplicates each index in P n times:

$$P = \{1, \dots, p\}, \text{ and } \bar{P} = \{n \text{ copies of } P\}. \quad (37)$$

The construction of the new hierarchy proceeds in a manner similar to that of Sect. 2. At any chosen level of relaxation $d \in \{1, \dots, n\}$, a higher-dimensional relaxation \bar{X}_d is constructed by considering the *S-factors* of order d defined as follows:

$$g(J) \equiv \prod_{i \in J} (g_i x - g_{0i}) \text{ for each distinct } J \subseteq \bar{P}, |J| = d. \quad (38)$$

Note that the S -factors of order d are composed by examining the collection of *constraint factors* $g_i x - g_{0i} \geq 0$, $i = 1, \dots, p$, and constructing the product of some d of these constraint factors, *including possible repetitions*, where the multi-set \bar{P} defined in (37) permits such repetitions as d varies from 1 to n . Since the relaxation described below will recover the convex hull representation for $d = n$, it is not necessary to consider $d > n$. Using $(d - 1)$ suitable dummy indices to represent duplications, it can be easily verified that there are a total of

$$\binom{p + d - 1}{d} = (p + d - 1)!/d!(p - 1)!$$

distinct factors of this type at level d .

These factors are then used in a *special structures reformulation – linearization technique*, abbreviated SSRLT, as stated below, for generating the relaxation \bar{X}_d .

- (a) *Reformulation Phase.* Multiply each inequality defining the feasible region (34) (including the constraints defining S) by each S -factor $g(J)$ of order d , and apply the identity $x_j^2 = x_j$ for all $j \in \{1, \dots, n\}$.
- (b) *Linearization Phase.* Linearize the resulting polynomial program by using the substitution defined in (39) below, to produce the d th level relaxation \bar{X}_d :

$$w_J = \prod_{j \in J} x_j, \quad \forall J \subseteq N \text{ and } v_{Jk} = y_k \prod_{j \in J} x_j, \quad \forall J \subseteq N, \quad \forall k = 1, \dots, m. \quad (39)$$

For conceptual purposes, as in Sect. 2, define the projection of \bar{X}_d onto the space of the original variables as

$$\bar{X}_{Pd} = \{(x, y) : (x, y, w, v) \in \bar{X}_d\}, \quad \forall d = 1, \dots, n. \quad (40)$$

Additionally, as before, denote $\bar{X}_{P0} \equiv \bar{X}_0$ (for $d = 0$) as the ordinary linear programming relaxation. Sherali et al. [90] then show that similar to (12),

$$\bar{X}_{P0} \equiv \bar{X}_0 \supseteq \bar{X}_{P1} \supseteq \bar{X}_{P2} \supseteq \dots \supseteq \bar{X}_{Pn} = \text{conv}(X), \quad (41)$$

where $\text{conv}(\cdot)$ denotes the convex hull of feasible solutions. Moreover, this hierarchy dominates that of Sect. 2 in the sense that

$$\bar{X}_d \subseteq X_d \text{ and } \bar{X}_{Pd} \subseteq X_{Pd}, \quad \forall d = 0, 1, \dots, n. \quad (42)$$

For convenience in notation, let $\{\cdot\}_L$ henceforth denote the process of linearizing a polynomial expression $\{\cdot\}$ in x and y via the substitution defined in (39), following the use of the identity $x_j^2 = x_j$, $\forall j = 1, \dots, n$.

Before proceeding further, some important comments pertaining to the application of SSRLT are worth highlighting. First, in an actual implementation, note that under the substitution $x_j^2 = x_j$ for all j , several terms defining the factors in (38)

might be zeros, either by definition or due to the restrictions of the set X in (34). For example, if $S \equiv \{x : 0 \leq x \leq e_n\}$, when $d = 2$, one such factor of type (38) is $x_j(1 - x_j)$ for $j \in \{1, \dots, n\}$, which is clearly null when x_j^2 is equated with x_j . Additionally, if $S \equiv \{x : 0 \leq x \leq e_n\}$, then multiplying the inequalities defining X in (34) by any factor $F_d(J_1, J_2)$ for which the remaining constraints of \bar{X}_d enforce $f_d(J_1, J_2) = 0$ will yield null constraints. Second, some of these factors might be implied in the sense that they can be reproduced as a nonnegative surrogate of other such factors that are generated in (38). For example, when $d = 3$ for $S \equiv \{x : 0 \leq x \leq e_n\}$, the factor $x_t^2 x_r \geq 0$ of order 3 is equivalent to $x_t x_r \geq 0$ of order 2, which is implied by other nonnegative factors of order 3 generated by the RLT constraints by [Property 1](#). All such null and implied factors and terms should be eliminated in an actual application of SSRLT. Third, if any constraint in \bar{X}_0 of (34) and (41) is implied by the remaining constraints, then this constraint can be removed from X without changing any resulting set \bar{X}_d . (This same logic holds relative to the foregoing RLT process and the sets X_d .) To illustrate, any single constraint in (21) or (22) can be removed from the QAP formulation of [Example 3](#) while preserving the strengths of the prescribed relaxations at levels 1 and 2, saving $1 + m(m - 1)$ and $1 + m(m - 1) + m(m - 1)^2(m - 2)$ constraints, respectively.

As evident from the foregoing comments, the RLT process described in [Sect. 2](#) is a special case of SSRLT when $S \equiv \{x : 0 \leq x \leq e_n\}$. Note that one obvious scheme for generating tighter relaxations via SSRLT is to include in the latter set S certain suitable additional constraints depending on the problem structure and hence generate S -factors at level d that include $f_d(J_1, J_2)$ of order d as defined in [Sect. 2](#), along with any collection of additional S -factors as obtained via (38). Eliminating any null terms or implied factors thus generated, a hierarchy of relaxations can be derived using SSRLT that would dominate the ordinary RLT hierarchy at each level. The focus in the discussion that follows largely resides on less obvious, and richer, instances where the set S possesses a special structure that implies the restrictions $0 \leq x \leq e_n$, without explicitly containing these bounding constraints.

Observe that SSRLT could be conceptually visualized as being an inductive process, with the relaxation at level $(d + 1)$ being produced by multiplying each of the constraints in the relaxation at level d with each S -factor defining S . Constraints produced by this process that effectively use null (zero) factor expressions $g(J)$ of order d are null constraints. Constraints produced by this process that effectively use factors $g(J)$ that are implied by other factors in (38) are themselves implied by the constraints generated using the latter factors. Hence, the processes of reducing the set of factors based on eliminating the use of null or implied factors at the reformulation step, or that of eliminating the corresponding redundant constraints generated by such factors, are equivalent steps. It follows that an equivalent relaxation at level d would be produced by using only the non-null, non-implied factors, recognizing any zero variable terms in the resulting relaxation as identified by the S -factors. Furthermore, such non-redundant/non-null factors can be generated inductively through the levels, recognizing zero terms revealed at previous levels. This latter relaxation is what should actually be generated in practice. [Sections 7](#) and [8](#) provide several examples. In a similar fashion to the

RLT process of Sect. 2, a hierarchy leading to the convex hull representation can be generated in a piecewise manner by sequentially enforcing binariness on sets of variables and constructing the highest level relaxation for each considered set in turn.

Finally, consider the treatment of equality constraints and equality factors. Note that whenever an equality constraint factor defines an S -factor, any resulting product constraint is an equality restriction. Consequently, in the presence of equality constrained factors, in general, it is only necessary to multiply the corresponding equality constraint factors simply with each x and y variable alone as well as by the constant 1, since the product with any other expression in x and y can be composed using these resulting products. Moreover, since the products with the x -variables are already being generated via other SSRLT constraints by virtue of the corresponding defining equality constraints of S already being included within X and since $x \geq 0$ is implied by the inequality restrictions of $x \in S$, only products using the y -variables are necessary.

Furthermore, in this connection, note that if X contains equality structural constraints, in general, then these can be treated as in Sect. 5. That is, at level d , these equality constraints would simply need to be multiplied by the factors $F_p(J, \emptyset)$ for $J \subseteq N$, $p \equiv |J| = 0, 1, \dots, d$. Naturally, factors $F_p(J, \emptyset)$ that are known to be zeros (and so we explicitly set $w_J \equiv 0$), that is, any such factor for which no feasible solution exists that has $x_j = 1$ for all $j \in J$, need not be used in constructing these product constraints and can be set equal to zero in the relaxation.

Example 4 For example, consider a set $S = \{x : e_n \cdot x = 1, x \geq 0\}$. Then the S -factors of order 1 are the expressions that define the restrictions

$$\{(1 - e_n \cdot x) = 0, x_1 \geq 0, \dots, x_n \geq 0\}, \quad (43)$$

which include the equality constraint factor along with the bound factors $x_1 \geq 0, \dots, x_n \geq 0$. To compose the S -factors of order 2, note that

$$x_t(1 - e_n \cdot x) = 0 \text{ yields } \sum_{j \neq t} w_{(jt)} = 0, \forall t, \quad (44)$$

upon using $x_t^2 \equiv x_t$ and substituting $w_{(jt)}$ for $x_j x_t$, $\forall j \neq t$ according to (39). (Note that $w_{(jt)}$ is defined only for $j < t$, and accordingly, the notation in (44) denotes $w_{(jt)} \equiv w_{jt}$ if $j < t$ and $w_{(jt)} \equiv w_{tj}$ if $t < j$.) Equation (44) along with

$$w_{(jt)} \equiv x_j x_t \geq 0, \forall j \neq t, \quad (45)$$

where (45) is produced by the other S -factors of order 2, imply that $w_{(jt)} \equiv 0, \forall j \neq t$, hence yielding null factors via (44) and (45). The only non-null S -factors of order 2 are therefore produced by pairwise self-products of the constraints defining S . But $(1 - e_n \cdot x)^2 = 0$ and $x_j^2 \geq 0, \forall j = 1, \dots, n$,

respectively, yield $(1 - e_n \cdot x) = 0$ and $x_j \geq 0$, $\forall j = 1, \dots, n$, upon using $x_j^2 = x_j$ and $x_j x_t = 0$, $\forall j \neq t$, as above. Hence, the reduced set of factors of order 2 is precisely the same as those of order 1, and this continues for all levels $2, \dots, n$. Consequently, by (41), the convex hull representation is produced at the first level itself for this example.

To produce this level-1 representation, all the constraints defining X (including the ones in S) are first multiplied by each factor $x_j \geq 0$, $j = 1, \dots, n$, from (43). However, for the equality factor $(1 - e_n \cdot x) = 0$, by the foregoing discussion, it is only necessary to construct the RLT constraints $\{y_k(1 - e_n \cdot x)\}_L = 0$, $\forall k = 1, \dots, m$, and retain $e_n \cdot x = 1$. The resulting relaxation produces the convex hull representation as asserted above.

To further reinforce some of the preceding ideas before presenting additional specific details, consider another example that includes an equality constraint in S but also explicitly includes the bound restrictions $0 \leq x \leq e_n$. As mentioned above, since the S -factors would now include the regular RLT bound factors, the use of any S -factors other than these bound factors is optional.

Example 5 Suppose that $n = 4$ and consider $S = \{x \in R^4 : x_1 + x_2 + x_3 + x_4 = 2$, $0 \leq x \leq e_4\}$. The following factors are derived that can be applied in SSRLT, noting the equality constraint defining S :

- (a) *Level-1 factors:* Bound factors $x_j \geq 0$ and $(1 - x_j) \geq 0$, $\forall j = 1, \dots, 4$, and optionally $(e_4 \cdot x - 2) = 0$ (to be multiplied by 1 and by each y -variable alone as noted above).
- (b) *Level-2 factors:* Bound factors of order 2 given by $\{x_i x_j, (1 - x_i)x_j, x_i(1 - x_j)$ and $(1 - x_i)(1 - x_j)$, $\forall 1 \leq i < j \leq 4\}$, and optionally any factors (to be applied to $y \geq 0$ alone) from the set $\{x_i - \sum_{j \neq i} x_i x_j = 0$, $\forall i = 1, \dots, 4$, obtained by multiplying $e_4 \cdot x - 2$ by each x_i , $i = 1, \dots, 4$, and $(e_4 \cdot x - 2) = 0$ itself, obtained from $(e_4 \cdot x - 2)^2 = 0$ upon using $\sum_{j \neq i} x_i x_j = x_i$, $\forall i\}$.
- (c) *Level-d factors, d=3, 4:* Bound factors $F_d(J_1, J_2) \geq 0$ of order d , with the additional restriction that all third and fourth order terms are zeros, plus optionally factors from the optional set at level 2. Note that the valid implication of polynomial terms of order 3 being zero, for example, is obtained through the RLT process by multiplying $x_i - \sum_{j \neq i} x_i x_j = 0$ with x_k , for each i, k , $i \neq k$. This gives $\sum_{j \neq i, k} x_i x_j x_k = 0$, which, by the nonnegativity of each triple product term, implies that $x_i x_j x_k = 0$, $\forall i \neq j \neq k$.

In a likewise fashion, for set partitioning problems, for example, any quadratic or higher-order products of variables that involve a pair of variables that appear together in any constraint are zeros. More generally, any product term that contains variables or their complements that cannot simultaneously take on a value of 1 in any feasible solution can be restricted to zero. Sherali and Lee [77] use this structure to present a specialization of RLT to derive explicit reduced level- d representations in their analysis of set partitioning problems.

7 Applications of RLT for Some Particular Special Structures

This section demonstrates how some specific special structures can be exploited in designing an application of the general framework of SSRLT. This discussion will also illuminate the relationship between RLT and SSRLT, beyond the simple dominance result stated in (42). For this purpose, various commonly occurring special structures are examined, such as generalized upper bounding (GUB) constraints, variable upper bounding (VUB) constraints, and, to a lesser degree of structure, problem sparsity. These illustrations are by no means exhaustive; the motivation here is to present the basic framework for this approach and encourage the reader to design similar constructs for other applications on a case-by-case basis.

7.1 Generalized Upper Bounding (GUB) or Multiple Choice Constraints

Suppose that the set S of (35) is given as follows:

$$S = \left\{ x : \sum_{j \in N_i} x_j \leq 1, \forall i \in Q \equiv \{1, \dots, q\}, x \geq 0 \right\}, \quad (46)$$

where $\bigcup_{i \in Q} N_i \equiv N \equiv \{1, \dots, n\}$. Problems possessing this particular special structure arise in various settings including maximum cardinality node packing, set packing, capital budgeting, and menu planning problems, among others (see Nemhauser and Wolsey [50]).

First, suppose that $q \equiv 1$ in (46), so that

$$S \equiv \{x : e_n \cdot x \leq 1, x \geq 0\}.$$

The S -factors of various orders for this particular set can be derived as follows:

- (a) *S -factors at level 1.* These factors are directly obtained from the set S via the constraint factors $(1 - e_n \cdot x) \geq 0$ and $x_j \geq 0, \forall j = 1, \dots, n$.
- (b) *S -factors at level 2.* The linearization operation $[x_t(1 - e_n \cdot x)]_L \geq 0$ produces an expression

$$\sum_{j \neq t} w_{(jt)} \leq 0, \forall t = 1, \dots, n,$$

where $w_{(jt)} \equiv w_{jt}$ if $j < t$ and $w_{(jt)} \equiv w_{tj}$ if $t < j$. Moreover, the pairwise products of x_j and x_t , $j \neq t$, produces factors of the type

$$(x_j x_t) \geq 0, \forall j \neq t, \text{ or } w_{(jt)} \geq 0, \forall j \neq t.$$

Similar to (44) and (45), the foregoing two sets of inequalities imply that

$$w_{(jt)} = 0, \forall j \neq t. \quad (47)$$

Consequently, under (47), the only non-null S -factors of order 2 that survive are self-product factors of the type $(x_t x_t) \geq 0$, $\forall t$, and $(1 - e_n \cdot x) \cdot (1 - e_n \cdot x) \geq 0$. These yield the same factors as at level 1, upon using $x_t^2 = x_t$, $\forall t$, along with (47), as seen in Sect. 6. Hence, the factors that need to be used for constructing the set \bar{X}_2 are

$$(1 - e_n \cdot x) \geq 0 \text{ and } x_j \geq 0, \quad j = 1, \dots, n.$$

Hence, notice that $\bar{X}_2 = \bar{X}_1$, and this equivalence relation continues through all levels of relaxations up to \bar{X}_n . Therefore, the first-level relaxation itself produces the convex hull representation in this case. There are two insightful points worthy of note in the context of this example. First, as illustrated next, although RLT recognizes that (47) holds true at each relaxation level, it may not produce the convex hull representation at the first level as does SSRLT.

Example 6 Let

$$X = \{(x_1, x_2) : 6x_1 + 3x_2 \geq 2, \quad x_1 + x_2 \leq 1, \quad x \text{ binary}\},$$

and consider the generation of the first-level RLT relaxation. Note that the factors used in this context are x_j and $(1 - x_j)$ for $j = 1, 2$. Examining the product of $x_1 + x_2 \leq 1$ with x_1 yields $w_{12} \leq 0$, which together with $w_{12} \equiv \{x_1 x_2\}_L \geq 0$ yields $w_{12} = 0$. Other products of the factors x_j and $(1 - x_j)$, $j = 1, 2$, with $x_1 + x_2 \leq 1$ and $0 \leq x \leq e_2$ simply reproduce these same latter constraints.

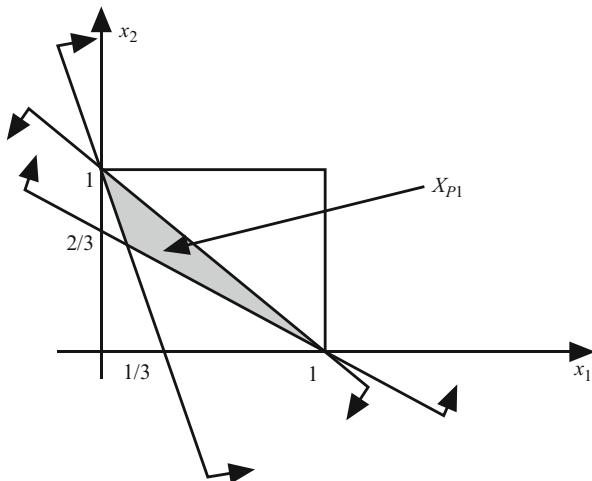
Examining the products of $6x_1 + 3x_2 \geq 2$ with these first-level factors yields nonredundant inequalities when the factors $(1 - x_1)$ and $(1 - x_2)$ are used, thus generating the constraints $2x_1 + 3x_2 \geq 2$ and $3x_1 + x_2 \geq 1$, respectively, since $w_{12} = 0$. Hence, the first-level relaxation (directly in projected form in this case) is obtained as follows:

$$X_{P1} = \{(x_1, x_2) : 2x_1 + 3x_2 \geq 2, \quad 3x_1 + x_2 \geq 1, \quad x_1 + x_2 \leq 1, \quad x \geq 0\}.$$

Figure 2 depicts the shaded region X_{P1} . However, using SSRLT and noting the above argument, results in $\bar{X}_{P1} = \text{conv}(X) \equiv \{x : x_1 + x_2 = 1, \quad x \geq 0\}$, which is a strict subset of X_{P1} .

A second point to note is that the generalized upper bounding inequality $e_n \cdot x \leq 1$ could have been written as an equality $e_n \cdot x + x_{n+1} = 1$ by introducing a slack variable x_{n+1} , and then, *recognizing the binariness of this slack variable*, it could have been used as an additional variable in composing the bound-factor products while applying RLT. Although this would have produced the same relaxation as with SSRLT, the process would have generated several more redundant constraints while applying the factors x_j and $(1 - x_j)$, $\forall j = 1, \dots, n + 1$, to the constraints, as opposed to using the fewer factors $(1 - e_n \cdot x)$ and x_j , $\forall j = 1, \dots, n$, as needed by SSRLT. However, in more general cases of the set S , such a transformation that yields the same representation using RLT as obtained via SSRLT may not be accessible (see Example 7 below, for instance).

Fig. 2 The first-level relaxation using RLT



Example 7 Next, consider the case of $q = 2$ in (46). For the sake of illustration, suppose that

$$S = \{x \in R^5 : x_1 + x_2 + x_3 \leq 1, x_3 + x_4 + x_5 \leq 1, \text{ and } x \geq 0\}. \quad (48)$$

- (a) *S-factors at level 1*: These are simply the constraints defining S .
- (b) *S-factors at level 2*: As before, the pairwise products within each GUB set of variables will reproduce the same factors as at the first level, since (47) holds true within each GUB set. However, examining mixed factors across the two GUB sets produces the nonnegative quadratic bound-factor products x_1x_4 , x_1x_5 , x_2x_4 , and x_2x_5 , along with the following factor products, recognizing that any quadratic product involving x_3 is zero because this variable appears in both GUB sets:

$$x_4 \cdot (1 - x_1 - x_2 - x_3) \geq 0 \text{ yielding } x_4 - x_1x_4 - x_2x_4 \geq 0,$$

$$x_5 \cdot (1 - x_1 - x_2 - x_3) \geq 0 \text{ yielding } x_5 - x_1x_5 - x_2x_5 \geq 0,$$

$$x_1 \cdot (1 - x_3 - x_4 - x_5) \geq 0 \text{ yielding } x_1 - x_1x_4 - x_1x_5 \geq 0,$$

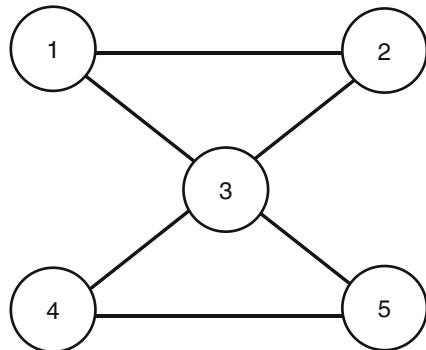
$$x_2 \cdot (1 - x_3 - x_4 - x_5) \geq 0 \text{ yielding } x_2 - x_2x_4 - x_2x_5 \geq 0,$$

and

$$(1 - x_1 - x_2 - x_3) \cdot (1 - x_3 - x_4 - x_5) \geq 0 \text{ yielding}$$

$$1 - x_1 - x_2 - x_3 - x_4 - x_5 + x_1x_4 + x_1x_5 + x_2x_4 + x_2x_5 \geq 0.$$

These can now be applied to the constraints defining X , recognizing the terms that have been identified to be zeros.

Fig. 3 Vertex packing graph

(c) *S-factors at levels ≥ 3* : Since there are only two GUB sets in this example and since any triple product of distinct factors must involve a pair of factors coming from the defining constraints corresponding to the same GUB set and the latter product is zero, all such products must vanish. Hence, all factors at level 3, and similarly at levels 4 and 5, coincide with those at level 2. In other words, the relaxation at level 2 (defined as \bar{X}_2) itself yields the convex hull representation.

In general, the *level equal to the independence number of the underlying intersection graph corresponding to the GUB constraints, which simply equals the maximum number of variables that can simultaneously be 1, is sufficient to generate the convex hull representation*. In the case of (46), the convex hull representation would be obtained at level q or earlier.

An enlightening special case of (46) deals with the vertex packing problem. Given a graph $G = (V, E)$ with vertex set $V = \{v_1, v_2, \dots, v_n\}$, an edge set E connecting pairs of vertices in V , and a weight c_j associated with each vertex v_j , the *vertex packing problem* is to select a maximum weighted subset of vertices such that no two vertices are connected by an edge. For each $j = 1, \dots, n$, by denoting the binary variable x_j to equal 1 if vertex j is chosen and 0 otherwise, the vertex packing problem can be stated as follows: Maximize $\{\sum_{j=1}^n c_j x_j : x_i + x_j \leq 1, \forall (i, j) \in E, x \text{ binary}\}$. The convex hull representation over any subset P of the variables can be obtained as above by considering any clique cover of the subgraph induced by the corresponding vertices, with each set N_i corresponding to the variables defining some clique i . In fact, given a cover that has q cliques, where each edge of E is included in some clique, the S -factors of level q themselves define the convex hull representation, since their products with the packing constraints, as well as with the nonnegativity restrictions on x , are implied by these factors. To illustrate, the inequalities of (48) can be considered as a (maximum cardinality) clique cover of the vertex packing problem on the graph in Fig. 3, and so, the stated S -factors of level 2 themselves define the convex hull representation. This general observation may have widespread applicability since, as noted by Garfinkel and Nemhauser [29], any finite integer linear program can be reformulated as a packing problem.

Table 1 Computational results for set packing problems

Ordinary LP relaxation				Level 1 via RLT			Level 1 via SSRLT		
(q, n)	Density	% gap	Iter	(m', n')	% gap	Iter	(m', n')	% gap	Iter
(15, 25)	60	34.7	23	(327, 25)	8.98	119	(277, 25)	0	17
(35, 65)	20	29.8	68	(3,706, 545)	6.17	1,012	(2,466, 545)	0	1,491
(35, 60)	25	25.7	90	(3,178, 167)	6.43	1,727	(2,202, 167)	0	1,968
(35, 65)	30	34.7	64	(3,294, 98)	16.7	916	(2,260, 98)	0	1,464
(55, 45)	27	37.0	103	(2,256, 45)	2.0	344	(1,896, 45)	0	171

To illustrate the computational benefits of SSRLT over RLT in this particular context, Sherali et al. [90] conducted the following experiment using pseudo-randomly generated set packing problems of the type: Maximize $\{\sum_{j=1}^n c_j x_j : \sum_{j \in N_i} x_j \leq 1, \forall i = 1, \dots, q, x \text{ binary}\}$. For several instances of such problems, the optimal value of the 0-1 packing problem, the optimal value of its ordinary LP relaxation, and the optimal values of the first-level relaxations produced by applying RLT and SSRLT were computed, where SSRLT was generated by using all of the defining clique constraints, together with $x \geq 0$, to represent the set S . **Table 1** gives the percentage gaps obtained for the latter three upper bounds with respect to the optimal 0-1 value, along with the sizes of the respective relaxations ($m' \equiv$ number of constraints, $n' \equiv$ number of variables), and the number of simplex iterations (Iter) needed by the OSL solver version 2.001, to achieve optimality. Note that for all instances, the first-level application of SSRLT was sufficient to solve the underlying integer program. On the other hand, although RLT appreciably improved the upper bound produced by the ordinary LP relaxation, it still left a significant gap that remains to be resolved in these problem instances. Moreover, the relatively simpler structure of SSRLT results in far fewer simplex iterations being required to solve this relaxation as compared with the effort required to solve RLT.

7.2 Variable Upper Bounding Constraints

This illustration points out that in the presence of variable upper bounding (VUB) types of restrictions, a further tightening of relaxations via SSRLT, beyond that of RLT, can be similarly produced.

Example 8 Consider a set S that is composed as follows in a particular problem instance:

$$S = \{x \in R^6 : 0 \leq x_1 \leq x_2 \leq x_3 \leq 1, 0 \leq x_4 \leq x_5 \leq 1, 0 \leq x_6 \leq 1\}.$$

The first-level factors for this instance are given by $x_1 \geq 0, x_2 - x_1 \geq 0, x_3 - x_2 \geq 0, 1 - x_3 \geq 0, x_4 \geq 0, x_5 - x_4 \geq 0, 1 - x_5 \geq 0, x_6 \geq 0$, and $1 - x_6 \geq 0$. Compared with the RLT factors, these yield tighter constraints as they imply the RLT factors.

For $d \in \{1, \dots, 6\}$, taking these factors d at a time, including self-products, and simplifying these factors by eliminating null or implied factors would produce the relaxation \bar{X}_d .

It is interesting to note in this connection that the VUB constraints of the type $0 \leq x_1 \leq x_2 \leq \dots \leq x_k \leq 1$ used in this example can be equivalently transformed into GUB constraints via the substitution $z_j = x_j - x_{j-1}$ for $j = 1, \dots, k$, where $x_0 \equiv 0$. The inverse transformation yields $x_j = \sum_{t=1}^j z_t$ for $j = 1, \dots, k$, thereby producing the equivalent representation $z_1 + z_2 + \dots + z_k \leq 1$, $z \geq 0$ for each such constraint. Under this transformation and imposing binary restrictions on all the z -variables, the reformulation strategies described above in Sect. 7.1 can be employed. However, note that the process of applying RLT to the original or to the transformed problem can produce different representations. This is illustrated next.

Example 9 Consider the set

$$X = \{(x_1, x_2) : -6x_1 + 3x_2 \leq 1, 0 \leq x_1 \leq x_2 \leq 1, x \text{ binary}\}.$$

The convex hull of feasible solutions is given by $0 \leq x_1 = x_2 \leq 1$ (see Fig. 4a). This representation is produced by the level-1 SSRLT relaxation using the VUB constraints to define S , where the relevant constraint $x_1 \geq x_2$, which yields $x_1 = x_2$, is obtained by noting that the factor products $\{x_1(x_2-x_1)\}_L \geq 0$ and $\{x_1(1-x_2)\}_L \geq 0$, respectively, give $w_{12} \geq x_1$ and $w_{12} \leq x_1$ or that $w_{12} = x_1$. This together with the constraint $\{(x_2-x_1)(1+6x_1-3x_2)\}_L \geq 0$ yields $-2(x_2-x_1) \geq 0$ or that $x_1 \geq x_2$.

On the other hand, constructing RLT at level 1 by applying the factors x_j and $(1-x_j)$, $j = 1, 2$, to the inequality restrictions of X produces the following relaxation (directly in projected form):

$$X_{P1} = \{(x_1, x_2) : 2x_1 - 3x_2 \geq -1, 3x_1 \geq x_2, 0 \leq x_1 \leq x_2 \leq 1\},$$

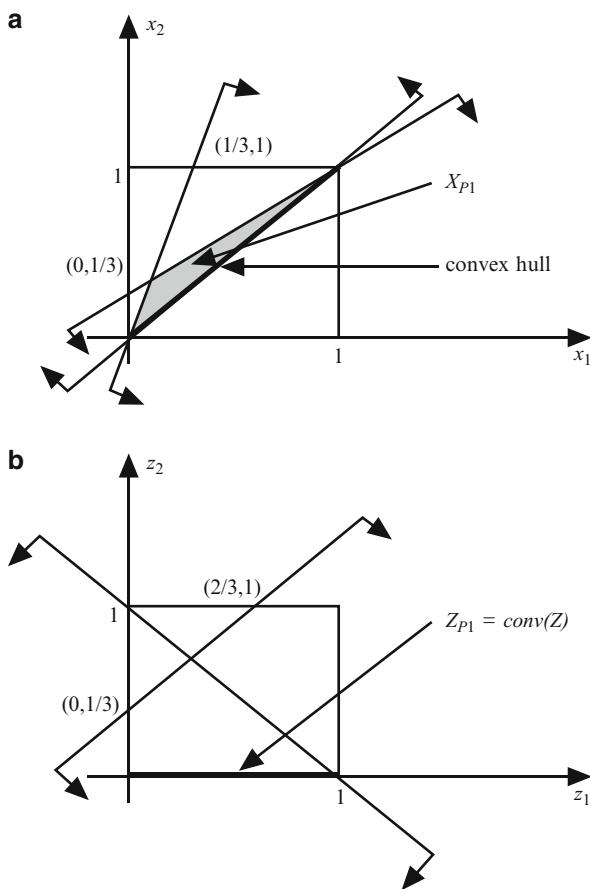
where $w_{12} = x_1$ is produced as with SSRLT and where the first two constraints defining X_{P1} result from the product constraints $\{(1+6x_1-3x_2)(1-x_1)\}_L \geq 0$ and $\{(1+6x_1-3x_2)x_2\}_L \geq 0$, respectively. The shaded area of Fig. 4a depicts the region defined by this relaxation.

However, applying the transformation $z_1 = x_1$, $z_2 = x_2 - x_1$ to X , where the inverse transformation is given by $x_1 = z_1$ and $x_2 = z_1 + z_2$, results in the following problem representation in z -space:

$$Z = \{(z_1, z_2) : -3z_1 + 3z_2 \leq 1, z_1 + z_2 \leq 1, z \text{ binary}\},$$

where the *binary restriction on z_2 has been additionally recognized*. Figure 4b illustrates that the set $\text{conv}(Z)$ is given by the constraints $0 \leq z_1 \leq 1$, $z_2 = 0$. Now, applying RLT to this transformed region, the relevant constraint $z_2 = 0$ is produced via $z_2 \geq 0$ and the first-level product constraint $\{(1+3z_1-3z_2)z_2\}_L \geq 0$, which

Fig. 4 Depiction of the first-level relaxations using RLT for Example 9



yields $-2z_2 \geq 0$, where $\{z_1 z_2\}_L = 0$, from $\{z_1 z_2\}_L \geq 0$ and $\{z_1(1 - z_1 - z_2)\}_L \geq 0$. Hence, for this transformed problem, RLT produces the first-level relaxation $Z_{P1} = \text{conv}(Z)$, while we had $X_{P1} \supset \text{conv}(X)$ when applying RLT to the original problem. However, as with Example 6 for the case $q = 1$ (treating that as the transformed z -variable problem), it is possible to obtain $Z_{P1} \supset \text{conv}(Z)$ as well, whereas SSRLT would necessarily produce the convex hull representation at level 1 in either case.

7.3 Sparse Constraints

This example illustrates how problem sparsity can be exploited. Consider a 0-1 mixed-integer programming problem that contains the following knapsack constraint (either inherent in the problem or implied by it): $2x_1 + x_2 + 2x_3 \geq 3$.

The facets of the convex hull of the set $\{(x_1, x_2, x_3) : 2x_1 + x_2 + 2x_3 \geq 3, x_i \text{ binary}, i = 1, 2, 3\}$ can be readily obtained as $\{x_1 + x_2 + x_3 \geq 2, x_1 \leq 1, x_2 \leq 1, x_3 \leq 1\}$. Similarly, another knapsack constraint in the problem might be of the type $x_4 + 2x_5 + 2x_6 \leq 2$, where the corresponding facets of the convex hull of feasible 0-1 solutions are given by $\{x_4 + x_5 + x_6 \leq 1, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0\}$. The set S can now be composed of these two sets of facets, along with other similar constraints involving the remaining variables on which binary restrictions are being enforced, including perhaps simple bound-factor constraints. Note that in order to generate valid tighter relaxations, binarity might simply be enforced only on a subset of variables that fractionate in the original linear programming relaxation in the present framework. Furthermore, entire convex hull representations of underlying knapsack polytopes are not necessary – simply, the condition (36) needs to be satisfied, perhaps by explicitly including simple bounding constraints. This extends the strategy of Crowder et al. [24] in using facets obtained as liftings of minimal covers from knapsack constraints within this framework, in order to generate tighter relaxations.

8 Using Conditional Logic to Strengthen RLT Constraints with Application to the Traveling Salesman Problem

In all of the foregoing discussion, depending on the structure of the problem, there is another idea that can be exploited to even further tighten the RLT constraints that are generated. This trick deals with the use of *conditional logic* in the generation of RLT-based constraints for tightening a relaxation at any level, which would otherwise have been possible only at some higher level in the hierarchy. In effect, this captures within the RLT process the concepts of branching and logical preprocessing, which are features that are critical to the efficient solution of discrete optimization problems.

To introduce the basic concept involved, for simplicity, consider the following first-level SSRLT constraint that is generated by multiplying a factor $(\alpha x - \beta) \geq 0$ with a constraint $(\gamma x - \delta) \geq 0$, where x is supposed to be binary-valued and where the data is all integer (similar extensions can be developed for mixed-integer constraints, as well as for higher-order SSRLT constraints in which some factor is being applied to some other valid constraint- or bound-factor product of order greater than one):

$$\{(\alpha x - \beta)(\gamma x - \delta)\}_L \geq 0. \quad (49)$$

Observe that if $\alpha x = \beta$, then $(\alpha x - \beta)(\gamma x - \delta) \geq 0$ is valid regardless of the validity of $\gamma x \geq \delta$. Otherwise, it must be that $\alpha x \geq \beta + 1$ (or possibly greater than $\beta + 1$, if the structure of $\alpha x \geq \beta$ so permits), and so, standard logical preprocessing tests (zero-one fixing, coefficient reduction, etc. – see, e.g., Nemhauser and Wolsey [50]) can be performed on the set of constraints $\alpha x \geq \beta + 1, \gamma x \geq \delta, x \text{ binary}$, along with possibly other problem-defining constraints, to tighten $\gamma x \geq \delta$ to the form $\gamma' x \geq \delta'$. For example, if $\alpha x \geq \beta$ is of the type $(1 - x_j) \geq 0$, for some $j \in \{1, \dots, n\}$,

then the restriction $\alpha x \geq \beta + 1$, x binary, asserts that $x_j = 0$, and so, $\gamma x \geq \delta$ can be tightened under the condition that $x_j = 0$. (Similarly, in a higher-order constraint, if a factor $F_d(J_1, J_2)$ multiplies $\gamma x \geq \delta$, then the latter constraint can be tightened under conditional logical tests based on setting $x_j = 1$, $\forall j \in J_1$, and $x_j = 0$, $\forall j \in J_2$.)

Additionally, the resulting constraint $\gamma'x \geq \delta'$ can potentially be further tightened by finding the maximum $\theta \geq 0$ for which $\gamma'x \geq \delta' + \theta$ is valid when $\alpha x \geq \beta + 1$ is imposed by considering the problem

$$\theta = -\delta' + \min \{ \gamma'x : \gamma'x \geq \delta', \alpha x \geq \beta + 1, \text{any other valid inequalities, } x \text{ binary} \}, \quad (50)$$

and by increasing δ' by this quantity θ . Note that a valid value of θ can be derived by simply solving the continuous relaxation of (50) and rounding the resulting value upward (using $\theta = 0$ if this value is negative). In either case, the following SSRLT constraint can be imposed in lieu of the relatively weaker restriction (49) within the underlying problem:

$$\{(\alpha x - \beta)(\gamma'x - \delta' - \theta)\}_L \geq 0. \quad (51)$$

Interestingly, the sequential lifting process studied by Balas and Zemel [18] for lifting a minimal cover inequality into a facet for a full dimensional knapsack polytope can be viewed as a consequence of this RLT construct (see Sherali et al. [90]).

Example 10 To illustrate, consider the following knapsack constraint in binary variables x_1 , x_2 , and x_3 : $2x_1 + 3x_2 + 3x_3 \geq 4$. Examine the RLT constraint

$$\{x_3(2x_1 + 3x_2 + 3x_3 - 4)\}_L \geq 0. \quad (52)$$

Applying the foregoing idea, (52) can be tightened under the restriction that $x_3 = 1$, knowing that it is always valid when $x_3 = 0$ regardless of the nonnegativity of any expression contained in (\cdot) . However, when $x_3 = 1$, the given knapsack constraint becomes $2x_1 + 3x_2 \geq 1$, which by coefficient reduction, can be tightened to $x_1 + x_2 \geq 1$. Hence, (52) can be replaced by the tighter restriction

$$\{x_3(x_1 + x_2 - 1)\}_L \geq 0. \quad (53)$$

Similarly, consider the RLT constraint

$$\{(1 - x_3)(2x_1 + 3x_2 + 3x_3 - 4)\}_L \geq 0. \quad (54)$$

This time, imposing $(1 - x_3) \geq 1$, that is, $x_3 = 0$, the knapsack constraint becomes $2x_1 + 3x_2 \geq 4$, which implies via standard logical tests that $x_1 = x_2 = 1$. Hence, the following *equality* restrictions can be imposed in lieu of (54), which is now implied:

$$\{(1 - x_3)(x_1 - 1)\}_L = 0 \text{ and } \{(1 - x_3)(x_2 - 1)\}_L = 0. \quad (55)$$

Observe that in this example, the sum of the RLT constraints in (53) and (55) yields $x_1 + x_2 + x_3 \geq 2$, which happens to define a facet of the knapsack polytope $\text{conv}\{x : 2x_1 + 3x_2 + 3x_3 \geq 4, x \text{ binary}\}$. This facet can alternatively be obtained by lifting the minimal cover inequality $x_1 + x_2 \geq 1$ as in Balas and Zemel [18].

Example 11 To illustrate the use of the foregoing conditional logic-based tightening procedure in the context of solving an optimization problem using general S -factors, consider the following problem:

$$\text{Minimize } \{x_1 + x_2 : 6x_1 + 3x_2 \geq 2, x \in S, x \text{ binary}\}, \quad (56)$$

$$\text{where } S \equiv \{(x_1, x_2) : 2x_1 + x_2 \leq 2, x_1 + 2x_2 \leq 2, x \geq 0\}.$$

Figure 5 depicts this problem graphically. The integer problem has the optimal value $v(\text{IP}) = 1$, attained at the solution $(1, 0)$ or $(0, 1)$. The ordinary LP relaxation, with extreme points $(0, 2/3)$, $(0, 1)$, $(2/3, 2/3)$, $(1, 0)$, and $(1/3, 0)$, has the optimal value $v(\text{LP}) = 1/3$, attained at the solution $(1/3, 0)$.

Next, consider the first-level relaxation (RLT-1) produced by RLT, using the factors x_j and $(1 - x_j)$, $j = 1, 2$, to multiply all the problem constraints. Note that $\{(2 - 2x_1 - x_2)x_1\}_L$ yields $w_{12} \leq 0$, which, together with $\{x_1x_2\}_L = w_{12} \geq 0$, gives $w_{12} = 0$. The other non-redundant constraints defining this relaxation are produced via the product constraints $\{(1 - x_1)(1 - x_2)\}_L \geq 0$, $\{(6x_1 + 3x_2 - 2)(1 - x_1)\}_L \geq 0$, and $\{(6x_1 + 3x_2 - 2)(1 - x_2)\}_L \geq 0$, which respectively yield (using $w_{12} = 0$) $x_1 + x_2 \leq 1$, $2x_1 + 3x_2 \geq 2$, and $3x_1 + x_2 \geq 1$. This gives (directly in projected form)

$$X_{P1} = \{(x_1, x_2) : x_1 + x_2 \leq 1, 2x_1 + 3x_2 \geq 2, 3x_1 + x_2 \geq 1\}.$$

Figure 5 depicts this region, with extreme points $(1, 0)$, $(0, 1)$, and $B = (1/7, 4/7)$. The optimal value using this relaxation is given by $v(\text{RLT-1}) = 5/7$, attained at the solution $(1/7, 4/7)$.

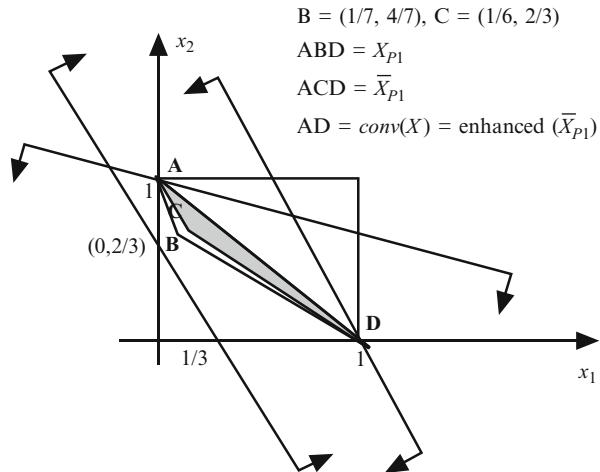
On the other hand, the first-level relaxation (SSRLT-1) produced by SSRLT would employ the S -factors given by the constraints in (56). As in RLT, this yields $w_{12} = 0$. Using this identity, the other nonredundant constraints defining the relaxation SSRLT-1 are produced via the following product constraints:

$$\{(2 - 2x_1 - x_2)(2 - x_1 - 2x_2)\}_L \geq 0,$$

$$\{(6x_1 + 3x_2 - 2)(2 - 2x_1 - x_2)\}_L \geq 0, \text{ and}$$

$$\{(6x_1 + 3x_2 - 2)(2 - x_1 - 2x_2)\}_L \geq 0,$$

Fig. 5 Depiction of the various first-level RLT relaxations



which respectively yield $x_1 + x_2 \leq 1$, $4x_1 + 5x_2 \geq 4$, and $2x_1 + x_2 \geq 1$. This gives (directly in projected form)

$$\bar{X}_{P1} = \{(x_1, x_2) : x_1 + x_2 \leq 1, 4x_1 + 5x_2 \geq 4, 2x_1 + x_2 \geq 1\}.$$

Figure 5 depicts this region, with extreme points $(1, 0)$, $(0, 1)$, and $C = (1/6, 2/3)$. Note that $\bar{X}_{P1} \subset X_{P1}$ and that the optimal value of this relaxation is given by $v(\text{SSRLT-1}) = 5/6 > v(\text{RLT-1})$ and is attained at the solution $(1/6, 2/3)$.

Now, consider an enhancement of SSRLT-1 using conditional logic (a similar enhancement can be exhibited for RLT-1). Specifically, consider the RLT product constraint $\{(2 - 2x_1 - x_2)(6x_1 + 3x_2 - 2)\}_L \geq 0$ of the form (49). Imposing $(2 - 2x_1 - x_2) \geq 1$ as for (49), that is, $2x_1 + x_2 \leq 1$, yields $x_1 = 0$ by a standard logical test. This, together with $6x_1 + 3x_2 \geq 2$, implies that $x_2 = 1$. Hence, the tightened form of the foregoing RLT product constraint is $(2 - 2x_1 - x_2)(x_1) = 0$ and $(2 - 2x_1 - x_2)(1 - x_2) = 0$. The first constraint asserts that $w_{12} = 0$ (as before), while the second constraint asserts that $x_1 + x_2 = 1$. This produces the convex hull representation, and so, this enhanced relaxation now recovers an optimal integer solution.

The paper by Lougee-Heimer and Adams [45] further explores the idea of conditional logic. It shows that the basic constructs provide a framework that can be used to motivate a variety of cut strengthening techniques, including single- and multiple-coefficient adjustments, which are found throughout the literature. Special structures arising within plant location, capacity expansion, resource allocation, variable upper bounds, cover inequalities, and SOS inequalities are examined. Relationships to the

foundation-penalty cuts of Glover and Sherali [33] and extensions to compound conditional logic that involve higher-order RLT products are also explored.

8.1 Application to the Traveling Salesman Problem

Sherali and Driscoll [74] have demonstrated the potential utility of the various concepts developed in Sects. 6–8 by applying them to the celebrated traveling salesman problem (TSP). To elucidate, consider the case of a general (asymmetric) TSP defined on a totally dense graph (see, e.g., Lawler et al. [42]). For this problem, Desrochers and Laporte [27] have derived a strengthened version of the Miller–Tucker–Zemlin (MTZ) formulation obtained by lifting the MTZ-subtour elimination constraints into facets of the underlying TSP polytope. While it is well known that the traditional MTZ formulation of TSP yields weak relaxations, Desrochers and Laporte exhibited computationally that their lifted-MTZ formulation significantly tightens this representation. The discussion below shows that an application of SSRLT concepts to the MTZ formulation of TSP, used in concert with the conditional logic-based strengthening procedure, *automatically recovers* the formulation of Desrochers and Laporte as well as affords a further tightening for this problem.

Toward this end, consider the following statement of the asymmetric traveling salesman problem, where $x_{ij} = 1$ if the tour proceeds from city i directly to city j and is 0 otherwise, for all $i, j = 1, \dots, n, i \neq j$:

$$\text{ATSP: Minimize } \sum_{i=1}^n \sum_{j \neq i} c_{ij} x_{ij}$$

$$\text{subject to } \sum_{j \neq i} x_{ij} = 1, \quad \forall i = 1, \dots, n, \quad (57)$$

$$\sum_{i \neq j} x_{ij} = 1, \quad \forall j = 1, \dots, n, \quad (58)$$

$$u_j \geq (u_i + 1) - (n - 1)(1 - x_{ij}), \quad \forall i, j \geq 2, i \neq j, \quad (59)$$

$$1 \leq u_j \leq (n - 1), \quad \forall j = 2, \dots, n, \quad (60)$$

$$x_{ij} \text{ binary}, \quad \forall i, j = 1, \dots, n, i \neq j. \quad (61)$$

Note that (57), (58), and (61) represent the assignment constraints and (59) and (60) are the MTZ-subtour elimination constraints. These latter constraints are derived based on letting u_j represent the rank order in which city j is visited, using $u_1 \equiv 0$, and enforcing that $u_j = u_i + 1$ whenever $x_{ij} = 1$ in any binary feasible solution. Now, in order to construct a suitable reformulation using SSRLT, compose the set S as follows and include this set of implied inequalities within the problem ATSP stated above:

$$S \equiv \{x : x_{ij} + x_{ji} \leq 1, \forall 2 \leq i < j \leq n; x_{1j} + x_{j1} \leq 1, \forall j \geq 2; \\ x_{ij} \geq 0, \forall i, j, i \neq j\}. \quad (62)$$

Note that S is comprised of simple two-city subtour elimination constraints of the form proposed by Dantzig, Fulkerson, and Johnson [26]. Next, construct the following selected S -factor constraint products. First, consider the product constraint generated by multiplying (59) with the S -factor x_{ij} . Using conditional logic as with (49) and noting that $u_j = (u_i + 1)$ when $x_{ij} = 1$, this yields the constraint

$$x_{ij} u_j = x_{ij}(u_i + 1), \forall i, j \geq 2, i \neq j. \quad (63)$$

Similarly, considering $\{x_{1j}(u_j - 1)\}_L \geq 0$ from (60) and enhancing this by the conditional logic that $u_j = 1$ when $x_{1j} = 1$ yields

$$x_{1j} u_j = x_{1j}, \forall j = 2, \dots, n. \quad (64)$$

Repeating this with the upper bounding constraint in (60), the restriction $\{x_{j1}(n - 1 - u_j)\}_L \geq 0$ can be enhanced to yield the following constraint, noting that $u_j = (n - 1)$ if $x_{j1} = 1$:

$$x_{j1} u_j = (n - 1)x_{j1}, \forall j = 2, \dots, n. \quad (65)$$

Next, consider the product of (59) with the S -factor $(1 - x_{ij} - x_{ji}) \geq 0$. This gives the constraint $\{(1 - x_{ij} - x_{ji})(u_j - u_i - 1 + (n - 1)(1 - x_{ij}))\}_L \geq 0$. Using $x_{ij}^2 = x_{ij}$ and $x_{ij} x_{ji} = 0$ (since $x_{ij} + x_{ji} \leq 1$), the foregoing constraint becomes

$$(u_j - u_i) \geq (u_j - u_i)(x_{ij} + x_{ji}) - (n - 2)(1 - x_{ij}) + (n - 2)x_{ji}.$$

Note that $(u_j - u_i)x_{ij} = x_{ij}$ from (63). Interchanging i and j , this yields $(u_j - u_i)x_{ji} = -x_{ji}$. Substituting this into the foregoing SSRLT constraint produces the valid inequality

$$u_j \geq (u_i + 1) - (n - 1)(1 - x_{ij}) + (n - 3)x_{ji}, \forall i, j \geq 2, i \neq j. \quad (66)$$

Similarly, multiplying (60) with the S -factor $(1 - x_{1j} - x_{j1}) \geq 0$ yields $\{(1 - x_{1j} - x_{j1})(u_j - 1)\}_L \geq 0$ and $\{(1 - x_{1j} - x_{j1})(n - 1 - u_j)\}_L \geq 0$. Using the conditional logic procedure, under $x_{1j} = x_{j1} = 0$, these constraints can be respectively tightened to $\{(1 - x_{1j} - x_{j1})(u_j - 2)\}_L \geq 0$ and $\{(1 - x_{1j} - x_{j1})(n - 2 - u_j)\}_L \geq 0$. Simplifying these products and using (64) and (65) yields the constraints

$$1 + (1 - x_{1j}) + (n - 3)x_{ji} \leq u_j \leq (n - 1) - (1 - x_{j1}) - (n - 3)x_{1j}. \quad (67)$$

Observe that (66) and (67) are tightened versions of (59) and (60), respectively, and are precisely the facet-defining, lifted-MTZ constraints derived by Desrochers and Laporte [27]. Hence, the above discussion reveals that a selected application

of SSRLT used in concert with the conditional logic-based strengthening procedure *automatically* generates this improved MTZ formulation. Sherali and Driscoll [74] have developed further enhancements that tighten and subsume this formulation for both the ordinary as well as for the precedence constrained version of the asymmetric traveling salesman problem, using these SSRLT constructs along with conditional logic implications. Using a different flow-based model, Sherali et al. [94] apply RLT concepts with conditional logic to derive a significantly tighter, yet polynomial length, lifted formulation for the ATSP.

9 Semidefinite Cuts and Extensions to Continuous Global Optimization

Motivated by the semidefinite relaxations of Lovasz and Schrijver [46] and Sherali and Fraticelli [75] derive a class of so-called *semidefinite cuts* that can be used to further tighten RLT relaxations while maintaining linearity, thus facilitating the use of robust, commercial linear programming software as routines for solving such relaxations within the framework of branch-and-cut algorithms. To illustrate,

consider the first-level relaxation for 0-1 MIPs, and let $x_{(1)} \equiv \begin{bmatrix} 1 \\ x \end{bmatrix}$, represent an $(n + 1)$ -dimensional extension of the binary vector x , and consider the matrix $M_1 \equiv \{x_{(1)}x_{(1)}^T\}_L$, whereas before, $\{\cdot\}_L$ denotes the linearization of the expression $\{\cdot\}$ under the RLT variable substitution process, including the application of the identity $x_j^2 = x_j$, $\forall j \in N$. Noting that $x_{(1)}x_{(1)}^T$ is symmetric and positive semidefinite (denoted $\succeq 0$), we could require that $M_1 \succeq 0$, that is,

$$\alpha^T M_1 \alpha = \{(\alpha^T x_{(1)})^2\}_L \geq 0, \quad \forall \alpha \in \mathbb{R}^{n+1}, \|\alpha\| = 1.$$

As such, letting \bar{M}_1 represent the evaluation of M_1 at a given solution to an underlying RLT relaxation in the lifted space of the original and new RLT variables, Sherali and Fraticelli [75] present a polynomial-time procedure, having a worst-case complexity $O(n^3)$, which either verifies that $\bar{M}_1 \succeq 0$ or else generates a suitable $\bar{\alpha} \in \mathbb{R}^{n+1}$ for which $\bar{\alpha}^T \bar{M}_1 \bar{\alpha} < 0$, thus yielding the semidefinite cut $\bar{\alpha}^T M_1 \bar{\alpha} = \{(\bar{\alpha}^T x_{(1)})^2\}_L \geq 0$. Furthermore, motivated in part by the moment matrices utilized by Lasserre [40, 41] in the context of continuous polynomial programs, Sherali and Desai [73] and Sherali et al. [95] extend this concept by replacing $x_{(1)}$ with a vector v that is comprised of the components of $x_{(1)}$ plus other suitable higher-order monomials and then invoking $\{vv^T\}_L \succeq 0$ to generate so-called v -semidefinite cuts. Different techniques are described by Sherali et al. [95] for generating various classes of such v -semidefinite cuts, along with encouraging computational results, actually, for continuous polynomial programming problems as discussed next.

The RLT methodology addressed herein provides a unifying framework for solving not only mixed-discrete polynomial programming problems but also continuous,

nonconvex polynomial and factorable optimization problems to global optimality (see Sherali and Tuncbilek [82] and Sherali and Wang [86]). Here, the *bound factors* ($x_j - \ell_j$) and ($u_j - x_j$) are defined with respect to the general bounding restrictions $\ell_j \leq x_j \leq u_j$, for all the variables x_j defining the given nonconvex program, and are used in concert with the aforementioned constraint factors to generate tight relaxations, which are then embedded within a branch-and-bound algorithm. These relaxations can be further augmented by semidefinite cuts (see Sherali et al. [95]) as well as with other valid inequalities based on grid factors, Lagrange interpolating polynomials, and Chebyshev polynomials (see Sherali and Tuncbilek [83, 84] and Sherali and Dalkiran [72]). By utilizing a suitable partitioning process, Sherali and Tuncbilek [82] and Sherali and Wang [86] show that such an enumeration process can be induced to converge to a global optimal solution. Sherali and Adams [68] provide additional reading on applications of RLT to several special problems (including bilinear, indefinite quadratic location – allocation, and linear complementarity problems) in the domain of both discrete and continuous nonconvex optimization.

10 Conclusion

The hierarchy of relaxations emerging from the RLT can be intuitively viewed as *stepping stones* between continuous and discrete sets, leading from the usual linear programming relaxation to the convex hull representation at level n . By inductively progressing along these stepping-stone formulations, Adams et al. [8] have studied *persistency* issues for certain constrained and unconstrained pseudo-Boolean programming problems. Given the tight linear programming relaxations afforded by RLT, a pertinent question that can be raised is that if the solution obtained for a particular d th level LP relaxation in the RLT hierarchy yields some of the n 0-1 variables as binary-valued, then can these binary values be expected to persist at optimality to the original problem? Adams et al. [8] derive sufficient conditions in terms of the dual solution that guarantee such a persistency result. For the unconstrained pseudo-Boolean program, it is proven that for $d = 1$ or for $d \geq n - 2$, persistency always holds. However, using an example with $d = 2$ and $n = 5$, it is shown that without the additional prescribed sufficient conditions, persistency will not hold in general. These results are also extended to constrained polynomial 0-1 programming problems. In particular, the analysis here reveals a class of 0-1 linear programs that possess the foregoing persistency property. Included within this class as a special case is the popular vertex packing problem, shown earlier in the literature to possess this property.

Thus far, the discussion has focused on binary discrete variables. Conceptually, for a more general *discrete integer program*, a binary transformation could be applied to rewrite the problem as a 0-1 mixed-integer program, to which the foregoing RLT constructs could be applied. Sherali and Adams [68] show that such a specialization can be equivalently translated into a novel direct approach applied to the original problem itself, which leads to more insightful and compact

representations. To see the basic form of this approach, consider the following feasible region of a discrete mixed-integer program:

$$X = \{(x, y) \in R^n \times R^m : Ax + Dy \geq b\}, \quad (68)$$

$$x_j \in S_j \equiv \{\theta_{jk}, k = 1, \dots, k_j\}, \forall j = 1, \dots, n, \quad (69)$$

$$y \geq 0\}, \quad (70)$$

where $\theta_{jk}, k = 1, \dots, k_j, j = 1, \dots, n$ are discrete real numbers (of either sign). As a generalization of the bound factors x_j and $(1 - x_j)$ that are used when x_j is binary, examine the *Lagrange interpolation polynomials* (LIPs) (see Volkov [102]):

$$L_{jk} = \frac{\prod_{p \neq k} (x_j - \theta_{jp})}{\prod_{p \neq k} (\theta_{jk} - \theta_{jp})}, \forall k = 1, \dots, k_j, j = 1, \dots, n. \quad (71)$$

Note that $L_{jk} = 1$ if $x_j = \theta_{jk}$ and $L_{jk} = 0$ if $x_j = \theta_{jp}$ for any $p = \{1, \dots, k_j\} \setminus \{k\}$, $\forall j = 1, \dots, n$. Hence, for each $j = 1, \dots, n$, the LIPs $L_{jk}, k = 1, \dots, k_j$, are akin to 0-1 weights attached to the respective discrete values $\theta_{jk}, k = 1, \dots, k_j$, where it follows from (71) upon algebraic simplifications that

$$\sum_{k=1}^{k_j} L_{jk} = 1, \quad \forall j = 1, \dots, n. \quad (72)$$

Over the discrete restrictions $x_j \in S_j, \forall j = 1, \dots, n$, it also holds that

$$\begin{aligned} & \{L_{jk_1} L_{jk_2} = 0, \forall k_1 \neq k_2\}, \{L_{jk}^2 = L_{jk}, \forall k\}, \text{ and} \\ & \{x_j L_{jk} = \theta_{jk} L_{jk}, \forall k\}, \forall j = 1, \dots, n. \end{aligned} \quad (73)$$

The RLT process in this case employs *LIP factors of order d* composed by selecting some d distinct indices j and, for each selected index, choosing some LIP $L_{jk}, k \in \{1, \dots, k_j\}$. The procedure then proceeds as follows for constructing the relaxation \tilde{X}_d , for any $d \in \{0, 1, \dots, n\}$:

Reformulation phase:

1. Multiply (68) and (70) by all possible LIP factors of order d .
2. Include nonnegativities on all possible LIP factors of order D , where $D \equiv \min\{d+1, n\}$.
3. Use the following identity in the resulting nonlinear program (see (73)):

$$x_j L_{jk} = \theta_{jk} L_{jk} \text{ (i.e., } (x_j - \theta_{jk})L_{jk} = 0\text{), } \forall k = 1, \dots, k_j, j = 1, \dots, n. \quad (74)$$

Linearization phase: Linearize the resulting problem obtained via Steps 1–3 of the reformulation phase by substituting a particular variable for each distinct product of the x -variables thus generated and similarly for each distinct product of y_t with the x -variables, $\forall t$. (Note that these product terms can include self-products of variables.) This produces the required polyhedral relaxation \tilde{X}_d in higher dimensions.

In regard to the foregoing RLT process, first of all, note that whenever x_j is a binary variable as in the previous analysis, (74) corresponds to the step of letting $x_j^2 = x_j$, that is, setting $x_j(1 - x_j) = 0$. Indeed if $S_j \equiv \{0, 1\}$, then L_{jk} in (71) reduces to $(1 - x_j)$ and x_j for $k = 1, 2$, respectively, and (74) yields the familiar identities $x_j(1 - x_j) = 0$ and $x_j^2 = x_j$ for $k = 1, 2$, respectively, which coincide in this binary case. Second, by (72) and (73), observe that for converting the polyhedral relaxation \tilde{X}_d into an equivalent *representation* (not relaxation) of the discrete set X , it suffices to require the LIPs L_{jk} to take on binary values, $\forall k = 1, \dots, k_j, j = 1, \dots, n$, for example, by equating them to distinct binary variables. Third, similar to (12), by defining \tilde{X}_{P_d} as the projection of \tilde{X}_d onto the space of the original variables (x, y) , it can be shown that this RLT process produces the following hierarchy of relaxations:

$$\tilde{X}_{P_0} \supseteq \tilde{X}_{P_1} \supseteq \dots \supseteq \tilde{X}_{P_n} = \text{conv}(X). \quad (75)$$

Moreover, the relationship between the binary transformed problem and the original discrete problem can be used to translate valid inequalities that are derived in the former space to those represented in the original (x, y) variable space. This construct can be useful in implementing a cutting plane or a branch-and-cut approach for general integer programs.

Adams and Sherali [6] provide an alternative elegant proof of (75) for the foregoing RLT procedure by recognizing that the collection of LIPs can be expressed as Kronecker products of matrices given by the inverses of the transposes of Vandermonde matrices. Noting that the Kronecker product of the inverses of a given set of matrices is the inverse of their Kronecker products directly leads to (75) and also thereby generalizes the application of RLT for 0-1 mixed-integer programs.

Furthermore, Sherali and Adams [69] have shown that (12) or (75) holds true even for general mixed-discrete, linear semi-infinite programs. As such, this enables the extension of RLT to bounded 0-1 mixed-integer and mixed-discrete *convex programs*, where such problems can be equivalently written (implicitly) as semi-infinite programs by using suitable supporting hyperplanes for the region defined by each convex constraint based on the first-order characterization of convexity (see Bazaraa et al. [20]). Sherali and Adams [69] present an inverse transformation mechanism whereby the resulting lifted semi-infinite relaxation at any level can be transformed back into an equivalent finitely constrained convex relaxation defined in terms of the original constraint functions. The hierarchy (75) is thus shown to hold true for the latter set of relaxations. In particular, for 0-1 mixed-integer

convex programs, this analysis recovers at the highest level the convex hull representation derived by Stubbs and Mehrotra [97]. Sherali and Adams [69] also present useful first-level lifted relaxations that can be judiciously applied to solve practical mixed-integer convex programs arising in a wide range of contexts such as robotics, optimal control, finance, and various chemical process and engineering design problems.

As far as the use of RLT as a practical computational aid for solving mixed-integer 0-1 problems is concerned, one may simply work with the relaxation X_1 itself, which has frequently proven to be beneficial or even utilize the reduced level-1 relaxation proposed by Sherali et al. [93], which can, in theory, achieve the same strength as the full level-1 relaxation. Using variations on the first-level RLT relaxation (a partial generation of X_1 enhanced by additional constraint products), Adams and Johnson [2], Adams and Sherali [3–5], Sherali and Adams [63, 66, 67], Sherali and Brown [70], Sherali et al. [89], and Sherali et al. [87] have shown how highly effective algorithms can be constructed for various classes of discrete problems and related applications. A study of special cases of this type has provided insights into useful implementation strategies based on tightening a given model formulation via first-order RLT product constraints. Additionally, techniques for generating tight valid inequalities that are implied by higher-order relaxations may be devised, or explicit convex hull representations or facetial inequalities could be generated by applying the highest order RLT scheme to various subsets of sparse constraints that involve a manageable number of variables. The lattermost strategy would be a generalization of using facets of the knapsack polytope derived from minimal covers (see Balas and Zemel [18] and Zemel [107, 108]), which were implemented by Crowder et al. [24] and Hoffman and Padberg [34] with great success. An alternative would be to apply a higher-order scheme on a subset of binary variables that turn out to be fractional in the initial linear programming solution. Letting x_j , $j \in J_f$, represent such a subset, it follows from Sherali and Adams [65] that by deriving $X_{|J_f|}$ based on this subset, which is manageable if $|J_f|$ is relatively small, the resulting linear program will yield binary values for x_j , $\forall j \in J_f$. This would be akin to employing judicious partial convex hull representations. Sherali and Smith [80] discuss how cuts implied by such (implicitly derived) higher-level RLT relaxations can be expeditiously generated. Moreover, the RLT-based formulations give rise to special structures that can be suitably exploited. Works of Sun [98] and Adams and Hadavas [1] have shown how select subsets of constraints that naturally arise in the level-1 RLT formulation can be transformed, via suitable substitutions of variables, into maximum flow networks.

Finally, consider the task of solving RLT-based relaxations. While the RLT process leads to tight linear programming relaxations for the underlying discrete problem being solved as discussed above, one has to contend with the repeated solutions of such large-scale linear programs. By the nature of the RLT process, these linear programs possess a special structure induced by the replicated products of the original problem constraints (or its subset) with certain designated variables.

At the same time, this process injects a high level of degeneracy in the problem since blocks of constraints automatically become active whenever the factor expression that generated them turns out to be zero at any feasible solution, and the condition number of the bases can become quite large. As a result, simplex-based procedures and even interior point methods experience difficulty in coping with such reformulated linear programs (see Adams and Sherali [5], for some related computational experience). On the other hand, a Lagrangian duality-based scheme can not only exploit the inherent special structures, but can quickly provide near optimal primal and dual solutions that serve the purpose of obtaining tight lower and upper bounds. In this vein, Sherali and Smith [81] present a dynamic Lagrangian dual scheme in which suitable level-1 RLT constraints are dynamically generated within a Lagrangian dual ascent process. However, for a successful use of this technique, there are two critical issues. First, an appropriate formulation of the underlying Lagrangian dual must be constructed (see Fisher [28]). Sherali and Myers [79] also discuss and test various strategies and provide guidelines for composing suitable Lagrangian dual formulations. Second, an appropriate nondifferentiable optimization technique must be employed to solve the Lagrangian dual problem. For the size of problems encountered in the context of RLT, it appears imperative to use conjugate subgradient methods as in Camerini et al. [22], Sherali and Ulular [85], and Sherali et al. [88], which employ higher-order information but in a manner involving minor additional effort and storage over traditional subgradient algorithms. For large-scale Lagrangian dual optimization problems, the proximal bundle algorithm of Kiwiel [39], the variable target value method of Sherali et al. [92] (see also Sherali and Lim [78] and Lim and Sherali [43]), and the trust region target value method of Lim and Sherali [44] might be more appropriate. Since these types of algorithms are not usually dual adequate [30], for algorithms that require primal solutions for partitioning purposes, some extra work becomes necessary. For this purpose, one can either use powerful LP solvers such as CPLEX [23] on suitable surrogate versions of the problem based on the derived dual solution or apply primal solution recovery procedures as in Shor [96] or Sherali and Choi [71] along with primal penalty function techniques as in Sherali and Ulular [85].

It is anticipated that automatic reformulation techniques, such as the RLT scheme discussed in this chapter, will continue to play a crucial role in enhancing problem solving capability. Ongoing advances in solving large-scale linear programming problems will provide a further impetus to such techniques, which typically tend to derive tighter representations in higher dimensions and with additional constraints. Furthermore, relaxations based on semidefinite programming approaches as motivated by the work of Lovasz and Shrijver [46], along with interior point approaches for solving such relaxations (see Overton and Wolkowicz [55], Anstreicher [11], and Terlaky [99]), hold promise for future advancements.

Acknowledgements Thanks are due to the research support of the *National Science Foundation* under Grant Nos. CMMI-969169 and CMMI-0968909.

Recommended Reading

1. W.P. Adams, P.T. Hadavas, A network approach for specially-structured linear programs arising in 0-1 quadratic optimization. *Discret. Appl. Math.* **156**(11), 2142–2165 (2008)
2. W.P. Adams, T.A. Johnson, Improved linear programming-based lower bounds for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, ed. by P.M. Pardalos, H. Wolkowicz. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (American Mathematical Society, Providence, 1994), pp. 43–75
3. W.P. Adams, H.D. Sherali, A tight linearization and an algorithm for zero-one quadratic programming problems. *Manag. Sci.* **32**(10), 1274–1290 (1986)
4. W.P. Adams, H.D. Sherali, Linearization strategies for a class of zero-one mixed integer programming problems. *Oper. Res.* **38**(2), 217–226 (1990)
5. W.P. Adams, H.D. Sherali, Mixed-integer bilinear programming problems. *Math. Program.* **59**(3), 279–305 (1993)
6. W.P. Adams, H.D. Sherali, A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems. *Ann. Oper. Res.* **140**(1), 21–47 (2005)
7. W.P. Adams, A. Billionnet, A. Sutter, Unconstrained 0-1 optimization and lagrangean relaxation. *Discret. Appl. Math.* **29**(2–3), 131–142 (1990)
8. W.P. Adams, J.B. Lassiter, H.D. Sherali, Persistency in 0-1 optimization. *Math. Oper. Res.* **23**(2), 359–389 (1998)
9. W.P. Adams, R.J. Forrester, F.W. Glover, Comparisons and enhancement strategies for linearizing mixed 0-1 quadratic programs. *Discret. Optim.* **1**(2), 99–120 (2004)
10. W.P. Adams, M. Guignard, P.M. Hahn, W.L. Hightower, A level-2 reformulation-linearization technique bound for the quadratic assignment problem. *Eur. J. Oper. Res.* **180**(3), 983–996 (2007)
11. K.M. Anstreicher, Interior point methods in theory and practice. *Math. Program. B* **76**(1–2), 1–263 (1997)
12. K.M. Anstreicher, N. Brixius, J.-P. Goux, J. Linderoth, Solving large quadratic assignment problems on computational grids. *Math. Program.* **91**(3), 563–588 (2002)
13. E. Balas, Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J. Algebr. Discret. Methods* **6**(3), 466–486 (1985)
14. E. Balas, J.B. Mazzola, Nonlinear 0-1 programming: I. Linearization techniques. *Math. Program.* **30**, 2–12 (1984a)
15. E. Balas, J.B. Mazzola, Nonlinear 0-1 programming: II. Dominance relations and algorithms. *Math. Program.* **30**, 22–45 (1984b)
16. E. Balas, M. Perregaard, Lift-and-project for mixed- 0-1 programming: recent progress. *Discret. Appl. Math.* **123**, 129–154 (2002)
17. E. Balas, M.A. Perregaard, Correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Math. Program. B* **94**, 221–245 (2003)
18. E. Balas, E. Zemel, Facets of the knapsack polytope from minimal covers. *SIAM J. Appl. Math.* **34**, 119–148 (1978)
19. E. Balas, S. Ceria, G. Cornuéjols, A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Program.* **58**(3), 295–324 (1993)
20. M.S. Bazaraa, H.D. Sherali, C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd edn. (Wiley, New York, 2006)
21. E. Boros, Y. Crama, P.L. Hammer, Upper bounds for quadratic 0-1 maximization. *Oper. Res. Lett.* **9**(2), 73–79 (1990)
22. P.M. Camerini, L. Fratta, F. Maffioli, On improving relaxation methods by modified gradient techniques, in *Mathematical Programming Study*, vol. 3 (North-Holland Publishing Co., New York, 1975), pp. 26–34
23. CPLEX, *Using the CPLEX Linear Optimizer* (CPLEX Optimization, Inc., Incline Village, 1990)

24. H. Crowder, E.L. Johnson, M.W. Padberg, Solving large-scale zero-one linear programming problems. *Oper. Res.* **31**(5), 803–834 (1983)
25. H. Crowder, M.W. Padberg, Solving large-scale symmetric traveling salesman problems to optimality. *Manag. Sci.* **26**, 495–509 (1980)
26. G. Dantzig, R. Fulkerson, S. Johnson, Solution of a large-scale traveling-salesman problem. *Oper. Res.* **2**(4), 393–410 (1954)
27. M. Desrochers, G. Laporte, Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Oper. Res. Lett.* **10**(1), 27–36 (1991)
28. M.L. Fisher, The Lagrangian relaxation method for solving integer programming problems. *Manag. Sci.* **27**(1), 1–18 (1981)
29. R.S. Garfinkel, G.L. Nemhauser, A survey of integer programming emphasizing computation and relations among models, in *Mathematical Programming: Proceedings of an Advanced Seminar*, ed. by T.C. Hu, S. Robinson (Academic, New York, 1973), pp. 77–155
30. A.M. Geoffrion, Lagrangian relaxation for integer programming, in *Mathematical Programming Study*, vol. 2, ed. by M.L. Balinski (North-Holland Publishing Co., Amsterdam, 1974), pp. 82–114
31. A.M. Geoffrion, R. McBryde, Lagrangian relaxation applied to facility location problems. *AIIE Trans.* **10**, 40–47 (1979)
32. F. Glover, Improved linear integer programming formulations of nonlinear integer problems. *Manag. Sci.* **22**(4), 455–460 (1975)
33. F. Glover, H.D. Sherali, Foundation penalty cuts for mixed-integer programs. *Oper. Res. Lett.* **31**(4), 245–253 (2003)
34. K.L. Hoffman, M. Padberg, Improving LP-representations of zero-one linear programs for branch-and-cut. *ORSA J. Comput.* **3**(2), 121–134 (1991)
35. R.G. Jeroslow, J.K. Lowe, Modeling with integer variables. *Math. Program. Study* **22**, 167–184 (1984)
36. R.G. Jeroslow, J.K. Lowe, Experimental results on new techniques for integer programming formulations. *J. Oper. Res. Soc.* **36**, 393–403 (1985)
37. E.L. Johnson, Modeling and strong linear programs for mixed integer programming, in *Algorithms and Model Formulations in Mathematical Programming*, ed. by S. Wallace. NATO ASI, vol. 51 (Springer, Berlin/New York, 1989), pp. 3–43
38. E.L. Johnson, M.M. Kostreva, U.H. Suhl, Solving 0-1 integer programming problems arising from large scale planning models. *Oper. Res.* **33**(4), 803–819 (1985)
39. K.C. Kiwiel, Proximity control in bundle methods for convex nondifferentiable minimization. *Math. Program.* **46**(1–3), 105–122 (1990)
40. J.B. Lasserre, Global optimization with polynomials and the problem of moments. *SIAM J. Optim.* **11**, 796–817 (2001)
41. J.B. Lasserre, Semidefinite programming vs. LP relaxations for polynomial programming. *Math. Oper. Res.* **27**(2), 347–360 (2002)
42. E. Lawler, J. Lenstra, A. Kan, D. Shmoys, *The Traveling Salesman Problem* (Wiley, New York, 1992)
43. C. Lim, H.D. Sherali, Convergence and computational analyses for some variable target value and subgradient deflection methods. *Comput. Optim. Appl.* **34**(3), 409–428 (2006a)
44. C. Lim, H.D. Sherali, A trust region target value method for optimizing nondifferentiable Lagrangian duals of linear programs. *Math. Methods Oper. Res.* **64**(1), 33–53 (2006b)
45. R. Lougee-Heimer, W.P. Adams, A conditional logic approach for strengthening mixed 0-1 linear programs. *Ann. Oper. Res.* **139**(1), 289–320 (2005)
46. L. Lovasz, A. Schrijver, Cones of matrices and set functions, and 0-1 optimization. *SIAM J. Optim.* **1**, 166–190 (1991)
47. T.L. Magnanti, R.T. Wong, Accelerating benders decomposition: algorithmic enhancement and model selection criteria. *Oper. Res.* **29**, 464–484 (1981)
48. K.R. Martin, Generating alternative mixed-integer programming models using variable redefinition. *Oper. Res.* **35**, 820–831 (1987)

49. R.R.A. Meyer, Theoretical and computational comparison of ‘Equivalent’ mixed-integer formulations. *Nav. Res. Logist. Q.* **28**, 115–131 (1981)
50. G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization* (Wiley, New York, 1988)
51. G.L. Nemhauser, L.A. Wolsey, A recursive procedure for generating all cuts for mixed-integer programs. *Math. Program.* **46**, 379–390 (1990)
52. G.L. Nemhauser, M. Savelsbergh, G. Sigismondi, *MINTO: A Mixed INTEGER Optimizer* (School of Industrial and Systems Engineering/Georgia Institute of Technology, Atlanta, 1991)
53. C.E. Nugent, T.E. Vollman, J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations. *Oper. Res.* **16**(1), 150–173 (1968)
54. L.A. Oley, R.J. Sjouquist, Automatic reformulation of mixed and pure integer models to reduce solution time in apex IV, in *Presented at the ORSA/TIMS Fall Meeting*, San Diego, 1982
55. M. Overton, H. Wolkowicz, Semidefinite programming. *Math. Program.* **77**(2), 105–110 (1997)
56. M.W. Padberg, (l, k) -configurations and facets for packing problems. *Math. Program.* **18**, 94–99 (1980)
57. M. Padberg, G. Rinaldi, A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.* **33**, 60–100 (1991)
58. B. Ramachandran, J.F. Pekny, Dynamic factorization methods for using formulations derived from higher order lifting techniques in the solution of the quadratic assignment problem, in *State of the Art in Global Optimization*, vol. 7, ed. by C.A. Floudas, P.M. Pardalos (Kluwer, Dordrecht/Boston/London, 1996), pp. 75–92
59. K.G. Ramakrishnan, M.G.C. Resende, P.M. Pardalos, A branch and bound algorithm for the quadratic assignment problem using a lower bound based on linear programming, in *State of the Art in Global Optimization*, vol. 7, ed. by C.A. Floudas, P.M. Pardalos (Kluwer, Dordrecht/Boston/London, 1996), pp. 57–74
60. M.G.C. Resende, K.G. Ramakrishnan, Z. Drezner, Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. *Oper. Res.* **5**(18), 781–791 (1995)
61. H.D. Sherali, Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *ALTA Math. Vietnam.* **22**(1), 245–270 (1997)
62. H.D. Sherali, RLT: a unified approach for discrete and continuous nonconvex optimization. *Ann. Oper. Res.* **149**, 184–193 (2007)
63. H.D. Sherali, W.P. Adams, A decomposition algorithm for a discrete location-allocation problem. *Oper. Res.* **32**(4), 878–900 (1984)
64. H.D. Sherali, W.P. Adams, A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM J. Discret. Math.* **3**(3), 411–430 (1990)
65. H.D. Sherali, W.P. Adams, A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discret. Appl. Math.* **52**(1), 83–106 (1994a). (Manuscript, 1989)
66. H.D. Sherali, W.P. Adams, A reformulation-linearization technique (RLT) for solving discrete and continuous nonconvex programming problems. *Math. Today, special issue on Recent Advances in Mathematical Programming*, O. Gupta (ed.), **XII-A**, 61–78 (1994b)
67. H.D. Sherali, W.P. Adams, Computational advances on using the reformulation-linearization technique (RLT) to solve various discrete and continuous nonconvex programming problems. *Optima Math. Program. Soc. Newslet.* **49**, 1–6 (1996)
68. H.D. Sherali, W.P. Adams, *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems* (Kluwer, Dordrecht/Boston/London, 1999)
69. H.D. Sherali, W.P. Adams, A reformulation-linearization technique (RLT) for semi-infinite and convex programs under mixed 0-1 and general discrete restrictions. *Discret. Appl. Math.* **157**(6), 1319–1333 (2009)

70. H.D. Sherali, E.L. Brown, A quadratic partial assignment and packing model and algorithm for the airline gate assignment problem, in *Quadratic Assignment and Related Problems*, ed. by P.M. Pardalos, H. Wolkowicz. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16 (American Mathematical Society, Providence, 1994), pp. 343–364
71. H.D. Sherali, G. Choi, Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. *Oper. Res. Lett.* **19**(3), 105–113 (1996)
72. H.D. Sherali, E. Dalkiran, Combined bound-grid-factor constraints for enhancing RLT relaxations for polynomial programs. Manuscript, Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2010
73. H.D. Sherali, J. Desai, On solving polynomial, factorable, and black-box optimization problems using the RLT methodology, in *Essays and Surveys on Global Optimization*, ed. by C. Audet, P. Hansen, G. Savard (Springer, New York, 2005), pp. 131–163
74. H.D. Sherali, P.J. Driscoll, On tightening the relaxation of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems. *Oper. Res.* **50**(4), 656–669 (2002)
75. H.D. Sherali, B.M.P. Fraticelli, Enhancing RLT relaxations via a new class of semidefinite cuts. *J. Glob. Optim.* **22**(1–4), 233–261 (2002)
76. H.D. Sherali, Y. Lee, Sequential and simultaneous liftings of minimal cover inequalities for generalized upper bound constrained knapsack polytopes. *SIAM J. Discret. Math.* **8**(1), 133–153 (1995)
77. H.D. Sherali, Y. Lee, Tighter representations for set partitioning problems. *Discret. Appl. Math.* **68**, 153–167 (1996)
78. H.D. Sherali, C. Lim, On embedding the volume algorithm in a variable target value method for solving lagrangian relaxations of linear programs. *Oper. Res. Lett.* **32**(5), 455–462 (2004)
79. H.D. Sherali, D.C. Myers, Dual formulations and subgradient optimization strategies for linear programming relaxations of mixed integer programs. *Discret. Appl. Math.* **20**(S-16), 51–68 (1989)
80. H.D. Sherali, J.C. Smith, Higher-level RLT or disjunctive cuts based on a partial enumeration strategy for 0-1 mixed-integer programs. *Optim. Lett.* **6**(1), 127–139 (2012)
81. H.D. Sherali, J.C. Smith, Dynamic Lagrangian dual and reduced RLT constructs for solving 0-1 mixed-integer programs. Manuscript, Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2010
82. H.D. Sherali, C.H. Tuncbilek, A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *J. Glob. Optim.* **2**, 101–112 (1992)
83. H.D. Sherali, C.H. Tuncbilek, A reformulation-convexification approach for solving nonconvex quadratic programming problems. *J. Glob. Optim.* **7**, 1–31 (1995)
84. H.D. Sherali, C.H. Tuncbilek, New reformulation-linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Oper. Res. Lett.* **21**(1), 1–9 (1997)
85. H.D. Sherali, O. Ulular, A primal-dual conjugate subgradient algorithm for specially structured linear and convex programming problems. *Appl. Math. Optim.* **20**, 193–221 (1989)
86. H.D. Sherali, H. Wang, Global optimization of nonconvex factorable programming problems. *Math. Program.* **89**(3), 459–478 (2001)
87. H.D. Sherali, S. Ramachandran, S. Kim, A localization and reformulation discrete programming approach for the rectilinear distance location-allocation problem. *Discret. Appl. Math.* **49**(1–3), 357–378 (1994)
88. H.D. Sherali, Y. Lee, W.P. Adams, A simultaneous lifting strategy for identifying new classes of facets for the Boolean quadric polytope. *Oper. Res. Lett.* **17**(1), 19–26 (1995)
89. H.D. Sherali, R.S. Krishnamurthy, F.A. Al-Khayyal, An enhanced intersection cutting plane approach for linear complementarity problems. *J. Optim. Theory Appl.* **90**(1), 183–201 (1996)
90. H.D. Sherali, W.P. Adams, P. Driscoll, Exploiting special structures in constructing a hierarchy of relaxations for 0-1 mixed integer problems. *Oper. Res.* **46**(3), 396–405 (1998a)
91. H.D. Sherali, R. Krishnamurthy, F.A. Al-Khayyal, Enumeration approach for linear complementarity problems based on a reformulation-linearization technique. *J. Optim. Theory Appl.* **99**(2), 481–507 (1998b)

92. H.D. Sherali, G. Choi, C.H. Tuncbilek, A variable target value method for nondifferentiable optimization. *Oper. Res. Lett.* **26**(1), 1–8 (2000a)
93. H.D. Sherali, J.C. Smith, W.P. Adams, Reduced first-level representations via the reformulation-linearization technique: results, counter-examples, and computations. *Discret. Appl. Math.* **101**(1), 247–267 (2000b)
94. H.D. Sherali, S.C. Sarin, P.F. Tsai, A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Discret. Optim.* **3**, 20–32 (2006)
95. H.D. Sherali, E. Dalkiran, J. Desai, A class of v -semidefinite Cuts. Manuscript, Grado Department of Industrial and Systems Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2009
96. N.Z. Shor, *Minimization Methods for Nondifferentiable Functions* (Springer, Berlin, 1985)
97. R.A. Stubbs, S. Mehrotra, A branch-and-cut method for 0-1 mixed convex programming. *Math. Program.* **86**, 515–532 (1999)
98. X. Sun, Combinatorial algorithms for boolean and pseudo-Boolean Functions, Ph.D. Dissertation, Rutgers University, 1992
99. T. Terlaky, *Interior Point Methods for Mathematical Programming* (Kluwer, Dordrecht/Boston/London, 1998)
100. T.J. Van Roy, L.A. Wolsey, Valid inequalities for mixed 0-1 programs, CORE Discussion Paper No. 8316, Center for Operations Research and Econometrics. Universite Catholique de Louvain, Belgium, 1983
101. T.J. Van Roy, L.A. Wolsey, Solving mixed integer programs by automatic reformulation. *Oper. Res.* **35**, 45–57 (1987)
102. E.A. Volkov, *Numerical Methods* (Hemisphere Publishing, New York, 1990)
103. H.P. Williams, *Model Building in Mathematical Programming*, 2nd edn. (Wiley (Wiley Interscience), New York, 1985)
104. L.A. Wolsey, Facets and strong valid inequalities for integer programs. *Oper. Res.* **24**, 367–373 (1976)
105. L.A. Wolsey, Strong formulations for mixed integer programming: a survey. *Math. Program.* **45**, 173–191 (1989)
106. L.A. Wolsey, Valid inequalities for 0-1 knapsacks and MIPs with generalized upper bound constraints. *Discret. Appl. Math.* **29**, 251–262 (1990)
107. E. Zemel, Lifting the facets of zero-one polytopes. *Math. Program.* **15**, 268–277 (1978)
108. E. Zemel, Easily computable facets of the knapsack problem. *Math. Oper. Res.* **14**, 760–774 (1989)

Resource Allocation Problems

Naoki Katoh, Akiyoshi Shioura and Toshihide Ibaraki

Contents

1	Introduction	2899
2	Preliminaries and Problem Classification	2902
2.1	Preliminaries	2902
2.2	Problem Classification	2903
3	Fundamental Algorithms	2907
3.1	SC/Simple/D	2908
3.2	SC/SM/D	2913
3.3	SC/GUB/D, SC/Nested/D, SC/Tree/D, and SC/Network/D	2925
3.4	Minimax and Maximin Problems	2927
3.5	Notes and References	2928
4	Proximity Theorems	2930
4.1	Proximity Theorem for General Linear Constraints	2931
4.2	Algorithm for Problem IP	2936
4.3	Proximity Theorem for Submodular Constraints	2938
4.4	Notes	2940
5	Lower Bounds on Time Complexity and Improved Algorithms	2941
5.1	Improved Algorithms for SC/SM/D and Its Special Cases	2942
5.2	Lower Bounds	2945
5.3	Strongly Polynomial Algorithms for Separable Quadratic Convex Objective Functions	2948
5.4	Notes and References	2951

N. Katoh (✉)

Department of Architecture and Architectural Engineering, Graduate School of Engineering,
Kyoto University, Kyoto, Japan
e-mail: naoki@archi.kyoto-u.ac.jp

A. Shioura

Department of System Information Sciences, Graduate School of Information Sciences,
Tohoku University, Sendai, Japan
e-mail: shioura@dais.is.tohoku.ac.jp

T. Ibaraki

The Kyoto College of Graduate Studies for Informatics, Kyoto, Japan
e-mail: ibaraki@ieee.org

6	Nonseparable Convex Resource Allocation.....	2951
6.1	<i>M</i> -Convex Functions.....	2952
6.2	A Polynomial Algorithm for Minimizing an <i>M</i> -Convex Function.....	2953
7	Applications.....	2956
7.1	Computer Science Applications.....	2956
7.2	Reliability Applications.....	2962
7.3	Production Planning Applications.....	2963
7.4	Other Applications.....	2965
7.5	Notes and References.....	2969
8	Further Topics.....	2970
8.1	Multiple Resource Allocation.....	2971
8.2	Multiperiod Resource Allocation.....	2975
8.3	Minimizing a Separable Convex Function Under a Nonlinear Constraint.....	2977
8.4	Other Variants of Resource Allocation Problems.....	2977
8.5	Notes and References.....	2980
9	Conclusion.....	2981
	Recommended Reading.....	2982

Abstract

The resource allocation problem seeks to find an optimal allocation of a fixed amount of resources to activities so as to minimize the cost incurred by the allocation. A simplest form of the problem is to minimize a separable convex function under a single constraint concerning the total amount of resource to be allocated. The amount of resource to be allocated to each activity is treated as a continuous or integer variable, depending on the situations. Hence, this problem can be viewed as a special case of the nonlinear programming problem or the nonlinear integer programming problem.

Due to its simple structure, the resource allocation problem is encountered in a variety of application areas, including load distribution, production planning, computer resource allocation, queueing control, portfolio selection, and apportionment. The first explicit investigation of the resource allocation problem was due to a paper by Koopman [89] published in 1953, where he discussed optimal distribution of efforts which arises from the problem of searching for an object whose position is a random variable. Since then, a great number of papers on the resource allocation problem have been published. Efficient algorithms have also been developed, depending on the types of objective functions, constraints, and variables (i.e., continuous or integer).

In 1988, two of the present authors, Ibaraki and Katoh, have published a book [69] that gave a comprehensive review of the state of the art of the resource allocation problem. Since then, more than 20 years have passed, during which, many papers on this problem have been published. A significant progress has been made on the algorithm side. Also, new generalizations and variants of the problem have been investigated, and new application fields have been discovered. The main purpose of this chapter is to give a brief overview of the recent progress on the theory and applications, putting emphasis on the cases with integer variables.

1 Introduction

The resource allocation problem seeks to find an optimal allocation of a fixed amount of resources to activities so as to minimize the cost incurred by the allocation. A simplest form of the problem is to minimize a separable convex function under a single constraint concerning the total amount of resource to be allocated. The amount of resource to be allocated to each activity is treated as a continuous or integer variable, depending on the situations. Hence, this problem can be viewed as a special case of the nonlinear programming problem or the nonlinear integer programming problem.

Due to its simple structure, this problem is encountered in a variety of application areas, including load distribution, production planning, computer resource allocation, queueing control, portfolio selection, and apportionment. The first explicit investigation of the resource allocation problem was due to a paper by Koopman [89] published in 1953, where he discussed optimal distribution of efforts which arises in the problem of searching for an object whose position is a random variable. Since then, a great number of papers have been published on the resource allocation problem. Efficient algorithms have also been developed, depending on the types of objective functions, constraints, and variables (i.e., continuous or integer).

In 1988, two of the present authors, Ibaraki and Katoh, have published a book [69] that gave a comprehensive review of the state of the art of the problems. Since then, more than 20 years have passed, during which, many papers have been published on resource allocation problems. A significant progress has been made on the algorithm side. Also new generalizations and variants of the problem have been investigated, and new application fields have been discovered. The main purpose of this chapter is to give a brief overview of the recent progress on the theory and applications, putting emphasis on the cases with integer variables. Notice that, in addition to the first edition of this chapter published in 1999, two survey papers have been recently published by Hochbaum [63] and by Patricksson [120].

To get an idea of some of topics included in this chapter, a perspective view of these years is briefly given here by using the simplest type of the problem, which is the separable convex resource allocation problem with integer variables described as follows:

$$\begin{aligned} & \text{minimize} \quad \sum_{j=1}^n f_j(x_j) \\ & \text{subject to} \quad \sum_{j=1}^n x_j = N, \\ & \quad x_j \geq 0, \quad \text{integer } j = 1, \dots, n. \end{aligned}$$

Because of its simple constraint, this problem is also referred to as *the simple resource allocation problem*. For this problem, a simple greedy-type algorithm was developed by Gross [54] (the same algorithm has been rediscovered by many

researchers, e.g., Fox [41] and Shih [128]). The running time of this algorithm is $O(N \log n + n)$. However, this time complexity is not polynomial but is pseudo-polynomial in the input size, which is $O(n + \log N)$. Thus, efficient polynomial-time algorithms have been studied by Frederickson and Johnson [42], Galil and Megiddo [48], and Katoh et al. [78]. The fastest among them is due to Frederickson and Johnson [42] and runs in $O(\max\{n, n \log(N/n)\})$ time.

In 1980s, the constraint in the above problem has been generalized, while the polynomial-time solvability is preserved. One such generalization is the problem with submodular constraint, which includes nested, tree, and network constraints as special cases. For this problem, polynomial-time algorithms have been developed in [43, 45, 53, 69]. It was shown that a greedy procedure for the simple allocation problem also works for the one with submodular constraint [39, 69]. The book by Fujishige [45] gives a comprehensive overview on the resource allocation problem under submodular constraint.

Later, Ando, Fujishige, and Naitoh [4, 5] considered the separable convex resource allocation problem with two types of constraints, one given by a bisubmodular system and the other given by a finite jump system; these constraints can be viewed as a generalization of submodular constraint. They developed greedy algorithms for such problems. For the case of a bisubmodular system, a polynomial-time algorithm has been developed by Fujishige [44]. Also, Hochbaum and Shanthikumar [65] showed that, for a class of general linear constraints, efficient algorithms can be developed. The running time of their algorithm depends on the maximum absolute value of the subdeterminants Δ of the constraint matrix, and if $\Delta = 1$ (i.e., the constraint matrix is totally unimodular), the running time becomes polynomial. The idea is based on the proximity theorem between integral and continuous optimal solutions. For the case of $\Delta = 1$, Karzanov and McCormick [76] proposed another polynomial-time algorithm. Hochbaum [61] strengthened the proximity result in Hochbaum and Shanthikumar [65] and improved the previous algorithm for the separable convex resource allocation problem with submodular constraint. It has been also shown that, if specialized to the separable convex resource allocation problem with a nested or tree constraint, the existing algorithms can be improved. Later, it was pointed out that the proximity theorems in Hochbaum [61] contained some errors, and they were fixed in Moriguchi and Shioura [107] and Moriguchi, Shioura, and Tsuchimura [108].

In addition to these efforts to generalize the constraints, several attempts have been directed to generalizing objective functions for which efficient algorithms can still be developed. One of them was done by Murota [109, 110] who identified a subclass of nonseparable convex functions, *M-convex functions*, which is defined on the base polyhedron of a submodular system. The *M*-convex functions can enjoy nice theorems of *discrete* convex analysis in a parallel manner to the traditional convex analysis. A polynomial-time algorithm has been developed for this class of problems [129].

Another interesting research field is the minimax resource allocation problem. The simplest of this type is described as follows:

$$\begin{aligned}
& \text{minimize} && \max_{1 \leq j \leq n} f_j(x_j) \\
& \text{subject to} && \sum_{j=1}^n x_j = N, \\
& && x_j \geq 0, \quad \text{integer } j = 1, \dots, n.
\end{aligned}$$

Here all $f_j(x_j)$ are nonincreasing (or nondecreasing) in x_j . This class of problem has been extensively studied due to its simple structure as well as rich applications. Theoretically, the minimax resource allocation problem can be equivalently transformed into the above simple resource allocation problem with a separable convex objective function. Therefore, equally efficient algorithms can be developed for minimax problems. As for the extensions and generalizations, minimax resource allocation problems with multiple resources and multiperiod resource allocation problems have been studied in connection with applications to production planning in high-tech industries [82–86, 88, 99].

To extend applications of resource allocation problems, a few new fields have been pointed out, such as (1) optimal allocation of computer resources, for example, determining optimal buffer size or optimal cache size; (2) computer scheduling; (3) software reliability; and (4) queueing control.

The organization of this chapter is as follows. [Section 2](#) prepares basic concepts and defines several types of resource allocation problems to be discussed in this chapter. [Section 3](#) gives fundamental algorithms for the separable convex and minimax resource allocation problems with several different types of constraints. All the algorithms presented in [Sect. 3](#) are more than 20 years old and were surveyed in the book of [69]. The correctness proofs given in [Sect. 3](#) for these problems with submodular constraint are however much simplified, based on the result by Murota [110]. [Sections 4 through 6](#) review rather new results developed in the 1990s and 2000s. [Sect. 4](#) explains proximity theorems between integer and continuous optimal solutions for separable convex resource allocation problems under submodular constraints as well as under linear constraints with totally unimodular constraint matrix. In [Sect. 5](#), improved algorithms, developed in Hochbaum [61] by using the proximity theorem, are presented for several types of separable convex resource allocation problems. All these algorithms, however, are not strongly polynomial time. Then, two lower bound results for the simple resource allocation problem are shown. This result, also due to Hochbaum [61], denies the existence of a strongly polynomial-time algorithm even if each f_j is a polynomial of degree 3. In addition, strongly polynomial-time algorithms are presented for several types of quadratic separable convex resource allocation problems. These algorithms are given by Hochbaum and Hong [64] and based on the proximity theorems. [Section 6](#) reviews a new development of *nonseparable* convex resource allocation problem related to minimizing an M-convex function. [Section 7](#) explains several new application areas. [Section 8](#) touches upon further related topics, such as allocation of multiple resources, multiperiod allocation, and other variants of resource allocation problems.

2 Preliminaries and Problem Classification

2.1 Preliminaries

Definitions and basic concepts, which are necessary for the discussion in this chapter, are given.

Let R and Z be the set of reals and the set of integers, respectively. A function $f : R^n \rightarrow R \cup \{+\infty\}$ is *convex* if $f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y)$ holds for every $x, y \in R^n$ and every α with $0 \leq \alpha \leq 1$. A function $f : Z^n \rightarrow R \cup \{+\infty\}$ is also defined to be *convex* if it can be extended to a convex function in the usual sense, that is, there exists a convex function $\tilde{f} : R^n \rightarrow R \cup \{+\infty\}$ such that $\tilde{f}(x) = f(x)$ for all $x \in Z^n$. For $l, u \in Z$ with $l \leq u$, let $[l, u]$ denote the closed interval from l to u . In the following, a function f of the form $f : [l, u] \cap Z \rightarrow R$ is often considered. Define

$$d(x) \equiv f(x) - f(x - 1) \text{ for } x \in [l + 1, u] \cap Z,$$

which is called the *increment* of f at x . A function $f : [l, u] \cap Z \rightarrow R$ is called *convex* if $d(x)$ is nondecreasing in x (this definition coincides with the above if $l = -\infty$ and $u = +\infty$).

Computation Model Throughout this chapter, $f(x)$ is assumed to be real-valued. Note that representing a real number may require infinite data length in the strict mathematical sense. On the other hand, in order to derive the computational complexity of an algorithm, it is assumed in this chapter (as is done in most of the complexity analysis) an evaluation of $f(x)$ can be done in a unit time, unless otherwise stated. It is also assumed that basic operations such as an arithmetic operation and a comparison operation concerning real numbers require a unit time. Although the assumptions may sound unrealistic, it does not produce serious troubles because an approximate evaluation of $f(x)$ (in finite digits) done in a unit of time is usually sufficient for practical purposes. Even if an evaluation of $f(x)$ requires not constant but $O(g(x))$ time, it is often straightforward to reevaluate the computation time for this more general situation.

Submodular System Let n be a positive integer and let

$$E = \{1, 2, \dots, n\}.$$

Let 2^E denote the family of all subsets of E , and $\mathcal{D} \subseteq 2^E$ be given. If \mathcal{D} is closed under union and intersection operations, \mathcal{D} is called a *distributive lattice*. Assume throughout this chapter that \mathcal{D} denotes a distributive lattice and that $\emptyset, E \in \mathcal{D}$, and $r(\emptyset) = 0$. A function $r : \mathcal{D} \rightarrow Z$ is *submodular* over \mathcal{D} if

$$r(X) + r(Y) \geq r(X \cup Y) + r(X \cap Y) \tag{1}$$

for all $X, Y \in \mathcal{D}$. Such a pair (\mathcal{D}, r) is called a *submodular system*. At the first glance, it may be difficult to understand the intuitive meaning of (1), but it captures the essence of combinatorial structures that are common to apparently different problems [45, 96]. In fact, as will be described in Sects. 2.2 and 3.2, the set of additional constraints that are often required in real applications can be defined in terms of a submodular system.

Define

$$R^E = \{x = (x_1, x_2, \dots, x_n) \mid x_j \in R, j \in E\}, \quad (2)$$

$$Z^E = \{x = (x_1, x_2, \dots, x_n) \mid x_j \in Z, j \in E\}. \quad (3)$$

The unit basis vector $e(j) = (e_1(j), \dots, e_n(j)) \in Z^E$ is defined by $e_i(j) = 1$ for $i = j$ and 0 for $i \neq j$. For vectors $x, y \in R^E$, write $x \leq y$ if $x_j \leq y_j$ for all $j \in E$, and write $x < y$ if $x \leq y$ and $x_j < y_j$ for some $j \in E$. For $x \in R^E$ and $S \subseteq E$, define

$$x(S) \equiv \sum_{j \in S} x_j,$$

where $x(\emptyset) = 0$ is assumed by convention. For a submodular system (\mathcal{D}, r) ,

$$P(r) = \{x \mid x \in R^E, x(S) \leq r(S), \text{ for all } S \in \mathcal{D}\} \quad (4)$$

is called the *submodular polyhedron* of (\mathcal{D}, r) . A subset of $P(r)$,

$$B(r) = \{x \mid x \in P(r) \text{ and } x(E) = r(E)\} \quad (5)$$

is called the *base polyhedron* of (\mathcal{D}, r) , and $x \in B(r)$ is a *base* of (\mathcal{D}, r) . In particular, $x \in B(r) \cap Z^E$ is called an *integral base* of (\mathcal{D}, r) . $B(r) \cap Z^E$ is called an *integral base polyhedron*. The notation $B(r)$ is also used to denote $B(r) \cap Z^E$, unless confusion occurs.

Section 3.2 provides basic properties of a submodular system that are necessary to develop efficient algorithms for the resource allocation problem with a submodular constraint.

2.2 Problem Classification

A generic form of the resource allocation problem discussed in this chapter is described as follows:

$$\text{RESOURCE: minimize } f(x_1, x_2, \dots, x_n) \quad (6)$$

$$\text{subject to } \sum_{j=1}^n x_j = N, \quad (7)$$

$$x_j \geq 0, \quad j = 1, 2, \dots, n. \quad (8)$$

That is, given one type of resource whose total amount is equal to N , it is required to allocate it to n activities so that the objective value $f(x_1, x_2, \dots, x_n)$ is minimized. The objective value may be interpreted as the cost or loss, or the profit or reward, incurred by the resulting allocation. In case of profit or reward, it is natural to maximize f ; in such a case, maximization problem is considered. The difference between maximization and minimization is not essential because maximizing f is equivalent to minimizing $-f$.

Each variable x_j represents the amount of resource allocated to activity j . If the resource is divisible, x_j is a continuous variable that can take any nonnegative value. If it represents persons, processors, or trucks, on the other hand, variable x_j becomes a discrete variable that takes nonnegative integer values, and the constraint

$$x_j : \text{integer}, \quad j = 1, 2, \dots, n \quad (9)$$

is added to (7) and (8). The resource allocation problem with this constraint is referred to as *discrete resource allocation problem*.

As for the objective function, it usually has some special structure according to the intended applications. Typically, the following special case, called *separable*, is often considered:

$$f(x) = \sum_{j=1}^n f_j(x_j). \quad (10)$$

If each f_j is convex, the objective function is called *separable convex*.

Resource allocation problems are classified according to the types of objective functions, constraints, and variables. The classification scheme used in this chapter as well as several types of problem formulations according to the classification scheme are described below. In general, the notation $\alpha/\beta/\gamma$ is used to denote the type of a resource allocation problem. Here α specifies the type of objective function, β the constraint type, and γ the variable type. Here

$$\gamma = D \quad \text{and} \quad \gamma = C$$

denote the integer variable and the continuous variable, respectively. The notations for α and β are given as follows.

α : Objective Functions The objective function $f(x_1, \dots, x_n)$ may take the following special structures:

1. *Separable* (S for short): $\sum_{j=1}^n f_j(x_j)$, where each f_j is a function of one variable.
2. *Separable and convex* (SC for short): $\sum_{j=1}^n f_j(x_j)$, where each f_j is a convex function of one variable. In particular, denote by SQC the subclass where each f_j is quadratic and convex.

3. *Minimax*: minimize $\max_{1 \leq j \leq n} f_j(x_j)$,
Maximin: maximize $\min_{1 \leq j \leq n} f_j(x_j)$,
where all f_j are monotone nondecreasing in x_j .
4. *Lexicographically minimax* (Lexico-Minimax for short): Since the objective value of Minimax is determined by the single-variable x_k^* satisfying $f_k(x_k^*) = \max_j f_j(x_j^*)$, there may be many optimal solutions. To remove such ambiguity, the lexicographical ordering is introduced for n -dimensional vectors: given $a = (a_1, \dots, a_n)$ and $b = (b_1, \dots, b_n)$, a is *lexicographically smaller* than b (i.e., b is *lexicographically greater* than a) if $a_j = b_j$ for $j = 1, \dots, k-1$ and $a_k < b_k$ some k . This is denoted by $a \leq_{lex} b$ or $b \geq_{lex} a$. For $a = (a_1, \dots, a_n)$, denote $DEC(a)$ (respectively, $INC(a)$) the n -tuple of $a_j, j = 1, \dots, n$, arranged in nonincreasing order (respectively, nondecreasing order) of their values (e.g., for $a = (4, 3, 1, 5)$, it holds that $DEC(a) = (5, 4, 3, 1)$ and $INC(a) = (1, 3, 4, 5)$). The objective of Lexico-Minimax is to find an allocation vector $x = (x_1, \dots, x_n)$ such that $DEC(x)$ is lexicographically minimum. Notice that an optimal solution to Lexico-Minimax is also optimal to Minimax, but the converse is not generally true. Similarly, define Lexico-Maximin as the one that lexicographically maximizes $INC(x)$.
5. *Fair*: minimize $g(\max_{1 \leq j \leq n} f_j(x_j), \min_{1 \leq j \leq n} f_j(x_j))$, where $g(u, v)$ is nondecreasing (resp. nonincreasing) in u (resp. v). Intuitively, this objective makes $\max_j f_j(x) - \min_j f_j(x)$ small in the criterion defined by g and is a generalization of Minimax and Maximin.

β : Constraints In addition to the simple resource constraint of (7), other additional constraints are also imposed. Typical additional constraints which appeared in various resource allocation problems are as follows. The case of no additional constraints other than (7) is referred to as Simple (see Fig. 1a).

1. *Lower and upper bounds* (LUB for short): $l_j \leq x_j \leq u_j, j = 1, \dots, n$.
2. *Generalized upper bounds* (GUB for short): $\sum_{j \in S_i} x_j \leq b_i, i = 1, \dots, m$, where S_1, S_2, \dots, S_m is a partition of $\{1, 2, \dots, n\}$ (see Fig. 1b).
3. *Nested constraints* (Nested for short): $\sum_{j \in S_i} x_j \leq b_i, i = 1, \dots, m$, where $S_1 \subset S_2 \subset \dots \subset S_m$. It can be assumed that $b_1 \leq b_2 \leq \dots \leq b_m$ since if $b_i > b_{i+1}$, the i -th constraint is redundant (see Fig. 1c).
4. *Tree constraints* (Tree for short): $\sum_{j \in S_i} x_j \leq b_i, i = 1, \dots, m$, where the sets S_i are derived by some hierarchical decomposition of E into disjoint subsets (see Fig. 1d).
5. *Network constraints* (Network for short): The constraint is defined in terms of a directed network with a single source and multiple sinks (see Fig. 1e). Given a directed graph $G = (V, A)$ with node set V and arc set A , let $s \in V$ be the source and $T \subseteq V$ be the set of sinks. The amount of supply from the source is $N > 0$, and the capacity of arc (u, v) is $c(u, v)$. Denote the flow vector by $\varphi = \{\varphi(u, v) \mid (u, v) \in A\}$. φ is a *feasible flow* in G if it satisfies

$$0 \leq \varphi(u, v) \leq c(u, v), \quad (u, v) \in A, \quad (11)$$

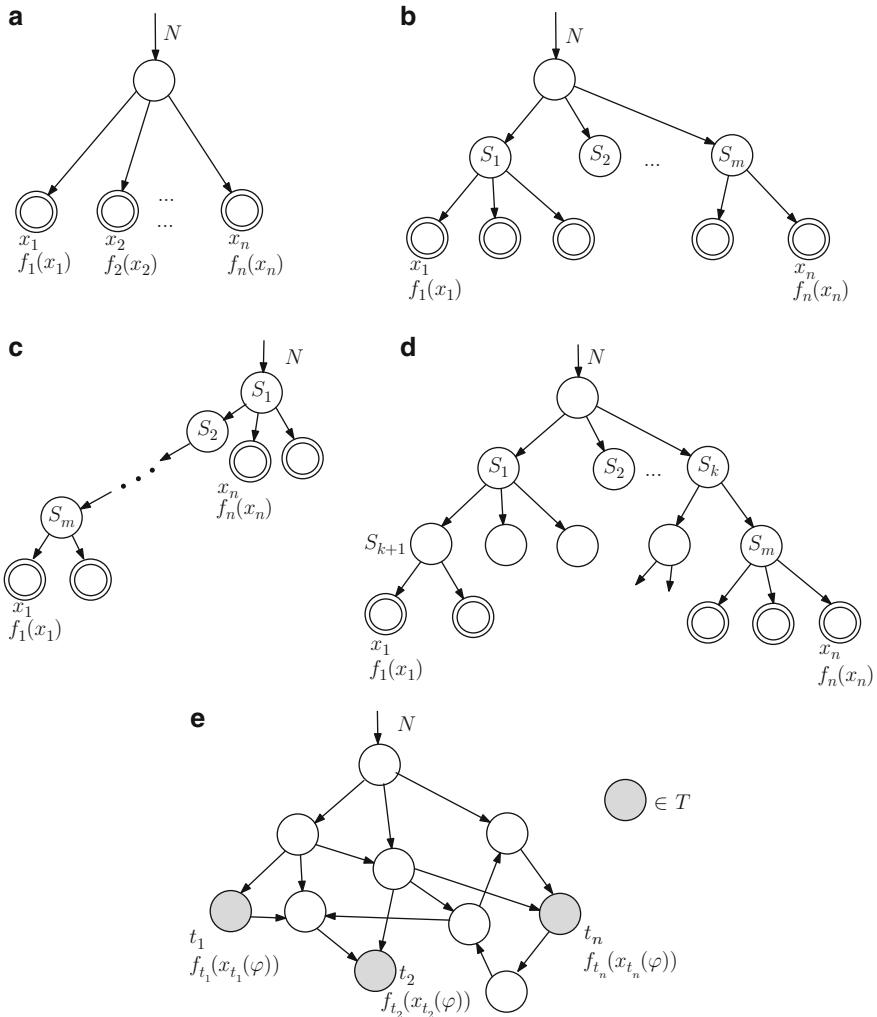


Fig. 1 Illustration of several constraints. (a) Simple. (b) GUB. (c) Nested. (d) Tree. (e) Network

$$\sum_{(v,w) \in A_+(v)} \varphi(v, w) - \sum_{(u,v) \in A_-(v)} \varphi(u, v) = 0, \quad v \in V - T - \{s\}, \quad (12)$$

$$\sum_{(s,v) \in A_+(s)} \varphi(s, v) - \sum_{(u,s) \in A_-(s)} \varphi(u, s) = N, \quad (13)$$

$$x_t(\varphi) \equiv \sum_{(u,t) \in A_-(t)} \varphi(u, t) - \sum_{(t,v) \in A_+(t)} \varphi(t, v) \geq 0, \quad t \in T, \quad (14)$$

$$\sum_t x_t(\varphi) = N, \quad (15)$$

where for $v \in V$, $A_+(v)$ (resp., $A_-(v)$) denotes the set of arcs leaving v (resp., entering v). The value $x_t(\varphi)$ denotes the amount of flow entering a sink $t \in T$. For a feasible flow φ , the vector $\{x_t(\varphi) | t \in T\}$ is called the *feasible flow vector* with respect to φ . For instance, the problem *SC/Network/C* (i.e., the separable convex resource allocation problem under network constraints) is defined as follows:

$$\begin{aligned} & \text{minimize: } \sum_{t \in T} f_t(x_t(\varphi)) \\ & \text{subject to } (11)-(15), \end{aligned} \quad (16)$$

where f_t for each $t \in T$ is a convex function.

6. *Submodular constraints* (SM for short): A set of feasible solutions x is defined by a base polyhedron $B(r)$ of (4) and (5) for a submodular system (\mathcal{D}, r) , that is,

$$x \in B(r). \quad (17)$$

In this case, notice that the constraint (7) is a part of constraint (17) since it is included as $x(E) = r(E)$ of the definition of $B(r)$. In the case of integer variables, the constraint becomes

$$x \in B(r) \cap Z^E.$$

It is assumed throughout this chapter that $B(r)$ is not explicitly given as an input, but is implicitly given through an *oracle* that tells us the value $r(S)$ whenever $S \in \mathcal{D}$ is input.

7. *General linear constraints* (Linear for short): Constraint is defined by a set of linear inequalities.

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, 2, \dots, m. \quad (18)$$

No other special assumption is imposed on the structure of the constraints.

Notice that all the constraints, LUB, GUB, Nested, Tree, Network, are special cases of SM (see [69]), and SM is a special case of Linear.

3 Fundamental Algorithms

This section first deals with the simplest problem SC/Simple/D and reviews two basic algorithms. Then, a more general problem SC/SM/D is considered, and two algorithms are presented. A relationship between SC/-/D and Minimax (or Maximin)/-/D is also discussed.

3.1 SC/Simple/D

3.1.1 Incremental Algorithm

An incremental algorithm for problem SC/Simple/D is first presented. It is assumed that each f_j is defined over the interval $[0, N]$. Since f_j is convex, it holds that

$$d_j(1) \leq d_j(2) \leq \cdots \leq d_j(N), \quad (19)$$

where

$$d_j(y_j) = f_j(y_j) - f(y_j - 1).$$

The incremental algorithm is a kind of *greedy algorithm* and is also called a *marginal allocation* method. Starting with the initial solution $x = (0, 0, \dots, 0)$, one unit of resource is allocated at each iteration to the most favorable activity (in the sense of minimizing the increase in the current objective value) until $\sum x_j = N$ is attained.

Procedure INCREMENT

Input: An instance of SC/Simple/D.

Output: An optimal solution x^* .

Let $x := (0, 0, \dots, 0)$ and $k := 0$;

while $k < N$ **do**

begin

 Find j^* such that $d_{j^*}(x_{j^*} + 1) = \min_{1 \leq j \leq n} d_j(x_j + 1)$;

$x_{j^*} := x_{j^*} + 1$;

$k := k + 1$

end;

Output x as x^* .

Theorem 1 Given an instance of SC/Simple/D, define the following matrix M :

$$M = \begin{bmatrix} d_1(1) & d_2(1) & \cdots & d_n(1) \\ d_1(2) & d_2(2) & \cdots & d_n(2) \\ \vdots & \vdots & \ddots & \vdots \\ d_1(N) & d_2(N) & \cdots & d_n(N) \end{bmatrix}. \quad (20)$$

Let X be the set of N smallest elements in M , where if $d_j(y-1) = d_j(y)$, $d_j(y-1)$ is given a higher priority to be chosen in X . Then the following x^* is an optimal solution of SC/Simple/D:

$$x_j^* = \begin{cases} 0, & d_j(1) \notin X, \\ N, & d_j(N) \in X, \\ y, & d_j(y) \in X \text{ and } d_j(y+1) \notin X. \end{cases} \quad (21)$$

Proof Suppose that x^* is not optimal, and let \hat{x} be the optimal solution such that

$$\delta = \sum_{j=1}^n |\hat{x}_j - x_j^*|$$

is minimum. Then from $\sum_{j=1}^n \hat{x}_j = \sum_{j=1}^n x_j^* = N$, there exist i and k such that $x_i^* > \hat{x}_i$ and $x_k^* < \hat{x}_k$. It follows from (19) that

$$d_i(\hat{x}_i) \leq d_i(\hat{x}_i + 1) \leq d_i(x_i^*), \quad d_k(x_k^*) \leq d_k(x_k^* + 1) \leq d_k(\hat{x}_k).$$

Since $d_i(x_i^*) \leq d_k(x_k^* + 1)$ from $d_k(x_k^* + 1) \notin X$ and $d_i(x_i^*) \in X$, it follows $d_i(\hat{x}_i + 1) \leq d_k(\hat{x}_k)$. For the feasible solution $x' = \hat{x} + e(i) - e(k)$, it holds that

$$\sum_{j=1}^n f_j(x'_j) - \sum_{j=1}^n f_j(\hat{x}_j) = d_i(\hat{x}_i + 1) - d_k(\hat{x}_k) \leq 0.$$

This says that x' is also an optimal solution of SC/Simple/D, contradicting the minimality of δ . \square

Theorem 2 Procedure INCREMENT correctly computes an optimal solution in $O(N \log n + n)$ time.

Proof Let x^k denote the solution after k increments are made in the **while** loop. Then it is an easy exercise to prove by induction that

$$X^k = \{d_j(y) \mid y = 1, 2, \dots, x_j^k, j = 1, 2, \dots, n\}$$

is a set of k smallest elements in M . Thus, the correctness immediately follows from **Theorem 1**. For the running time, the set

$$S = \{d_j(x_j + 1) \mid j = 1, 2, \dots, n\}$$

is maintained in an appropriate data structure such as heap. In each iteration of **while** loop, the minimum element $d_{j^*}(x_{j^*} + 1)$ is chosen and S is thereby updated. Since the initial construction of S requires $O(n)$ time and finding the minimum element of S and the update of S require $O(\log n)$ time, and **while** loop is iterated N times, the running time of $O(N \log n + n)$ is derived. \square

Algorithm INCREMENT is easy to be implemented and efficient if N is not too large. However, it is not polynomial in the input size because the input size is $O(n + \log N)$, where it is assumed that the description of each f_j requires a constant input length.

3.1.2 Polynomial-Time Algorithms

Several polynomial-time algorithms have been developed for problem SC/Simple/D [42, 48, 78]. The fastest among them is proposed by Frederickson and Johnson [42]. Its running time is $O(\max\{n, n \log(N/n)\})$, but the algorithm is very complicated. For this reason, the $O(n(\log N)^2)$ time algorithm by Galil and Megiddo [48] is presented here, which is a *divide-and-conquer* method to find the N -th smallest element in matrix M of (20). Recall that each column M_j of M is already sorted in the nondecreasing order by property (19).

Once the N -th smallest element $d_{j^*}(y^*)$ in M is found, let $\lambda = d_{j^*}(y^*)$ and construct an optimal solution x^* as follows, where care is needed in the situation that many $d_j(y)$ may satisfy $d_j(y) = \lambda$. First, compute the following $p_j(\lambda)$ and $q_j(\lambda)$ for each j with $j = 1, 2, \dots, n$ by binary search.

$$\begin{aligned} p_j(\lambda) &= \max\{y \mid y \in \{1, 2, \dots, N\} \text{ and } d_j(y) < \lambda\}, \\ q_j(\lambda) &= \max\{y \mid y \in \{1, 2, \dots, N\} \text{ and } d_j(y) \leq \lambda\}. \end{aligned} \tag{22}$$

Then, compute the index j' such that

$$j' = \max \left\{ j \mid \sum_{k=1}^j q_k(\lambda) + \sum_{k=j+1}^n p_k(\lambda) \leq N \right\},$$

and let

$$x_j^* = \begin{cases} q_j(\lambda), & j \leq j' \\ N - \left(\sum_{k=1}^{j'} q_k(\lambda) + \sum_{k=j'+2}^n p_k(\lambda) \right), & j = j' + 1 \\ p_j(\lambda), & j \geq j' + 2. \end{cases} \tag{23}$$

Notice that, after $\lambda = d_{j^*}(y^*)$ is computed, the time required to compute x^* in this way is $O(n \log N)$ because the computation of $p_j(\lambda)$ and $q_j(\lambda)$ can be done in $O(\log N)$ time for each j by applying the binary search over $[1, N]$. Then it is easy to see that set $\{d_j(y) \mid 1 \leq y \leq x_j^*, j = 1, \dots, n\}$ is the set of N smallest elements of M . Thus, the x^* defined above is optimal to SC/Simple/D.

Now, it is explained how to compute the N -th smallest element $d_{j^*}(y^*)$ in M . Let s denote the size of M (initially, $s = nN$). First, compute the medians $d_j(y_j^m)$ of columns M_j for all j , which can be computed in $O(1)$ time for each j because each M_j is already sorted. Sort these medians in the nondecreasing order

$$d_{j_1}(y_{j_1}^m) \leq d_{j_2}(y_{j_2}^m) \leq \cdots \leq d_{j_n}(y_{j_n}^m)$$

and compute the k such that

$$\sum_{i=1}^{k-1} |M_{j_i}| < s/2 \text{ and } \sum_{i=1}^k |M_{j_i}| \geq s/2. \quad (24)$$

Then, choose

$$\lambda = d_{j_k}(y_{j_k}^m)$$

as a candidate for the N -th smallest element of M . For this λ , define

$$\begin{aligned} M^1 &= \{d_j(y) \in M \mid d_j(y) < \lambda\}, \\ M^2 &= \{d_j(y) \in M \mid d_j(y) = \lambda\}, \\ M^3 &= \{d_j(y) \in M \mid d_j(y) > \lambda\}. \end{aligned}$$

If $|M^1| \geq N$, the N -th smallest element of M must be in M^1 , and the above procedure is recursively applied to M^1 . If $|M^1| + |M^2| \geq N > |M^1|$, $\lambda = d_{j_k}(y_{j_k}^m)$ is the N -th smallest element of M . If $|M^1| + |M^2| < N$, the N -th smallest element is in M^3 . In this case, the above procedure is recursively applied to M^3 after setting $N = N - (|M^1| + |M^2|)$.

At each iteration, the size of the current matrix M systematically decreases, while maintaining the property that it contains the target M^i selected in the previous iterations; this property implies that eventually the N -th smallest element of M is found. It is claimed that, at each iterations; this property implies that at most three-fourths of the elements in the current matrix M remain in the M of the next iteration. To see this, first note that by the choice of k , the set M' of Fig. 2 contains $\sum_{i=1}^k \lceil M_{j_i}/2 \rceil (\geq |M|/4)$ elements. Since each element in M' is not larger than $\lambda = d_{j_k}(y_{j_k}^m)$, $M' \subseteq M^1 \cup M^2$ follows. Hence,

$$|M^1| + |M^2| \geq |M'| \geq |M|/4$$

holds. A similar argument applied to M'' of Fig. 2 implies that

$$|M^2| + |M^3| \geq |M|/4.$$

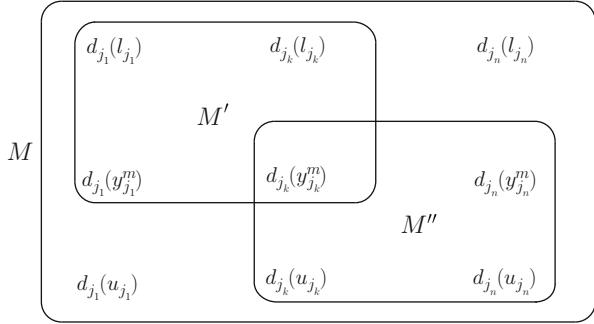
From these, it is concluded that

$$|M^1| \leq 3|M|/4 \text{ and } |M^3| \leq 3|M|/4,$$

proving the above claim.

From this observation, it is concluded that $O(\log N)$ iterations are required before $|M| \leq n$ holds. At this point, a linear-time selection algorithm is directly applied to the current M of size $O(n)$ to identify the N -th smallest element. The algorithm is described as follows:

Fig. 2 Illustration of M , M' and M''



Procedure SELECT

Input: An instance of problem SC/Simple/D.

Output: An optimal solution x^* .

Let $l_j := 1$ and $u_j := N$ for $j = 1, 2, \dots, n$;

Let $L_j := u_j - l_j + 1$ for $j = 1, 2, \dots, n$ and $s := \sum_{j=1}^n L_j$;

while $s > n$ **do**

begin

 Find the medians of all $d_j(y_j^m)$ in M_j , $j = 1, 2, \dots, n$, and sort them in nondecreasing order, $d_{j_1}(y_{j_1}^m) \leq d_{j_2}(y_{j_2}^m) \leq \dots \leq d_{j_n}(y_{j_n}^m)$;

 Compute the k such that $\sum_{i=1}^{k-1} |M_{j_i}| < s/2$ and $\sum_{i=1}^k |M_{j_i}| \geq s/2$, and let $\lambda = d_{j_k}(y_{j_k}^m)$;

 Compute $p_j(\lambda)$ and $q_j(\lambda)$ of (22) for $j = 1, 2, \dots, n$, and let

$L^1 := \sum_{j=1}^n (p_j(\lambda) - l_j + 1)$, and $L^2 := \sum_{j=1}^n (q_j(\lambda) - l_j + 1)$;

if $L^1 \leq N \leq L^2$ **then** output $\lambda = d_{j_k}(y_{j_k}^m)$ and halt.

else

if $L^1 \geq N$ **then** let $u_j := p_j(\lambda)$, $j = 1, \dots, n$

else let $N := N - L^2$ and $l_j := q_j(\lambda) + 1$, $j = 1, \dots, n$.

end;

Find the N -th smallest element $\lambda = d_{j^*}(y^*)$ in M and construct an optimal solution x^* by (23).

It is assumed in the algorithm that M_j denotes the j -th column of the current M with lower and upper bounds, that is, $M_j = \{d_j(l_j), \dots, d_j(u_j)\}$. Thus, the current M is given by $M = \cup_{1 \leq j \leq n} M_j$.

Theorem 3 Given an instance of problem SC/Simple/D, procedure SELECT correctly finds the N -th smallest element in M of (20) in $O(n(\log N)^2)$ time.

Proof The correctness directly follows from the discussion prior to the description of the algorithm. The running time is analyzed as follows. First, the medians of $d_j(y_j^m)$ of M_j , $j = 1, 2, \dots, n$ are computed in $O(n)$ time because each

column is already sorted as (19). Sorting these n medians requires $O(n \log n)$ time. Computation of $p_j(\lambda)$ and $q_j(\lambda)$ for each j is done by binary search and requires $O(\log |M_j|) = O(\log N)$ time. Thus, $O(n \log N)$ time is required in total. As mentioned earlier, the **while** loop of SELECT is iterated $O(\log N)$ times, and the total running time is

$$O((n \log N + n \log n) \log N).$$

This running time is further improved by repeatedly applying linear-time median finding algorithm for identifying the k of (24) and $\lambda = d_{j_k}(y_{j_k}^m)$, instead of sorting all medians $d_j(y_j^m)$. The details are omitted (see [48] or the book [69] for details). Using this strategy, the running time is reduced to

$$O((n \log N + n) \log N) = O(n(\log N)^2).$$

□

It is clear that the two algorithms INCREMENT and SELECT can be generalized in a straightforward manner to deal with the constraints of LUB without changing the time complexity.

3.2 SC/SM/D

The problem SC/SM/D is described as follows:

$$\begin{aligned} & \text{minimize} \sum_{j \in E} f_j(x_j), \\ & \text{subject to } x = (x_1, x_2, \dots, x_n) \in B(r), \\ & \quad x_j : \text{nonnegative integer}, \quad j \in E, \end{aligned} \tag{25}$$

where $E = \{1, \dots, n\}$ and $B(r)$ denotes the base polyhedron of a submodular system, as defined in Sect. 2.1.

Before describing algorithms for solving (25), some definitions and properties concerning a submodular system are given, which are used to develop the algorithms. Proofs of the properties are omitted except for Lemma 4.

Lemma 1 *For a submodular system (\mathcal{D}, r) and a vector $y \in R^E$, define a function $r^y : 2^E \rightarrow R$ by*

$$r^y(X) = \min\{r(Z) + y(X - Z) \mid Z \subseteq X, Z \in \mathcal{D}\}, \quad X \subseteq E. \tag{26}$$

Then r^y is a submodular function defined over 2^E .

□

This submodular system $(2^E, r^y)$ is called the *restriction of (\mathcal{D}, r) by vector y* . The base polyhedron $B(r^y)$ of $(2^E, r^y)$ is given by

$$\{x \mid x \in B(r), x \leq y\}. \quad (27)$$

Lemma 2 For a submodular system (\mathcal{D}, r) and a vector $y \in R^E$, define a function $r_y : 2^E \rightarrow R$ by

$$r_y(X) = \min\{r(Z) - y(Z - X) \mid X \subseteq Z, Z \in \mathcal{D}\}, \quad X \subseteq E. \quad (28)$$

Then r_y is a submodular function defined over 2^E . \square

This submodular system $(2^E, r_y)$ is called the *contraction* of (\mathcal{D}, r) by vector y . The base polyhedron $B(r_y)$ of $(2^E, r_y)$ is given by

$$\{x \mid x \in B(r), x \geq y\}. \quad (29)$$

Since nonnegative integer x_j is mainly considered in this section, it is assumed throughout this section that the underlying submodular system is equal to $(2^E, r_0)$ which is obtained by contracting a submodular system (\mathcal{D}, r) by zero vector $\mathbf{0}$. Therefore, $B_0(r)$ denotes in what follows the base polyhedron corresponding to $(2^E, r_0)$, and the following problem is considered instead of (25):

$$\begin{aligned} & \text{minimize} \sum_{j \in E} f_j(x_j) \\ & \text{subject to } x : \text{an integral base of } B_0(r). \end{aligned} \quad (30)$$

For a submodular system (\mathcal{D}, r) and a vector $x \in P(r)$ (see (4) for the definition of $P(r)$), define the *saturation function* $sat : P(r) \rightarrow 2^E$ by

$$sat(x) = \{j \mid j \in E, x + d \cdot e(j) \notin P(r) \text{ for any } d > 0\}. \quad (31)$$

Also, define *dependence function* $dep : P(r) \times E \rightarrow 2^E$ by

$$dep(x, j) = \begin{cases} \{j' \mid j' \in E, x + d \cdot (e(j) - e(j')) \in P(r) \text{ for some } d > 0\} \\ \emptyset \text{ otherwise.} \end{cases} \quad (32)$$

The following lemma characterizes sat and dep (see [45] for its proof).

Lemma 3 (i) For $x \in P(r)$, it holds that

$$sat(x) = \bigcup_{X \in \mathcal{D}: x(X)=r(X)} X. \quad (33)$$

(ii) For $x \in P(r)$ and $j \in E$, it holds that

$$dep(x, j) = \bigcap_{X \in \mathcal{D}: j \in X, x(X) = r(X)} X. \quad (34)$$

□

For $x \in P(r)$ and $j \in E$, define the *saturation capacity* as

$$\hat{c}(x, j) = \max\{d \mid d \geq 0, x + d \cdot e(j) \in P(r)\}. \quad (35)$$

Similarly, for $x \in P(r)$, $j \in sat(x)$, and $j' \in dep(x, j) - \{j\}$, define the *exchange capacity* as

$$\tilde{c}(x, j, j') = \max\{d \mid d > 0, x + d \cdot (e(j) - e(j')) \in P(r)\}. \quad (36)$$

The following lemma is a basis of the algorithms presented in this section and [Sect. 6](#).

Lemma 4 (i) For any integral base x of $B_0(r)$, $j \in E$, and $j' \in dep(x, j) - \{j\}$,

$$x + e(j) - e(j')$$

is also an integral base of $B_0(r)$.

(ii) For any two distinct integral bases $x, y \in B_0(r)$, and for any $i \in E$ with $x_i > y_i$, there exists $j \in E$ with $x_j < y_j$ such that

$$x - e(i) + e(j) \in B_0(r) \text{ and } y + e(i) - e(j) \in B_0(r).$$

Proof Only the proof of (ii) is given since the proof of (i) is done in almost the same fashion as that of [Claim 1](#) below. It starts with the following two claims.

Claim 1 For any two distinct integral bases $x, y \in B_0(r)$, and for any $i \in E$ with $x_i > y_i$, there exists $j \in E$ with $x_j < y_j$ such that

$$y + e(i) - e(j) \in B_0(r).$$

Proof Let $Y = dep(y, i)$. First, it is shown that $y(Y) = r(Y)$ holds. Notice that, for any X, X' with $i \in X \cap X'$ such that $y(X) = r(X)$ and $y(X') = r(X')$ hold, $y(X \cap X') = r(X \cap X')$ also holds because

$$\begin{aligned} y(X \cap X') &\leq r(X \cap X') \\ &\leq r(X) + r(X') - r(X \cup X') \quad (\text{from submodularity of } r(\cdot)) \\ &\leq r(X) + r(X') - y(X \cup X') \\ &= y(X) + y(X') - y(X \cup X') \quad (\text{from } y(X) = r(X) \text{ and}) \end{aligned}$$

$$\begin{aligned} & y(X') = r(X')) \\ &= y(X \cap X'). \end{aligned}$$

Therefore, repeating this argument, $y(Y) = r(Y)$ follows from (34). Also, $|Y| \geq 2$ holds since otherwise $Y = \{i\}$ holds and thus $r(Y) = y(Y) = y_i < x_i = x(Y)$ holds, violating $x \in B_0(r)$. Thus, there exists $j \in Y$ with $j \neq i$. Then it is shown that $y + e(i) - e(j) \in B_0(r)$. From the definition of Y , all X with $i \in X$ and $y(X) = r(X)$ satisfies $X \supseteq Y$. Thus, $(y + e(i) - e(j))(X) = r(X)$ for all such X . For other X , either $y(X) < r(X)$ or $i \notin X$ and $y(X) = r(X)$ holds, and hence, $(y + e(i) - e(j))(X) \leq r(X)$ follows from the integrality of y and r . This proves the claim. \square

Claim 2 For any two distinct integral bases $x, y \in B_0(r)$ with $\|x - y\| (= \sum_j |x_j - y_j|) = 4$, the lemma assertion (ii) holds.

Proof The claim is proved by case analysis. There are two cases.

- Case 1.* There exists exactly one i such that $x_i > y_i$. From $\|x - y\| = 4$, $x_i - y_i = 2$. From [Claim 1](#), there exists j with $x_j < y_j$ such that $y + e(i) - e(j) \in B_0(r)$. For the j , applying [Claim 1](#) to y , $x - e(i) + e(j) \in B_0(r)$ must follow because the i is the unique element such that $x_i > y_i$.
- Case 2.* There are i, i' with $i \neq i'$ such that $x_i > y_i$ and $x_{i'} > y_{i'}$. From $\|x - y\| = 4$, $x_i - y_i = 1$ and $x_{i'} - y_{i'} = 1$. Then, the claim can be proved in a manner similar to Case 1. \square

Now the lemma shall be proven. Assume, to the contrary, that the following set is not empty:

$$\begin{aligned} \mathcal{S} = \{(x, y) \mid x, y \in B_0(r), \exists i^* \text{ with } x_{i^*} > y_{i^*} \text{ such that} \\ \forall j \text{ with } x_j < y_j, \text{ it holds that } x - e(i^*) + e(j) \notin B_0(r) \text{ or} \\ y + e(i^*) - e(j) \notin B_0(r)\}. \end{aligned} \tag{37}$$

Choose (x, y) from \mathcal{S} such that $\|x - y\|$ is minimum. It is assumed that $\|x - y\| \geq 6$ since otherwise the lemma is immediate from [Claims 1](#) and [2](#). Let i^* be the index satisfying the condition of (37) for the (x, y) . Consider first the case such that i^* is the unique element satisfying $x_{i^*} > y_{i^*}$. From [Claim 1](#), there exists j_0 with $y + e(i^*) - e(j_0) \in B_0(r)$, and again by applying [Claim 1](#) with roles of x and y interchanged, $x - e(i^*) + e(j_0) \in B_0(r)$ follows since i^* is the unique element satisfying $x_{i^*} > y_{i^*}$. Thus, the lemma follows. Therefore, assume that there exists i_0 with $i_0 \neq i^*$ and $x_{i_0} > y_{i_0}$. Let

$$\begin{aligned} X &= \{j \in E \mid x - e(i^*) + e(j) \in B_0(r)\}, \\ Y &= \{j_0 \in E \mid x_{j_0} < y_{j_0}, y + e(i_0) - e(j_0) \in B_0(r)\}. \end{aligned}$$

Recall that $Y \neq \emptyset$ holds from [Claim 1](#).

Case 1: $X \cap Y \neq \emptyset$. Fix a j_0 such that $j_0 \in X \cap Y$. Let $y' = y + e(i_0) - e(j_0)$. A contradiction shall be derived by showing $(x, y') \in \mathcal{S}$ because $\|x - y'\| = \|x - y\| - 2$. Consider an arbitrary j with $x_j < y'_j$ and $x - e(i^*) + e(j) \in B_0(r)$, and let

$$y'' = y' + e(i^*) - e(j) = y + e(i_0) + e(i^*) - e(j_0) - e(j).$$

Notice that $y + e(i^*) - e(j_0) \notin B_0(r)$ and $y + e(i^*) - e(j) \notin B_0(r)$ follow from $(x, y) \in \mathcal{S}$. Thus, $y'' \notin B_0(r)$ follows since otherwise at least one of $y + e(i^*) - e(j_0)$ or $y + e(i^*) - e(j)$ belongs to $B_0(r)$, if [Claim 1](#) is applied to y and y'' .

Case 2: $X \cap Y = \emptyset$. Let j and y'' be chosen as in case 1. Then, $y + e(i^*) - e(j) \notin B_0(r)$ holds from $(x, y) \in \mathcal{S}$, and $y + e(i_0) - e(j) \notin B_0(r)$ holds from $X \cap Y = \emptyset$ and the definition of Y . Thus, $y'' \notin B_0(r)$ follows since otherwise at least one of $y + e(i^*) - e(j)$ or $y + e(i_0) - e(j)$ belongs to $B_0(r)$ if [Claim 2](#) is applied to y and y'' . \square

The following lemma is well known (see [45, 55, 69] for its proof).

Lemma 5 *Given a submodular system (\mathcal{D}, r) , the following computations can be done in polynomial time in $|E|$ and in $\log r(E)$, provided that an oracle to compute $r(X)$ for a given X in polynomial time is available:*

1. *Membership test of $x \in P(r)$ for a given $x \in R^E$.*
2. *Computation of $\text{sat}(x)$ and $\text{dep}(x, j)$ for a given $x \in P(r)$ and $j \in E$.*
3. *Computation of $\hat{c}(x, j)$ of (36) for given $x \in P(r)$ and $j \in E$.*
4. *Computation of $r^x(X)$ of (26) and $r_x(X)$ of (27) for $x \in R^E$ and $X \in 2^E$.* \square

For a general submodular system, polynomial algorithms based on ellipsoid method can be used for all the above computations (see [55]). However, for most of particular cases, more efficient polynomial-time algorithms are available.

The following fundamental results are useful to develop efficient algorithms, which will be presented below.

Lemma 6 *Consider the base polyhedron $B_0(r)$ of a submodular system $(2^E, r_0)$ and a function $f(x) = \sum_{j \in E} f_j(x_j)$, where $f_j(x_j)$ is convex for all j . Then, for any two distinct integral bases $x, y \in B_0(r)$ and for any $i, j \in E$ with $x_i > y_i$ and $x_j < y_j$ such that*

$$x - e(i) + e(j), y + e(i) - e(j) \in B_0(r), \quad (38)$$

it holds that

$$f(x) + f(y) \geq f(x - e(i) + e(j)) + f(y + e(i) - e(j)). \quad (39)$$

Proof Inequality (39) follows from

$$\begin{aligned} & f(x) + f(y) - (f(x - e(i) + e(j)) + f(y + e(i) - e(j))) \\ &= d_i(x_i) - d_i(y_i + 1) + d_j(y_j) - d_j(x_j + 1), \end{aligned}$$

and the convexity of f_i and f_j . \square

The following theorem demonstrates that local optimality guarantees global optimality for a submodular system.

Theorem 4 Let $B_0(r)$ be the base polyhedron of a submodular system $(2^E, r_0)$. An integral base $x \in B_0(r)$ is an optimal solution of problem SC/SM/D if and only if

$$f(x - e(i) + e(j)) \geq f(x) \text{ (i.e., } d_j(x_j + 1) \geq d_i(x_i)) \quad (40)$$

holds for all $i, j \in E$ with $x - e(i) + e(j) \in B_0(r)$.

Proof Since the *only if* part is obvious, only the *if* part will be proven, that is,

$$f(y) \geq f(x) \text{ for all } y \in B_0(r). \quad (41)$$

Suppose, to the contrary, that the x satisfying assumption (40) is not optimal. Let y be an optimal solution such that $\|x - y\| (= \sum_{j \in E} |x_j - y_j|)$ is minimum. By Lemma 4 (ii), there exist two integral bases $x' = x - e(i) + e(j)$ and $y' = y + e(i) - e(j)$ with $x_i > y_i$ and $x_j < y_j$. Then, $f(x) + f(y) \geq f(x') + f(y')$ holds from (39). Since $f(x') \geq f(x)$ from (40), $f(y') \leq f(y)$ must follow. Thus, $f(y')$ is also an optimal solution and $\|x - y'\| < \|x - y\|$ holds, contradicting the minimality of $\|x - y\|$. \square

Two algorithms for finding an optimal solution of problem SC/SM/D will be presented below.

3.2.1 Incremental Algorithm

It will be shown that the incremental algorithm presented in Sect. 3.1.1 also works for problem SC/SM/D. In this case, among all the elements such that $x + e(j) \in P(r)$ (i.e., feasible except for the constraint $(x + e(j))(E) = r(E)$), the x_j with the minimum increase in $f_j(x_j)$ is incremented by one. This process is repeated until $x(E) = r(E)$ is finally attained.

Procedure SM-INCREMENT

Input: An instance of problem SC/SM/D.

Output: An optimal solution x^* .

```

Let  $x_j := 0$  for all  $j \in E$ ,  $E' := E$  and  $k := 0$ ;
while  $k < r(E)$  do
  begin
    Find  $j^* \in E'$  such that  $d_{j^*}(x_{j^*} + 1) = \min\{d_j(x_j + 1) \mid j \in E'\}$ .
    if  $x + e(j^*) \in P(r)$  then
      let  $x_{j^*} := x_{j^*} + 1$ ,  $k := k + 1$ ;
    else delete  $j^*$  from  $E'$ 
  end;
  Output  $x$  as  $x^*$ .

```

To prove the correctness of SM-INCREMENT, it is remarked here that, during the execution of the algorithm, it arrives at the following situation (A) from time to time:

- (A) For the solution x obtained in the **while** loop, some $X \in 2^E$ becomes newly saturated (i.e., $x(X) = r(X)$), as a result of increasing x_i by one (where $i = j^*$).

Once (A) occurs, such x_i will never be increased again. Let E_A denote the set of all i for which (A) occurs before the algorithm halts. Without loss of generality, assume $E_A = \{1, 2, \dots, p\}$, for which (A) has occurred in the order of $i = 1, 2, \dots, p$. Define

$$S_i = \bigcup_{x(Y)=r(Y), Y \in 2^E} Y, \quad (42)$$

whenever condition (A) has occurred for solution x and $i \in E$, where $S_0 = \emptyset$ by convention. Thus, S_i satisfies $x(S_i) = r(S_i)$ by the submodularity of r . When SM-INCREMENT halts, $i = p$ and $S_p = E$ hold. Now three lemmas and two theorems will be given, in which x^* denotes the solution obtained by SM-INCREMENT.

Lemma 7 *If condition (A) holds for the solution x and $i \in E$ during the execution of SM-INCREMENT, the optimal solution x^* output by SM-INCREMENT satisfies $x_j^* = x_j$ for all $j \in S_i$, and hence, $x^*(S_i) = r(S_i)$.*

Proof Once condition (A) holds with $i \in E$, x_j for any $j \in S_i$ cannot be incremented later by the test $x + e(j^*) \in P(r)$ in the algorithm. Thus, the lemma follows. \square

Lemma 8 *For the subsets $S_i, i = 1, 2, \dots, p$, constructed in SM-INCREMENT, the following properties hold:*

- (i) $S_0 \subset S_1 \subset \dots \subset S_p = E$.

(ii) For all i with $1 \leq i \leq p$, $\{x_j^* \mid j \in S_i - S_{i-1}\}$ is an optimal solution of the following problem of SC/Simple/D:

$$\begin{aligned} Q_i : & \text{minimize} \sum_{j \in S_i - S_{i-1}} f_j(x_j) \\ \text{subject to} & \sum_{j \in S_i - S_{i-1}} x_j = r(S_i) - r(S_{i-1}), \\ & x_j : \text{nonnegative integer}, \quad j \in S_i - S_{i-1}. \end{aligned} \quad (43)$$

Proof (i) Immediate from the definition of S_i , $i = 1, \dots, p$.

(ii) Before S_i becomes saturated, it holds $j \notin \text{sat}(x)$ for all $j \in S_i - S_{i-1}$. Hence, $x + e(j) \in P(r)$. Thus, SM-INCREMENT selects the $j^* \in S_i - S_{i-1}$ such that

$$d_{j^*}(x_{j^*} + 1) = \min\{d_j(x_j + 1) \mid j \in S_i - S_{i-1}\} \quad (44)$$

in the succeeding situations. Since $x^*(S_i) = r(S_i)$ and $x^*(S_{i-1}) = r(S_{i-1})$ hold, it holds that $x^*(S_i - S_{i-1}) = r(S_i) - r(S_{i-1})$. Thus, what SM-INCREMENT does to the variable x_j for $j \in S_i - S_{i-1}$ is exactly the same as what procedure INCREMENT does if problem Q_i of (43) is input. The correctness of INCREMENT then proves property (ii). \square

Lemma 9 In the execution of SM-INCREMENT,

$$d_j(x_j^*) \leq \min\{d_k(x_k^* + 1) \mid k \in E - S_{i-1}\} \quad (45)$$

holds for all $i \in E_A$ and any $j \in S_i - S_{i-1}$.

Proof For a $j \in S_i - S_{i-1}$, let $\hat{x} = \{\hat{x}_k \mid k \in E\}$ denote the vector obtained just before $x_j = x_j^* - 1$ is increased to $x_j = x_j^*$. Then all $k \in E - S_{i-1}$ satisfy $\hat{x} + e(k) \in P(r)$ by the definition of S_i and S_{i-1} , and thus,

$$d_j(x_j^*) = d_j(\hat{x}_j + 1) = \min\{d_k(\hat{x}_k + 1) \mid k \in E - S_{i-1}\}$$

holds. This proves the lemma because $d_k(\hat{x}_k + 1) \leq d_k(x_k^* + 1)$ holds for all $k \in E - S_{i-1}$ by $\hat{x}_k \leq x_k^*$. \square

Theorem 5 SM-INCREMENT correctly computes an optimal solution of SC/SM/D in $O(r(E) \log n + (r(E) + n)\tau)$ time, where τ denotes the time required to perform the membership test of $x \in P(r)$.

Proof Let x^* be the solution output by SM-INCREMENT. This x^* is clearly a feasible solution of SC/SM/D. From [Theorem 4](#), it suffices to show that

$$d_j(x_j^* + 1) \geq d_i(x_i^*) \quad (46)$$

holds for all $i, j \in E$ with $x^* - e(i) + e(j) \in B_0(r)$. Let $x' = x^* - e(i) + e(j)$. Suppose $i \in S_k - S_{k-1}$. Then $j \in E - S_{k-1}$ must hold since otherwise $x'(S_{k-1}) > x^*(S_{k-1}) = r(S_{k-1})$ holds from $i \notin S_{k-1}$ and $j \in S_{k-1}$ (the equality is from Lemma 7), contradicting the feasibility of x' . Then (46) follows from Lemma 9.

As for the running time, it is needed to test whether $x + e(j^*) \in P(r)$. Once j^* is removed from E' , j^* is no more considered. Thus, the number of executions of such feasible test is at most $r(E) + n$. The other part of the procedure is analyzed in a manner similar to the analysis of procedure INCREMENT. This completes the proof. \square

Notice that the running time of the above procedure SM-INCREMENT is not polynomial but pseudo-polynomial if $r(E)$ is given as an input.

3.2.2 Polynomial-Time Algorithm

This section presents a polynomial algorithm for solving SC/SM/D. It first solves a problem of SC/Simple/D type, which is obtained from the original problem by considering only the simple constraint $x(E) = r(E)$ but disregarding the rest. If the obtained solution y is feasible, that is, $y \in P(r)$, then it is an optimal solution of the original problem. Otherwise, the problem is decomposed into subproblems using the information obtained from the vector y and the submodular constraints. For this, a maximal vector $v \in R^E$ satisfying below is first computed:

$$v \in P(r) \quad \text{and} \quad v \leq y. \quad (47)$$

If $v = y$, y is a feasible solution of SC/SM/D, that is, y is an optimal solution. Otherwise, let

$$E_1 = \text{sat}(v) \quad \text{and} \quad E_2 = E - E_1,$$

and define vectors $l^1, u^1 \in R^{E_1}$ and $l^2, u^2 \in R^{E_2}$ by

$$\begin{aligned} l_j^1 &= 0, & u_j^1 &= y_j \quad \text{for } j \in E_1, \\ l_j^2 &= y_j, & u_j^2 &= +\infty \quad \text{for } j \in E_2. \end{aligned} \quad (48)$$

As will be proved later, there exists an optimal solution x^* of SC/SM/D such that

$$\begin{aligned} l_j^1 &\leq x_j^* \leq u_j^1 \quad \text{for } j \in E_1, \\ l_j^2 &\leq x_j^* \leq u_j^2 \quad \text{for } j \in E_2, \\ x^*(E_1) &= r(E_1), \\ x^*(E_2) &= r(E) - r(E_1). \end{aligned} \quad (49)$$

This means that the original problem is equivalently solved by the following two subproblems, Q_1 and Q_2 :

$$\begin{aligned} Q_1 : & \text{minimize } \sum_{j \in E_1} f_j(x_j) \\ \text{subject to } & x(E_1) = r(E_1), \\ & x(X) \leq r(X), X \in 2^{E_1}, \\ & l_j^1 \leq x_j \leq u_j^1, j \in E_1, \\ & x_j : \text{nonnegative integer, } j \in E_1, \end{aligned} \quad (50)$$

$$\begin{aligned} Q_2 : & \text{minimize } \sum_{j \in E_2} f_j(x_j) \\ \text{subject to } & x(E_2) = r(E) - r(E_1), \\ & x(X) \leq r(X \cup E_1) - r(E_1), X \in 2^{E_2}, \\ & l_j^2 \leq x_j \leq u_j^2, j \in E_2, \\ & x_j : \text{nonnegative integer, } j \in E_2. \end{aligned} \quad (51)$$

Note that the constraint of problem Q_1 is to add constraints $x_j \leq y_j$, $j \in E_1$, to the original constraint $x \in B_0(r)$, and the constraint of problem Q_2 is to add constraints $x_j \geq y_j$, $j \in E_2$, to the original constraint $x \in B_0(r)$. The constraints of Q_1 and Q_2 are again submodular, because the former is equivalent to the restriction of $(2^{E_1}, r)$ by vector y (see Sect. 3.2), and the latter is equivalent to the contraction of $(2^{E_2}, r)$ by vector y (see Sect. 3.2). Thus, these problems can be solved by recursively executing the algorithm. In a general step of the recursion, it solves the following problem $SM(E', S, l, u)$ of SC/SM/D for some $E' \subset E$ and $l, u \in R^{E'}$, by first solving its relaxation, $SI(E', S, l, u)$:

$$\begin{aligned} SM(E', S, l, u) : & \text{minimize } \sum_{j \in E'} f_j(x_j) \\ \text{subject to } & x \in R^{E'}, \\ & x(E') = r(E' \cup S) - r(S), \\ & x(X) \leq r(X \cup S) - r(S), X \in 2^{E'}, \\ & l \leq x \leq u, \\ & x_j : \text{integer, } j \in E'. \end{aligned} \quad (52)$$

$$\begin{aligned} SI(E', S, l, u) : & \text{minimize } \sum_{j \in E'} f_j(x_j) \\ \text{subject to } & x \in R^{E'}, \\ & x(E') = r(E' \cup S) - r(S), \\ & l \leq x \leq u, \\ & x_j : \text{integer, } j \in E'. \end{aligned} \quad (53)$$

The entire procedure is called SM, in which subroutine $DA(E', S, l, u)$ solves the above problem $SM(E', S, l, u)$.

Procedure SM

Input: An instance of problem SC/SM/D.

Output: An optimal solution x^* .

Step 1: Let $l_j := 0$ and $u_j := +\infty$ for $j \in E$.

Step 2: Call procedure DA(E, \emptyset, l, u) to obtain a solution x^* .

Step 3: Output x^* and halt.

Procedure DA(E' , S , l , u)

Step 1: Compute an optimal solution y of SI(E' , S , l , u).

Step 2: Find a maximal vector $v \in P'(r)$ such that $v \leq y$, where $P'(r)$ is the submodular polyhedron of the underlying submodular system of problem SM(E' , S , l , u).

Step 3: If $v = y$, let $x^* := y$ and return x^* .

Step 4: Compute $\text{sat}(v)$ for $P'(r)$, and let $E'_1 := \text{sat}(v)$ and $E'_2 := E' - E'_1$.

Step 5: Define two vectors $l^1, u^1 \in Z^{E'_1}$ by $l_j^1 := l_j, u_j^1 := y_j$ for $j \in E'_1$, and call DA(E'_1, S, l^1, u^1) to obtain an optimal solution x^1 of $Q'_1 = \text{SM}(E'_1, S, l^1, u^1)$ (which is the above Q_1 in the case of the first call).

Step 6: Define two vectors $l^2, u^2 \in Z^{E'_2}$ by $l_j^2 := y_j, u_j^2 := u_j$ for $j \in E'_2$, and call DA($E'_2, S \cup E'_1, l^2, u^2$) to obtain an optimal solution x^2 of $Q'_2 = \text{SM}(E'_2, S \cup E'_1, l^2, u^2)$ (which is the above Q_2 in the first call).

Step 7: Return an optimal solution $x^* = (x_j^1, j \in E'_1; x_j^2, j \in E'_2)$.

Before proving the correctness of procedure SM, the following lemma is proved:

Lemma 10 *If DA(E' , S , l , u) returns x^* in Step 3, it is an optimal solution of problem SM(E' , S , l , u).*

Proof Step 1 of DA(E' , S , l , u) computes an optimal solution y of problem SI(E' , S , l , u), which belongs to class SC/LUB/D. Since the problem SI(E' , S , l , u) is a relaxation of problem SM(E' , S , l , u), the y is optimal to SM(E' , S , l , u) if it is feasible to SM(E' , S , l , u). \square

Theorem 6 *Procedure SM correctly computes an optimal solution of SC/SM/D.*

Proof Consider the computation of DA(E', \emptyset, l, u). If $v < y$, E' is partitioned into E'_1 and E'_2 , and the original problem is accordingly decomposed into two subproblems. This decomposition is recursively applied, if necessary, and the entire process is represented by a decomposition tree T_E , where $E = \{1, 2, \dots, n\}$. The proof is done by the induction on the height of T_E . At a leaf of T_E , the corresponding subset E' is no more decomposed because the condition of Step 3 holds. Thus, due to Lemma 10, the subproblem can be correctly solved at the leaf. Consider an intermediate node of T_E , and let SM(E' , S , l , u) denote the problem considered at the node. Let Q'_1 and Q'_2 denote the subproblems of SC/SM/D considered in Steps 5 and 6 of DA(E' , S , l , u), respectively. By induction hypothesis, assume that problems Q'_1 and Q'_2 are correctly solved, and let $\{x_j^* \mid j \in E'_1\}$ (resp.,

$\{x_j^* \mid j \in E'_2\}$) denote the obtained optimal solution of Q'_1 (resp., Q'_2). It will be proven that $\{x_j^* \mid j \in E' (= E'_1 \cup E'_2)\}$ is an optimal solution of $\text{SM}(E', S, l, u)$.

From [Theorem 4](#), it suffices to show

$$d_j(x_j^* + 1) \geq d_i(x_i^*) \quad (54)$$

for all $i, j \in E'$ with $x^* - e(i) + e(j) \in B_0(r)$. Let $x' = x^* - e(i) + e(j)$. If $i, j \in E'_1$ or $i, j \in E'_2$, the inequality (54) clearly holds from the induction hypothesis. If $i \in E'_2$ and $j \in E'_1$, it holds that

$$x'(E'_1) > x^*(E'_1) = r(E'_1) - r(S)$$

(the right equality follows from the constraint of (52)), violating the feasibility of Q'_2 . Thus, consider the case of $i \in E'_1$ and $j \in E'_2$. From the way of decomposition used in Steps 5 and 6,

$$x_i^* \leq y_i \text{ and } x_j^* \geq y_j \quad (55)$$

hold. Since y is an optimal solution of the simple resource allocation problem $\text{SI}(E', S, l, u)$,

$$d_i(y_i) \leq d_j(y_j + 1) \quad (56)$$

holds from [Theorem 1](#). Then from the convexity of f_i and f_j , and from (55), it follows that

$$d_i(x_i^*) \leq d_j(x_j^* + 1).$$

This completes the proof. \square

Theorem 7 *The running time of procedure SM is polynomial in $|E|$ and $\log r(E)$. The time complexity is $O(\tau_1|E| + \tau_2|E|^2 \log r(E) + \tau_3|E|)$ time, where τ_1 , τ_2 , and τ_3 denote the running times to solve $\text{SI}(E', S, l, u)$, to test $x \in P(r)$ for a given $x \in R^E$, and to compute $\text{sat}(X)$ for a given $X \subseteq E$, respectively.*

Proof The running time of each execution of procedure $\text{DA}(E', S, l, u)$ is first evaluated. Step 1 can be solved in τ_1 time. Computation of a maximal vector v in Step 2 is done by iteratively increasing the j -th component of v by d_j^{\max} , $j \in E$, where

$$d_j^{\max} = \max\{d \mid v + d \cdot e(j) \in P(r)\}.$$

The computation of this d_j^{\max} can be done by binary search (i.e., in $O(\log r(E'))$ iterations), and testing whether $v + d \cdot e(j) \in P(r)$ or not can be done in τ_2 time. Therefore, Step 2 requires $O(\tau_2|E'| \log r(E'))$ time. Finally, $\text{sat}(v)$ is computed in τ_3 time. These computations clearly dominate all the other parts of DA, showing that the time to execute all steps of $\text{DA}(E', S, l, u)$ once is $O(\tau_1 + \tau_2|E'| \log r(E') + \tau_3)$. Since the number of vertices in the decomposition tree T_E is $O(|E|)$, DA is called

$O(|E|)$ times. Also, by $|E'| \leq |E|$ and $r(E') \leq r(E)$, the stated time complexity of SM follows.

From [Theorem 3](#), τ_1 is polynomial in $|E|$ and $\log r(E)$. Also, from [Lemma 5](#), τ_2 and τ_3 are polynomial in $|E|$ and $\log r(E)$. Therefore, the running time of procedure SM is polynomial in $|E|$ and $\log r(E)$. \square

3.3 SC/GUB/D, SC/Nested/D, SC/Tree/D, and SC/Network/D

It shall be briefly explained how separable convex resource allocation problems under generalized upper bounds (GUB), nested (Nested), tree (Tree), and network (Network) constraints can be solved efficiently, when the decomposition algorithm SM in the previous subsection is specialized to these cases. Since GUB, Nested, and Tree are special cases of Network, only problem SC/Network/D shall be considered.

Call a vector $\{x_t(\varphi) | x \in T\}$ for a feasible flow φ of (11)–(15) an *inflow vector*. For $X \in T$, define

$$w(X) = \max \left\{ \sum_{t \in X} x_t(\varphi) \mid \{x_t(\varphi) | x \in T\} \text{ is an inflow vector} \right\}. \quad (57)$$

Then, it is easy to see from the well-known max-flow min-cut theorem [40] that

$$w(X) = \min\{c(U, V - U) \mid s \in U, X \subseteq V - U\} \quad (58)$$

holds, where $c(U, V - U) = \sum_{u \in U, v \in V - U} c(u, v)$ represents the capacity of a cut (i.e., a partition) $(U, V - U)$, separating X from the sources. It is also well known in the network theory (see [68]) that the submodular propFerty holds for $w(X)$, that is,

$$W(X) + W(Y) \geq W(X \cup Y) + W(X \cap Y), \quad \text{for } X, Y \in T.$$

Then, $\{x_t(\varphi) \mid x \in T\}$ is an inflow vector of some feasible flow φ if and only if

$$x(X) \leq w(X) \quad \text{for all } X \subseteq T,$$

and

$$x(T) = N.$$

(See, e.g., [69] for the proof.)

Now, the running time of procedure SM is analyzed when procedure SM is specialized to SC/Network/D. Step 1 of DA can be executed by solving a problem of SC/Simple/D. For the obtained optimal solution y , Step 2 computes a maximal vector $v \leq y$. In the case of SC/Network/D, this is done by introducing a supersink t^* as well as arcs (t, t^*) for all $t \in T$ with capacity $c(t, t^*) = y_t$ and finding a maximum flow φ^* from s to t^* in the modified network. (To be more precise, while constructing the modified network, the lower bound on the flow value of the arc

(t, t^*) should be introduced in order to reflect the constraints implied by a vector l .) Then a vector

$$\{v_t = \varphi^*(t, t^*) \mid t \in T\}$$

is obtained. Thus, the v can be computed by applying an appropriate max-flow algorithm. The time required for other steps is dominated by that of Steps 1 and 2. Thus, the following theorem is obtained.

Theorem 8 *Problem SC/Network/D can be solved in $O(|T|(\tau(n, m, C_{\max}) + |T| \log(N/|T|)))$ time, where $\tau(n, m, C_{\max})$ denotes the running time for the maximum flow algorithm for a network with n vertices, m arcs, and the maximum arc capacity C_{\max} .*

Proof Since SC/Network/D is a special case of SC/SM/D, the running time is analyzed by following the proof of [Theorem 7](#). Since a maximum flow algorithm requires $\Omega(n)$ time, $|T| \leq \tau(n, m, C_{\max})$ clearly holds. The computation of vector y of Step 1 in procedure DA reduces to solving problem SC/Simple/D, which can be done in $O(|T| \log(N/|T|))$ time by using Frederickson and Johnson's algorithm [42]. The computation of vector $v \leq y$ of Step 2 is done by computing a maximum flow, which requires $\tau(n, m, C_{\max})$ time. These computations clearly dominate all the rest of DA. Thus, the theorem follows from [Theorem 7](#) (recall that $|E| = |T|$ and $r(E) = N$ hold for this case). \square

Existing max-flow algorithms run in time $\tau(n, m, C_{\max}) = O(nm \log(n^2/m))$, $O(n^3/\log n)$, and $O(n^{2/3}m \log(n^2/m) \log C_{\max})$ [124]. Since the maximum flow for a tree network can be computed in $O(m)$ time (see [69]) and $m = O(n)$ holds for a tree, the following corollary holds.

Corollary 1 *Problems SC/GUB/D, SC/Nested/D, and SC/Tree/D can be solved in $O(n^2 \log(N/n))$ time.* \square

Faster algorithms have been developed for SC/GUB/D, SC/Nested/D, and SC/Tree/D. Hochbaum [61] showed that SC/GUB/D can be solved in the same running time as SC/Simple/D by using the observation that every instance of SC/GUB/D can be reduced to a family of *disjoint* instances of SC/Simple/D. This implies, in particular, that SC/GUB/D can be solved in $O(n \log(N/n))$ time by using the algorithm of Frederickson and Johnson [42] repeatedly.

For SC/Nested/D, Dyer and Walker's algorithm [35] runs in $O(n \log n \log^2(N/n))$ time, and for SC/Tree/D, Dyer and Frieze [34] developed an $O(n \log^2 n \log N)$ time algorithm. Hochbaum [61] further improved the running time of these two algorithms to $O(n \log n \log(N/n))$. The idea of the improvement is based on a general proximity theorem between SC/SM/D and its *scaled* version. Proximity theorems and the resulting efficient algorithms are described in [Sects. 4](#) and [5](#).

3.4 Minimax and Maximin Problems

It will be shown here that minimax and maximin resource allocation problems, Minimax/-/D and Maximin/-/D, are equivalently transformed into problems of SC/-/D, where \cdot denotes any type of constraints defined in Sect. 2.2. Therefore, equally efficient algorithms can be developed for minimax and maximin problems. It suffices to show this fact for the most general case, that is, Minimax/SM/D and Maximin/SM/D, which are described as follows:

$$\begin{aligned} \text{MINIMAX: } & \text{minimize } \max_{j \in E} f_j(x_j), \\ & \text{subject to } x : \text{an integral base of } B_0(r), \end{aligned} \quad (59)$$

$$\begin{aligned} \text{MAXIMIN: } & \text{maximize } \min_{j \in E} f_j(x_j), \\ & \text{subject to } x : \text{an integral base of } B_0(r). \end{aligned} \quad (60)$$

Here all f_j , $j \in E$, are assumed to be nondecreasing. When all f_j are nonincreasing, problems MINIMAX and MAXIMIN are mutually transformed into MAXIMIN and MINIMAX, respectively, by the following identities:

$$\begin{aligned} -\min_x \max_{j \in E} f_j(x_j) &= \max_x \min_{j \in E} -f_j(x_j), \\ -\max_x \min_{j \in E} f_j(x_j) &= \min_x \max_{j \in E} -f_j(x_j). \end{aligned}$$

Define for $j \in E$

$$\begin{aligned} g_j(x_j) &= \sum_{y=0}^{x_j} f_j(y), \quad x_j = 0, 1, 2, \dots, \\ h_j(x_j) &= \sum_{y=-1}^{x_j} f_j(y), \quad x_j = 0, 1, 2, \dots, \end{aligned} \quad (61)$$

where $f_j(-1) = f_j(0)$ is assumed. Note that

$$\begin{aligned} g_j(x_j) - g_j(x_j - 1) &= f_j(x_j), \\ h_j(x_j) - h_j(x_j - 1) &= f_j(x_j - 1) \end{aligned} \quad (62)$$

hold for each $x_j = 0, 1, \dots$. From the nondecreasingness of f_j , it follows that g_j and h_j are convex over the nonnegative integers. Now consider the following problems of SC/SM/D:

$$Q_g : \min \left\{ \sum_{j \in E} g_j(x_j) \mid x : \text{an integral base of } B_0(r) \right\}, \quad (63)$$

$$\mathcal{Q}_h : \min \left\{ \sum_{j \in E} h_j(x_j) \mid x : \text{an integral base of } B_0(r) \right\}. \quad (64)$$

Theorem 9 Every optimal solution of problem \mathcal{Q}_g (resp., \mathcal{Q}_h) is optimal to MINIMAX (resp., MAXMIN).

Proof Only the proof for \mathcal{Q}_g is given (the case of \mathcal{Q}_h can be similarly treated). Let x^g be an optimal solution of \mathcal{Q}_g . By Theorem 4 and property $f_j(x_j) = g_j(x_j) - g_j(x_j - 1)$ by (62), it holds

$$f_j(x_j^g + 1) \geq f_i(x_i^g) \quad (65)$$

for all $i, j \in E$ with $x^g - e(i) + e(j) \in B_0(r)$. Suppose that x^g is not optimal to MINIMAX, and let x^* be an optimal solution such that $\|x^g - x^*\| (= \sum_{j \in E} |x_j^g - x_j^*|)$ is minimum. By (39) with $x = x^g$ and $y = x^*$, it holds that

$$f_i(x_i^g) - f_i(x_i^* + 1) + f_j(x_j^*) - f_j(x_j^g + 1) \geq 0 \quad (66)$$

for any i, j with $x_i^g > x_i^*$, $x_j^g < x_j^*$ such that $x^g - e(i) + e(j), x^* + e(i) - e(j) \in B_0(r)$. Hence, together with (65), it holds that

$$f_j(x_j^*) - f_i(x_i^* + 1) \geq 0. \quad (67)$$

Let $\hat{x} = x^* + e(i) - e(j)$. Since $f_j(x_j)$ is nondecreasing, $f_j(\hat{x}_j) = f_j(x_j^* - 1) \leq f_j(x_j^*)$ holds. In addition, it holds that

$$f_j(x_j^*) \geq f_i(x_i^* + 1) = f_i(\hat{x}_i). \quad (68)$$

Since $f_k(\hat{x}_k) = f_k(x_k^*)$ for $k \neq i, j$, it follows

$$\max_{j \in E} f_j(x_j^*) \geq \max_{j \in E} f_j(\hat{x}_j).$$

This implies that \hat{x} is also an optimal solution to MINIMAX, contradicting the minimality of $\|x^* - x^g\|$. \square

3.5 Notes and References

Most of the proofs of fundamental properties of a submodular system can be found in [45, 69]. The proof of Lemma 4 (ii) is due to Murota [111] although this property has been known as a folklore among researchers in this field. Also, Murota [111]

has shown that the condition of [Lemma 4](#) (ii) is a necessary and sufficient condition that a set $B \subseteq R^E$ is a base polyhedron of a submodular system.

Incremental and decomposition algorithms of [Sects. 3.2.1](#) and [3.2.2](#) are due to Federgruen and Groenevelt [39] and Groenevelt [53]. The original idea of the decomposition algorithm is due to Fujishige [43] who considered a special case of SC/SM/C in which $f_j(x_j) = w_j x_j^2$ ($w_j > 0$) holds for each j . The correctness proof given here is based on [Theorem 4](#), which is due to Murota [109]. [Theorem 4](#) helps simplify the proofs given in the original papers. The proof of [Theorem 9](#) for MINIMAX is also based on the result of [109] for M -convex functions.

The incremental algorithm for SC/SM/D was extended to more general constraints, for example, to the separable convex resource allocation problem for a bisubmodular system and for a finite jump system by Ando, Fujishige, and Naitoh (see papers [4, 5, 44] for the details). Notice that a finite jump system is a generalization of a bisubmodular system. The existence of a polynomial-time algorithm similar to SM of [Sect. 3.2.2](#) was known for a bisubmodular system [44], and for a finite jump system, a polynomial-time algorithm was proposed by Shioura and Tanaka [131].

Resource allocation problem with continuous variables has also been extensively studied, although the details are omitted in this chapter.

The decomposition algorithm of [Sect. 3.2.2](#) works also for the case of continuous variables, that is, problem SC/SM/C, with a minor modification in Step 1 of procedure DA that problem SC/Simple/C is solved instead of SC/Simple/D (see the book by Fujishige [45]). Fujishige [43, 45] showed the equivalence between Lexico-Minimax/SM/C and SC/SM/C in a sense similar to [Theorem 9](#). Also, it has been shown in [43, 45] that an optimal solution of Minimax of (59) obtained through problem Q_g of (63) is lexicographically minimax.

For general S/Simple/D with nonconvex objective function, a parallel algorithm has been proposed by Shao and Rao [127], and a metaheuristic algorithm based on ant-colony optimization was proposed by Yin and Wang [146].

Minimax/Simple/C and Minimax/Simple/D have been further studied in [21, 22, 37, 75, 91, 100, 147] since the book by Ibaraki and Katoh [69] has been published. In particular, Luss [100] studied lexicographic minimax approach while mentioning several application fields including production planning and scheduling, workforce management, allocation of emergency services, distribution of strategic resource, and telecommunications network design.

Fujishige, Katoh, and Ichimori [46] proposed an efficient algorithm for Fair/SM/D by making use of the proximity result that relates an optimal solution of Fair/SM/D and optimal solutions of Minimax/SM/D and Maximin/SM/D. Namikawa and Ibaraki [112] also derived an efficient algorithm for Fair/SM/D using the continuous relaxation.

The separable convex quadratic resource allocation problem with continuous variables has been well studied, where, instead of the constraint $\sum_{j=1}^n x_j = N$, the following knapsack constraint is considered:

$$\sum_{j=1}^n a_j x_j \leq N, \quad (69)$$

where all a_j are nonnegative. This problem is called the *quadratic knapsack problem*. Notice that the problem with this constraint can be transformed into SQC/Simple/C because, due to continuous variables, introducing new variables $x'_j = a_j x_j$ and a slack variable s turns the above knapsack constraint into the equality constraint of the form $\sum_{j=1}^n x_j + s = N$ without increasing the computational difficulty. The LUB constraint can also be easily incorporated without increasing the difficulty. For this problem, Brucker [23] developed a linear-time algorithm. Later, Nielsen and Zenios [114], Pardalos and Kovoov [118], Pardalos, Ye, and Han [119], and Robinson, Jiang, and Lerme [123] further studied this problem. Problem SQC/GUB/C has been also studied by Brethauer and Shetty [15] who performed extensive computational experiments for various problem instances with up to hundred thousand variables and showed that their algorithm is extremely efficient compared with a general-purpose software for nonlinear programs. Megiddo and Tamir [105] developed a linear-time algorithm for the separable convex quadratic resource allocation problem with more general constraints. Problem SQC/-/C will be utilized to solve SQC/-/D in Sect. 5.3.

4 Proximity Theorems

Proximity theorem refers to a theoretical bound on the distance between two optimal solutions to an optimization problem and to its related (relaxed or approximated) problem. In this section, proximity theorems are presented for the two types of problems. The first theorem deals with SC/Linear/D, due to Hochbaum and Shanthikumar [65], where Linear is defined by linear inequalities $Ax \geq b$ for $m \times n$ integer matrix A , and the second deals with SC/SM/D, due to Hochbaum [61], Moriguchi and Shioura [107], and Moriguchi, Shioura, and Tsuchimura [108].

The first proximity result is described in terms of the maximum absolute value of the subdeterminants of the constraint matrix A , denoted by Δ . If $\Delta = 1$ (i.e., constraint matrix is *totally unimodular*), the theorem leads to a polynomial-time algorithm for the problem with integer variables. Thus, this result identifies a new class of resource allocation problems with integer variables for which a polynomial-time algorithm can be developed. Notice that a system of linear inequalities defined by a totally unimodular matrix is different from the one defined by submodular constraints, because the constraint matrix defined by submodular functions (i.e., (5)) is *totally dual integral* but is not totally unimodular (see, e.g., [36, 67]).

From the second proximity theorem for SC/SM/D, Hochbaum [61] developed efficient algorithms for SC/GUB/D, SC/Nested/D, and SC/Tree/D, improving the running time of the algorithms given in Sect. 3.2. The algorithms use the

so-called scaling techniques in which the scaled versions of incremental algorithms are iteratively applied. These algorithms will be described in [Sect. 5](#).

4.1 Proximity Theorem for General Linear Constraints

For simplicity, denote

$$f(x) = \sum_{j=1}^n f_j(x_j), \quad (70)$$

where all $f_j : R \rightarrow R$, $j = 1, \dots, n$ are convex. Consider the following problem,

$$\begin{aligned} \text{IP : minimize } & f(z) \\ \text{subject to } & Az \geq b, \\ & z \geq 0, \\ & z : \text{integer}, \end{aligned} \quad (71)$$

as well as its continuous version,

$$\begin{aligned} \text{RP : minimize } & f(x) \\ \text{subject to } & Ax \geq b, \\ & x \geq 0. \end{aligned} \quad (72)$$

Here, A is an $m \times n$ integer matrix and b is an m -dimensional integer vector. It is assumed that the polyhedron defined by $\{x \mid Ax \geq b, x \in R^n\}$ is bounded, and thus, there exists an optimal solution for each of IP and RP. For a scaling parameter s (a positive integer), let the scaled problem IP/ s be defined by

$$\begin{aligned} \text{IP}/s : \text{minimize } & f(sy) \\ \text{subject to } & Ay \geq b/s, \\ & y \geq 0, \\ & y : \text{integer}. \end{aligned} \quad (73)$$

In problem IP/ s , the solution $x = sy$ is an integer multiple of s . Although RP and a continuous version of IP/ s are the same, it is needed to define the linearized functions $f_j^{L:s}$ and $f^{L:s}$ for s .

$f_j^{L:s} : R \rightarrow R$, a piecewise linear convex function such that

$$f_j^{L:s}(sy_j) = f_j(sy_j) \text{ holds for all integers } y_j,$$

$$f^{L:s}(x) = \sum_{j=1}^n f_j^{L:s}(x_j). \quad (74)$$

Then an optimal solution to the following problem,

$$\begin{aligned} \text{IP}'/s : & \text{minimize } f^{L:s}(sy) \\ & \text{subject to } Ay \geq b/s, \\ & y \geq 0, \\ & y : \text{integer}, \end{aligned} \tag{75}$$

is optimal to IP/s, since $f^{L:s}(sy) = f(sy)$ holds for all integer vectors y . Define also the following continuous problem:

$$\begin{aligned} \text{LP}/s : & \text{minimize } f^{L:s}(x) \\ & \text{subject to } Ax \geq b, \\ & x \geq 0. \end{aligned} \tag{76}$$

Now assume, without loss of generality (see [117]), that lower and upper bounds on variables are available such that an optimal solution exists in the interval given by the bounds, that is,

$$l_j \leq x_j \leq u_j, \quad j = 1, 2, \dots, n.$$

Remark 1 LP/s can be formulated as a linear programming problem by introducing N_j continuous variables for each j corresponding to the line segments of the piecewise linear convex function $f_j^{L:s}(sy_j)$, where $N_j = (u_j - l_j)/s$ (see [33]).

Use the notation $x \vee y$ and $x \wedge y$ for $x, y \in R^n$ to denote

$$\begin{aligned} x \vee y &= (\max\{x_1, y_1\}, \dots, \max\{x_n, y_n\}), \\ x \wedge y &= (\min\{x_1, y_1\}, \dots, \min\{x_n, y_n\}). \end{aligned}$$

Lemma 11 *For the function f of (70) and for any $x, x', y, y' \in R^n$ satisfying*

- (i) $x \wedge x' \leq y \leq x \vee x'$,
 - (ii) $x \wedge x' \leq y' \leq x \vee x'$,
 - (iii) $x + x' = y + y'$,
- it holds that*

$$f(x) + f(x') \geq f(y) + f(y').$$

Proof Let $z = x \wedge x'$ and $z' = x \vee x'$. Since either $x_j = z_j$, $x'_j = z'_j$ or $x_j = z'_j$, $x'_j = z_j$ holds for all j from the definition of z and z' ,

$$z_j + z'_j = x_j + x'_j, \tag{77}$$

$$f_j(z_j) + f_j(z'_j) = f_j(x_j) + f_j(x'_j) \tag{78}$$

hold for all j . Then from (i), (ii), (iii), and (77), it follows that $z_j \leq y_j \leq z'_j$, $z_j \leq y'_j \leq z'_j$ and $z_j + z'_j = y_j + y'_j$ for all j . From the convexity of f_j , it is easy to see

$$f_j(z_j) + f_j(z'_j) \geq f_j(y_j) + f_j(y'_j),$$

and hence,

$$f(z) + f(z') \geq f(y) + f(y').$$

Since $f(z) + f(z') = f(x) + f(x')$ holds from (78), the lemma follows. \square

Now let Δ be defined as above, and

$$\|x\|_\infty = \max_{1 \leq j \leq n} |x_j|$$

be the L_∞ norm of $x \in R^n$. The proximity theorem between IP and RP is stated as follows.

Theorem 10 (i) For every optimal solution \hat{x} of RP, there exists an optimal solution z^* of IP such that $\|\hat{x} - z^*\|_\infty \leq n\Delta$.
(ii) For every optimal solution \hat{z} of IP, there exists an optimal solution x^* of RP such that $\|\hat{z} - x^*\|_\infty \leq n\Delta$.

Proof First note that both IP and RP have optimal solutions as assumed at the beginning of Sect. 4.1. First, a cone with respect to \hat{z} and \hat{x} is defined. Let (S_1, S_2) be the partition of $\{1, \dots, n\}$ such that $\hat{z}_j \geq \hat{x}_j$ for all $j \in S_1$ and $\hat{z}_j < \hat{x}_j$ for all $j \in S_2$. Also, partition the matrix A into submatrices

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}$$

such that $A_1\hat{z} < A_1\hat{x}$ and $A_2\hat{z} \geq A_2\hat{x}$. Define the cone $C = \{y \in R^n \mid A_1y \leq 0, A_2y > 0, y_j \geq 0, j \in S_1, y_j \leq 0, j \in S_2\}$. Let $U \subset C$ be a finite set of integer vectors that generate C . Then, because of the integrality of the components of A ,

$$\|u\|_\infty \leq \Delta$$

holds for all $u \in U$, and since $\hat{z} - \hat{x} \in C$, there exist t ($\leq n$) vectors $u^{(i)} \in U$ and $\alpha_i \geq 0$, $i = 1, \dots, t$ such that $\hat{z} - \hat{x} = \sum_{i=1}^t \alpha_i u^{(i)}$ (see [31]), that is,

$$\hat{z} = \hat{x} + \sum_{i=1}^t \alpha_i u^{(i)}. \quad (79)$$

Let

$$\beta_i = \alpha_i - \lfloor \alpha_i \rfloor, \quad i = 1, \dots, t$$

and define

$$z^* = \hat{x} + \sum_{i=1}^t \beta_i u^{(i)} \quad (80)$$

and

$$x^* = \hat{z} - \sum_{i=1}^t \beta_i u^{(i)}. \quad (81)$$

From (79), (80), and (81), it follows that

$$z^* = \hat{z} - \sum_{i=1}^t (\alpha_i - \beta_i) u^{(i)}, \quad (82)$$

$$x^* = \hat{x} + \sum_{i=1}^t (\alpha_i - \beta_i) u^{(i)}. \quad (83)$$

Now, it is shown that these z^* and x^* are feasible for IP and RP, respectively. Observe from (82) that

$$A_1 z^* = A_1 \hat{z} - \sum_{i=1}^t (\alpha_i - \beta_i) A_1 u^{(i)} \geq A_1 \hat{z}, \quad (84)$$

since $\alpha_i - \beta_i \geq 0$ and $u^{(i)} \in C$ (i.e., $A_1 u^{(i)} \leq 0, i = 1, \dots, t$). Similarly, from (80),

$$A_2 z^* = A_2 \hat{x} + \sum_{i=1}^t \beta_i A_2 u^{(i)} \geq A_2 \hat{x}. \quad (85)$$

Hence, $Az^* \geq b$ holds from $A\hat{z} \geq b$ and $A\hat{x} \geq b$. Since $\alpha_i - \beta_i = \lfloor \alpha_i \rfloor$ and $u^{(i)}$ are integral, the integrality of z^* follows from (82). Analogously,

$$A_1 x^* = A_1 \hat{z} - \sum_{i=1}^t \beta_i A_1 u^{(i)} \geq A_1 \hat{z}, \quad (86)$$

$$A_2 x^* = A_2 \hat{x} + \sum_{i=1}^t (\alpha_i - \beta_i) A_2 u^{(i)} \geq A_2 \hat{x}. \quad (87)$$

Hence, $Ax^* \geq b$. Since $u^{(i)} \in C$, it holds that $u_j^{(i)} \geq 0$ for $j \in S_1$ and $u_j^{(i)} \leq 0$ for $j \in S_2$. Note also that $\hat{z}_j \geq \hat{x}_j$ for $j \in S_1$ and $\hat{z}_j < \hat{x}_j$ for $j \in S_2$. Hence, it follows from (80) and (82) that

$$\begin{aligned} \hat{z}_j &\geq z_j^* \geq \hat{x}_j, \quad j \in S_1, \\ \hat{z}_j &\leq z_j^* \leq \hat{x}_j, \quad j \in S_2, \end{aligned} \quad (88)$$

and similarly from (81) and (83) that

$$\begin{aligned}\hat{z}_j &\geq x_j^* \geq \hat{x}_j, \quad j \in S_1, \\ \hat{z}_j &\leq x_j^* \leq \hat{x}_j, \quad j \in S_2.\end{aligned}\tag{89}$$

The inequalities (88) and (89) imply

$$\hat{z} \wedge \hat{x} \leq z^* \leq \hat{z} \vee \hat{x}\tag{90}$$

and

$$\hat{z} \wedge \hat{x} \leq x^* \leq \hat{z} \vee \hat{x}.\tag{91}$$

Furthermore, (80) and (81) imply

$$\hat{z} + \hat{x} = z^* + x^*.\tag{92}$$

It then follows from (90), (91), (92), and Lemma 11 that

$$f(\hat{z}) + f(\hat{x}) \geq f(z^*) + f(x^*).\tag{93}$$

Therefore,

$$f(z^*) \leq f(\hat{z}) + f(\hat{x}) - f(x^*) \leq f(\hat{z}),$$

since \hat{x} is an optimal solution of RP and x^* is feasible to RP. Hence, z^* is also optimal to IP and $f(z^*) = f(\hat{z})$ holds. Therefore, $f(\hat{x}) = f(x^*)$ follows, and x^* is also optimal to RP. Finally, by (80) and (81), it holds that

$$\begin{aligned}\|\hat{x} - z^*\| &= \left\| \sum_{i=1}^t \beta_i u^{(i)} \right\|_\infty \leq n\Delta, \\ \|\hat{z} - x^*\| &= \left\| \sum_{i=1}^t \beta_i u^{(i)} \right\|_\infty \leq n\Delta.\end{aligned}$$

□

The proofs for the following proximity theorems, between IP/s and IP, and between IP and LP/s, can also been done by using the arguments similar to the above theorem. These will be used to develop an algorithm in the next subsection.

- Theorem 11** (i) For every optimal solution \hat{y} of IP/s, there exists an optimal solution z^* of IP such that $\|s\hat{y} - z^*\|_\infty \leq ns\Delta$.
(ii) For every optimal solution \hat{z} of IP, there exists an optimal solution y^* of IP/s such that $\|\hat{z} - sy^*\|_\infty \leq ns\Delta$. □

- Theorem 12** For every optimal solution \hat{x} of LP/s, there exists an optimal solution z^* of IP such that

$$\|\hat{x} - z^*\|_\infty \leq 2ns\Delta. \quad (94)$$

Proof Since LP/s is a continuous relaxation of IP'/s with $x = sy$, it follows from [Theorem 10](#) that there exists an optimal solution y^* of IP'/s (and hence of IP/s) such that $\|\hat{x}/s - y^*\|_\infty \leq n\Delta$. From [Theorem 11](#), there exists an optimal solution z^* of IP such that $\|sy^* - z^*\|_\infty \leq ns\Delta$. Combining these proximity results implies (94). \square

4.2 Algorithm for Problem IP

As stated at the beginning of [Sect. 4.1](#), it is assumed that the polyhedron defined by the $\{x \mid Ax \geq b, x \in R^n\}$ is bounded. This implies that there exists a positive integer γ such that an optimal solution of the problem,

$$\begin{aligned} \text{IP}^{(1)} : & \text{minimize } f(z) \\ & \text{subject to } Az \geq b, \\ & \quad -2^{\gamma+1}n\Delta e \leq z \leq 2^{\gamma+1}n\Delta e, \\ & \quad z : \text{integer}, \end{aligned}$$

where $e = (1, \dots, 1)^T$, is also optimal to IP (see the book [117] for the proof of the existence of such γ). In particular, it can be chosen that

$$\gamma = \lceil \log_2((m/n)\|b\|_\infty) \rceil - 1.$$

Now the algorithm for IP is described as follows:

Algorithm SOLVE_IP

Input: An instance of problem IP.

Output: An optimal solution \hat{z} .

Step 1: Let $s_k := 2^{\gamma-k}$, $k = 0, \dots, \gamma$, and $\hat{x}^{(0)} := 0$.

Step 2: For each of $k = 1, \dots, \gamma$, compute an optimal solution $\hat{x}^{(k)}$ of the following problem:

$$\begin{aligned} \text{LP}^{(k)}/s_k : & \text{minimize } f^{L:s_k}(x) \\ & \text{subject to } Ax \geq b, \\ & \quad \hat{x}^{(k-1)} - 2ns_{k-1}\Delta e \leq x \leq \hat{x}^{(k-1)} + 2ns_{k-1}\Delta e. \end{aligned}$$

Step 3: Obtain an optimal solution \hat{z} by solving the following integer program with lower bound $L_j = x_j^{(\gamma)} - 2n\Delta$ and upper bound $U_j = x_j^{(\gamma)} + 2n\Delta$ on z_j :

$$\begin{aligned} \text{IP}^* : & \text{minimize } f(z) \\ & \text{subject to } Az \geq b, \\ & \quad \hat{x}^{(\gamma)} - 2n\Delta e \leq z \leq \hat{x}^{(\gamma)} + 2n\Delta e \\ & \quad z : \text{integer}. \end{aligned}$$

Output \hat{z} as an optimal solution of IP.

Let $T(n, m, \Delta)$ denote the running time of a linear programming algorithm for solving $\min\{cx \mid Ax \geq b, 0 \leq x_j \leq 1 \text{ for } j = 1, 2, \dots, n\}$, where Δ denotes the maximum absolute value of the subdeterminants of A , and let $TI(n, m, A)$ denote the running time to solve the integer linear programming problem $\min\{cx \mid Ax \geq b, 0 \leq x_j \leq 1 \text{ for } j = 1, 2, \dots, n, x : \text{integer}\}$. Let A^k denote the $m \times kn$ matrix in which each column of A appears k times.

Theorem 13 *The solution \hat{z} obtained by Algorithm SOLVE_IP is an optimal solution of problem IP. The running time of SOLVE_IP is*

$$\log_2((m/n)\|b\|_\infty) \cdot T(8n^2\Delta, m, \Delta) + TI(4n^2\Delta, m, A^{4n\Delta}).$$

Proof Consider the problem $LP^{(1)}/s_1$ with $s_1 = 2^{\gamma-1}$, which is solved in Step 1 for $k = 1$. This $LP^{(1)}/s_1$ is a relaxation of $IP^{(1)}$ since $f_j^{L,s_1}(x)$ is a piecewise linearization of the original function $f(x)$. By [Theorem 12](#), there exists an optimal solution $z^{(1)}$ of $IP^{(1)}$ such that

$$\|\hat{x}^{(1)} - z^{(1)}\|_\infty \leq 2ns_1\Delta = 2^\gamma n\Delta.$$

Therefore, an optimal solution of the integer program,

$$\begin{aligned} IP^{(2)} : & \text{minimize } f(z) \\ & \text{subject to } Az \geq b, \\ & \hat{x}^{(1)} - 2^\gamma n\Delta e \leq z \leq \hat{x}^{(1)} + 2^\gamma n\Delta e \\ & z : \text{integer}, \end{aligned}$$

is also optimal to $IP^{(1)}$ (and hence of IP). Observe that problem $LP^{(2)}/s_2$ with $s_2 = 2^{\gamma-2}$ is a linearized relaxation of $IP^{(2)}$. Hence, again from [Theorem 12](#), it follows that there exists an optimal solution $z^{(2)}$ of $IP^{(2)}$ (and hence of IP) such that

$$\|\hat{x}^{(2)} - z^{(2)}\|_\infty \leq 2ns_2\Delta = 2^{\gamma-1}n\Delta.$$

Repeating this argument, it can be seen that, for each $k = 1, \dots, \gamma$, there exists an optimal solution $z^{(k)}$ of IP such that

$$\|\hat{x}^{(k)} - z^{(k)}\|_\infty \leq 2^{\gamma+1-k}n\Delta.$$

Therefore, for each $k = 1, \dots, \gamma$, an optimal solution of $IP^{(k)}$ (defined similarly to $IP^{(2)}$) is also optimal to IP. Since [Theorem 12](#) tells that there exists an optimal solution \hat{z} of IP such that $\|\hat{x}^{(\gamma)} - \hat{z}\|_\infty \leq 2n\Delta$, the correctness of Algorithm SOLVE_IP follows.

As for the time complexity, Step 2 is repeated γ times. Here,

$$\gamma = \lceil \log_2((m/n)\|b\|_\infty) \rceil - 1 < \log_2((m/n)\|b\|_\infty).$$

At every iteration, each variable x_j is replaced by $8n\Delta$ variables (see Remark 1) because x_j is in an interval of length $4ns_{k-1}\Delta$ from definition of $\text{LP}^{(k)}/s_k$, and the number of variables used for linearization is $4ns_{k-1}\Delta/s_k = 8n\Delta$. Thus, the time complexity for solving the corresponding linear program is $T(8n^2\Delta, m, \Delta)$. In order to solve the problem IP^* in Step 3, it is transformed into an integer linear program first. For this transformation, $4n\Delta$ 0-1 variables are needed since x_j for all j is in an interval of length $4n\Delta$. Thus, the number of resulting 0-1 variables is $4n^2\Delta$, and the matrix A has each column duplicated $4n\Delta$ times. Hence, Step 3 requires $TI(4n^2\Delta, m, A^{4n\Delta})$ time. \square

Corollary 2 *If A is totally unimodular, problem IP can be solved in polynomial time. The time complexity is $\log_2((m/n)\|b\|_\infty) \cdot T(8n^2, m, 1)$.*

Proof If A is totally unimodular, problem IP^* can be solved by applying a polynomial-time algorithm for the linear program. The theorem then immediately follows from $\Delta = 1$. \square

4.3 Proximity Theorem for Submodular Constraints

As remarked at the beginning of this section, the constraint matrix defining the base polyhedron of a submodular system is not totally unimodular. Thus, Algorithm `SOLVE_IP` in the previous subsection cannot be applied directly to SC/SM/D in order to obtain a polynomial-time algorithm. However, as will be shown below, stronger proximity results for this case can be established.

Recall the formulation of the problem SC/SM/D.

$$\begin{aligned} Q : & \text{minimize } \sum_{j \in E} f_j(x_j), \\ & \text{subject to } x : \text{ an integer base of } B(r). \end{aligned} \tag{95}$$

Let x^* denote an optimal solution of Q . For a positive integer s , the following algorithm `SM-INCREMENT(s)` obtained by modifying an incremental algorithm of Sect. 3.2.1 produces a feasible solution $x^{(s)}$ of Q . This $x^{(s)}$ is then used to establish a strong proximity theorem. This algorithm will be used as a subroutine in the algorithm presented in Sect. 5.3. The description of the following algorithm reflects a correction of an error in the original version by Hochbaum [61] (fixed in Moriguchi and Shioura [107]). Recall the definition of the saturation capacity $\hat{c}(x, j)$ for $x \in P(r)$ and $j \in E$, which is given by (35). δ computed in the algorithm is used only for the proof of the proximity theorem (Theorem 14).

Procedure `SM-INCREMENT(s)`

Input: An instance Q of problem SC/SM/D and a positive integer s .

Output: A feasible solution $x^{(s)}$ of SC/SM/D.

Let $x_j := 0$ and $\delta_j := 0$ for all $j \in E$, $E' := E$ and $k := 0$.

```

while  $E' \neq \emptyset$  do
  begin
    Find  $j^* \in E'$  such that
     $d_{j^*}(x_{j^*} + 1) = \min\{d_j(x_j + 1) \mid j \in E'\};$ 
    if  $x + e(j^*) \notin P(r)$  then  $E' := E' - \{j^*\}$ 
    else if  $x + s \cdot e(j^*) \notin P(r)$  then  $E' := E' - \{j^*\}, \alpha := \hat{c}(x, j^*),$ 
       $x_{j^*} := x_{j^*} + \alpha, k := k + \alpha$  and  $\delta_{j^*} := \alpha$ 
    else  $x_{j^*} := x_{j^*} + s, k := k + s$ , and  $\delta_{j^*} := s$ 
  end;
  Output  $x^{(s)} := x$ .

```

Note that the output $x^{(s)}$ of SM-INCREMENT(s) satisfies $x^{(s)} + e(j) \notin P(r)$ for all $j \in E$, and therefore, $x^{(s)} \in B(r)$ holds; in particular, $\sum_{j \in E} x_j^{(s)} = r(E)$ holds.

A proximity theorem of the following form holds for SC/SMD. Notice that the vector δ records the last increments executed by SM-INCREMENT(s), that is, $\delta_j = \alpha$ implies that $x^{(s)}$ is equal to α modulo s .

Theorem 14 *For the solution $x^{(s)}$ and vector δ output by SM-INCREMENT(s), there exists an optimal solution x^* of problem Q such that $x^* \geq x^{(s)} - \delta \geq x^{(s)} - se$.*

This proximity theorem follows immediately from the next result by Girlich, Kovalev, and Zaporozhets [51]. Notice that the next result also gives an alternative proof for the validity of the incremental algorithm for SC/SMD presented in Sect. 3.2.1.

Theorem 15 *Let $x \in P(r)$ and $y \in \mathbb{Z}^E$ be a vector with $y \leq x$ such that there exists an optimal solution x^* of problem Q satisfying $x^* \geq y$. Suppose that $j^* \in E$ satisfies $x + e(j^*) \in P(r)$ and*

$$d_{j^*}(x_{j^*} + 1) = \min\{d_j(x_j + 1) \mid j \in E, x + e(j) \in P(r)\}.$$

Then, there exists an optimal solution \tilde{x} of problem Q satisfying both of $\tilde{x} \geq y$ and $\tilde{x}_{j^} > x_{j^*}$.*

Proof Let x^* be an optimal solution of problem Q with $x^* \geq y$ such that the value of $x_{j^*}^*$ is the maximum among all such vectors. To derive a contradiction, assume that $x_{j^*}^* \leq x_{j^*}$. Let $x' = x + e(j^*)$. Then, it holds that $x'_{j^*} = x_{j^*} + 1 > x_{j^*}^*$. Since x' and x^* are in the submodular polyhedron $P(r)$ and satisfy $\sum_{j \in E} x'_j \leq r(E) = \sum_{j \in E} x_j^*$, it can be shown in a similar way as in Lemmas 4 and 6 that there exists some $i \in E$ with $x'_i < x_i^*$ such that $x' - e(j_*) + e(i) \in P(r)$, $x^* + e(j_*) - e(i) \in P(r)$, and

$$f(x') + f(x^*) \geq f(x' - e(j_*) + e(i)) + f(x^* + e(j_*) - e(i)). \quad (96)$$

Notice that $x' - e(j_*) + e(i) = x + e(i)$, which implies that $f(x' - e(j_*) + e(i)) \geq f(x')$ due to the choice of j^* . It follows from this inequality and the inequality (96) that $f(x^*) \geq f(x^* + e(j_*) - e(i))$. Since $x^* + e(j_*) - e(i) \in B(r)$ holds, the vector $\tilde{x} \equiv x^* + e(j_*) - e(i)$ is also an optimal solution of problem Q , which satisfies $\tilde{x} \geq y$; recall that $\tilde{x}_i = x_i^* - 1 \geq x'_i \geq y_i$. In addition, \tilde{x} satisfies $\tilde{x}_{j_*} > x_{j_*}^*$, a contradiction to the choice of x^* . \square

A proximity theorem for SC/SM/D and its continuous relaxation, that is, SC/SM/C are now considered. The following statement is shown by Moriguchi, Shioura, and Tsuchimura [108], which corrects an error in the corresponding statement of Hochbaum [61].

Theorem 16 *For any optimal solution x^* of Q , there exists an optimal solution \tilde{x} of the continuous relaxation of Q such that*

$$\|\tilde{x} - x^*\|_1 \leq 2(n - 1), \quad (97)$$

and, vice versa, for any optimal solution \tilde{x} of the continuous relaxation of Q , there exists an optimal solution x^ of Q that satisfies (97). Here $\|y\|_1$ denotes the l_1 -norm $\sum_{j \in E} |y_j|$.* \square

The proof of this theorem is omitted. From Theorem 16, the following result can be obtained by using the fact that $\sum_{j \in E} x_j^* = \sum_{j \in E} \tilde{x}_j$.

Corollary 3 *For any optimal solution x^* of Q , there exists an optimal solution \tilde{x} of the continuous relaxation of Q such that*

$$\|\tilde{x} - x^*\|_\infty \leq n - 1, \quad (98)$$

and, vice versa, for any optimal solution \tilde{x} of the continuous relaxation of Q , there exists an optimal solution x^ of Q that satisfies (98).*

The bound in (98) is better than the one given in Theorem 10, which says $\|x^* - \tilde{x}\|_\infty \leq n\Delta$, since Δ for the submodular constraints is generally greater than one.

These proximity results shown in this section will be used to develop efficient algorithms for SC/-/D and SQC/-/D in the next section.

4.4 Notes

Proximity results between continuous and integer optimal solutions for some optimization problems have been known before. In Chap. 4 of the book by Ibaraki and Katoh [69], a proximity theorem between SC/Simple/C and SC/Simple/D was shown, based on which an efficient algorithm for SC/Simple/D was developed. An efficient algorithm for Fair/SM/D by Fujishige, Katoh, and Ichimori [46] is based on

the proximity theorem developed therein among Fair/SM/D, Minimax/SM/D, and Maximin/SM/D. Namikawa and Ibaraki [112] proved a similar theorem between Fair/SM/C and Fair/SM/D, to derive an efficient algorithm for Fair/SM/D.

These proximity results come from the special structures of objective functions and/or constraints. By extending this line of development, many other types of proximity results may be obtained.

Notice that Algorithm SOLVE.IP explained in Sect. 4.2 does not specify an algorithm that solves the linear program $\text{LP}^{(k)}/s_k$. When the constraint matrix represents a network, Hochbaum [60] proposed a network flow algorithm for this case which is an adaptation of the *successive shortest path method* for minimum-cost flow problems. For a general totally unimodular constraint matrix, Karzanov and McCormick [76] proposed another polynomial-time algorithm. The idea of their algorithm is a generalization of the *minimum mean cancellation method* developed for minimum-cost flow problems. Thus, the algorithm does not rely on linear programming, either. Karzanov and McCormick [76] further considered specializations and generalizations by changing the form of the objective functions and the constraints.

5 Lower Bounds on Time Complexity and Improved Algorithms

This section first explains an alternative, faster algorithm for SC/SM/D based on the proximity theorem (Theorem 14), which is developed by Hochbaum [61] and later fixed by Moriguchi and Shioura [107]. This algorithm, when specialized to problems SC/Nested/D and SC/Tree/D, gives better time bounds than previous algorithms. The running times of the resulting algorithms are $O(n \log n \log(N/n))$ for both of SC/Nested/D and SC/Tree/D. None of these algorithms, however, is strongly polynomial, where a polynomial-time algorithm is called *strongly polynomial* if its running time depends only on the dimension of the problem (i.e., the number of the input, e.g., n in the resource allocation problem), instead of the input size, and is called *weakly polynomial* otherwise. For instance, the $O(n \log(N/n))$ time bound of the SC/Simple/D is weakly polynomial because the number of the input is $O(n)$, but the running time depends on N .

Then, a lower bound result for SC/Simple/D obtained by Hochbaum [61] is shown (see also Hochbaum [63]), which says that, under a certain computation model, there cannot exist a strongly polynomial algorithm even for SC/Simple/D, if it is assumed that f_j 's are general convex functions (e.g., polynomials of degree at least 3). This is then contrasted with the cases of linear and quadratic objective functions. In the latter cases, there are strongly polynomial algorithms for SC/-/D, where \cdot denotes Simple, GUB, Nested, Tree, and Network, as demonstrated by Hochbaum and Hong [64] (see also Moriguchi, Shioura, and Tsuchimura [108]).

This section is mainly devoted to describing the underlying basic ideas of how the above results are derived.

5.1 Improved Algorithms for SC/SM/D and Its Special Cases

An algorithm for SC/SM/D is described below, which is based on the scaling technique and a proximity theorem ([Theorem 14](#)) and runs in $O(n(\log n + \tilde{\tau}) \log(r(E)/n))$ time, where $\tilde{\tau}$ denotes the time required to compute the saturation capacity $\hat{c}(x, j)$ for given $x \in P(r)$ and $j \in E$.

Procedure SCALING

Input: An instance of problem SC/SM/D.

Output: An optimal solution x^* .

```

Let  $s := \lceil r(E)/2n \rceil$  and  $l := x := (0, 0, \dots, 0)$ ;
while  $s > 1$  do
begin
    Call SM-INCREMENT( $s, l$ ), and let  $x^{(s)}$  be its output;
    Let  $l$  be defined by  $l_i := \max\{x_i^{(s)} - s, 0\}$  for each  $i \in E$ ;
    Let  $s := \lceil s/2 \rceil$ 
end;
Call SM-INCREMENT with  $x^{(s)}$  as an initial solution;
Output the obtained solution as  $x^*$ .

```

In the above procedure, $e = (1, 1, \dots, 1)$ and SM-INCREMENT(s, l) is the same as procedure SM-INCREMENT(s) of [Sect. 4.3](#) except that it starts with $x = l$ as an initial solution.

Theorem 17 *Given an instance of SC/SM/D, procedure SCALING correctly finds an optimal solution x^* in $O(n(\log n + \tilde{\tau}) \log(r(E)/n))$ time.*

Proof Since the vector $x^{(s)} - se$ is a lower bound of an optimal solution by [Theorem 14](#), addition of the constraint $x \geq x^{(s)} - se$ does not change the optimal value of the problem. Thus, the correctness of the algorithm follows immediately.

Then, the running time is analyzed. The algorithm calls procedure SM-INCREMENT(s, l) $O(\log \lceil r(E)/2n \rceil)$ times in the while loop and procedure SM-INCREMENT only once. In each call to SM-INCREMENT(s, l), at most $\lfloor (r(E) - \sum_{j \in E} l_j)/s \rfloor + n$ increments are executed. Let s' be the value of s in the last iteration in the while loop. Then, $s = \lceil s'/2 \rceil$ and $l = x^{(s')} - s'e$ hold. Since $\sum_{j \in E} x_j^{(s')} = r(E)$, it holds that

$$\sum_{j \in E} l_j = \sum_{j \in E} (x_j^{(s')} - s') = \sum_{j \in E} x_j^{(s')} - s'n \geq r(E) - 2sn.$$

Hence, the number of increments in SM-INCREMENT(s, l) is

$$\left\lfloor \frac{r(E) - \sum_{j \in E} l_j}{s} \right\rfloor + n \leq \left\lfloor \frac{2sn}{s} \right\rfloor + n = 3n = O(n).$$

The call to SM-INCREMENT after the while loop also requires $O(n)$ increments. Thus, it can be shown in a similar way as in [Theorem 5](#) that the running time of procedure SCALING is $O(n(\log n + \tilde{\tau}) \log(r(E)/n))$. \square

[Theorem 17](#) implies that when specialized to problems SC/Nested/D or SC/Tree/D, the algorithm SCALING runs in $O(n^2 \log(N/n))$ time since there exist $O(n)$ constraints, and hence, the following operations for $x \in P(r)$ and $i \in E$ can be done in $O(n)$ time:

- (a) Check the feasibility of $x + e(i)$, and if $x + e(i)$ is feasible, update x to $x + e(i)$.
- (b) For a given integer $s \geq 2$, compute the saturation capacity $\hat{c}(x, i)$, and update x to $x + \alpha e(i)$, where $\alpha = \min\{\hat{c}(x, i), s\}$.

That is, $\tilde{\tau} = O(n)$ holds if a naive implementation is used. In contrast, Hochbaum [61] improved upon this running time by using the special structure of the problems. The results obtained by Hochbaum [61] are summarized as follows:

Theorem 18 *Problems SC/Nested/D and SC/Tree/D can be solved in $O(n \log n \log (N/n))$ time.* \square

This improvement is attained by devising a systematic implementation of the operations (a) and (b) above, which makes it possible to perform the two operations in $O(1)$ time on the average. Instead of giving their details, a key idea is illustrated by briefly explaining how to implement the operation (a) (i.e., feasibility check of $x + e(i)$) for the case of problem SC/Nested/D.

Suppose that the following n nested constraints are given:

$$\sum_{j \in S_i} x_j \leq b_i, \quad i = 1, \dots, n, \quad (99)$$

where $S_i = \{1, 2, \dots, i\}$ ($i = 1, \dots, n$) are assumed for simplicity. Also, assume without loss of generality that $b_1 \leq b_2 \leq \dots \leq b_n$.

It is now explained how to check the feasibility of a vector $x + e(i)$ for given feasible vector x and $i \in E$. For this purpose, a nonnegative *slack* σ_i is associated with each variable x_i (or i -th inequality in (99)). The value of each slack σ_i is determined as follows by using the current feasible vector x :

$$\sigma_1 = \bar{b}_1, \quad \sigma_i = \bar{b}_i - \bar{b}_{i-1} \quad (i = 2, 3, \dots, n), \quad (100)$$

where \bar{b}_i is recursively defined by

$$\bar{b}_i = \begin{cases} b_n - \sum_{j=1}^n x_j & (i = n), \\ \min\{b_i - \sum_{j=1}^i x_j, \bar{b}_{i+1}\} & (i = n-1, n-2, \dots, 1). \end{cases} \quad (101)$$

For example, if $x = (0, 0, \dots, 0)$, then it holds that $\sigma_1 = b_1$ and $\sigma_i = b_i - b_{i-1}$ for $i = 2, 3, \dots, n$. Note that $\sigma_i \geq 0$ for all i since \bar{b}_i is monotone nondecreasing in i by definition and that the set of constraints (99) is equivalent to

$$\bar{b}_i = \sigma_1 + \sigma_2 + \dots + \sigma_i \geq 0, \quad i = 1, 2, \dots, n.$$

During the execution, the algorithm implicitly maintains the feasibility of constraints (99) through the slack vector $\sigma = (\sigma_1, \dots, \sigma_n)$.

Suppose that the current solution x is feasible and the vector σ of (100) is given. To check whether the increment of x_i by one unit is feasible or not, the value

$$k(i) := \max\{j \mid 1 \leq j \leq i, \sigma_j > 0\} \quad (102)$$

is computed, where it is understood that $k(i) = 0$ if there exists no j with $1 \leq j \leq i$ such that $\sigma_j > 0$. If $k(i) = 0$, then it holds that $\sigma_1 = \dots = \sigma_i = 0$, that is, $\bar{b}_i = 0$ holds, and therefore, the increment of x_i by one unit is clearly infeasible from definition of \bar{b} . Otherwise, $x + e(i)$ is feasible since $\bar{b}_i > 0$ holds. According to the update of x to $x + e(i)$, σ is also updated as

$$\sigma_{k(i)} := \sigma_{k(i)} - 1. \quad (103)$$

It is easy to see that this update correctly maintains the condition (100) for the new feasible solution $x + e(i)$; indeed, before the update $\bar{b}_{k(i)} = \dots = \bar{b}_i$ holds from the definition of $k(i)$, and updating σ by (103) implicitly decreases all $\bar{b}_{k(i)}, \dots, \bar{b}_i$ by one unit, which maintains the correct \bar{b} defined by (101) for the new feasible solution $x + e(i)$. It shall be explained later how this update of the vector σ can be done in $O(1)$ time on the average.

The feasibility check algorithm is summarized as follows. The input of the algorithm is a feasible vector x , the index i of the variable to be increased, the current slack vector σ , and a 0-1 labeling vector $label$ defined as

$$label_i = \begin{cases} 1 & \text{if } \sigma_i > 0, \\ 0 & \text{if } \sigma_i = 0. \end{cases} \quad (104)$$

As is made clear later, the vector $label$ plays an important role in efficient implementation of the algorithm.

Procedure FEASIBILITY-CHECK

Input: An instance of SC/Nested/D, a feasible solution x , an index i , a slack vector σ , and a 0-1 labeling vector $label$.

Output: “feasible” and the updated slack vector σ if $x + e(i)$ is feasible, and “infeasible” otherwise.

Compute $k(i)$ by (102);

if $k(i) > 0$ **then do**

begin

Let $\sigma_{k(i)} := \sigma_{k(i)} - 1$ and $label_{k(i)} := \min\{label_{k(i)}, \sigma_{k(i)}\}$;

Output “feasible” and $(\sigma_1, \dots, \sigma_n)$

end

else output “infeasible”

The implementation of FEASIBILITY-CHECK requires the labeling of $label_j$ for all $j \in E$ and the identification of index $k(i)$ for all $i \in E$. This can be done in linear time as follows. If the vector $label$ is regarded as a one-dimensional array, $label$ can be seen as the intervals of 0’s, separated by 1’s. Such intervals are maintained so that the interval containing a given index i can be identified efficiently. Left-end and right-end indices of the interval are associated with the interval ID, which makes it possible to find $k(i)$ efficiently. When $label_{k(i)}$ is updated from 1 to 0, the adjacent two intervals are merged. Hence, this can be seen a special case of a *union-find* problem, and the sequence of $O(n)$ such operations can be executed in $O(n)$ time using Gabow and Tarjan’s algorithm [47]. Thus, feasibility check of a vector $x + e(i)$ can be done in $O(1)$ time on the average.

In a similar way, the other operation (b) can be implemented so that it can be performed in $O(1)$ time on the average. Therefore, from [Theorem 17](#), the total complexity of algorithm SCALING for SC/Nested/D becomes $O((n \log n + n) \log(N/n)) = O(n \log n \log(N/n))$.

5.2 Lower Bounds

Two types of weakly polynomial lower bounds for SC/Simple/D are presented, which tell that no strongly polynomial algorithm exists for SC/Simple/D. The first one is based on a comparison model and the second on an algebraic tree model.

5.2.1 Lower Bound on a Comparison Model

A comparison computation model allows only the operations of comparisons and branchings. The lower bound proof establishes that no algorithm exists that solves SC/Simple/D in less than $\log_2 N$ comparisons. The proof relies on a result of information theory according to which there is no algorithm that finds a value in a monotone nondecreasing n -array that is the first to be smaller than some specified constant in less than $\log_2 n$ time. Consider first a lower bound result for SC/Simple/D in two variables, $Q(2)$:

$$Q(2) : \text{minimize } f_1(x_1) + c \cdot x_2,$$

$$\text{subject to } x_1 + x_2 = N,$$

$$x_j : \text{nonnegative integer, } j = 1, 2.$$

Let the function $f_1(x_1)$ be given as an array of N increments at the N integer points. Namely, if $f_1(i) = a_i$, the array of increments is $\{a_0, a_1 - a_0, a_2 - a_1, \dots, a_N - a_{N-1}\}$. Since the function is convex, the entries in the array are monotone nondecreasing, $a_{i+1} - a_i \geq a_i - a_{i-1}$. The optimal solution to this problem is $x_1 = j$ and $x_2 = N - j$, where j is the largest index such that $a_j - a_{j-1} \leq c$. Since the entries in the array are monotone nondecreasing, determining the index j can be done using binary search in $\log_2 N$ comparisons. The information theoretic lower bound is also $\log_2 N$ comparisons. This is because the comparison tree has N leaves so the path of comparisons leading from the root to a leaf could be as long as $\log_2 N$ (see Knuth [87]). Suppose the problem could be solved independently of N , then given a monotone nondecreasing array and a value c , it has a corresponding convex function f_1 such that the solution to the $Q(2)$ is independent of N . Consequently, the required entry in the vector could be found independently of N , which is a contradiction to the comparison tree lower bound.

The same arguments can be extended to prove that in the comparison model, the problem of SC/Simple/D on $n + 1$ variables has complexity $\Omega(n \log_2 \lfloor \frac{N}{n} \rfloor)$.

Consider the following problem $Q(n + 1)$ of SC/Simple/D, where $c < 0$ is a given constant:

$$\begin{aligned} Q(n + 1) : \text{minimize } & \sum_{j=1}^n f_j(x_j) + c \cdot x_{n+1}, \\ \text{subject to } & \sum_{j=1}^{n+1} x_j = N, \\ & x_j : \text{nonnegative integer, } j = 1, 2, \dots, n + 1. \end{aligned}$$

Let the functions f_j be monotone decreasing in interval $[0, \lfloor \frac{N}{n} \rfloor]$, and constant in $[\lfloor \frac{N}{n} \rfloor, N]$. Define a vector $\hat{x} = (\hat{x}_1, \hat{x}_2, \dots, \hat{x}_{n+1})$ as follows:

$$\begin{aligned} \hat{x}_j &= \max\{y \mid d_j(y) \leq c, y \in [1, \lfloor \frac{N}{n} \rfloor]\}, \\ \hat{x}_{n+1} &= N - \sum_{j=1}^n \hat{x}_j. \end{aligned} \tag{105}$$

It is claimed that \hat{x} is an optimal solution of $Q(n + 1)$. First note that \hat{x} is feasible to Q from (105). Since $d_{n+1}(y) = c$ holds for all y , condition (105) tells that \hat{x} satisfies the optimality condition of [Theorem 4](#), that is,

$$d_i(\hat{x}_i) \leq d_i(\hat{x}_j + 1) \tag{106}$$

holds for all i, j with $1 \leq i, j \leq n + 1$.

Thus, the above claim holds. Therefore, solving problem $Q(n + 1)$ is equivalent to determining in n arrays of length $\lfloor \frac{N}{n} \rfloor$ each, the largest entry of value $\leq c$. Since the arrays are independent, the information theory lower bound is $\Omega(n \log_2 \lfloor \frac{N}{n} \rfloor)$.

Theorem 19 *In a comparison model, a lower bound on the time complexity for SC/Simple/D is $\Omega(n \log \lfloor \frac{N}{n} \rfloor)$.* \square

5.2.2 Lower Bound on an Algebraic Tree Model

The comparison model may be considered too restrictive. If it is allowed to use arithmetic operations such as addition, subtraction, multiplication, and division, there may be a possibility of constructing a strongly polynomial algorithm for SC/Simple/D. In fact, if all f_j are quadratic, an $O(n)$ time algorithm can be developed for SQC/Simple/D (SQC means *separable quadratic convex*) by making use of a continuous optimal solution of SQC/Simple/C (see Sect. 4.6 of [69]), which can be obtained in $O(n)$ time by the algorithm for SQC/Simple/C of Brucker [23]. This approach, however, is denied in the general case, as will be shown below, even if f_j are polynomials of degree 3.

The lower bound proof is based on Renegar's result [122] saying that $\Omega(\log \log(R/\epsilon))$ is a lower bound for the time complexity of finding an ϵ -accurate single real root of a polynomial of a single variable of degree ≥ 2 in an interval $[0, R]$, even if the polynomial is monotone increasing in that interval. Here $\hat{x} \in R$ is called an ϵ -*accurate root* of an equation $p(x) = 0$ if \hat{x} satisfies $|\hat{x} - x^*| \leq \epsilon$ for an exact root x^* of $p(x) = 0$.

Let $p_1(x), \dots, p_n(x)$ be n monotone increasing polynomials, each with a single root in interval $[0, N/n]$ for the equation $p_j(x) = c$, where c is a fixed constant. Since the choice of these polynomials is arbitrary, the lower bound on finding the n roots of these n equations is $\Omega(n \log \log(N/n\epsilon))$. Let

$$f_j(x_j) = \int_0^{x_j} p_j(x) dx.$$

These f_j are then polynomials of degree ≥ 3 .

Now, consider the following problem Q_ϵ for $\epsilon > 0$ such that $1/\epsilon$ is an integer:

$$Q_\epsilon : \text{minimize } \sum_{j=1}^n f_j(\epsilon \cdot x_j) + c \cdot \epsilon \cdot x_{n+1},$$

$$\text{subject to } \sum_{j=1}^{n+1} x_j = \frac{N}{\epsilon}, \tag{107}$$

$$x_j : \text{nonnegative integer}, \quad j = 1, \dots, n+1.$$

Also, let R_ϵ be a continuous version of Q_ϵ . Problems Q_ϵ and R_ϵ are instances of SC/Simple/D and SC/Simple/C, respectively. It is claimed below that, for any

optimal solution x^* of Q_ϵ , $y = \epsilon \cdot x^*$ is an ϵ -accurate vector of roots for the following system of nonlinear equations:

$$\begin{cases} p_1(y_1) = c, \\ p_2(y_2) = c, \\ \vdots \\ p_n(y_n) = c. \end{cases} \quad (108)$$

This means that the computational complexity is not less than finding an ϵ -accurate vector for the above system of equations. Since it is assumed that for each j with $1 \leq j \leq n$ there exists a single root in interval $[0, N/n]$ for the equation $p_j(x) = c$, it follows from the Karush–Kuhn–Tucker condition that an optimal solution y^* of R_ϵ satisfies (108) with $y = y^*$.

It follows from [Theorem 4](#) that

$$d_j(x_j^* + 1) \geq d_{n+1}(x_{n+1}^*) \quad (= c \cdot \epsilon)$$

and $d_j(x_j^*) \leq d_{n+1}(x_{n+1}^* + 1) \quad (= c \cdot \epsilon)$

for all j , that is,

$$\int_{\epsilon x_j^*}^{\epsilon x_j^* + \epsilon} p_j(x) dx \geq c \cdot \epsilon \text{ and}$$

$$\int_{\epsilon x_j^* - \epsilon}^{\epsilon x_j^*} p_j(x) dx \leq c \cdot \epsilon,$$

for all j , implying that there are $y_j \in [\epsilon x_j^* - \epsilon, \epsilon x_j^* + \epsilon]$ such that $p_j(y_j) = c$ from the monotonicity of $p_j(y_j)$. This proves the above claim. Hence, a lower bound for the time complexity of solving Q_ϵ is $\Omega(n \log \log(N/n\epsilon))$. For $\epsilon = 1$, Q_ϵ is an instance of SC/Simple/D, and thus, the following result is obtained.

Theorem 20 *In the algebraic computation tree model, a lower bound on the computational complexity for SC/Simple/D is $\Omega(n \log \log(N/n))$. \square*

5.3 Strongly Polynomial Algorithms for Separable Quadratic Convex Objective Functions

Although no strongly polynomial-time algorithm for SC/Simple/D exists even for polynomials f_j of degree 3, as proved in [Sect. 5.2.2](#), there remains a possibility of a strongly polynomial algorithm for the case of a separable *quadratic* objective function. Indeed, SQC/Simple/D can be solved in $O(n)$ time, as mentioned in [Sect. 5.2.2](#). This result also implies the linear-time solvability of SQC/GUB/D (see [Sect. 3.3](#)).

In addition, Hochbaum and Hong [64] constructed such algorithms for several cases of constraints. Results are summarized as follows.

Theorem 21 *Let n be the number of variables.*

- (i) *SQC/Nested/D can be solved in $O(n^2 \log n)$ time.*
- (ii) *SQC/Tree/D can be solved in $O(n^2 \log n)$ time.*
- (iii) *SQC/Network/D can be solved in $O(n^2 M + NM \log(N^2/M))$ time, where N and M denote the numbers of nodes and arcs in a network.* \square

It is noted that the statements in [Theorem 21](#) are corrected version of the corresponding statements in Hochbaum and Hong [64], which rely on an incorrect proximity statement. However, [Theorem 21](#) can be obtained by using an alternative proximity theorem, as explained below (see [108, Sect. 2] for details).

The underlying idea to derive [Theorem 21](#) is now explained. Consider SQC/SM/D because all the problems listed in [Theorem 21](#) are special cases of SQC/SM/D. The first idea is to make use of an optimal solution of the corresponding continuous problem SQC/SM/C. It is here assumed that an optimal solution of SQC/SM/C is available; as shown below, the continuous versions of all the above three problems can be solved in strongly polynomial time if the objective function is quadratic. The continuous optimal solution can then be used to bound an integer optimal solution from below, based on the proximity theorem given in [Sect. 4](#). The algorithm then applies procedure SM-INCREMENT of [Sect. 3.2.1](#) in order to solve SQC/SM/D. This runs in strongly polynomial time since the lower bound obtained is sufficiently close to the integer optimal solution, as will be seen below.

Let \bar{x} be a continuous optimal solution of SQC/SM/C. From the proximity result in [Corollary 3](#), there exists an optimal solution x^* of SQC/SM/D such that

$$x^* \geq l, \quad \text{where } l_j = \lceil \bar{x}_j \rceil - (n-1) \text{ for } j \in E. \quad (109)$$

Notice that the vector l belongs to the submodular polyhedron $P(r)$. Then, procedure SM-INCREMENT of [Sect. 3.2.1](#) is applied, where the vector l is used as an initial solution. Procedure SM-INCREMENT terminates in $O(n^2)$ iterations since

$$r(E) - \sum_{j \in E} l_j = \sum_{j \in E} \bar{x}_j - \sum_{j \in E} \{\lceil \bar{x}_j \rceil - (n-1)\} \leq n(n-1).$$

Hence, [Theorem 5](#) implies the following lemma. Recall that feasibility check of a vector $x + e(i)$ for $x \in P(r)$ and $i \in E$ can be done in $O(1)$ time on the average for SQC/Nested/D and for SQC/Tree/D (see [Sect. 5.1](#)).

Lemma 12 *Suppose that an optimal solution of SQC/Nested/C (resp., SQC/Nested/C) is given. Then SQC/Nest/D (resp., SQC/Tree/D) can be solved in $O(n^2 \log n)$ time.* \square

In case of SQC/Network/D, feasibility check of a vector $x + e(i)$ can be done in $O(M)$ time, as is well known in the network flow theory. Hence, the next lemma follows.

Lemma 13 *Given an optimal solution of SQC/Network/C, problem SQC/Network/D can be solved in $O(n^2M)$ time, where M is the number of arcs in a network.* \square

In order to efficiently compute a continuous optimal solution for the three problems under consideration, it is possible to use a technique based on the Karush–Kuhn–Tucker condition involving the original variables and Lagrange multipliers. Since the objective function is quadratic, the Karush–Kuhn–Tucker condition contains only linear terms with respect to variables, and optimal Lagrange multipliers can be efficiently computed using the special structure of the constraints. In particular, Hochbaum and Hong [64] showed that optimal Lagrange multipliers for SQC/Network/C can be computed by reduction to the parametric flow problem, which is solved by a generalization of the algorithm given by Gallo, Grigoriadis, and Tarjan [49] (The generalization is necessary since the latter algorithm is applicable only to the case in which each quadratic function $f_j(x_j)$ does not contain a linear term. In addition, Hochbaum and Hong [64] found a flaw in the algorithm of [49] and fixed it).

Further details about how this task can be done efficiently are omitted and only the results are stated here (see [64] for details).

Theorem 22 *Let n be the number of variables.*

- (i) *SQC/Nested/C can be solved in $O(n \log n)$ time.*
- (ii) *SQC/Tree/C can be solved in $O(n \log n)$ time.*
- (iii) *SQC/Network/C can be solved in $O(NM \log(N^2/M))$ time, where N and M denote the numbers of nodes and arcs in a network, respectively.* \square

Theorem 21 can be obtained by combining Theorem 22 with Lemmas 12 and 13.

Finally, some improved results for SQC/Nested/D, SQC/Tree/D, and SQC/Network/D will be explained which were obtained by Moriguchi, Shioura, and Tsuchimura [108]. The results are summarized as follows.

Theorem 23 *Let n be the number of variables.*

- (i) *SQC/Nested/D can be solved in $O(n^2)$ time.*
- (ii) *SQC/Tree/D can be solved in $O(n^2)$ time.*
- (iii) *SQC/Network/D can be solved in $O(NM \log(N^2/M))$ time, where N and M denote the numbers of nodes and arcs in a network.* \square

As in the approach by Hochbaum and Hong [64], the approach in [108] also uses a continuous optimal solution and a proximity theorem, but the approach in [108] uses Theorem 16 rather than Corollary 3; in particular, it uses neither of upper

nor lower bounds for an integer optimal solution. Their approach firstly transforms a continuous optimal solution $\bar{x} \in B(r) \cap R^E$ into an integer feasible solution $x \in B(r) \cap Z^E$ which is sufficiently close to \bar{x} . Then, the vector x is iteratively updated by incrementing some x_i by one unit and decrementing some x_k by one unit until an integer optimal solution is found. It can be shown that the number of iterations required by this algorithm is proportional to the L_1 distance between integer and continuous optimal solutions. Hence, the algorithm terminates in $O(n)$ iterations by [Theorem 16](#). For more details, see [108].

5.4 Notes and References

Strongly polynomial algorithms have also been developed for problem SQC/Linear/C(or D) which includes special constraints other than submodular constraints, for example, the transportation problem with a fixed number of sources (or sinks) [32] and its extension [105] and the minimum-cost flow in a series-parallel network with a single source and sink [133]. In these cases, the constraint matrix is totally unimodular, and the problems can be viewed as separable convex cost flow problems. A strongly polynomial algorithm has also been developed for the transportation problem with a certain type of nonseparable quadratic convex objective function [66].

6 Nonseparable Convex Resource Allocation

In this section, a class of nonseparable convex resource allocation problems is considered. Recall that a separable convex function f defined over a base polyhedron $B(r)$ satisfies [Lemma 6](#), that is,

$$f(x) + f(y) \geq f(x - e(i) + e(j)) + f(y + e(i) - e(j))$$

for any bases $x, y \in B(r)$ and any $i, j \in E$ with $x_i > y_i$ and $x_j < y_j$ such that $x - e(i) + e(j), y + e(i) - e(j) \in B(r)$. The class of functions f (which is not separable in general) that satisfy this property is defined as an *M-convex function*, which was introduced by Murota [109]. There are several beautiful theorems concerning an *M-convex* function that can be viewed as a discrete analogue of theorems in classical convex analysis. Since the class of *M-convex* functions provides a potentially much wider class of problems than the separable convex functions, it is interesting to ask whether there exists an efficient general algorithm for minimizing an *M-convex* function over a base polyhedron of a submodular system. Shioura [129] answers this question affirmatively by giving a polynomial-time algorithm for this problem, which will be explained in this section after preparing some notions and properties in the next subsection.

6.1 M -Convex Functions

Let $E = \{1, 2, \dots, n\}$. A function $f : Z^E \rightarrow R \cup \{\infty\}$ is said to be M -convex if it satisfies the following property:

(M-EXC): For any $x, y \in \text{dom } f$, and for any $i \in \text{supp}^+(x - y)$, there exists $j \in \text{supp}^-(x - y)$ such that

$$f(x) + f(y) \geq f(x - e(i) + e(j)) + f(y + e(i) - e(j)), \quad (110)$$

where

$$\begin{aligned}\text{dom } f &= \{x \in Z^E \mid f(x) < +\infty\}, \\ \text{supp}^+(x - y) &= \{k \in E \mid x_k > y_k\}, \\ \text{supp}^-(x - y) &= \{k \in E \mid x_k < y_k\}.\end{aligned}$$

Minimizing an M -convex function f over $\text{dom } f$ is called MC/SM/D. The property (M-EXC) implies that $\text{dom } f$ is an integral base polyhedron as shown by Murota [109] because $B \subseteq Z^E$ is an integral base polyhedron if and only if B satisfies the following property (see Lemma 4 (ii)):

(B-EXC): For any $x, y \in \text{dom } f$, and for any $i \in \text{supp}^+(x - y)$, there exists $j \in \text{supp}^-(x - y)$ such that $x - e(i) + e(j), y + e(i) - e(j) \in B$.

Before describing an algorithm for minimizing an M -convex function, the following results are first shown. A vector $x \in Z^E$ that minimizes an M -convex function f is called a *minimizer* of f , which is assumed to exist in the following discussion. Let the underlying base polyhedron be denoted by B . Also, assume that B is bounded.

Lemma 14 *Given an $x \in \text{dom } f$, it holds $f(x) \leq f(y)$ for all $y \in \text{dom } f$ if and only if $f(x) \leq f(x - e(i) + e(j))$ for all $i, j \in E$.*

Proof The proof is done essentially in the same manner as the proof of Theorem 4. \square

Lemma 15 *Let $f : Z^E \rightarrow R \cup \{\infty\}$ be an M -convex function.*

(i) *Given an $x \in \text{dom } f$ and a $j \in E$, let $i \in E$ satisfy*

$$f(x - e(i) + e(j)) = \min_{h \in E} f(x - e(h) + e(j)).$$

Then, for the vector $x' = x - e(i) + e(j)$, there exists a minimizer x^ of f that satisfies $x_i^* \leq x'_i$.*

(ii) *Given an $x \in \text{dom } f$ and an $i \in E$, let $j \in E$ satisfy*

$$f(x - e(i) + e(j)) = \min_{l \in E} f(x - e(i) + e(l)).$$

Then, for the vector $x' = x - e(i) + e(j)$, there exists a minimizer x^ of f that satisfies $x_j^* \geq x'_j$.*

Proof Only (i) is proved. Let x^* be a minimizer of f such that x_i^* is minimum among all minimizers of f . Suppose $x_i^* > x'_i$. From (M-EXC), there exists some $k \in \text{supp}^-(x^* - x')$ such that

$$f(x^*) + f(x') \geq f(x^* - e(i) + e(k)) + f(x' + e(i) - e(k)). \quad (111)$$

From the assumption on x' , it holds that

$$f(x' + e(i) - e(k)) = f(x - e(k) + e(j)) \geq f(x - e(i) + e(j)).$$

This, combined with (111), implies that $f(x^* - e(i) + e(k)) \leq f(x^*)$, from which it follows that $x^* - e(i) + e(k)$ is also a minimizer of f . This contradicts the assumption on x^* . \square

Corollary 4 *Let $x \in \text{dom } f$ be not a minimizer of an M -convex function f , for which $i, j \in E$ satisfy*

$$f(x - e(i) + e(j)) = \min_{h,l \in E} f(x - e(h) + e(l)).$$

Then, there exists a minimizer x^ of f satisfying $x_i^* \leq x_i - 1$ and $x_j^* \geq x_j + 1$.* \square

6.2 A Polynomial Algorithm for Minimizing an M -Convex Function

Lemma 14 suggests a simple algorithm for finding a minimizer of f in a manner similar to the *steepest-descent* method for minimizing a convex function over continuous variables. That is, starting from $x \in B$, the algorithm checks whether

$$f(x) = \min_{h,l \in E} f(x - e(h) + e(l))$$

holds or not. If so, x is a minimizer of f by **Lemma 14**. Otherwise, repeat the above procedure after updating $x := x - e(i) + e(j)$, where the $i, j \in E$ satisfy $f(x - e(i) + e(j)) = \min_{h,l \in E} f(x - e(h) + e(l))$. Since the value of $f(x)$ strictly decreases in every iteration, a minimizer of f can be eventually found. This simple approach, however, cannot guarantee the convergence in polynomial time, and a more sophisticated method is needed.

For this, the *peeled base polyhedron* $B_{\text{peeled}} \subseteq B$ is defined as follows. For each $j \in E$, define

$$l_j = \min_{x \in B} x_j, \quad u_j = \max_{x \in B} x_j, \quad (112)$$

and then

$$l'_j = \lfloor (1 - 1/n)l_j + (1/n)u_j \rfloor, \quad u'_j = \lceil (1/n)l_j + (1 - 1/n)u_j \rceil. \quad (113)$$

Then, B_{peeled} is defined as

$$B_{\text{peeled}} = \{x \in B \mid l'_j \leq x_j \leq u'_j, \quad j \in E\}.$$

In other words, B_{peeled} is the base polyhedron obtained from B by performing the restriction by $u' = (u'_1, \dots, u'_n)$ and the contraction by $l' = (l'_1, \dots, l'_n)$ to B .

Theorem 24 *For a base polyhedron B ($\neq \emptyset$), the peeled base polyhedron B_{peeled} satisfies $B_{\text{peeled}} \neq \emptyset$.*

Proof Let $r: 2^E \rightarrow Z$ be a submodular function such that $B = B(r)$. Note that $l_j = r(E) - r(E - \{j\})$ and $u_j = r(j)$ hold for all $j \in E$ (see [45]). It suffices to show the following (see [45]):

- (i) $l'_j(X) \leq r(X)$ for all $X \subseteq E$.
- (ii) $u'_j(X) \geq r(E) - r(E - X)$ for all $X \subseteq E$.

Since (ii) can be proved similarly, only (i) is proved. Let $|X| = k$. It is claimed that

$$k \cdot r(X) + k \sum_{j \in X} (r(E - \{j\}) - r(E)) \geq \sum_{j \in X} (r(j) + r(E - \{j\}) - r(E)). \quad (114)$$

Indeed, by the submodularity of r , it holds that

(the left-hand side of (114))

$$\begin{aligned} &= k \cdot r(X) + \sum_{j \in X} \sum_{k \in X - \{j\}} (r(E - \{k\}) - r(E)) + \sum_{j \in X} (r(E - \{j\}) - r(E)) \\ &\geq k \cdot r(X) + \sum_{j \in X} (r(E - (X - \{j\})) - r(E)) + \sum_{j \in X} (r(E - \{j\}) - r(E)) \\ &\geq (\text{the right-hand side of (114)}). \end{aligned}$$

Since the right-hand side of (114) is nonnegative, the k in (114) can be replaced by n ($\geq k$). Thus, it holds that

$$\begin{aligned} r(X) &\geq (1 - 1/n) \sum_{j \in X} [r(E - \{j\}) - r(E)] + \sum_{j \in X} r(j) \\ &\geq l'_j(X). \end{aligned}$$

□

The following algorithm for finding a minimizer x^* of f reduces B iteratively, until B contains a single base.

Procedure REDUCTION

Input: M -convex function f .

Output: A minimizer x^* of f .

Let $B := \text{dom } f$, and compute vectors l and u by (112);

while $u_j - l_j \geq 1$ for some j **do**

begin

Compute vectors l' and u' of (113) to define B_{peeled} ;

Find a vector $x \in B_{peeled}$;

Let $f(x - e(i) + e(j)) := \min_{h,l \in E} f(x - e(h) + e(l))$;

if $f(x) = f(x - e(i) + e(j))$ **then** output $x^* := x$ and halt

else let $u_i := x_i - 1$, $l_j := y_j + 1$, and

$B := B \cup \{y \in Z^E \mid y_i \leq u_i, y_j \geq l_j\}$;

end.

Output the unique vector in B .

The correctness immediately follows from Corollary 4 and Theorem 24. The running time is analyzed below.

Lemma 16 Let B^k , l^k , and u^k denote the set B , l , and u at the beginning of the k -th iteration of the while loop of REDUCTION, respectively. Then $u_h^{k+1} - l_h^{k+1} < (1 - 1/n)(u_h^k - l_h^k)$ holds for $h = i$ and j , where $i, j \in E$ are the elements found in the k -th execution of the while loop.

Proof Only the case of $h = i$ is considered (the case of $h = j$ is similar), and let $x \in B_{peeled}^k$ be the vector chosen in the while loop. Then,

$$\begin{aligned} u_i^{k+1} - l_i^{k+1} &\leq x_i - 1 - l_i^k \\ &\leq \lceil (1/n)l_i^k + (1 - 1/n)u_i^k \rceil - 1 - l_i^k \\ &< (1 - 1/n)(u_i^k - l_i^k). \end{aligned}$$

□

Lemma 17 Procedure REDUCTION halts in $O(n^2 \log L)$ iterations, where

$$L = \max_{j \in E} (u_j^1 - l_j^1). \quad (115)$$

Proof Since $u_j^k - l_j^k$ is a nonnegative integer for every $j \in E$, the procedure halts if $u_j^k - l_j^k < 1$ holds for all $j \in E$. Let k be the minimum integer such that $(1 - 1/n)^k (u_j^1 - l_j^1) < 1$. Then,

$$\begin{aligned} k &\leq -\ln(u_j^1 - l_j^1)/\ln(1 - 1/n) + 1 \\ &\leq n \ln(u_j^1 - l_j^1) + 1 \leq n \ln L + 1 \end{aligned}$$

by the inequality $\ln z \leq z - 1$ for $z > 0$. Thus, the lemma follows. \square

The most time-consuming part of the above procedure is the computation of $x \in B_{peeled}$. It is assumed that a subroutine to test whether a vector x belongs to the submodular polyhedron P or not is available. Notice that this test can be done in polynomial time by Lemma 5. So, the l and u satisfying (112) can be computed by applying the subroutine $O(n^2 \log L)$ times, because l_j and u_j for each j can be computed by $O(n \log L)$ applications of the subroutine. The computation of $x \in B_{peeled}$ is done by starting from $x \in B$ and iteratively updating x as $x + \alpha(e(i) - e(j))$ for appropriately chosen i, j , and α (see [45] for details of this procedure). Therefore, $x \in B_{peeled}$ can be computed by applying the subroutine $O(n^2 \log L)$ times. Hence, the following theorem follows from Lemma 17.

Theorem 25 *A minimizer of an M -convex function f with n variables can be computed in $O(n^4(\log L)^2 F)$ time, where L is given by (115) and F is the time required to evaluate the function value of f .* \square

7 Applications

As mentioned in Introduction, the resource allocation problem has a wide range of applications. Here, several interesting applications will be explained, to supplement standard ones (see, e.g., [69]).

7.1 Computer Science Applications

1. Optimal Batching Policies for Video-on-Demand Storage Server (Aggarwal, Wolf and Yu [1])

In a video-on-demand environment, *batching* of video requests is often used to reduce I/O demand and improve throughput. The basic idea of batching is to delay the requests of some videos for a certain amount of time so that more requests for the same video arriving during the current *batching interval* may be serviced using the same stream. Thus, when a stream becomes available at the server end, a question arises as to which video is best to schedule at that particular time moment.

There may be several scheduling policies for implementing the batching. Since viewers may defect if they experience long waits, a good video scheduling policy needs to consider not only the batch size but also the viewer defection probabilities and wait times. Two conventional scheduling policies for batching are the *first-come-first-served* (FCFS) policy that always schedules the video with the longest waiting request and the *maximum queue length* (MQL) policy which selects the video with the maximum number of waiting requests. Neither of these policies lead to entirely satisfactory results. MQL tends to be too aggressive in scheduling popular videos, while FCFS has the opposite effect by completely ignoring the queue length.

The problem of determining a best batching policy is formulated as an optimization problem after introducing the following performance measure: Suppose that the server contains a database of n different videos from which the clients may make requests.

- *Average Latency Time:* The latency of a viewer is the period that elapses between the arrival of the video request and the time when the service to the display device is actually initiated.

Assume that the frequency of requests for the j -th video is known and is denoted by f_j . If L_j denotes the length of video j , then $L = \left(\sum_{j=1}^n f_j \cdot L_j \right) / \left(\sum_{j=1}^n f_j \right)$ is the average video length. Assume that the server capacity (in terms of the number of streams) is S . Consequently, the average number of streams which are scheduled by the server every minute at full capacity is S/L . Suppose that t_1, t_2, \dots, t_n are the average time intervals at which the videos $1, 2, \dots, n$ are batched. Then the average latency for a video of type j is equal to $t_j/2$. The objective in this model is to minimize the average latency of the viewers. Thus, modulo a constant that can be ignored, it is desired to minimize the sum of the expected latency times of all the requests which arrive in a unit interval of time: $\sum_{j=1}^n f_j \cdot t_j/2$.

A constraint is needed that limits the number of video streams which can be scheduled at any moment of time. On average, the number of streams for video j scheduled per unit of time is approximately equal to $1/t_j$. (This is not entirely rigorous because $E[1/X] \neq 1/E[X]$ holds for a random variable X , in general. However, if X has a relatively small deviation, the approximation $E[1/X] \approx 1/E[X]$ is fairly accurate.) Thus, the total number of streams of all video types scheduled per unit of time at full capacity is $\sum_{j=1}^n 1/t_j$, which must be equal to S/L . Thus, the following constrained nonlinear program can be obtained:

$$\begin{aligned} & \text{minimize} \quad \sum_{j=1}^n f_j \cdot t_j, \\ & \text{subject to} \quad \sum_{j=1}^n 1/t_j = S/L, \\ & \quad t_j \geq 0, \quad j = 1, \dots, n. \end{aligned}$$

This is a typical problem of SC/Simple/C. There exists the unique optimal solution, which satisfies

$$t_1 \cdot \sqrt{f_1} = t_2 \cdot \sqrt{f_2} = \cdots = t_n \cdot \sqrt{f_n}.$$

In the absence of any defections from the system, the average queue length q_j of the j -th video at the time of batching is equal to $q_j = t_j \cdot f_j$. Thus, from the above equations, it holds the following relationship for the queue lengths at the time of batching:

$$\frac{q_1}{\sqrt{f_1}} = \frac{q_2}{\sqrt{f_2}} = \cdots = \frac{q_n}{\sqrt{f_n}}.$$

In other words, the following intuitive result can be obtained.

Theorem 26 *In order to achieve the best average latency time at the above video-on-demand server, the queue lengths q_j at the time of batching should be approximately proportional to $\sqrt{f_j}$.* \square

Theorem 26 suggests the following greedy scheduling policy, called *maximum factored queue length policy*, or *MFQ policy*, for the video-on-demand problem.

Whenever the server capacity becomes available for scheduling a stream, choose the video with the largest $q_j / \sqrt{f_j}$.

Aggarwal, Wolf, and Yu [1] performed a simulation study to establish the effectiveness of the proposed MFQ policy against existing MQL and FCFS policies from several points of view. Empirical simulations indicate that MFQ yields excellent empirical results in terms of standard performance measures such as average latency time, defection rates, and fairness.

2. Optimal Buffer Partition of the Nested Block Join Algorithm (Wolf, Iyer, Pattipati and Yu [143])

Nested loop is the well-known method for query processing in a relational database, when queries require complex join predicates. There are two alternatives for efficient execution of join operations: *sort merge join* and *nested loop join*. It is said that nested loop join is preferable when only a small amount of main memory is available. Nested loop join is executed as follows: for every row of the outer relation, the inner relation is examined in order to seek for a match. The inner relation is looped over once for every row of the outer relation. This generalizes to the case of more than two relations. If no relation fits into main memory, it is required to fetch each row of the inner relation from disk for each row of the outer relation. This requires too much I/O access.

In order to reduce the number of I/O accesses, nested block join has been proposed by Kim [80]. Suppose that a block (which is usually called a page) is employed as the unit of I/O access. Each relation is partitioned into consecutive blocks of rows of the same size so that the sum of block sizes of two relations does not exceed memory size. One block of each relation is brought into memory. A naive loop join is performed among the rows contained within the blocks in

memory. Suppose that there exist R relations and a total buffer size in memory is B . Let p_i and x_i denote the size of relation i and its block size, respectively, that is, relation i is divided into $\lceil p_i/x_i \rceil$ blocks. Let us assume that relation 1 is outermost, and so on. Since relation 1 is fetched into memory once, the number of I/Os for relation 1 is p_1 . For each block of relation 1, it is needed to fetch relation 2 into memory once. Thus, the number of I/Os for relation 2 is $p_2 \lceil p_1/x_1 \rceil$. Similarly, the number of I/Os for relation 3 is thus $p_3 \lceil p_2/x_2 \rceil \lceil p_1/x_1 \rceil$, and so forth. Thus, the total number of I/Os is given by

$$\sum_{i=1}^R p_i \prod_{j=1}^{i-1} \lceil p_j/x_j \rceil. \quad (116)$$

It is desired to minimize this objective function under the buffer size B . It is easy to see from the form of the objective function (116) that x_R can be set to 1. Thus, the problem is formulated as follows:

$$Q : \text{minimize } \sum_{i=1}^R p_i \prod_{j=1}^{i-1} \lceil p_j/x_j \rceil \quad (117)$$

$$\text{subject to } \sum_{i=1}^{R-1} x_i = B - 1, \quad (118)$$

$$x_i : \text{positive integer, } i = 1, \dots, n. \quad (119)$$

It is assumed here for simplicity that the ordering of relations is fixed. Since the objective function of Q is not separable, it needed a general algorithm such as a dynamic programming algorithm of an $O(RB^2)$ time bound [69], which can be improved to $O(RB^{1.5})$ by using a clever analysis based on the number theory [143]. Here, instead of the original problem, consider the following approximate problem:

$$Q' : \text{minimize } \sum_{i=1}^R p_i \prod_{j=1}^{i-1} p_j/x_j \quad (120)$$

$$\text{subject to } \sum_{i=1}^{R-1} x_i = B - 1, \quad (121)$$

$$x_i : \text{positive integer, } i = 1, \dots, n. \quad (122)$$

Although the objective function is still nonseparable, surprisingly, the incremental algorithm of Sect. 3.1 works for this problem, as will be seen below. Thus, this is another class of problems for which an incremental algorithm works.

Now a class of problems shall be described, including problem Q' , for which the incremental algorithm works. Let a_1, \dots, a_n and b_1, \dots, b_n be integers satisfying

$a_i \leq b_i$ for all i . Let $F(x_1, \dots, x_n)$ be a function defined on $\{a_1, a_1 + 1, \dots, b_1\} \times \{a_2, a_2 + 1, \dots, b_2\} \times \dots \times \{a_n, a_n + 1, \dots, b_n\}$, and let K be an integer satisfying $\sum_{i=1}^n a_i \leq K \leq \sum_{i=1}^n b_i$. Consider the problem R_K of minimizing

$$R_K : \text{minimize } F(x_1, \dots, x_n) \quad (123)$$

$$\text{subject to } \sum_{i=1}^n x_i = K, \quad (124)$$

$$x_i \in \{a_i, a_i + 1, \dots, b_i\}, \quad i = 1, \dots, n. \quad (125)$$

An incremental algorithm for problem R_K starts from $(x_1, \dots, x_n) = (a_1, \dots, a_n)$ and, in each iteration, increments the variable x_{i^*} by one, where i^* is chosen so that the cost increase incurred by the increment of x_{i^*} is minimum among all variables x_i with $x_i < b_i$. This process is repeated until $\sum_{i=1}^n x_i = K$ is satisfied.

Definition 1 Given a function $F(x_1, \dots, x_n)$ with the domain as described above, define, for each K with $\sum_{i=1}^n a_i \leq K \leq \sum_{i=1}^n b_i$, the expression $\hat{F}(K)$ to be equal to the optimal objective value of problem R_K . Thus, $\hat{F}(K)$ is a single-variable function and is called the *surrogate function* of F .

The following theorem can be viewed as describing how to glue together those functions which can be solved by incremental algorithms into a new function which can again be solved by an incremental algorithm.

Theorem 27 (i) For each $i = 1, \dots, M$, let $F_i(x_{i,1}, \dots, x_{i,n_i})$ be a function for which an incremental algorithm can be applied, and suppose that each of the resulting surrogate function $\hat{F}_i(K)$ is convex in K . Then, an incremental algorithm can also be applied to

$$F_\Sigma(\{x_{ij}\}) = \sum_{i=1}^M F_i(x_{i,1}, \dots, x_{i,n_i}),$$

and the surrogate function \hat{F}_Σ of F_Σ is also convex.

(ii) For each $i = 1, \dots, M$, let $F_i(x_{i,1}, \dots, x_{i,n_i})$ be a positive-valued function for which an incremental algorithm can be applied, and suppose that each of the resulting surrogate function \hat{F}_i is logarithmic convex, that is, $\log \hat{F}_i(K)$ is convex in K . Then an incremental algorithm can be applied to

$$F_\Pi(\{x_{ij}\}) = \prod_{i=1}^M F_i(x_{i,1}, \dots, x_{i,n_i}),$$

and the surrogate function \hat{F}_Π is also logarithmic convex.

Proof (i) The fact that an incremental algorithm can be applied is evident by applying [Theorem 2](#) to the objective function, which can be rewritten in convex, separable form as $\sum_{i=1}^M \hat{F}_i(y_i)$, where $y_i = \sum_{j=1}^{n_i} x_{ij}$. To show the convexity of \hat{F}_Σ , let us look at the values of the surrogate function \hat{F}_Σ at three consecutive integers, say K , $K + 1$, and $K + 2$. Suppose

$$\hat{F}_\Sigma(K) = \sum_{j=1}^M \hat{F}_i(y_i)$$

for some choice (y_1, \dots, y_M) whose sum satisfies $\sum_{i=1}^M y_i = K$. Since an incremental algorithm can be applied to the right-hand side, there exists an index $i_1 \in \{1, \dots, M\}$ for which $\hat{F}_\Sigma(K + 1) = \hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1} + 1) + \dots + \hat{F}_M(y_M)$. Similarly, there will exist an index $i_2 \in \{1, \dots, M\}$ which corresponds to the value of $\hat{F}_\Sigma(K + 2)$ when incremented by one more from $(y_1, \dots, y_{i_1} + 1, \dots, y_M)$. There are two cases: either $i_1 = i_2$ or $i_1 \neq i_2$. In the first case, it holds that

$$\begin{aligned} & \hat{F}_\Sigma(K + 2) - \hat{F}_\Sigma(K + 1) \\ &= \hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1} + 2) + \dots + \hat{F}_M(y_M) \\ &\quad - [\hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1} + 1) + \dots + \hat{F}_M(y_M)] \\ &= \hat{F}_{i_1}(y_{i_1} + 2) - \hat{F}_{i_1}(y_{i_1} + 1) \\ &\geq \hat{F}_{i_1}(y_{i_1} + 1) - \hat{F}_{i_1}(y_{i_1}) \\ &= [\hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1} + 1) + \dots + \hat{F}_M(y_M)] \\ &\quad - [\hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1}) + \dots + \hat{F}_M(y_M)] \\ &= \hat{F}_\Sigma(K + 1) - \hat{F}_\Sigma(K). \end{aligned} \tag{126}$$

Here, the inequality at the center is due to the convexity of \hat{F}_{i_1} . In the second case, analogously,

$$\begin{aligned} & \hat{F}_\Sigma(K + 2) - \hat{F}_\Sigma(K + 1) \\ &= [\hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1} + 1) + \dots + \hat{F}_{i_2}(y_{i_2} + 1) + \dots + \hat{F}_M(y_M)] \\ &\quad - [\hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1} + 1) + \dots + \hat{F}_{i_2}(y_{i_2}) + \dots + \hat{F}_M(y_M)] \\ &= \hat{F}_{i_2}(y_{i_2} + 1) - \hat{F}_{i_2}(y_{i_2}) \\ &\geq \hat{F}_{i_1}(y_{i_1} + 1) - \hat{F}_{i_1}(y_{i_1}) \\ &= [\hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1} + 1) + \dots + \hat{F}_M(y_M)] \\ &\quad - [\hat{F}_1(y_1) + \dots + \hat{F}_{i_1}(y_{i_1}) + \dots + \hat{F}_M(y_M)] \\ &= \hat{F}_\Sigma(K + 1) - \hat{F}_\Sigma(K). \end{aligned} \tag{127}$$

Here, the inequality at the center is due to the rule of the incremental algorithm.

The results (126) and (127) tell that $\hat{F}_\Sigma(K)$ is convex in K .

- (ii) Note that minimizing a function amounts to minimizing the logarithm of the function. Since

$$\log \prod_{i=1}^M \hat{F}_i(y_i) = \sum_{i=1}^M \log \hat{F}_i(y_i), \quad (128)$$

and $\log \hat{F}_i(y_i)$ is convex by assumption, part (ii) follows from part (i). \square

A positive logarithmic convex function over the integers can be shown to be convex. This follows easily from the fact that the arithmetic mean of two numbers exceeds the geometric mean. But the converse is not true. So the convexity condition in part (ii) of [Theorem 27](#) is slightly stronger than that of part (i). Note that, functions of the form p/K , where K is a positive integer, are logarithmic convex. Furthermore, adding a positive constant to a logarithmic convex function yields a new logarithmic convex function. Hence the following theorem is established.

Theorem 28 *An incremental algorithm of [Sect. 3.1](#) correctly solves problem Q' of (120).*

Proof Rewrite the objective function of problem Q' of (120) as

$$p_1 + \frac{p_1}{x_1} \{ p_2 + \frac{p_2}{x_2} \{ p_3 + \frac{p_3}{x_3} \{ p_4 + \cdots + \frac{p_{R-2}}{x_{R-2}} \{ p_{R-1} + \frac{p_{R-1}}{x_{R-1}} p_R \} \cdots \} \} \cdots \}. \quad (129)$$

Notice that a function $f(x) = a/x + b$ for constants $a > 0$ and $b \geq 0$ is logarithmic convex and that the product of two such functions is also logarithmic convex. Thus, applying this argument from the innermost terms of (129) out to the outermost, it follows that the function of (129) is logarithmic convex. Thus, from [Theorem 27](#) (ii), the theorem follows.

The computational complexity of this procedure is $O(RB)$ since employing [Theorem 27](#) (ii) at one stage takes $O(R)$ time, and there are $O(B)$ such stages.

Notice that the polynomial algorithm of [Sect. 3.1.2](#) for SC/Simple/D cannot be applied to problem Q' and it is not known yet whether Q' can be solved in polynomial time or not. However, to the authors' knowledge, Q' is the first nonseparable convex resource allocation problem that has been shown to be solvable by an incremental algorithm.

7.2 Reliability Applications

1. Optimal Allocation for Software-Testing Resources (Ohtera and Yamada [115])

During the module testing stage in the software development process, the manager has to determine an optimal allocation of resources such as manpower,

CPU hours, test cases to be executed, and so forth, so that the software quality and reliability are maximized. This problem is modeled as a resource allocation problem on the basis of the software reliability growth model developed by Kubat and Koch [90], which describes the relationship between the applied resource and the detected software errors. In the model of [90], the average of the cumulative number of software errors $m(t)$ detected in the time interval $(0, t]$ is expressed in terms of $W(t)$, the cumulative resource expenditures used until time t , that is,

$$m(t) = a(1 - \exp[-rW(t)]), \quad (130)$$

where a denotes the mean number of initial errors and r (> 0) denotes the error detection rate per unit resource expenditure.

Based on this model, a resource allocation problem can be defined, which minimizes software errors under the following assumptions:

1. The software system is composed of M independent modules. The number of software errors remaining in each module can be estimated by the model of [90].
2. The manager has to allocate the total amount of testing-resource R to each software module to minimize the number of software errors remaining in the system.

From (130), the number of software errors remaining in module j is estimated as:

$$z_j = a_j \cdot \exp[-r_j q_j], \quad j = 1, 2, \dots, M, \quad (131)$$

where a_j and r_j denote the average of initial number of errors in module j and the error detection rate per unit resource expenditure for module j , respectively.

Thus, the testing-resource allocation problem is formulated as

$$\text{minimize} \sum_{j=1}^M v_j a_j \cdot \exp[-r_j q_j] \quad (132)$$

$$\text{subject to} \sum_{j=1}^M q_j = R, \quad (133)$$

$$q_j \geq 0, \quad j = 1, 2, \dots, M, \quad (134)$$

where v_j stands for the relative importance of module j . This problem is a special case of SC/Simple/C.

7.3 Production Planning Applications

To illustrate rich applications in production planning as mentioned in [69], a new application is mentioned below.

1. Service-Constrained Optimal Policy for Spare Parts (Cohen, Kleindorfer and Lee [30])

The problem is to determine the optimal policy for a facility that stocks various spare parts for repairs to satisfy an overall service level requirement.

Suppose that a product is made up of n field replaceable units (FRUs for short), denoted by subscripts $i \in N = \{1, 2, \dots, n\}$. Demands for parts in any given period are represented by nonnegative integer-valued independent random variables $\{D_i \mid i \in N\}$ whose probability density function and cumulative distribution function are denoted by f_i and F_i , respectively. It is asked to determine the amounts of stocks $S = \{S_i \mid i \in N\}$ at the beginning of a period, where S_i are nonnegative integers, so as to minimize the expected holding and ordering costs while meeting the specified service levels for products and customers. Stock replenishment occurs instantaneously (i.e., zero lead time) at the end of each period to restore the stock levels depleted through the period demands.

The objective is to minimize the expected total cost per period:

$$\text{minimize } G(S) = \sum_{i \in N} G_i(S_i), \quad (135)$$

where $G_i(S_i)$ is the expected cost of FRU $_i$ per period defined by

$$\begin{aligned} G_i(S_i) = & E\{K_i \delta(D_i)\} && \text{ordering cost} \\ & + (C_{ih}/2)[S_i + (S_i - D_i)^+] && \text{holding cost} \\ & + C_{it} \min[S_i, D_i] && \text{transportation cost} \\ & + C_{is}(D_i - S_i)^+\}. && \text{shortage cost} \end{aligned} \quad (136)$$

Here, $x^+ = \max(x, 0)$ and $\delta(x) = 0$ (respectively, 1) if $x \leq 0$ (respectively, if $x > 0$). In (136), K_i is a fixed order cost, which will be neglected since it is independent of the stock policy S_i . The holding cost is the average of the initial inventory S_i and the final inventory $(S_i - D_i)^+$ times the holding cost per unit C_{ih} . Transportation cost per unit C_{it} denotes the cost of issuing an FRU, if available, and transporting it to the customer. Finally, the shortage cost C_{is} is associated with the incremental cost of meeting the demand from a remote location. It is assumed that

$$C_{is} \geq C_{it} > 0 \text{ and } C_{ih} > 0 \quad (137)$$

hold for all i . It is desired to minimize (135) subject to the following service constraints:

$$Pr\left\{\sum_{i \in N}(D_i - S_i)^+ > 0\right\} \leq \beta. \quad (138)$$

This tells that, in the long run, the excess demand occurs for at most a fraction β of the period. This type of constraint is often used in this field. Now note that the event $\{\sum_{i \in N}(D_i - S_i)^+ > 0\}$ is equivalent to the event $\{\cup_{i \in N}[D_i > S_i]\}$. Moreover, $Pr\{\sum_{i \in N} D_i > 0\} = 1 - \prod_{i \in N} F_i(0)$. Thus, (138) can be expressed as

$$1 - \prod_{i \in N} F_i(S_i) \leq \beta[1 - \prod_{i \in N} F_i(0)].$$

Assuming that $\alpha = 1 - \beta[1 - \prod_{i \in N} F_i(0)]$ is nonzero, the following equivalent, separable constraint can be obtained:

$$\sum_{i \in N} \log(F_i(S_i)) \geq \log \alpha. \quad (139)$$

Therefore, the above problem has a separable objective function and a separable nonlinear constraint. Since the derivative of $G_i(S_i)$ is shown to be

$$(C_{is} - C_{it} + C_{ih}/2)F_i(S_i) - (C_{is} - C_{it} - C_{ih}/2),$$

it follows that $G_i(S_i)$ is convex because $C_{is} - C_{it} + C_{ih}/2 > 0$ from (137) and $F_i(S_i)$ is nondecreasing in S_i .

Efficient solution algorithm similar to the incremental algorithm has been developed by Cohen et al. [30] although the constraint (139) is nonlinear.

7.4 Other Applications

7.4.1 An Optimal Scheduling of the Mass Screening Tests

The policy maker is faced with designing the mass screening programs for contagious diseases by taking into account the trade-off between the frequency of test applications and the types of tests used (as different technologies may have different reliability characteristics and costs), against the benefits to be achieved from detecting the defect in an earlier stage of development. Moreover, because different subpopulations may have different susceptibility to the disease, the problem of optimal allocation of a fixed testing budget among subpopulations must be considered [93]. This problem will be formulated as SC/Simple/C under the assumptions stated below:

1. The latent period of the disease is negligible.
2. The infectious period of the disease is exponentially distributed. Note that the termination of the infectious period can be a result of natural deaths.
3. The size of the population is large enough so that the number of susceptibles is relatively constant.
4. The incidence rates and the rates of transmission of the disease are stationary over time and are independent of each other.
5. The probability of an increase in the number of infected units at time t is directly proportional to the number of units in the infectious period of the disease at time t .
6. Once an infected unit is identified as having the disease after a mass screening test, it will be removed from the population and subsequently treated. As a result, it is assumed to be no longer infectious.

The assumption that the infectious periods are exponentially distributed is common in most of the related models. Assumption 5 is also generally used in epidemics. The following notation is used:

λ : natural incidence rate of the disease for the unit

γ : rate of transmission of the disease from a contagious unit to a susceptible

μ : rate of infected units ending the infectious period

In the time interval $[t, t']$ consisting of two consecutive mass screening tests, the number of contagious units is considered to be a stochastic process described by

$$X(t + \Delta t) = \lambda \Delta t + \gamma X(t) \Delta t - \mu X(t) \Delta t.$$

From this equation and the above assumptions, a closed-form expression can be obtained for the average number of infected units over $[t, t']$. Now suppose that it is required to perform M mass screening tests over the planning time horizon $[0, T]$. In [93], it is shown that the best scheduling policy for M tests is to perform the tests with equal interval lengths. Then, it has been shown that the average number of infected units in the population over a planning horizon of T time units can be written as

$$[\lambda/(\gamma - \mu)^2](M/T)\{\exp[(\gamma - \mu)(T/M)] - 1\} - [\lambda/(\gamma - \mu)], \quad (140)$$

where M is the number of mass screening tests performed over the planning time horizon $[0, T]$. As easily verified, given T , (140) is decreasing in M .

Now, suppose that each mass screening costs C , and B is the amount that the society can afford or is willing to pay per unit time for mass screening. Then, given a budget, the society would try to increase M as much as possible. For argument's sake, suppose M is increased so that $M/T = B/C$ holds. The average number of infected units in the population under the optimal screening schedule becomes as follows:

$$\psi(B) = [\lambda/(\gamma - \mu)^2](B/C)\{\exp[(\gamma - \mu)(C/B)] - 1\} - [\lambda/(\gamma - \mu)]. \quad (141)$$

As $T \rightarrow \infty$, (141) can be viewed as the long-range average number of infected units in the population under the optimal mass screening schedule. Notice that $\psi(B)$ is convex in B .

Suppose that there are J subpopulations, where there is little or no interaction among the subpopulations as far as the transmission of the disease is concerned. Thus, each subpopulation may be viewed as an independent population in terms of the etiology of the disease under screening. Let $D_j(\cdot)$ be the respective utility function for subpopulations j . Suppose B_j and C_j are the amount per unit time allotted to mass screening and the cost of each mass screening for subpopulation j , respectively. Let B be the amount per unit time available to all the subpopulations.

Thus, the problem facing the policy maker is to determine the B_j 's by solving the following resource allocation problem of type SC/Simple/C to maximize the objective function $f(B_1, B_2, \dots, B_J)$ under the assumption that $D_j(y)$ is concave in y (i.e., to minimize a convex function $-f(B_1, B_2, \dots, B_J)$):

$$\begin{aligned} & \text{maximize } f(B_1, B_2, \dots, B_J) \\ &= \sum_{j=1}^J D_j \left(\frac{\lambda_j}{(\gamma_j - \mu_j)^2} \left(\frac{B_j}{C_j} \right) \cdot \left[\exp \left\{ (\gamma_j - \mu_j) \left(\frac{C_j}{B_j} \right) \right\} - 1 \right] - \frac{\lambda_j}{\gamma_j - \mu_j} \right) \\ & \text{subject to } \sum_{j=1}^J B_j = B, \\ & \quad B_j \geq 0; \quad j = 1, \dots, J. \end{aligned}$$

7.4.2 Minimum Entropy Orientation Problem

Most of the resource allocation problems discussed so far mainly deal with separable *convex* functions as objective functions, which are to be minimized. In contrast, the application explained below gives an interesting example of a resource allocation problem, where a separable *concave* function is minimized.

Let $G = (V, E)$ be an undirected graph which may have parallel edges but no self loop. A directed graph $D_G = (V, A)$ is called an *orientation* of G if D_G is obtained from G by orienting each edge $(u, v) \in E$ as a (directed) arc (u, v) or (v, u) . For the directed graph D_G and a vertex $v \in V$, denote by $\rho_{D_G}(v)$ the in-degree of v in D_G , that is, the number of arcs in D_G which enter the vertex v . A vector $p \in R^V$ given by $p_v = \rho_{D_G}(v)/|E|$ for $v \in E$ is called the *in-degree distribution* of D_G .

The *minimum entropy orientation* problem (MINEO) is the problem of finding an orientation of G for which the in-degree distribution is as unbalanced as possible. More formally, the objective function of MINEO is given by

$$f(D_G) = - \sum_{v \in V} \frac{\rho_{D_G}(v)}{|E|} \log \frac{\rho_{D_G}(v)}{|E|},$$

which is minimized among all orientations D_G of G . Notice that $g(\alpha) = -(\alpha/|E|) \log(\alpha/|E|)$ is a concave function in α .

The problem MINEO is firstly considered by Cardinal et al. [25] as a special case of a problem called the minimum entropy set cover problem (MINESC). The input of MINESC is a ground set U and a family $\mathcal{S} = \{S_1, S_2, \dots, S_q\}$ of subsets of U . The output is an assignment (or a function) $\pi : U \rightarrow \{1, \dots, q\}$, which minimizes the *entropy*

$$-\sum_{i=1}^q \frac{|U_i|}{|U|} \log \frac{|U_i|}{|U|},$$

where $S_i = \{v \in S \mid \pi(v) = i\}$ ($i = 1, \dots, q$). It is not difficult to see that MINEO is a special case of MINESC by setting $U = E$ and $\mathcal{S} = \{S_v \mid v \in V\}$, where $S_v = \{e \in E \mid e \text{ is incident to } v\}$ for $v \in V$.

MINESC is introduced by Halperin and Karp [59], due to a motivation to solve a certain haplotyping problem, which is of importance in computational biology. The haplotyping problem involves covering a set U of *partial haplotypes* of length d , defined as words in the set $\{0, 1, *\}^d$, by *complete haplotypes*, defined as words in $\{0, 1\}^d$. Each subset S_i in \mathcal{S} corresponds to a complete haplotype $a \in \{0, 1\}^d$ and is given as the set of partial haplotypes b in U that are *compatible* with a , that is, $b_j = a_j$ for all $j = 1, \dots, d$ such that $b_j \neq *$. It is shown that, under some probabilistic assumptions, minimizing the entropy amounts to maximizing its likelihood [59].

In this context, MINEO corresponds to a special case of the haplotyping problem, where each partial haplotype has exactly one “*” in it. In this case, partial haplotypes correspond to edges of a d -dimensional hypercube, and the subsets U_i correspond to vertices of the hypercube. That is, this special case of the haplotyping problem is a minimum entropy orientation problem in a partial hypercube.

Cardinal et al. [25] show that MINEO is NP-hard by using the reduction of the 1-in-3 satisfiability problem. On the other hand, it is shown that an approximate solution with additive error of one (i.e., the objective function value is at most [the optimal value of MINEO] + 1) can be computed in linear time by a simple algorithm. In addition, it is presented in [25] that MINEO can be formulated as the minimization of a separable *concave* function on a base polyhedron, as explained below.

Consider an undirected graph $G = (V, E)$ which is an instance of MINEO. For $S \subseteq V$, let $r(S)$ be the number of edges in E incident to S . Then, $r : 2^V \rightarrow \mathbb{Z}$ is a nondecreasing submodular function. The set of orientations of G is deeply related to the base polyhedron $B(r)$ of the submodular system $(2^V, r)$, and it is known that an integral vector $x \in \mathbb{Z}^V$ is in $B(r)$ (i.e., x is an integral base of $B(r)$) if and only if there exists an orientation D_G of G such that $\rho_{D_G}(v) = x_v$ for all $v \in V$ (see, e.g., [124, Chap. 61]). Hence, MINEO can be formulated as follows:

$$\begin{aligned} & \text{minimize} \sum_{v \in V} (-1) \cdot \frac{x_v}{|E|} \log \frac{x_v}{|E|} \\ & \text{subject to } x : \text{ an integral base of } B(r), \end{aligned}$$

which is the minimization of a separable concave function under submodular constraints. It should be noted that the maximization version of MINEO is a special case of SC/SM/D and therefore can be solved in polynomial time by using the results in Sect. 4.

7.5 Notes and References

There are many other applications of the resource allocation problem. Among them, applications to computer science are found in [2, 28, 95, 101, 136–141, 144, 150].

Biller et al. [10] considered an optimal dynamic pricing problem which can be formulated as a problem of SC/SM/D.

Baldick and Wu [7] considered the problem of an optimal design of coordination of capacitors and regulators in which they applied the algorithm by Hochbaum and Shanthikumar [65].

Carrizosa and Morales [26] studied the sample allocation problem for stratified simple random sampling that is to determine sample sizes drawn from each stratum. The paper considered two objectives: the total sampling cost and the variance of the so-called Horvitz–Thompson estimator. It then proposed a method to compute a set of Pareto optimal solutions.

A biobjective method for sample allocation in stratified sampling, Eu [38] considered an optimal allocation of sample sizes in a real-time monitoring system to detect errors of digital signals in telecommunication systems, by sampling, so as to minimize the sum of the so-called bit error ratios of multiple signals. He formulated the problem as SC/LUB/C.

Li and Haimes [94] considered an optimization problem that arises in designing a high-reliability large-scale system that is also viewed as a variant of the resource allocation problem.

Shanthikumar and Yao [126] considered the optimal control problem for a class of multi-class queueing systems. They found a certain submodular structure of performance vectors that allows us to efficiently compute an optimal performance vector.

Shanthikumar and Yao [125] studied the resource allocation problem in queueing networks, which models certain types of manufacturing systems where each node represents a machine or a workstation. It seeks to find an optimal allocation of servers to the stations in order to maximize the throughput, under the constraint that the total number of servers to be allocated is fixed.

Bitran and Tirupati [12, 13] studied a manufacturing capacity planning problem in manufacturing systems that are also regarded as queueing networks (see also Buzacott and Shanthikumar [24]). The problem involves the minimum-cost selection of the service rate (or capacity) at each workstation, subject to an upper limit on the total dollar value of work in process in the system. Under certain assumptions, they have formulated the problem as the separable convex resource allocation problem under a separable convex constraint. They presented efficient heuristic algorithms for the continuous variable case [12] and for the discrete variable case [13].

Resource allocation problems with nonlinear constraints in production planning under uncertainty are often encountered. One such application is found in [145]. This topic will be discussed again in Sect. 8.3.

The quickest flow problem on a grid network which arises in determining an optimal evacuation planning in cities was studied by Kamiyama, Katoh, and Takizawa [73]. The paper formulated the problem to find the optimal allocation of evacuees to paths to a safety place as Minimax/Simple/D and gave a polynomial-time algorithm.

See also [9, 18, 27, 29, 121, 132, 134] for other applications of resource allocation problems.

8 Further Topics

This section deals with some extensions and generalizations of resource allocation problems not discussed so far.

The first topic is on the multiple resource allocation problem in which each activity requires more than one type of resource. The modeling capability is substantially enhanced by this generalization. An application of this problem is typically found in production planning to find an optimal allocation of many types of resources such as raw materials, machines, and manpower among competing production lines so as to meet the demands. This problem has been well studied [82–86, 97–99, 102, 104, 135]. For most of multiple resource allocation problems discussed in the literature, the performance of each activity (the production level of a certain product in the context of production planning) is measured by the deviation of the planned activity level (the amount of products to be produced) from a given demand. Minimax objective is usually adopted because minimizing the maximum deviation among all activities balances deviations over all activities and is thus intuitively appealing to practitioners engaged in production planning. Also, decision variables can be usually regarded as continuous. Efficient algorithms for such problems have been developed. Among them two algorithms proposed by Luss [97, 98] shall be reviewed in the next subsection.

The second topic is on the multiperiod resource allocation problem. A typical application is also found in production planning. For each activity, a cumulative target level is prespecified. Each decision variable represents the cumulative level selected for an activity up to some period. Similarly to the multiple resource allocation problem, this type of problem is usually formulated as the one with minimax objective function and continuous variables. This will be discussed in Sect. 8.2.

The third topic deals with the resource allocation problem under a single nonlinear constraint. An application of this problem is mentioned in Sect. 7.3, and, as pointed out in the survey paper [17] by Brethauer and Shetty, this type of problem has many applications in various fields. This topic will be dealt with in Sect. 8.3.

Finally, in Sect. 8.4 miscellaneous problems related to resource allocation shall be discussed, including (1) multiple resource allocation problem with smoothing objectives and (2) minimum variance resource allocation problem.

8.1 Multiple Resource Allocation

As remarked above, the multiple resource allocation problem usually has a minimax objective function and continuous variables. A problem formulation is given first, and then two algorithms are explained briefly. The following notation is used:

- i : index for resources, $i = 1, 2, \dots, m$
- j : index for activities, $j = 1, 2, \dots, n$
- a_{ij} : amount of resource i used by one unit of activity j ; $a_{ij} \geq 0$
- b_i : amount of available resource i ; $b_i > 0$
- x_j : activity level to be chosen for j ; $x_j \geq 0$
- $f_j(x_j)$: cost function associated with activity j

The problem is formulated as follows:

$$\begin{aligned} MRP : \quad & \text{minimize} \quad \max_{1 \leq j \leq n} f_j(x_j) \\ & \text{subject to} \quad \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, 2, \dots, m, \\ & \quad x_j \geq 0, \quad j = 1, 2, \dots, n. \end{aligned} \tag{142}$$

It is assumed that $f_j(x_j)$ is strictly decreasing and continuous. Denote the inverse function of $f_j(x_j)$ by $f_j^{-1}(\cdot)$, that is, $f_j(x_j) = y$ implies $f_j^{-1}(y) = x_j$. Note that $f_j^{-1}(y)$ is also strictly decreasing and continuous. Let the indices j be reordered so that

$$f_1(0) \leq f_2(0) \leq \dots \leq f_n(0)$$

holds. In the following, denote the optimal objective value by V^* .

Theorem 29 *There exists an optimal solution x^* of MRP with the objective value V^* , such that there exists a set $J^* = \{j^*, j^* + 1, \dots, n\}$ satisfying*

$$x_j^* > 0, \quad j \in J^*, \tag{143}$$

$$x_j^* = 0, \quad j \notin J^*, \tag{144}$$

and

$$f_j(x_j^*) = V^*, \quad j \in J^*, \tag{145}$$

$$f_j(0) \leq V^*, \quad j \notin J^*. \tag{146}$$

Proof Suppose there is an optimal solution x^* of problem MRP with $x_{j_1}^* = 0$ and $x_j^* > 0$ for some $j < j_1$. Since $f_j(0) \leq f_{j_1}(0) \leq V^*$ for all $j \leq j_1$ and all a_{ij} are nonnegative, reducing x_j^* to zero for all $j < j_1$ preserves feasibility and optimality; so (143) and (144) hold for some J^* . Now, suppose $f_j(x_j^*) < V^*$ for some $j \in J^*$. Reducing each such x_j^* until $f_j(x_j^*) = V^*$ or until x_j^* reaches zero,

whichever occurs first, preserves feasibility and optimality; so (145) holds. (Note that if $x_{j_1}^*$ is reduced to zero, then all x_j^* 's for $j \leq j_1$ are also reduced to zero, so that (143) and (144) holds.) Since $f_j(x_j^*) \leq V^*$ for all j , (144) implies (146). \square

An optimal solution for MRP is not necessarily unique. However, for an optimal solution that satisfies (143)–(146) none of the x_j^* 's can be reduced without violating the nonnegativity constraints or optimality. Thus, there is a unique optimal set J^* that satisfies (143)–(146). Since $f_j(x_j)$ is strictly decreasing, (143)–(146) also imply

$$f_{j^*}(0) > V^*, \quad (147)$$

$$f_{j^*-1}(0) \leq V^*. \quad (148)$$

If $j^* = 1$ holds here, only (147) is relevant. Note that the following property holds by definition of J^* :

$$f_{j^*}(0) = \min_{j \in J^*} f_j(0) \text{ and } f_{j^*-1}(0) = \max_{j \notin J^*} f_j(0).$$

The first algorithm solves the problem by identifying the above set J^* , by considering a sequence of relaxed versions MRP_h , $h = 1, 2, \dots, n$, each obtained by removing nonnegativity constraints $x_j \geq 0$. That is, MRP_h is defined on set $J_h = \{h, h+1, \dots, n\}$ of activities as follows:

$$\text{MRP}_h : \text{ minimize } \max_{j \in J_h} f_j(x_j) \quad (149)$$

$$\text{subject to } \sum_{j \in J_h} a_{ij} x_j \leq b_i, \quad i = 1, 2, \dots, m. \quad (150)$$

Since all f_j are decreasing and all a_{ij} are nonnegative, at least one of the constraints (150) holds with equality for an optimal solution of MRP_h . Thus, from (145), in order to solve MRP_h , values k_i ($i = 1, 2, \dots, m$) are first computed such that

$$\sum_{j \in J_h} a_{ij} f_j^{-1}(k_i) = b_i, \quad i = 1, 2, \dots, m. \quad (151)$$

The optimal objective value V_h^* of MRP_h is then obtained by

$$V_h^* = \max_i k_i.$$

If

$$f_h(0) > V_h^* \text{ and } f_{h-1}(0) \leq V_h^* \quad (152)$$

hold for $J_h = \{h, h+1, \dots, n\}$, then this J_h gives the optimal set J^* . Furthermore, the optimal objective value V^* of MRP is equal to V_h^* , and the positive x_j^* 's are

given by $x_j^* = f_j^{-1}(V_h^*)$. By [Theorem 29](#), there exists an h satisfying (152), thus yielding an algorithm to compute J^* .

As shown in [97], for certain nonlinear functions, the k_i 's are given as closed-form expressions. For example, this is possible if

$$f_j(x_j) = \beta_j(x_j + \gamma_j)^{-\delta} \text{ for all } j,$$

where $\beta_j > 0$, $\gamma_j \geq 0$ and $\delta > 0$. However, in general (e.g., the above δ is replaced by δ_j), solving equation (151) requires numerical methods such as Newton method.

Another algorithm is now explained that facilitates the search for J^* without explicitly solving equations of the type (151). Some properties used for this purpose are presented below.

Theorem 30 *The optimal set $J^* = \{j^*, j^* + 1, \dots, n\}$ of problem MRP satisfies the following conditions:*

$$\sum_{j \in J^*} a_{ij} f_j^{-1}(f_{j^*}(0)) < b_i \text{ for } i = 1, 2, \dots, m. \quad (153)$$

Furthermore, if $j^* > 1$, it holds that

$$\sum_{j \in J^*} a_{ij} f_j^{-1}(f_{j^*-1}(0)) \geq b_i \text{ for some } i. \quad (154)$$

Proof By (142) and (145) and the nonnegativity of a_{ij} , it holds that

$$\sum_{j \in J^*} a_{ij} x_j^* = \sum_{j \in J^*} a_{ij} f_j^{-1}(V^*) \leq b_i$$

for all i . Since $f_{j^*}(0) > V^*$ holds by (147), $f_j^{-1}(y)$ is strictly decreasing, and all a_{ij} 's are nonnegative, it follows

$$\sum_{j \in J^*} a_{ij} f_j^{-1}(f_{j^*}(0)) < \sum_{j \in J^*} a_{ij} f_j^{-1}(V^*) \leq b_i,$$

which then implies (153). To prove (154), note that there is some i , say i' , such that

$$\sum_{j \in J^*} a_{i'j} x_j^* = b_{i'}.$$

Furthermore, since (148) implies $f_{j^*-1}(0) \leq V^*$, it holds that

$$f_j^{-1}(f_{j^*-1}(0)) \geq f_j^{-1}(V^*).$$

Therefore,

$$\begin{aligned} \sum_{j \in J^*} a_{i'j} f_j^{-1}(f_{j^*-1}(0)) &\geq \sum_{j \in J^*} a_{i'j} f_j^{-1}(V^*) \\ &= \sum_{j \in J^*} a_{i'j} x_j^* = b_{i'} \end{aligned}$$

follows. \square

Remark 2 If $J^* = \{n\}$, condition (153) is always satisfied since $b_i > 0$ for all i and $f_{j^*}^{-1}(f_{j^*}(0)) = 0$.

Theorem 31 For problem MRP, if a set $J_h = \{h, h+1, \dots, n\}$ does not satisfy (153) (i.e., $\sum_{j \in J_h} a_{ij} f_j^{-1}(f_h(0)) \geq b_i$ for some i), it holds that

$$J^* \subset J_h.$$

Proof Let $J_l = \{l, l+1, \dots, n\}$ be any set such that $J_l \supset J_h$ (i.e., $l < h$) and suppose that $J_l = J^*$, that is, J_l satisfies (153) (i.e., $\sum_{j \in J_l} a_{ij} f_j^{-1}(f_l(0)) < b_i$ for all i). This implies that

$$\begin{aligned} \sum_{j \in J_h} a_{ij} f_j^{-1}(f_h(0)) &\leq \sum_{j \in J_h} a_{ij} f_j^{-1}(f_l(0)) \\ &\leq \sum_{j \in J_l} a_{ij} f_j^{-1}(f_l(0)) < b_i \end{aligned}$$

holds for all i . The first inequality follows since $l < h$ implies $f_l(0) \leq f_h(0)$ and $f_i^{-1}(\cdot)$ is strictly decreasing. The second inequality follows since

$$f_j^{-1}(f_l(0)) \geq 0 \text{ for all } j \geq l$$

holds from $f_j^{-1}(f_j(0)) = 0$, $f_j(0) \geq f_l(0)$ for $j \geq l$ and the monotone decreasingness of $f_i^{-1}(\cdot)$. These inequalities tell that J_h also satisfies (153), which is a contradiction.

Hence, $J^* \neq J_l$ holds for all $l < h$, from which $J^* \subset J_h$ follows since $J^* \neq J_h$. \square

Theorem 32 For problem MRP, if a set $J_h = \{h, h+1, \dots, n\}$ (for $h \geq 2$) does not satisfy condition (154) (i.e., $\sum_{j \in J_h} a_{ij} f_j^{-1}(f_{h-1}(0)) < b_i$ for all i), then it holds that

$$J^* \supset J_h.$$

Proof Let $J_l = \{l, l+1, \dots, n\}$ satisfy $J_l \subset J_h$ (i.e., $l > h$). Then

$$\begin{aligned} \sum_{j \in J_l} a_{ij} f_j^{-1}(f_{l-1}(0)) &\leq \sum_{j \in J_l} a_{ij} f_j^{-1}(f_{h-1}(0)) \\ &\leq \sum_{j \in J_h} a_{ij} f_j^{-1}(f_{h-1}(0)) < b_i \end{aligned}$$

holds for all i . The first inequality is from $f_{h-1}(0) \leq f_{l-1}(0)$. The second inequality holds since

$$f_j^{-1}(f_{h-1}(0)) \geq 0 \text{ for } j \geq h$$

by assumption $f_j^{-1}(f_j(0)) = 0$, $f_{h-1}(0) \leq f_j(0)$ for $j \geq h$ and the monotone decreasingness of $f_j^{-1}(\cdot)$. Hence, J_l does not satisfy (154), implying that $J^* \supset J_h$. \square

As a direct consequence of [Theorems 31](#) and [32](#), J^* can be computed efficiently by applying the binary search over the index set $\{1, 2, \dots, n\}$ using (153) and (154). It should be remarked here that, unlike conditions (147) and (148), testing whether a set J^* satisfies (153) and (154) does not require the costly computation of V_h^* , which is obtained by solving nonlinear equations (151) as in the first algorithm. Instead, it suffices to solve equations (151) only once, in order to determine V^* after the J^* is found. This will greatly reduce the computation time.

8.2 Multiperiod Resource Allocation

Consider the multiperiod resource allocation problem that seeks to find an optimal allocation of resources over multiple periods so as to minimize the cost incurred by the resulting allocation.

As in the problem discussed in the previous subsection, activities may require more than one resource type. It is assumed that all resources are storable, that is, excess resources in one period can be used in the subsequent periods.

As mentioned at the beginning of this section, a cumulative target level is prespecified for each activity. This means that each decision variable represents the cumulative level selected for the corresponding activity up to some period, implying that there exists an ordering constraint on the decision variables for each activity telling that the cumulative activity levels are nondecreasing over time. All variables are assumed to be continuous. The objective is again Minimax, as in the previous subsection, that is, the maximum among all costs of activities is minimized.

Suppose that a set of resources I and a set of activities A are given. Resources in I are represented by index i , and activities in A are represented by index j . The model considers discrete time periods $t = 1, 2, \dots, \tau$ and $T = \{1, 2, \dots, \tau\}$. Activity i at time t is represented by two indices (j, t) for $j \in A$ and $t \in T$. The set of all activity-period pairs is denoted by AP . The following additional notations are used:

- a_{ijt} : amount of resource i used by one unit of activity (j, t) ; $a_{ijt} \geq 0$
- b_i : amount available of resource i ; $b_i > 0$

- x_{jt} : cumulative level selected for activity (j, t)
 l_{jt} : lower bound on the level selected for activity (j, t) ; $l_{jt} \geq 0$
 u_{jt} : upper bound on the level selected for activity (j, t) ; $u_{jt} \geq 0$
 $f_{jt}(x_{jt})$: performance function associated with activity (j, t)

The multiperiod minimax resource allocation problem is formulated as follows:

$$\text{MPMRP : minimize } \max_{(j,t) \in AP} f_{jt}(x_{jt}) \quad (155)$$

$$\text{subject to } \sum_{(j,t) \in AP} a_{ijt} x_{jt} \leq b_i, \quad i \in I, \quad (156)$$

$$x_{jt} \geq x_{j,t-1}, \quad (j, t) \in AP, \quad (157)$$

$$l_{jt} \leq x_{jt} \leq u_{jt}, \quad (j, t) \in AP, \quad (158)$$

where $x_{j0} = 0$ is assumed for all $j \in A$ by convention. Denote the optimal objective value by V^* and assume that $f_{jt}(x_{jt})$ is strictly decreasing and continuous in x_{jt} . The inverse function of f_{jt} is denoted by f_{jt}^{-1} .

Betts, Brown, and Luss [8] has developed an efficient algorithm for this problem, based on the following theorem (the proof is omitted here; see [8] for its proof). For a given constant $V \in R$, we call V *feasible* if there exists a feasible solution x of MPMRP such that the objective value is less than or equal to V .

Theorem 33 *Given a $V \in R$ for problem MPMRP, let $x(V) = \{x_{jt}(V) \mid (j, t) \in AP\}$ satisfy the following:*

$$x_{jt}(V) = \max\{l_{jt}, x_{j,t-1}(V), f_{jt}^{-1}(V)\}, \quad (j, t) \in AP. \quad (159)$$

(This $x(V)$ can be obtained by executing (159) in the increasing order of t , starting from $x_{j0}(V) = 0$, $j \in A$.) Then

- (i) *V is feasible to problem MPMRP if and only if $x(V)$ satisfies constraints (156)–(158).*
- (ii) *If $V = V^1$ is feasible to problem MPMRP and $V = V^2$ is not, then $V^2 < V^1$ holds.* \square

Theorem 33 immediately suggests a binary search method to find an optimal solution of MPMRP. Since we can compute $x(V)$ using the recursive formula (159), the entire algorithm is efficient.

8.3 Minimizing a Separable Convex Function Under a Nonlinear Constraint

The problem considered here has a single nonlinear constraint together with lower and upper bounds on all variables:

$$\text{SC/NLP/D : } \underset{\mathbf{x}}{\text{minimize}} \quad \sum_{j=1}^n f_j(x_j) \quad (160)$$

$$\text{subject to} \quad \sum_{j=1}^n g_j(x_j) \leq b, \quad (161)$$

$$l_j \leq x_j \leq u_j, \quad j = 1, \dots, n, \quad (162)$$

$$x_j : \text{integer}, \quad j = 1, \dots, n, \quad (163)$$

where $f_j(x_j)$ and $g_j(x_j)$, $j = 1, \dots, n$ are differentiable convex functions over R , b is a constant, and l_j and u_j ($l_j < u_j$), $j = 1, \dots, n$ are lower and upper bounds on the integer variables. We assume that the feasible region is nonempty and bounded. Note that the assumptions on $f_j(x_j)$ and $g_j(x_j)$ imply that the continuous relaxation of SC/NLP/D (denoted by SC/NLP/C) is a convex program.

Bretthauer and Shetty [14, 17] pointed out that both problems SC/NLP/D and SC/NLP/C have a wide variety of applications, including production planning, capital budgeting, capacity planning in manufacturing networks, capacity planning in computer networks, stratified sampling, and marketing. They proposed a branch-and-bound algorithm based on the continuous relaxation and reported that the proposed algorithm could solve problem instances with up to 200 variables, all of which were taken from the real applications as mentioned above.

Kodialam and Luss [88] studied SC/NLP/C and developed an algorithm by generalizing the algorithms for SC/LUB/C developed by Bitran and Hax [11] and Zipkin [151]. They carried out computational experiments for problem instances taken from production planning applications and showed that the proposed algorithms are practically efficient.

Hochbaum [62] proposed a fully polynomial-time approximation scheme for SC/NLP/D with running time $O(\frac{1}{\epsilon^2}(n \log N + \log \frac{1}{\epsilon} \log n + \frac{1}{\epsilon^2} \log \frac{1}{\epsilon}))$. She also proposed an $O(n \log(N/\epsilon))$ time algorithm that computes an ϵ -accurate solution for SC/NLP/C. Both algorithms assume that g_j are nondecreasing.

See also [16, 20] for other results related to the problem SC/NLP/D.

8.4 Other Variants of Resource Allocation Problems

8.4.1 Resource Allocation Problem with Smoothing Objectives

This problem is a variant of the multiperiod resource allocation problem discussed in Sect. 8.2. For simplicity, let us assume that we have a single resource type.

The objective is to allocate the resource as smoothly as possible over the T periods while satisfying the demands in all periods. More specifically, we attempt to minimize the sum of the absolute values of the changes in resource allocation in adjacent periods. A perfectly smooth allocation of the resource implies that the same amount of resource is made available in each period. However, this ideal situation is, in general, infeasible, as it may incur temporary shortages or require additional resources that will become surplus at the end of the planning horizon. There are two models. The first model assumes that the allocation decisions are continuous variables, and the second model considers discrete variables. Both models have been studied by Luss and Rosenwein [102].

As indicated in [102], an application of such smoothing problems can be found, for example, in a manufacture-to-order environment, where the orders are known for a finite planning horizon. Consider a complex product like a communication switching system. Each system is custom-made and requires a different amount of many subassemblies. The sequence of systems assembled in the final assembly shop during the planning horizon imposes varying demands on each feeder shop that manufactures a subassembly for that system. The question then arises how smooth can the production of each subassembly be, subject to meeting the demands in all periods without having unnecessary inventory.

Here a problem formulation in the simplest case is presented in which the following notations are used:

T : the planning horizon

t, τ : indices for time periods; $t, \tau = 1, 2, \dots, T$

d_t : demand for the product in period t ; $d_t \geq 0$

x_t : amount of resource made available in period t

In production planning applications, d_t is the demand for the product (sub-assembly or component) at period t , and x_t is the amount of the product produced in period t . The model assumes that resource allocations and demands occur instantaneously and simultaneously at the beginning of each period. Further, no resources are available prior to the first period, and the resource made available in period t is storable. That is, it can be used to satisfy the demand of period t or of later periods. Demands must be met on time, that is, the demand of period t must be satisfied by resources made available in period t or earlier. The objective is to allocate the resource as smoothly as possible among successive periods. More precisely, the objective is to minimize the sum of the absolute value of the changes in the amounts of resource made available in adjacent periods. The formulation of this model is as follows:

$$\text{minimize} \quad \sum_{t=2}^T |x_{t-1} - x_t| \quad (164)$$

$$\text{subject to} \quad \sum_{\tau=1}^t x_{\tau} \geq \sum_{\tau=1}^t d_{\tau}, \quad t = 1, 2, \dots, T-1, \quad (165)$$

$$\sum_{\tau=1}^T x_\tau = \sum_{\tau=1}^T d_\tau, \quad (166)$$

$$x_t \geq 0, \quad t = 1, 2, \dots, T. \quad (167)$$

The case for discrete variables is similarly defined. Luss and Rosenwein [102] proposed an $O(T^2)$ time algorithm for the continuous case. For the discrete case, they proposed an algorithm with the same running time that converts a continuous optimal solution to the discrete one with an extra postprocessing time of $O(T)$.

8.4.2 Minimum Variance Resource Allocation Problem

All the objectives of Minimax, Maximin, Lexico-Minimax, Lexico-Maximin, and Fair may be viewed as those seeking for a *fair* allocation of resources to activities. However, in an allocation of discrete resources, the *perfect fairness* may not be attained in general. Consider here *variance* as an alternative measure to express the degree of the *unfairness* or *imbalance* of the resulting allocation. Suppose that function $f_j(x_j)$ represents the profit resulting from allocating x_j amount of resources to activity j . Then the variance of the profit vector $(f_1(x_1), f_2(x_2), \dots, f_n(x_n))$ for an allocation vector $x = (x_1, x_2, \dots, x_n)$ is defined as

$$\text{Var}(x) = \sum_{j=1}^n \frac{1}{n} [f_j(x_j) - \frac{1}{n} \sum_{k=1}^n f_k(x_k)]^2. \quad (168)$$

Then, the minimum variance resource allocation problem under a simple constraint is formulated as follows:

$$\begin{aligned} Q : & \text{minimize } \text{Var}(x) \\ & \text{subject to } \sum_{j=1}^n x_j = N, \\ & x_j \geq 0, \quad \text{integer } j = 1, \dots, n. \end{aligned}$$

This problem has been studied by Katoh [77] and a pseudo-polynomial-time algorithm was presented. The proposed algorithm is based on the technique that transforms the problem into an equivalent parametric problem where the objective function becomes separable, that is,

$$\begin{aligned} Q(\lambda) : & \text{minimize } \sum_{j=1}^n ([f_j(x_j)]^2 - \lambda f_j(x_j)) \\ & \text{subject to } \sum_{j=1}^n x_j = N, \\ & x_j \geq 0, \quad \text{integer } j = 1, \dots, n. \end{aligned}$$

Katoh [77] showed that there exists a parameter $\lambda = \lambda^*$ such that $Q(\lambda^*)$ is optimal to Q . Notice here that each term $[f_j(x_j)]^2 - \lambda f_j(x_j)$ in the objective function may not be convex even if $f_j(x_j)$ is convex. Because of this, no efficient method to search for λ^* has been developed. Thus, it is necessary to solve $Q(\lambda)$ for all λ in a certain range. It was shown in [69] that, in continuously changing the parameter λ , the number of different optimal solutions for $Q(\lambda)$ is $O(n^2 N^2)$. Therefore, all optimal solutions in the range can be obtained in $O(n^2 N^2 \cdot \tau)$ time, where τ denotes the time to solve $Q(\lambda)$ for a fixed λ . Although τ is not polynomial in general, we can apply dynamic programming to solve $Q(\lambda)$ in $O(nN^2)$ time (see, e.g., [69]). (This is possible even if the objective function is not convex.) Thus, problem Q can be solved in a pseudo-polynomial time. This idea can be generalized to the problems with more general constraints such as submodular constraints.

Ichimori and Katoh [72] (see also [71]) studied a special case of Q where each $f_j(x_j)$ is x_j/s_j ($s_j > 0$). In this case, $[f_j(x_j)]^2 - \lambda f_j(x_j)$ is convex, and hence, $Q(\lambda)$ reduces to problem SC/Simple/D. They proposed a very efficient algorithm for this case by devising a technique similar to branch-and-bound procedures. They exhibited its efficiency through computational experiments.

8.5 Notes and References

We give here some additional references not mentioned in the previous sections. We start with multiple resource allocation problem MRP. King [81] developed an interactive software tool for solving problem MRP with linear objective function in support of manufacturing resource planning (MRP II) systems. Ichimori [70] developed a polynomial-time algorithm for a special class of two-resource allocation problem with quadratic convex objective function. Klein and Luss [82] and Klein, Luss, and Rothblum [84] considered an extension of problem MRP in which certain substitutions among resources are possible. They proposed efficient algorithms for the problem. It was mentioned in [82, 84] that such substitution is an important function in the manufacturing of high-tech electronic products because shortage of parts is often incurred due to rapid changes in technology. Nguyen and Stone [113] and Pang and Yu [116] considered similar minimax resource allocation problems with substitutional resources.

A special case of problem MPMRP was examined by Luss and Smith [103] and Klein, Luss, and Smith [83], where $f_{jt}(x_{jt})$ is linear in x_{jt} . In [83], the objective is Lexico-Minimax, and efficient algorithms have been developed for this case in [83, 103]. The algorithms in [83, 103] first solve a relaxed problem obtained by neglecting the constraints (157) and (158). When variables \hat{x}_{jt} in the optimal solution \hat{x} of the relaxed problem violate the constraint (157) or the lower bound constraint of (158), x_{jt} is fixed to $x_{jt} = \max\{l_{jt}, x_{j,t-1}\}$ or $x_{jt} = l_{jt}$, respectively. We then repeat this process by solving the relaxed problem with the above additional equality constraints until all variables are fixed. It was shown in [83, 103] that this algorithm produces an optimal solution. This algorithm can be viewed in a certain sense as a generalization of the algorithm for problem SC/LUB/C by Bitran and

Hax [11] and Zipkin [151] (see also Chap. 2 of [69]). Luss [99] developed post-optimization schemes and parametric analysis, which may be employed once an optimal solution of the problem is obtained.

Klein, Luss, and Rothblum [85] have developed a general methodology to solve multiple resource minimax problems including MRP and MPMRP discussed in this section, based on the so-called relaxation method.

Betts, Brown, and Luss [8] considered minimax resource allocation problems with ordering constraints by generalizing the constraints (157) of problem MPMRP and proposed a practically efficient algorithm that solves a sequence of problems without ordering constraints.

Hackman and Platzman [58] considered another generalization of MRP in which a set of resources required by an activity is not fixed, but such allocation of resource types to an activity is also a decision variable. This problem is formulated as a nonlinear mixed-integer program, which may be solved by a branch-and-bound algorithm. This was motivated by positioning stock problem in a large distribution center (see [57] for the details).

Karabati, Kouvelis, and Yu [74, 75] considered the so-called min-max-sum resource allocation problem with discrete variables that include as special cases both SC/Simple/D and Minimax/Simple/D. Yüceer [149] proposed a variant of the incremental algorithm for the resource allocation problem with a nonseparable objective function and a single knapsack constraint (69), which produces a non-dominated solution. He discussed an application of this problem to a spare-kit problem. Moré and Vavasis [106] considered a separable concave discrete minimization with a simple constraint $\sum_{j=1}^n x_j = N$ and developed an algorithm for the problem.

Recently, a decentralized algorithm to solve SC/Simple/C was proposed [92] in which it is assumed that agents communicate through a network of dynamic topology in order to solve the problem. The proposed algorithm is shown to converge under a mild condition on the objective function (e.g., $f_j(x_j)$ has bounded subgradients).

Also, a parallel algorithm for S/Simple/D was proposed by Shao and Rao [127]. See also [3, 6, 19, 50, 56, 79, 148] for other results related to resource allocation problems.

9 Conclusion

This chapter reviewed the state of the art in the theory and algorithms of the resource allocation problems, mainly with integer variables. In addition, various applications of the resource allocation problems as well as related problems were presented.

The resource allocation problem is one of the most fundamental problem in nonlinear integer programming. Due to the recent advances in the research of nonlinear integer programming, the resource allocation problem has still been studied extensively, although it is a classical problem which dates back to the 1950s. Indeed, applications and generalizations of resource allocation problems

often appear in the literature, as mentioned in this chapter. Therefore, a further advance is desired in the research on the resource allocation problem.

One important open problem is to detect a general class of resource allocation problems which can be solved in polynomial time. One such class is the nonseparable convex resource allocation problem with an M-convex objective functions discussed in Sect. 6. This generalization, however, is not sufficient from the viewpoint of application since it does not fully cover interesting applications of resource allocation problems. Some attempts in this research direction are done by one of the present authors [130, 131], but further investigation is still required.

As mentioned in Sects. 7 and 8, there are many applications and generalizations of resource allocation problems which are known to be NP-hard. Another important open problem is to devise approximation algorithms with theoretical approximation guarantee for such NP-hard problems. For example, Hochbaum [62] proposed a fully polynomial-time approximation scheme for SC/NLP/D (see Sect. 8.3). However, there are only a few results in this research direction so far. From this viewpoint, an interesting (possibly NP-hard) problem is the maximization of a separable convex quadratic function under submodular constraints considered in Goemans et al. [52], where they devise a $1/O(\log n)$ -approximation algorithm. This problem arises as a subproblem for the approximation of a monotone submodular function, and an approximation algorithm for the subproblem with a better approximation ratio would provide a better approximation of a monotone submodular function.

Recommended Reading

1. C.C. Aggarwal, J.L. Wolf, P.S. Yu, On optimal batching policies for video-on-demand storage servers, in *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Hiroshima, Japan, 1996
2. C.C. Aggarwal, J.L. Wolf, P.S. Yu, Optimization issues in multimedia systems. *Int. J. Intell. Syst.* **13**(12), 1113–1135 (1998)
3. A. Andersson, F. Ygge, Efficient resource allocation with non-concave objective functions. *Comput. Optim. Appl.* **20**(3), 281–298 (2001)
4. K. Ando, S. Fujishige, T. Naitoh, A greedy algorithm for minimizing a separable convex function over an integral bisubmodular polyhedron. *J. Oper. Res. Soc. Jpn.* **37**, 188–196 (1994)
5. K. Ando, S. Fujishige, T. Naitoh, A greedy algorithm for minimizing a separable convex function over a finite jump system. *J. Oper. Res. Soc. Jpn.* **38**, 362–375 (1995)
6. I. Averbakh, Minmax regret linear resource allocation problems. *Oper. Res. Lett.* **32**(2), 174–180 (2004)
7. R. Baldick, F.F. Wu, Efficient integer optimization algorithms for optimal coordination of capacitors and regulators. *IEEE Trans. Power Syst.* **5**, 805–812 (1990)
8. L.M. Betts, J.R. Brown, H. Luss, Minimax resource allocation for problems with ordering constraints. *Nav. Res. Logist.* **41**, 719–738 (1994)
9. P.P. Bhattacharya, L. Georgiadis, P. Tsoucas, I. Viniotis, Adaptive lexicographic optimization in multi-class M/GI/1 Queues. *Math. Oper. Res.* **18**(3), 705–740 (1993)

10. S. Biller, L.M.A. Chan, D. Simchi-Levi, J. Swann, Dynamic pricing and the direct-to-customer model in the automotive industry. *Electron. Commer. Res.* **5**(2), 309–334 (2005)
11. G.R. Bitran, A.C. Hax, Disaggregation and resource allocation using convex knapsack problems with bounded variables. *Manag. Sci.* **27**, 431–441 (1981)
12. G.R. Bitran, D. Tirupati, Tradeoff curves, targeting and balancing in manufacturing queueing networks. *Oper. Res.* **37**, 547–564 (1989)
13. G.R. Bitran, D. Tirupati, Capacity planning in manufacturing networks. *Ann. Oper. Res.* **17**, 119–135 (1989)
14. K.M. Bretthauer, B. Shetty, The nonlinear resource allocation problem. *Oper. Res.* **43**(4), 670–683 (1995)
15. K.M. Bretthauer, B. Shetty, Quadratic resource allocation with generalized upper bounds. *Oper. Res. Lett.* **20**, 51–57 (1997)
16. K.M. Bretthauer, B. Shetty, A pegging algorithm for the nonlinear resource allocation problem. *Comput. Oper. Res.* **29**(5), 505–527 (2002)
17. K.M. Bretthauer, B. Shetty, The nonlinear knapsack problem – algorithms and applications. *Eur. J. Oper. Res.* **138**(3), 459–472 (2002)
18. K.M. Bretthauer, B. Shetty, S. Syam, S. White, A model for resource constrained production and inventory management. *Decis. Sci.* **25**, 561–580 (1994)
19. K.M. Bretthauer, B. Shetty, S. Syam, Branch and bound algorithm for integer quadratic knapsack problem. *ORSA J. Comput.* **7**, 109–116 (1995)
20. K.M. Bretthauer, B. Shetty, S. Syam, A specially structured nonlinear integer resource allocation problem. *Nav. Res. Logist.* **50**(7), 770–792 (2003)
21. J.R. Brown, Solving knapsack sharing with general tradeoff functions. *Math. Program.* **51**, 55–73 (1991)
22. J.R. Brown, Bounded knapsack sharing. *Math. Program.* **67**(3), 343–382 (1994)
23. P. Brucker, An $O(n)$ algorithm for quadratic knapsack problems. *Oper. Res. Lett.* **3**, 163–166 (1984)
24. J.A. Buzacott, J.G. Shanthikumar, *Stochastic Models for Manufacturing Systems* (Prentice-Hall, New Jersey, 1993)
25. J. Cardinal, S. Fiorini, G. Joret, Minimum entropy orientations. *Oper. Res. Lett.* **36**, 680–683 (2008)
26. E. Carrizosa, D. Romero Morales, A biobjective method for sample allocation in stratified sampling. *Eur. J. Oper. Res.* **177**(2), 1074–1089 (2007)
27. C.G. Cassandras, L. Dai, C.G. Panayiotou, Ordinal optimization for a class of deterministic and stochastic discrete resource allocation problems. *IEEE Trans. Automa. Control* **43**(7), 881–900 (1998)
28. C.S. Chang, Z. Liu, A bandwidth sharing theory for a large number of HTTP-like connections. *IEEE/ACM Trans. Netw.* **12**(5), 952–962 (2004)
29. X. Chao, L. Liu, S. Zheng, Resource allocation in multisite service systems with intersite customer flows. *Manag. Sci.* **49**(12), 1739–1752 (2003)
30. M.A. Cohen, P.R. Kleindorfer, H.L. Lee, Near-optimal service-constrained stocking policies for spare parts. *Oper. Res.* **37**(1), 104–117 (1989)
31. W. Cook, A.M.H. Gerards, A. Schrijver, E. Tardos, Sensitivity results in integer linear programming. *Math. Program.* **34**, 251–264 (1986)
32. S. Cosares, D.S. Hochbaum, Strongly polynomial algorithms for the quadratic transportation problem with fixed number of sources. *Math. Oper. Res.* **19**(1), 94–111 (1994)
33. G.B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, 1963)
34. M.E. Dyer, A.M. Frieze, On an optimization problem with nested constraints. *Discret. Appl. Math.* **26**, 159–173 (1990)
35. M.E. Dyer, J. Walker, An algorithm for a separable integer programming problem with cumulatively bounded variables. *Discret. Appl. Math.* **16**, 135–149 (1987)
36. J. Edmonds, R. Giles, A mini-max relation for submodular functions on graphs. *Ann. Discret. Math.* **1**, 185–204 (1977)

37. H.A. Eiselt, Continuous maximin knapsack problems with GLB constraints. *Math. Program.* **36**, 114–121 (1986)
38. J.E. Eu, The sampling resource allocation problem. *IEEE Trans. Commun.* **39**(9), 1277–1279 (1991)
39. A. Federgruen, H. Groenevelt, The greedy procedure for resource allocation problems – necessary and sufficient conditions for optimality. *Oper. Res.* **34**, 909–918 (1986)
40. L.R. Ford, D.R. Fulkerson, *Flows in Networks* (Princeton University Press, Princeton, 1962)
41. B.L. Fox, Discrete optimization via marginal analysis. *Manag. Sci.* **13**, 210–216 (1966)
42. G.N. Frederickson, D.B. Johnson, The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *J. Comput. Syst. Sci.* **24**, 197–208 (1982)
43. S. Fujishige, Lexicographically optimal base of a polymatroid with respect to a weight vector. *Math. Oper. Res.* **21**, 186–196 (1980)
44. S. Fujishige, A min-max theorem for bisubmodular polyhedra. *SIAM J. Discret. Math.* **10**(2), 294–308 (1997)
45. S. Fujishige, *Submodular Functions and Optimization*, 2nd edn. (Elsevier, Amsterdam, 2005)
46. S. Fujishige, N. Katoh, T. Ichimori, The fair resource allocation problem with submodular constraints. *Math. Oper. Res.* **13**(1), 164–173 (1988)
47. H.N. Gabow, R.E. Tarjan, Linear time algorithm for a special case of disjoint set union. *J. Comput. Syst. Sci.* **30**, 209–221 (1985)
48. Z. Galil, N. Megiddo, A fast selection algorithm and the problem of optimum distribution of effort. *J. ACM* **26**, 58–64 (1979)
49. G. Gallo, M.E. Grigoriadis, R.E. Tarjan, A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* **18**, 30–55 (1989)
50. A. Galperin, Z. Waksman, A separable integer programming problem equivalent to its continual version. *J. Comput. Appl. Math.* **7**, 173–179 (1981)
51. E. Gurlich, M. Kovalev, A. Zaporozhets, A polynomial algorithm for resource allocation problems with polymatroid constraints. *Optimization* **37**, 73–86 (1996)
52. M.X. Goemans, N.J.A. Harvey, S. Iwata, V. Mirrokni, Approximating submodular functions everywhere, in *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms*, New York, USA, 2009, pp. 535–544
53. H. Groenevelt, Two algorithms for maximizing a separable concave function over a polymatroid feasible region. *Eur. J. Oper. Res.* **54**, 227–236 (1991)
54. O. Gross, *A Class of Discrete Type Minimization Problems*, RM-1644 (RAND-Corporation, Santa Monica, 1956)
55. M. Grötschel, L. Lovász, A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Algorithms and Combinatorics, vol. 2 (Springer, Berlin, 1988)
56. O.K. Gupta, A. Ravindran, Branch and bound experiments in convex nonlinear integer programming problems. *Manag. Sci.* **31**, 1533–1546 (1985)
57. S.T. Hackman, L.K. Platzman, Allocating items to an automated storage and retrieval systems. *IIE Trans.* **22**, 7–14 (1989)
58. S.T. Hackman, L.K. Platzman, Near-optimal solution of generalized resource allocation problems with large capacities. *Oper. Res.* **38**(5), 902–910 (1990)
59. E. Halperin, R.M. Karp, The minimum-entropy set covering problem. *Theor. Comput. Sci.* **348**, 240–250 (2005)
60. D.S. Hochbaum, Polynomial algorithms for convex network optimization, in *Network Optimization Problems: Algorithms, Complexity and Applications*, ed. by D. Du, P.M. Pardalos (World Scientific, Singapore/River Edge, 1993), pp. 63–92
61. D.S. Hochbaum, Lower and upper bounds for the allocation problem and other nonlinear optimization problems. *Math. Oper. Res.* **19**(2), 390–409 (1994)
62. D.S. Hochbaum, A nonlinear knapsack problem. *Oper. Res. Lett.* **17**, 103–110 (1995)
63. D.S. Hochbaum, Complexity and algorithms for convex network optimization and other nonlinear problems. *Ann. Oper. Res.* **153**, 257–296 (2007)
64. D.S. Hochbaum, S. Hong, About strongly polynomial time algorithms for quadratic optimization over submodular constraints. *Math. Program.* **69**, 269–309 (1995)

65. D.S. Hochbaum, J.G. Shanthikumar, Nonlinear separable optimization is not much harder than linear optimization. *J. ACM* **37**(4), 843–862 (1990)
66. D.S. Hochbaum, R. Shamir, J.G. Shanthikumar, A polynomial algorithm for an integer quadratic nonseparable transportation problem. *Math. Program.* **55**(3), 359–372 (1992)
67. A.J. Hoffman, A generalization of max-flow min-cut. *Math. Program.* **6**, 352–359 (1974)
68. T.C. Hu, *Integer Programming and Network Flows* (Addison-Wesley, New-York, 1969)
69. T. Ibaraki, N. Katoh, *Resource Allocation Problems: Algorithmic Approaches* (MIT, Cambridge, 1988)
70. T. Ichimori, A two-resource allocation problem with a quadratic objective function. *Trans. Jpn. Soc. Ind. Appl. Math.* **3**(3), 199–215 (1993) (in Japanese)
71. T. Ichimori, N. Katoh, A two-commodity sharing problem on networks. *Networks* **21**, 547–564 (1991)
72. T. Ichimori, N. Katoh, Minimum variance discrete resource allocation problem. *Trans. Jpn. Soc. Ind. Appl. Math.* **8**, 389–404 (1998)
73. N. Kamiyama, N. Katoh, A. Takizawa, An efficient algorithm for evacuation problem in dynamic network flows with uniform arc capacity. *IEICE Trans. Fundam. Inst. Electron. Inf. Commun. Eng.* **89**, 2372–2379 (2006)
74. S. Karabati, P. Kouvelis, G. Yu, The discrete allocation problem in flow lines. *Manag. Sci.* **41**(9), 1417–1430 (1995)
75. S. Karabati, P. Kouvelis, G. Yu, A min-max-sum resource allocation problem and its application. *Oper. Res.* **(49)**, 913–922 (2001)
76. A.V. Karzanov, S.T. McCormick, Polynomial methods for separable convex optimization in totally unimodular linear spaces with applications. *SIAM J. Comput.* **26**(4), 1245–1275 (1997)
77. N. Katoh, An ϵ -approximation scheme for minimum variance problems. *J. Oper. Res. Soc. Jpn.* **33**(1), 46–65 (1990)
78. N. Katoh, T. Ibaraki, H. Mine, A polynomial time algorithm for the resource allocation problem with a convex objective function. *J. Oper. Res. Soc.* **30**, 449–455 (1979)
79. N. Katoh, H. Tamaki, T. Tokuyama, Parametric polymatroid optimization and its geometric applications. *Int. J. Comput. Geom. Appl.* **12**(5), 429–443 (2002)
80. W. Kim, A new way to compute the product and join of relations, in *Proceedings of the ACM SIGMOD Conference*, Santa Monica, CA, 1980
81. J.H. King, Allocation of scarce resources in manufacturing facilities, *AT&T Tech. J.* **68**(3), 103–113 (1989)
82. R.S. Klein, H. Luss, Minimax resource allocation with tree structured substitutable resources. *Oper. Res.* **39**(2), 285–295 (1991)
83. R.S. Klein, H. Luss, D.R. Smith, A lexicographic minimax algorithm for multiperiod resource allocation. *Math. Program.* **55**, 213–234 (1992)
84. R.S. Klein, H. Luss, U.G. Rothblum, Minimax resource allocation problems with resource-substitutions represented by graphs. *Oper. Res.* **41**, 959–971 (1993)
85. R.S. Klein, H. Luss, U.G. Rothblum, Relaxation-based algorithms for minimax optimization problems with resource allocation applications. *Math. Program.* **64**, 337–363 (1994)
86. R.S. Klein, H. Luss, U.G. Rothblum, Multiperiod allocation of substitutable resources. *Eur. J. Oper. Res.* **85**, 488–503 (1995)
87. D.E. Knuth, *The Art of Computer Programming: Vol. 3. Sorting and Searching* (Addison Wesley, Reading, 1973)
88. M.S. Kodialam, H. Luss, Algorithms for separable nonlinear resource allocation problems. *Oper. Res.* **46**, 272–284 (1998)
89. B.O. Koopman, The optimum distribution of effort. *Oper. Res.* **1**, 52–63 (1953)
90. P. Kubat, H.S. Koch, Managing test-procedures to achieve reliable software. *IEEE Trans. Reliab.* **R-32**, 299–303 (1983)
91. T. Kuno, H. Konno, E. Zemel, A linear time algorithm for solving continuous knapsack problems. *Oper. Res. Lett.* **10**, 23–26 (1991)

92. H. Lakshmanan, D.P. de Farias, Decentralized resource allocation in dynamic networks of agents. *SIAM J. Optim.* **19**(2), 911–940 (2008)
93. H.L. Lee, W.P. Pierskalla, Mass screening models for contagious diseases with no latent period. *Oper. Res.* **36**(1), 917–928 (1988)
94. D. Li, Y. Y. Haimes, A decomposition method for optimization of large-system reliability. *IEEE Trans. Reliab.* **41**(2), 183–189 (1992)
95. D.H. Lorenz, A. Orda, D. Raz, Y. Shavitt, Efficient QoS partition and routing of unicast and multicast. *IEEE/ACM Trans. Netw.* **14**(6), 1336–1347 (2006)
96. L. Lovász, Submodular functions and convexity, in *Mathematical Programming – The State of the Art*, ed. by A. Bachem, M. Grötschel, B. Korte (Springer, Berlin, 1983), pp. 235–257
97. H. Luss, An algorithm for separable nonlinear minimax problems. *Oper. Res. Lett.* **6**, 159–162 (1987)
98. H. Luss, A nonlinear minimax allocation problem with multiple knapsack constraints. *Oper. Res. Lett.* **10**, 183–187 (1991)
99. H. Luss, Minimax resource allocation problems: optimization and parametric analysis. *Eur. J. Oper. Res.* **60**, 76–86 (1992)
100. H. Luss, On equitable resource allocation problems: a lexicographic minimax approach. *Oper. Res.* **47**(3), 361–378 (1999)
101. H. Luss, Allocation of resources of modular sizes with an application to internet protocol (IP) address allocation. *Nav. Res. Logist.* **52**(8), 713–723 (2005)
102. H. Luss, M.B. Rosenwein, Multiperiod resource allocation with a smoothing objective. *Nav. Res. Logist.* **42**, 1007–1020 (1995)
103. H. Luss, D.R. Smith, Resource allocation among competing activities: a lexicographic minimax approach. *Oper. Res. Lett.* **5**, 227–231 (1986)
104. H. Luss, D.R. Smith, Multiperiod allocation of limited resources: a minimax approach. *Nav. Res. Logist.* **35**, 493–501 (1988)
105. N. Megiddo, A. Tamir, Linear time algorithms for some separable quadratic programming problems. *Oper. Res. Lett.* **13**, 203–211 (1993)
106. J.J. Moré, S.A. Vavasis, On the solution of concave knapsack problems. *Math. Program.* **49**, 397–411 (1991)
107. S. Moriguchi, A. Shioura, On Hochbaum’s proximity-scaling algorithm for the general resource allocation problem. *Math. Oper. Res.* **29**, 394–397 (2004)
108. S. Moriguchi, A. Shioura, N. Tsuchimura, M-convex function minimization by continuous relaxation approach: proximity theorem and algorithm. *SIAM J. Optim.* **21**, 633–668 (2011)
109. K. Murota, Convexity and Steinitz’s exchange property. *Adv. Math.* **124**, 272–311 (1996)
110. K. Murota, Discrete convex analysis. *Math. Program.* **83**, 313–371 (1998)
111. K. Murota, *Discrete Convex Analysis* (SIAM, Philadelphia, 2003)
112. K. Namikawa, T. Ibaraki, An algorithm for the fair resource allocation problem with a submodular constraint. *Jpn. J. Ind. Appl. Math.* **8**, 377–387 (1991)
113. Q.C. Nguyen, R.E. Stone, A multiperiod resource allocation problem with storable and substitutable resources. *Manag. Sci.* **39**, 964–974 (1993)
114. S.S. Nielsen, S.A. Zenios, Massively parallel algorithms for singly constrained convex program. *ORSA J. Comput.* **4**, 166–181 (1992)
115. H. Ohtera, S. Yamada, Optimal allocation & control problems for software-testing resources. *IEEE Trans. Reliab.* **39**(2), 171–176 (1990)
116. J.-S. Pang, C.-S. Yu, A min-max resource allocation problem with substitutions. *Eur. J. Oper. Res.* **41**, 218–223 (1989)
117. C.H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity* (Prentice-Hall, Englewood Cliffs, 1982)
118. P.M. Pardalos, N. Kovoor, An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Math. Program.* **46**, 321–328 (1990)
119. P.M. Pardalos, Y. Ye, C.G. Han, Algorithms for the solution of quadratic knapsack problems. *Linear Algebra Appl.* **152**, 69–91 (1991)

120. M. Patriksson, A survey on the continuous nonlinear resource allocation problem. *Eur. J. Oper. Res.* **185**, 1–46 (2008)
121. C. Qu, P. Wang, Optimization in production quota problem with convex cost function. *Appl. Math. Comput.* **177**(2), 652–658 (2006)
122. J. Renegar, On the worst case arithmetic complexity of approximation zeroes of polynomials. *J. Complex.* **3**, 9–113 (1987)
123. A.G. Robinson, N. Jiang, C.S. Lerme, On the continuous quadratic knapsack problem. *Math. Program.* **55**, 99–108 (1992)
124. A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency* (Springer, Berlin, 2004)
125. J.G. Shanthikumar, D.D. Yao, Second-order stochastic properties in queueing systems. *Proc. IEEE* **77**(1), 162–170 (1989)
126. J.G. Shanthikumar, D.D. Yao, Multiclass queueing systems: polymatroidal structure and optimal scheduling control. *Oper. Res.* **40**(2), 293–299 (1992)
127. B.B.M. Shao, H.R. Rao, A parallel hypercube algorithm for discrete resource allocation problems. *IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum.* **36**(1), 233–242 (2006)
128. W. Shih, A new application of incremental analysis in resource allocations. *Oper. Res. Q.* **25**, 587–597 (1974)
129. A. Shioura, Minimization of an M-convex function. *Discret. Appl. Math.* **84**, 215–220 (1998)
130. A. Shioura, Neighbor systems, jump systems, and bisubmodular polyhedra. *SIAM J. Discret. Math.* **26**, 114–144 (2012)
131. A. Shioura, K. Tanaka, Polynomial-time algorithms for linear and convex optimization on jump systems. *SIAM J. Discret. Math.* **21**, 504–522 (2007)
132. H.S. Stone, J. Turek, J.L. Wolf, Optimal partitioning of cache memory. *IEEE Trans. Comput.* **41**(9), 1054–1068 (1992)
133. A. Tamir, A strongly polynomial algorithm for minimum convex separable quadratic cost flow problems on series-parallel networks. *Math. Program.* **59**, 117–132 (1993)
134. T. Tamir, B. Vakseniser, Algorithms for storage allocation based on client preferences. *J. Comb. Optim.* **19**(3), 304–324 (2010)
135. C.S. Tang, A max-min allocation problem: its solutions and applications. *Oper. Res.* **36**, 359–367 (1988)
136. A.N. Tantawi, D. Towsley, J.L. Wolf, *An Algorithm for a Class Constrained Resource Allocation Problem* (IBM Thomas Watson Research Center, Yorktown Heights, 1987)
137. D. Thiébaut, H.S. Stone, J.L. Wolf, Improving disk cache hit-ratios through cache partitioning. *IEEE Trans. Comput.* **41**(9), 665–676 (1992)
138. J. Turek, J.L. Wolf, K.R. Pattipati, P.S. Yu, Scheduling parallelizable tasks: putting it all on the shelf, in *Proceedings of ACM SIGMETRICS '92*, Newport, RI, June 1992, pp. 225–236
139. J. Turek, J.L. Wolf, P.S. Yu, Approximate algorithms for scheduling parallelizable tasks, in *Proceedings of the 4th Annual Symposium on Parallel Algorithms and Architectures*, San Diego, CA, June 1992, pp. 323–332
140. J. Turek, W. Ludwig, J.L. Wolf, L. Fleischer, P. Tiwari, J. Glasgow, U. Schwiegelsohn, P.S. Yu, Scheduling parallelizable tasks to minimize average response time, in *Proceedings of ACM Symposium on Parallel Algorithms and Architectures*, Cape May, NJ, 1994
141. D.A. Turner, K.W. Ross, Adaptive streaming of layer-encoded multimedia presentations. *J. VLSI Signal Process.* **34**(1), 83–99 (2003)
142. J.A. Ventura, C.M. Klein, A note on multi-item inventory systems with limited capacity. *Oper. Res. Lett.* **7**, 71–75 (1988)
143. J.L. Wolf, B.R. Iyer, K.R. Pattipati, P.S. Yu, Optimal buffer partitioning for the nested block join algorithm, in *Proceedings of 7th International Conference on Data Engineering*, Kobe, Japan, 1991, pp. 510–519
144. J.L. Wolf, P.L. Yu, H. Shachnai, DASD dancing: a disk load balancing optimization scheme for video-on-demand computer systems, in *Proceedings of ACM Sigmetrics Conference*, Ottawa, Canada, 1995

145. D.D. Yao, Optimal run quantities for an assembly system with random yields. *IIE Trans.* **20**(4), 399–403 (1988)
146. P.Y. Yin, J.Y. Wang, Ant colony optimization for the nonlinear resource allocation problem. *Appl. Math. Comput.* **174**(2), 1438–1453 (2006)
147. G. Yu, Min-max optimization of several classical discrete optimization problems. *J. Optim. Theory Appl.* **98**(1), 221–242 (1998)
148. G. Yu, P. Kouvelis, On min-max optimization of a collection of classical discrete optimization problems, in *Minimax and Applications*, ed. by D.-Z. Du, P.M. Pardalos, (Kluwer, Dordrecht/Boston, 1995), pp. 157–171
149. Ü. Yüceer, Marginal allocation algorithm for nonseparable functions. *INFOR* **37**, 97–113 (1999)
150. Tak-S. Yum, Mon-S. Chen, Yiu-W. Leung, Bandwidth allocation for multimedia teleconferences, in *Proceedings of IEEE International Conference on Communications*, Denver, CO, 1991, pp. 852–858
151. P.H. Zipkin, Simple ranking methods for allocation of one resource. *Manag. Sci.* **26**, 34–43 (1980)

Rollout Algorithms for Discrete Optimization: A Survey

Dimitri P. Bertsekas

Contents

1	Introduction	2989
2	The Basic Rollout Algorithm for Discrete Optimization.....	2993
2.1	Termination and Sequential Consistency.....	2996
3	Applications in Discrete Optimization.....	3007
4	Further Reading.....	3011
	Recommended Reading.....	3012

Abstract

This chapter discusses rollout algorithms, a sequential approach to optimization problems, whereby the optimization variables are optimized one after the other. A rollout algorithm starts from some given heuristic and constructs another heuristic with better performance than the original. The method is particularly simple to implement and is often surprisingly effective. This chapter explains the method and its properties for discrete deterministic optimization problems.

1 Introduction

Rollout is a form of sequential optimization that originated in dynamic programming (DP for short). It may be viewed as a single iteration of the fundamental method of policy iteration. The starting point is a given policy (called the base policy), whose performance is evaluated in some way, possibly by simulation. Based on the evaluation, an improved policy is obtained by one-step lookahead. When the problem is discrete and deterministic, as will be assumed in this chapter, the

D.P. Bertsekas (✉)

Laboratory for Information and Decision Systems, Massachusetts Institute of Technology,
Cambridge, MA, USA
e-mail: dimitrib@mit.edu

method is very simple to implement: the base policy is just some heuristic, and the rollout policy consists of repeated application of this heuristic. The rollout policy is guaranteed to improve the performance of the base policy, often very substantially in practice. In this chapter, rather than using the dynamic programming formalism, the method is explained starting from first principles.

Consider the minimization of a function

$$g(x_1, \dots, x_N)$$

of the discrete variables x_1, \dots, x_N , assumed to take values in some finite set. This can be done (at least conceptually) by minimizing over x_1 the result of minimization of g over the remaining variables x_2, \dots, x_N , with x_1 held fixed, that is, finding

$$x_1^* \in \arg \min_{x_1} J_1(x_1)$$

where the function J_1 is defined by

$$J_1(x_1) = \min_{x_2, \dots, x_N} g(x_1, x_2, \dots, x_N).$$

Leaving aside for the moment the difficulty of calculating $J_1(x_1)$, given the optimal value x_1^* , the optimal value of x_2 can be found by the minimization

$$x_2^* \in \arg \min_{x_2} J_2(x_1^*, x_2),$$

where the function J_2 is defined by

$$J_2(x_1, x_2) = \min_{x_3, \dots, x_N} g(x_1, x_2, x_3, \dots, x_N).$$

Similarly, for every $k = 1, \dots, n$, given the optimal values x_1^*, \dots, x_{k-1}^* , the optimal value of x_k can be found by the minimization

$$x_k^* \in \arg \min_{x_k} J_k(x_1^*, \dots, x_{k-1}^*, x_k), \quad k = 1, \dots, N, \quad (1)$$

where the function J_k is defined by

$$J_k(x_1, \dots, x_k) = \min_{x_{k+1}, \dots, x_N} g(x_1, \dots, x_k, x_{k+1}, \dots, x_N). \quad (2)$$

This is a calculation that is typical of DP, where the functions J_k are called the *optimal cost-to-go functions*, and are defined by the recursion

$$J_k(x_1, \dots, x_k) = \min_{x_{k+1}} J_{k+1}(x_1, \dots, x_k, x_{k+1}),$$

starting from the boundary condition

$$J_N(x_1, \dots, x_N) = g(x_1, \dots, x_N).$$

[The validity of this recursion can be seen from the definition of J_k .]

Unfortunately, there is a serious difficulty with the preceding approach: calculating numerically and storing the functions J_k in tables is impossible in practice (except for very special problems; if each of the variables x_1, \dots, x_k may take m distinct values, the storage of J_k requires a table of size m^k). In DP this is known as *the curse of dimensionality*.

In the rollout approach, J_k is *approximated* by a function that is more easily calculated and does not require excessive storage. In particular, for any given x_1, \dots, x_k , an easily implementable heuristic (called *base heuristic*) is used to approximate the minimization (2). If $H_k(x_1, \dots, x_k)$ denotes the corresponding approximately optimal value, the *rollout algorithm* obtains a suboptimal solution by replacing J_k with H_k in Eq. (1):

$$x_k \in \arg \min_{x_k} H_k(\tilde{x}_1, \dots, \tilde{x}_{k-1}, x_k), \quad k = 1, \dots, N.$$

These minimizations are carried out in sequence, first obtaining $\tilde{x}_1 \in \arg \min_{x_1} H_1(x_1)$, then obtaining $\tilde{x}_2 \in \arg \min_{x_2} H_2(\tilde{x}_1, x_2)$, and proceeding with $\tilde{x}_3, \dots, \tilde{x}_N$ in that order.

With some analysis and algorithmic refinement, it can be shown that rollout results in no performance deterioration over just applying the base heuristic once to the original problem. Most importantly, experimentation has shown that rollout typically results in significant and often dramatic improvement over the common approach of just applying the base heuristic. This improvement comes at the expense of a substantial but often tractable (polynomial) increase in computational requirements. The following example provides some insight into the nature of cost improvement.

Example 1 (The Breakthrough Problem) Consider a binary tree with N stages as shown in Fig. 1. Stage k of the tree has 2^k nodes. There are two types of tree arcs: *free* and *blocked*. A free (or blocked) arc can (cannot, respectively) be traversed in the direction from the root to the leaves. The objective is to break through the graph with a sequence of free arcs (a free path) starting from the root and ending at one of the leaves.

One may use a DP-like algorithm to discover a free path (if one exists) by starting from the last stage and by proceeding backward to the root node. The k th step of the algorithm determines for each node of stage $N - k$ whether there is a free path from that node to some leaf node, by using the results of the preceding step. The amount of calculation at the k th step is $O(2^{N-k})$. Adding the calculations for the N stages, it can be seen that the total amount of calculation is $O(N2^N)$, so it increases exponentially with the number of stages.

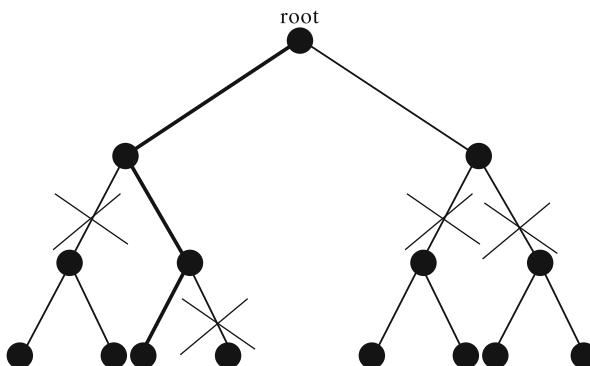


Fig. 1 Binary tree with N stages for the breakthrough problem. Each arc is either free or is blocked (crossed out in the figure). The problem is to find a path from the root to one of the leaves, which is free (such as the one shown with *thick lines*)

As an alternative, one may suboptimally use a *greedy* algorithm, which starts at the root node, selects a free outgoing arc (if one is available), and tries to construct a free path by adding successively nodes to the path. Generally, at the current node, if one of the outgoing arcs is free and the other is blocked, the greedy algorithm selects the free arc. Otherwise, it selects one of the two outgoing arcs according to some fixed rule that depends only on the current node (and not on the status of other arcs). Clearly, the greedy algorithm may fail to find a free path even if such a path exists, as can be seen from Fig. 1 (e.g., if the choice at the root node is the right-hand side arc). On the other hand the amount of computation associated with the greedy algorithm is $O(N)$, which is much faster than the $O(N2^N)$ computation of the optimal algorithm. Thus, one may view the greedy algorithm as a fast heuristic, which is suboptimal in the sense that there are problem instances where it fails while the DP algorithm succeeds.

Consider also the rollout algorithm that uses the greedy algorithm as the base heuristic. This algorithm starts at the root and tries to construct a free path by exploring alternative paths constructed by the greedy algorithm. At the current node, it proceeds according to the following two cases:

- (a) If at least one of the two outgoing arcs of the current node is blocked, the rollout algorithm adds to the current path the arc that the greedy algorithm would select at the current node.
 - (b) If both outgoing arcs of the current node are free, the rollout algorithm considers the two end nodes of these arcs, and from each of them it runs the greedy algorithm. If the greedy algorithm succeeds in finding a free path that starts from at least one of these nodes, the rollout algorithm stops with a free path having been found; otherwise, the rollout algorithm moves to the node that the greedy algorithm would select at the current node.

Thus, when both outgoing arcs are free, the rollout algorithm explores further the suitability of these arcs, as in case (b) above. Because of this additional

discriminatory capability, the rollout algorithm always does at least as well as the greedy (it always finds a free path when the greedy algorithm does, and it also finds a free path in some cases where the greedy algorithm does not). This is consistent with our earlier discussion of the generic cost improvement property of the rollout algorithm over the base heuristic. On the other hand, the rollout algorithm applies the greedy heuristic as many as $2N$ times, so that it requires $O(N^2)$ amount of computation – this is intermediate between the $O(N)$ computation of the greedy and the $O(N2^N)$ computation of the DP algorithm.

The greedy and the rollout algorithms may be evaluated by calculating the probabilities that they will find a free path given a randomly chosen breakthrough problem. In particular, the graph of the problem may be generated randomly, by selecting each of its arcs to be free with probability p , independently of the other arcs. If the corresponding probabilities of success for the greedy and the rollout algorithms are calculated, it follows that the rollout algorithm has an $O(N)$ times larger probability of finding a free path than the greedy algorithm, while requiring $O(N)$ times more computation (see [3], Example 6.4.2). This type of tradeoff is qualitatively typical: the rollout algorithm achieves a substantial performance improvement over the base heuristic at the expense of extra computation that is equal to the computation time of the base heuristic times a factor that is a low order polynomial of the problem size.

2 The Basic Rollout Algorithm for Discrete Optimization

The rollout algorithm will now be formalized by introducing a graph search problem that can serve as a general model for discrete optimization. A graph is given that has a finite set of nodes \mathcal{N} , a finite set of arcs \mathcal{A} , and a special node s , called the *origin*. The arcs are directed in the sense that arc (i, j) is distinct from arc (j, i) . A subset $\bar{\mathcal{N}}$ of nodes is also given, called *destinations* and a cost $g(i)$ for each destination i . The destination nodes are terminal in the sense that they have no outgoing arcs. The problem is to find a path that starts at the origin s , ends at one of the destination nodes $i \in \bar{\mathcal{N}}$, and is such that the cost $g(i)$ is minimized.

In our terminology, a path is a sequence of arcs

$$(i_1, i_2), (i_2, i_3), \dots, (i_{m-1}, i_m),$$

all of which are oriented in the forward direction. The nodes i_1 and i_m are called the *start node* and the *end node* of the path, respectively. For convenience, and without loss of generality,¹ it will be assumed that given an ordered pair of nodes (i, j) , there is at most one arc with start node i and end node j , which (if it exists) will be

¹In the case where there are multiple arcs connecting a node pair, all these arcs can be merged to a single arc, since the set of destination nodes that can be reached from any non-destination node will not be affected.

denoted by (i, j) . In this way, a path consisting of arcs $(i_1, i_2), (i_2, i_3), \dots, (i_{m-1}, i_m)$ is unambiguously specified as the sequence of nodes (i_1, i_2, \dots, i_m) .

Let us assume the availability of a heuristic path construction algorithm, denoted \mathcal{H} , which given a non-destination node $i \notin \bar{\mathcal{N}}$, constructs a path $(i, i_1, \dots, i_m, \bar{i})$ starting at i and ending at one of the destination nodes \bar{i} . Implicit in this assumption is that for every non-destination node, there exists at least one path starting at that node and ending at some destination node. The algorithm \mathcal{H} is referred to as the *base heuristic* and will be used as the basic building block for constructing the rollout algorithm to be introduced shortly.

The end node \bar{i} of the path constructed by the base heuristic \mathcal{H} is completely specified by the start node i . The node \bar{i} is called the *projection of i under \mathcal{H}* and is denoted by $p(i)$. The corresponding cost is denoted by $H(i)$,

$$H(i) = g(p(i)).$$

The projection of a destination node is the node itself by convention, so that $i = p(i)$ and $H(i) = g(i)$ for all $i \in \bar{\mathcal{N}}$. Note that while the base heuristic \mathcal{H} will generally yield a suboptimal solution, the path that it constructs may involve a fairly sophisticated suboptimization. For example, \mathcal{H} may construct several paths ending at destination nodes according to some heuristics and then select the path that yields minimal cost.

One possibility for suboptimal solution of the problem is to start at the origin s and use the base heuristic \mathcal{H} to obtain the projection $p(s)$. It is instead proposed to use \mathcal{H} to construct a path to a destination node sequentially. At the typical step, a path that starts at s and ends at a node i is available, the base heuristic \mathcal{H} is run starting from each of the downstream neighbors j of i , and the corresponding projections and costs are obtained. The neighbor that gives the best projection is chosen as the next node in the current path. This sequential version of \mathcal{H} is called the *rollout algorithm based on \mathcal{H}* and is denoted by \mathcal{RH} .

Formally, let $N(i)$ denote the set of downstream neighbors of a non-destination node i ,

$$N(i) = \{j \mid (i, j) \text{ is an arc}\}.$$

The rollout algorithm \mathcal{RH} starts with the origin node s . At the typical step, given a node sequence (s, i_1, \dots, i_m) , where i_m is not a destination, \mathcal{RH} adds to the sequence a node i_{m+1} such that

$$i_{m+1} \in \arg \min_{j \in N(i_m)} H(j). \quad (3)$$

If i_{m+1} is a destination node, \mathcal{RH} terminates. Otherwise, the process is repeated with the sequence $(s, i_1, \dots, i_m, i_{m+1})$ replacing (s, i_1, \dots, i_m) ; see Fig. 2.

Once \mathcal{RH} has terminated with a path (s, i_1, \dots, i_m) , the projection $p(i_k)$ of each of the nodes i_k , $k = 1, \dots, m$, will have been obtained. The best of these projections yields a cost

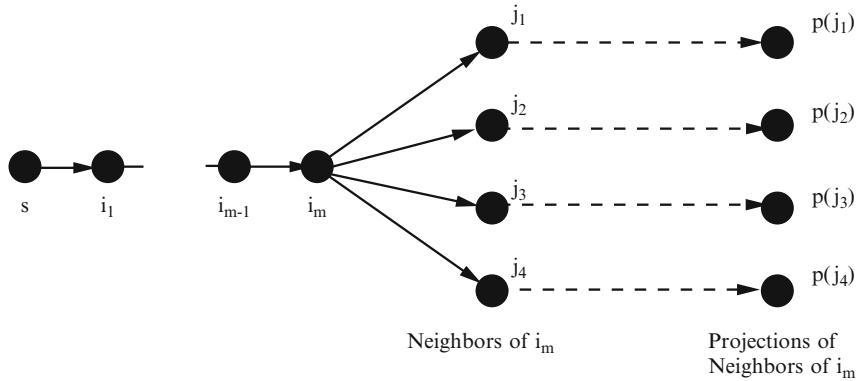


Fig. 2 Illustration of the rollout algorithm. After m steps, the algorithm has constructed the path (s, i_1, \dots, i_m) . To extend this path at the next step, the set $N(i_m)$ of neighbors of the terminal node i_m is generated, and the neighbor that has the best projection is selected from this set, that is, $i_{m+1} \in \arg \min_{j \in N(i_m)} H(j) \in \arg \min_{j \in N(i_m)} g(p(j))$

$$\min_{k=1,\dots,m} H(i_k) = \min_{k=1,\dots,m} g(p(i_k)),$$

and the projection that corresponds to the minimum above may be taken as the final (suboptimal) solution produced by the rollout algorithm. The above minimal cost may also be compared with the cost $g(p(s))$ of the projection $p(s)$ of the origin, so that $p(s)$ is used as the final solution if it produces a smaller cost. This will ensure that the rollout algorithm will produce a solution that is no worse than the one produced by the base heuristic. Note that while the best neighbor of i_m is i_{m+1} according to Eq. (3), it does not necessarily follow that i_{m+1} has a better projection than i_m under \mathcal{H} , that is, that

$$H(i_{m+1}) \leq H(i_m). \quad (4)$$

The reason is that if the path constructed by \mathcal{H} starting from i_m is $(i_m, j_1, \dots, j_k, \bar{i})$, it is not necessarily true that the path constructed by \mathcal{H} starting from j_1 is $(j_1, \dots, j_k, \bar{i})$. If this were so, then Eq. (4) would hold, since i_{m+1} would have no worse projection than j_1 according to Eq. (3). This argument will be the basis for further analysis to be given later (see Proposition 1).

Example 2 (Traveling Salesman Problem) Let us consider the traveling salesman problem, whereby a salesman wants to find a minimum mileage/cost tour that visits each of N given cities exactly once and returns to the city he started from. With each city $i = 1, \dots, N$, a node is associated and an arc (i, j) with traversal cost a_{ij} is introduced for each ordered pair of nodes i and j . Note that the graph is assumed complete; that is, there exists an arc for each ordered pair of nodes. There is no loss of generality in doing so because a very high cost a_{ij} can be assigned to an arc

(i, j) that is precluded from participation in the solution. The problem is to find a cycle that goes through all the nodes exactly once and whose sum of arc costs is minimum.

There are many heuristic approaches for solving the traveling salesman problem. For illustration purposes, consider a simple *nearest neighbor* heuristic. It starts from a path consisting of just a single node i_1 , and at each iteration, it enlarges the path with a node that does not close a cycle and minimizes the cost of the enlargement. In particular, after k iterations, a path $\{i_1, \dots, i_k\}$ consisting of distinct nodes has been constructed, and at the next iteration, an arc (i_k, i_{k+1}) that minimizes $a_{i_k i}$ over all arcs (i_k, i) with $i \neq i_1, \dots, i_k$, is added. After $N - 1$ iterations, all nodes are included in the path, which is then converted to a tour by adding the final arc (i_N, i_1) .

The traveling salesman problem can be formulated as a graph search problem as follows: A starting city, say i_1 , is chosen corresponding to the origin of the graph search problem. Each node of the graph search problem corresponds to a path (i_1, i_2, \dots, i_k) , where i_1, i_2, \dots, i_k are distinct cities. The neighbor nodes of the path (i_1, i_2, \dots, i_k) are paths of the form $(i_1, i_2, \dots, i_k, i_{k+1})$ that correspond to adding one more unvisited city $i_{k+1} \neq i_1, i_2, \dots, i_k$ at the end of the path. The destinations are the cycles of the form (i_1, i_2, \dots, i_N) , and the cost of a destination in the graph search problem is the cost of the corresponding cycle. Thus, a path from the origin to a destination in the graph search problem corresponds to constructing a cycle in $N - 1$ arc addition steps and at the end incurring the cost of the cycle.

Let us now use as base heuristic the nearest neighbor method. The corresponding rollout algorithm operates as follows: After k iterations, a path $\{i_1, \dots, i_k\}$ consisting of distinct nodes has been constructed. At the next iteration, the nearest neighbor heuristic is run starting from each of the paths of the form $\{i_1, \dots, i_k, i\}$ where $i \neq i_1, \dots, i_k$, and a corresponding cycle is obtained. The node i_{k+1} of the path is selected to be the node i that corresponds to the best cycle thus obtained.

2.1 Termination and Sequential Consistency

The rollout algorithm \mathcal{RH} is said to be *terminating* if it is guaranteed to terminate finitely starting from any node. Contrary to the base heuristic \mathcal{H} , which by definition, has the property that it yields a path terminating at a destination starting from any node, the rollout algorithm \mathcal{RH} need not have this property in the absence of additional conditions. The termination question can usually be resolved quite easily, and a few different methods by which this can be done will now be discussed.

One important case where \mathcal{RH} is terminating is when the graph is acyclic, since then the nodes of the path generated by \mathcal{RH} cannot be repeated within the path, and their number is bounded by the number of nodes in \mathcal{N} . As a first step toward developing another case where \mathcal{RH} is terminating, consider the following definition, which will also set the stage for further analysis of the properties of \mathcal{RH} .

Definition 1 The base heuristic \mathcal{H} is said to be *sequentially consistent* if for every node i , it has the following property: If \mathcal{H} generates the path $(i, i_1, \dots, i_m, \bar{i})$ when it starts at i , it generates the path $(i_1, \dots, i_m, \bar{i})$ when it starts at the node i_1 .

Thus, \mathcal{H} is sequentially consistent if all the nodes of a path that it generates have the same projection. There are many examples of sequentially consistent algorithms that are used as heuristics in combinatorial optimization, including the following.

Example 3 (Greedy Algorithms as Base Heuristics) Consider a function F , which for each node i , provides a scalar estimate $F(i)$ of the optimal cost starting from i , that is, the minimal cost $g(\bar{i})$, that can be obtained with a path that starts at i and ends at one of the destination nodes $\bar{i} \in \bar{\mathcal{N}}$. Then, F can be used to define a base heuristic, called the *greedy algorithm with respect to F* , as follows:

The greedy algorithm starts at a node i with the (degenerate) path that consists of just node i . At the typical step, given a path (i, i_1, \dots, i_m) , where i_m is not a destination, the algorithm adds to the path a node i_{m+1} such that

$$i_{m+1} \in \arg \min_{j \in N(i_m)} F(j). \quad (5)$$

If i_{m+1} is a destination, the algorithm terminates with the path $(i, i_1, \dots, i_m, i_{m+1})$. Otherwise, the process is repeated with the path $(i, i_1, \dots, i_m, i_{m+1})$ replacing (i, i_1, \dots, i_m) .

An example of a greedy algorithm is the nearest neighbor heuristic for the traveling salesman problem (cf. [Example 2](#)). Recall from that example that nodes of the graph search problem correspond to paths (sequences of distinct cities), and a transition to a neighbor node corresponds to adding one more unvisited city to the end of the current path. The function F in the nearest neighbor heuristic specifies the cost of the addition of the new city.

It is also interesting to note that by viewing F as a cost-to-go approximation, the greedy algorithm may be considered to be a special type of one-step lookahead policy. Furthermore, if $F(j)$ is chosen to be the cost obtained by some base heuristic starting from j , then the greedy algorithm becomes the corresponding rollout algorithm. Thus, it may be said that the rollout algorithm is a special case of a greedy algorithm. However, the particular choice of F used in the rollout algorithm is responsible for special properties that are not shared by other types of greedy algorithms.

Let us denote by \mathcal{H} the greedy algorithm described above and assume that it terminates starting from every node (this has to be verified independently). Let us also assume that whenever there is a tie in the minimization of Eq. (5), \mathcal{H} resolves the tie in a manner that is fixed and independent of the starting node i of the path, for example, by resolving the tie in favor of the numerically smallest node j that attains the minimum in Eq. (5). Then, it can be seen that \mathcal{H} is sequentially consistent, since by construction, every node on a path generated by \mathcal{H} has the same projection.

For a sequentially consistent base heuristic \mathcal{H} , a restriction will be imposed in the way the rollout algorithm \mathcal{RH} resolves ties in selecting the next node on its path; this restriction will guarantee that \mathcal{RH} is terminating. In particular, suppose that after m steps, \mathcal{RH} has produced the node sequence (s, i_1, \dots, i_m) , and that the path generated by \mathcal{H} starting from i_m is $(i_m, i_{m+1}, i_{m+2}, \dots, \bar{i})$. Suppose that among the neighbor set $N(i_m)$, the node i_{m+1} attains the minimum in the selection test

$$\min_{j \in N(i_m)} H(j), \quad (6)$$

but there are also some other nodes, in addition to i_{m+1} , that attain this minimum. Then, the tie is broken in favor of i_{m+1} , that is, the next node added to the current sequence (s, i_1, \dots, i_m) is i_{m+1} . Under this convention for tie-breaking, the following proposition shows that the rollout algorithm \mathcal{RH} terminates at a destination and yields a cost that is no larger than the cost yielded by the base heuristic \mathcal{H} .²

Proposition 1 *Let the base heuristic \mathcal{H} be sequentially consistent. Then, the rollout algorithm \mathcal{RH} is terminating. Furthermore, if $(i_1, \dots, i_{\tilde{m}})$ is the path generated by \mathcal{RH} starting from a non-destination node i_1 and ending at a destination node $i_{\tilde{m}}$, the cost of \mathcal{RH} starting from i_1 is less or equal to the cost of \mathcal{H} starting from i_1 . In particular,*

$$H(i_1) \geq H(i_2) \geq \dots \geq H(i_{\tilde{m}-1}) \geq H(i_{\tilde{m}}). \quad (7)$$

Furthermore, for all $m = 1, \dots, \tilde{m}$,

$$H(i_m) = \min \left\{ H(i_1), \min_{j \in N(i_1)} H(j), \dots, \min_{j \in N(i_{m-1})} H(j) \right\}. \quad (8)$$

Proof Let i_m and i_{m+1} be two successive nodes generated by \mathcal{RH} , and let $(i_m, i'_{m+1}, i'_{m+2}, \dots, \bar{i}_m)$ be the path generated by \mathcal{H} starting from i_m , where \bar{i}_m is the projection of i_m . Then, since \mathcal{H} is sequentially consistent,

$$H(i_m) = H(i'_{m+1}) = g(\bar{i}_m).$$

Furthermore, since $i'_{m+1} \in N(i_m)$, using the definition of \mathcal{RH} [cf. Eq. (3)],

²For an example where this convention for tie-breaking is not observed and as a consequence \mathcal{RH} does not terminate, assume that there is a single destination d and that all other nodes are arranged in a cycle. Each non-destination node i has two outgoing arcs: one arc that belongs to the cycle and another arc which is (i, d) . Let \mathcal{H} be the (sequentially consistent) base heuristic that, starting from a node $i \neq d$, generates the path (i, d) . When the terminal node of the path is node i , the rollout algorithm \mathcal{RH} compares the two neighbors of i , which are d and the node next to i on the cycle, call it j . Both neighbors have d as their projection, so there is tie in Eq. (6). It can be seen that if \mathcal{RH} breaks ties in favor of the neighbor j that lies on the cycle, then \mathcal{RH} continually repeats the cycle and never terminates.

$$H(i'_{m+1}) \geq \min_{j \in N(i_m)} H(j) = H(i_{m+1}).$$

Combining the last two relations,

$$H(i_m) \geq H(i_{m+1}) = \min_{j \in N(i_m)} H(j). \quad (9)$$

To show that \mathcal{RH} is terminating, note that in view of Eq. (9), either $H(i_m) > H(i_{m+1})$ or else $H(i_m) = H(i_{m+1})$. In the latter case, in view of the convention for breaking ties that occur in Eq. (6) and the sequential consistency of \mathcal{H} , the path generated by \mathcal{H} starting from i_{m+1} is the tail portion of the path generated by \mathcal{H} starting from i_m and has one arc less. Thus, the number of nodes generated by \mathcal{RH} between successive times that the inequality $H(i_m) > H(i_{m+1})$ holds is finite. On the other hand, the inequality $H(i_m) > H(i_{m+1})$ can occur only a finite number of times, since the number of destination nodes is finite, and the destination node of the path generated by \mathcal{H} starting from i_m cannot be repeated if the inequality $H(i_m) > H(i_{m+1})$ holds. Therefore, \mathcal{RH} is terminating.

Finally, if $(i_1, \dots, i_{\tilde{m}})$ is the path generated by \mathcal{RH} , the relation (9) implies the desired relations (7) and (8). **Q. E. D.**

Proposition 1 shows that in the sequentially consistent case, the rollout algorithm \mathcal{RH} has an important “automatic cost sorting” property, whereby it follows the best path generated by the base heuristic \mathcal{H} . In particular, when \mathcal{RH} generates a path $(i_1, \dots, i_{\tilde{m}})$, it does so by using \mathcal{H} to generate a collection of other paths and corresponding projections starting from all the successor nodes of the intermediate nodes $i_1, \dots, i_{\tilde{m}-1}$. However, $(i_1, \dots, i_{\tilde{m}})$ is guaranteed to be the best among this path collection, and $i_{\tilde{m}}$ has minimal cost among all generated projections [cf. Eq. (8)]. Of course, this does not guarantee that the path generated by \mathcal{RH} will be a near-optimal path, because the collection of paths generated by \mathcal{H} may be “poor.” Still, the property whereby \mathcal{RH} at all times follows the best path found so far is intuitively reassuring.

The following example illustrates the preceding concepts.

Example 4 (One-Dimensional Walk) Consider a person who walks on a straight line and at each time period takes either a unit step to the left or a unit step to the right. There is a cost function assigning cost $g(i)$ to each integer i . Given an integer starting point on the line, the person wants to minimize the cost of the point where he will end up after a given and fixed number N of steps.

This problem can be formulated as a graph search problem of the type discussed in the preceding section. In particular, without loss of generality, assume that the starting point is the origin, so that the person’s position after n steps will be some integer in the interval $[-n, n]$. The nodes of the graph are identified with pairs (k, m) , where k is the number of steps taken so far ($k = 1, \dots, N$) and m is the person’s position ($m \in [-k, k]$). A node (k, m) with $k < N$ has two outgoing arcs

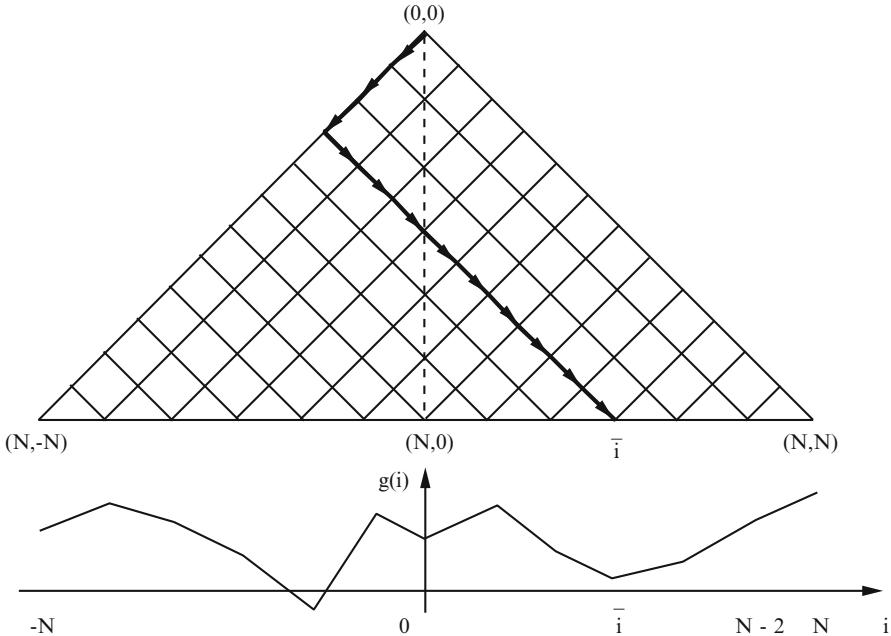


Fig. 3 Illustration of the path generated by the rollout algorithm \mathcal{RH} in [Example 4](#). The algorithm keeps moving to the *left* up to the time where the base heuristic \mathcal{H} generates two destinations (N, \bar{i}) and $(N, \bar{i} - 2)$ with $g(\bar{i}) \leq g(\bar{i} - 2)$. Then it continues to move to the *right* ending at the destination (N, \bar{i}) , which corresponds to the local minimum closest to N

with end nodes $(k+1, m-1)$ (corresponding to a left step) and $(k+1, m+1)$ (corresponding to a right step). The starting state is $(0, 0)$ and the terminating states are of the form (N, m) , where m is of the form $N-2l$ and $l \in [0, N]$ is the number of left steps taken.

Let the base heuristic \mathcal{H} be defined as the algorithm, which, starting at a node (k, m) , takes $N-k$ successive steps to the right and terminates at the node $(N, m+N-k)$. Note that \mathcal{H} is sequentially consistent. The rollout algorithm \mathcal{RH} , at node (k, m) , compares the cost of the destination node $(N, m+N-k)$ (corresponding to taking a step to the right and then following \mathcal{H}) and the cost of the destination node $(N, m+N-k-2)$ (corresponding to taking a step to the left and then following \mathcal{H}).

Let us say that an integer $i \in [-N+2, N-2]$ is a *local minimum* if $g(i-2) \geq g(i)$ and $g(i) \leq g(i+2)$. Let us also say that N (or $-N$) is a local minimum if $g(N-2) \leq g(N)$ [or $g(-N) \leq g(-N+2)$, respectively]. Then, it can be seen that starting from the origin $(0, 0)$, \mathcal{RH} obtains the local minimum that is closest to N (see [Fig. 3](#)). This is no worse (and typically better) than the integer N obtained by \mathcal{H} . This example illustrates how \mathcal{RH} may exhibit “intelligence” that is totally lacking from \mathcal{H} and is in agreement with the result of [Proposition 1](#).

2.1.1 Sequential Improvement

It is possible to show that the rollout algorithm improves on the base heuristic (cf. [Proposition 1](#)) under weaker conditions. To this end the following definition is introduced.

Definition 2 The base heuristic \mathcal{H} is said to be *sequentially improving* if for every non-destination node i ,

$$H(i) \geq \min_{j \in N(i)} H(j). \quad (10)$$

It can be seen that a sequentially consistent \mathcal{H} is also sequentially improving, since sequential consistency implies that $H(i)$ is equal to one of the values $H(j)$, $j \in N(i)$. The following proposition generalizes [Proposition 1](#).

Proposition 2 Let the base heuristic \mathcal{H} be sequentially improving and assume that the rollout algorithm \mathcal{RH} is terminating. Let $(i_1, \dots, i_{\tilde{m}})$ be the path generated by \mathcal{RH} starting from a non-destination node i_1 and ending at a destination node $i_{\tilde{m}}$. Then, the cost of \mathcal{RH} starting from i_1 is less or equal to the cost of \mathcal{H} starting from i_1 . In particular, for all $m = 1, \dots, \tilde{m}$,

$$H(i_m) = \min \left\{ H(i_1), \min_{j \in N(i_1)} H(j), \dots, \min_{j \in N(i_{m-1})} H(j) \right\}. \quad (11)$$

Proof For each $m = 1, \dots, \tilde{m} - 1$,

$$H(i_m) \geq \min_{j \in N(i_m)} H(j),$$

by the sequential improvement assumption, while

$$\min_{j \in N(i_m)} H(j) = H(i_{m+1}),$$

by the definition of the rollout algorithm. These two relations imply Eq. (11). Since the cost of \mathcal{RH} starting from i_1 is $H(i_{\tilde{m}})$, the result follows. **Q. E. D.**

Example 5 (One-Dimensional Walk: Continued) Consider the one-dimensional walk problem of [Example 4](#), and let \mathcal{H} be defined as the algorithm that, starting at a node (k, m) , compares the cost $g(m + N - k)$ (corresponding to taking all of the remaining $N - k$ steps to the right) and the cost $g(m - N + k)$ (corresponding to taking all of the remaining $N - k$ steps to the left) and accordingly moves to node

$$(N, m + N - k) \quad \text{if} \quad g(m + N - k) \leq g(m - N + k),$$

or to node

$$(N, m - N + k) \quad \text{if} \quad g(m - N + k) < g(m + N - k).$$

It can be seen that \mathcal{H} is not sequentially consistent, but is instead sequentially improving. Using Eq. (11), it follows that starting from the origin $(0, 0)$, \mathcal{RH} obtains the global minimum of g in the interval $[-N, N]$, while \mathcal{H} obtains the better of the two points $-N$ and N .

Proposition 2 actually follows from a general equation for the cost of the path generated by the rollout algorithm, which holds for any base heuristic (not necessarily one that is sequentially improving). This is given in the following proposition.

Proposition 3 Assume that the rollout algorithm \mathcal{RH} is terminating. Let $(i_1, \dots, i_{\tilde{m}})$ be the path generated by \mathcal{RH} starting from a non-destination node i_1 and ending at a destination node $i_{\tilde{m}}$. Then, the cost of \mathcal{RH} starting from i_1 is equal to

$$H(i_1) + \delta_{i_1} + \dots + \delta_{i_{\tilde{m}-1}},$$

where for every non-destination node i ,

$$\delta_i = \min_{j \in N(i)} H(j) - H(i).$$

Proof By the definition of the rollout algorithm

$$H(i_m) + \delta_{i_m} = \min_{j \in N(i_m)} H(j) = H(i_{m+1}), \quad m = 1, \dots, \tilde{m} - 1.$$

By adding these equations over m ,

$$H(i_1) + \delta_{i_1} + \dots + \delta_{i_{\tilde{m}-1}} = H(i_{\tilde{m}}).$$

Since the cost of \mathcal{RH} starting from i_1 is $H(i_{\tilde{m}})$, the result follows. **Q. E. D.**

If the base heuristic is sequentially improving, there holds $\delta_i \leq 0$ for all non-destination nodes i , so it follows from **Proposition 3** that the cost of the rollout algorithm is less or equal to the cost of the base heuristic (cf. **Proposition 2**).

2.1.2 The Fortified Rollout Algorithm

A variant of the rollout algorithm will now be described, which implicitly uses a sequentially improving base heuristic, so that it has the cost improvement property of **Proposition 2**. This variant, called the *fortified rollout algorithm*, and denoted by \mathcal{RH} , starts at the origin s , and after m steps, maintains, in addition to the current sequence of nodes (s, i_1, \dots, i_m) , a path

$$P(i_m) = (i_m, i'_{m+1}, \dots, i'_k),$$

ending at a destination i'_k . Roughly speaking, the path $P(i_m)$ is the tail portion of the best path found after the first m steps of the algorithm, in the sense that the destination i'_k has minimal cost over all the projections of nodes calculated thus far.

In particular, initially $P(s)$ is the path generated by the base heuristic \mathcal{H} starting from s . At the typical step of the fortified rollout algorithm \mathcal{RH} , a node sequence (s, i_1, \dots, i_m) has been constructed and the path $P(i_m) = (i_m, i'_{m+1}, \dots, i'_k)$ is available, where i_m is not a destination. Then, if

$$\min_{j \in N(i_m)} H(j) < g(i'_k), \quad (12)$$

\mathcal{RH} adds to the node sequence (s, i_1, \dots, i_m) the node

$$i_{m+1} \in \arg \min_{j \in N(i_m)} H(j)$$

and sets $P(i_{m+1})$ to the path generated by \mathcal{H} , starting from i_{m+1} . On the other hand, if

$$\min_{j \in N(i_m)} H(j) \geq g(i'_k), \quad (13)$$

\mathcal{RH} adds to the node sequence (s, i_1, \dots, i_m) the node

$$i_{m+1} = i'_{m+1}$$

and sets $P(i_{m+1})$ to the path $(i_{m+1}, i'_{m+2}, \dots, i'_k)$. If i_{m+1} is a destination, \mathcal{RH} terminates, and otherwise, \mathcal{RH} repeats the process with (s, i_1, \dots, i_{m+1}) replacing (s, i_1, \dots, i_m) and $P(i_{m+1})$ replacing $P(i_m)$, respectively.

The idea behind the construction of \mathcal{RH} is to follow the path $P(i_m)$ unless a path of lower cost is discovered through Eq. (12). It can be shown that \mathcal{RH} may be viewed as the rollout algorithm \mathcal{RH} corresponding to a modified version of \mathcal{H} , called *fortified* \mathcal{H} , and denoted $\bar{\mathcal{H}}$. This algorithm is applied to a slightly modified version of the original problem, which involves an additional downstream neighbor for each node i_m that is generated in the course of the algorithm \mathcal{RH} and for which the condition (13) holds. For every such node i_m , the additional neighbor is a copy of i'_{m+1} , and the path generated by $\bar{\mathcal{H}}$ starting from this copy is (i'_{m+1}, \dots, i'_k) . From every other node, the path generated by $\bar{\mathcal{H}}$ is the same as the path generated by \mathcal{H} .

It can be seen that $\bar{\mathcal{H}}$ is sequentially improving, so that \mathcal{RH} is terminating and has the automatic cost sorting property of [Proposition 2](#); that is,

$$H(i_m) = \min \left\{ H(i_1), \min_{j \in N(i_1)} H(j), \dots, \min_{j \in N(i_{m-1})} H(j) \right\}.$$

The above property can also be easily verified directly, using the definition of \mathcal{RH} . Finally, it can be seen that when \mathcal{H} is sequentially consistent, the rollout algorithm \mathcal{RH} and its fortified version \mathcal{RH} coincide.

2.1.3 Using Multiple Base Heuristics: Parallel Rollout

In many problems, several promising path construction heuristics may be available. It is then possible to use all of these heuristics in parallel within the rollout framework, essentially by combining them into a single “superheuristic.” In particular, let us assume that K algorithms $\mathcal{H}_1, \dots, \mathcal{H}_K$ are available. The k th of these algorithms, given a non-destination node i , produces a path $(i, i_1, \dots, i_m, \bar{i})$ that ends at a destination node \bar{i} , and the corresponding cost is denoted by $H_k(i) = g(\bar{i})$. The K algorithms can be incorporated in a generalized version of the rollout algorithm, which uses the minimal cost

$$H(i) = \min_{k=1, \dots, K} H_k(i), \quad (14)$$

in place of the cost obtained by any one of the K algorithms $\mathcal{H}_1, \dots, \mathcal{H}_K$.

In particular, the algorithm starts with the origin node s . At the typical step, given a node sequence (s, i_1, \dots, i_m) , where i_m is not a destination, the algorithm adds to the sequence a node i_{m+1} such that

$$i_{m+1} \in \arg \min_{j \in N(i_m)} H(j).$$

If i_{m+1} is a destination node, the algorithm terminates, and otherwise, the process is repeated with the sequence $(s, i_1, \dots, i_m, i_{m+1})$ replacing (s, i_1, \dots, i_m) .

An interesting property, which can be readily verified by using the definitions, is that if all the algorithms $\mathcal{H}_1, \dots, \mathcal{H}_K$ are sequentially improving, the same is true for \mathcal{H} . This is evident from the fact that the heuristics run in parallel from a given node may be viewed as a single heuristic, which has the cost improvement property of [Definition 2](#). The fortified version of the rollout algorithm \mathcal{RH} easily generalizes for the case of Eq. (14), by defining the path generated starting from a node i as the path generated by the path construction algorithm, which attains the minimum in Eq. (14).

In an alternative version of the rollout algorithm that uses multiple path construction heuristics, the results of the K algorithms $\mathcal{H}_1, \dots, \mathcal{H}_K$ are weighted with some fixed scalar weights r_k to compute $H(i)$ for use in Eq. (3):

$$H(i) = \sum_{k=1}^K r_k H_k(i). \quad (15)$$

The weights r_k may be adjusted by trial and error or more sophisticated techniques which may be found in the literature (see e.g., [6]).

2.1.4 Extension for Intermediate Arc Costs

Let us consider a variant of the graph search problem where in addition to the terminal cost $g(i)$, there is a cost $c(i, j)$ for a path to traverse an arc (i, j) . Within this context, the cost of a path (i_1, i_2, \dots, i_n) that starts at i_1 and ends at a destination node i_n is redefined to be

$$g(i_n) + \sum_{k=1}^{n-1} c(i_k, i_{k+1}). \quad (16)$$

Note that when the cost $g(i)$ is zero for all destination nodes i , this is the problem of finding a shortest path from the origin node s to one of the destination nodes, with $c(i, j)$ viewed as the length of arc (i, j) . However, here we are interested in problems where the number of nodes is very large, and the use of the shortest path algorithms is impractical.

One way to transform the problem with arc costs into one involving a terminal cost only is to redefine the graph of the problem so that nodes correspond to sequences of nodes in the original problem graph. Thus, having arrived at node i_k using path (i_1, \dots, i_k) , the choice of i_{k+1} as the next node is viewed as a transition from (i_1, \dots, i_k) to $(i_1, \dots, i_k, i_{k+1})$. Both nodes (i_1, \dots, i_k) and $(i_1, \dots, i_k, i_{k+1})$ are viewed as nodes of a redefined graph. Furthermore, in this redefined graph, a destination node has the form (i_1, i_2, \dots, i_n) , where i_n is a destination node of the original graph, and has a cost given by Eq. (16).

After the details are worked out, it can be seen that to recover our earlier algorithms and analysis, the cost of the heuristic algorithm \mathcal{H} needs to be modified as follows: If the path (i_1, \dots, i_n) is generated by \mathcal{H} starting at i_1 , then

$$H(i_1) = g(i_n) + \sum_{k=1}^{n-1} c(i_k, i_{k+1}).$$

Furthermore, the rollout algorithm \mathcal{RH} at node i_m selects as next node i_{m+1} the node

$$i_{m+1} \in \arg \min_{j \in N(i_m)} [c(i_m, j) + H(j)];$$

[cf. Eq. (3)]. The definition of a sequentially consistent algorithm remains unchanged. Furthermore, [Proposition 1](#) remains unchanged except that Eqs. (7) and (8) are modified to read

$$H(i_k) \geq c(i_k, i_{k+1}) + H(i_{k+1}) = \min_{j \in N(i_k)} [c(i_k, j) + H(j)], \quad k = 1, \dots, m-1.$$

A sequentially improving algorithm should now be characterized by the property

$$H(i_k) \geq c(i_k, i_{k+1}) + H(i_{k+1}),$$

where i_{k+1} is the next node on the path generated by \mathcal{H} starting from i_k . Furthermore, [Proposition 2](#) remains unchanged, except that Eq. (11) is modified to read

$$H(i_k) \geq \min_{j \in N(i_k)} [c(i_k, j) + H(j)], \quad k = 1, \dots, m-1.$$

Finally, the criterion $\min_{j \in N(i_m)} H(j) < g(i'_k)$ [cf. Eq. (12)] used in the fortified rollout algorithm, given the sequence (s, i_1, \dots, i_m) , where $i_m \notin \bar{\mathcal{N}}$, and the path $P(i_m) = (i_m, i'_{m+1}, \dots, i'_k)$, should be replaced by

$$\min_{j \in N(i_m)} [c(i_m, j) + H(j)] < g(i'_k) + c(i_m, i'_{m+1}) + \sum_{l=m+1}^{k-1} c(i'_l, i'_{l+1}).$$

2.1.5 Rollout Algorithms with Multistep Lookahead

It is possible to incorporate *multistep lookahead* into the rollout framework. To describe the case of 2-step lookahead, suppose that after m steps of the rollout algorithm, the current node sequence is (s, i_1, \dots, i_m) . Then, the set of all 2-step-ahead neighbors of i_m is considered, defined as

$$N_2(i_m) = \{j \in \mathcal{N} \mid j \in N(i_m) \text{ and } j \in \bar{\mathcal{N}}, \text{ or } j \in N(n) \text{ for some } n \in N(i_m)\}.$$

The base heuristic \mathcal{H} is then run starting from each $j \in N_2(i_m)$, and the node $\bar{j} \in N_2(i_m)$ that has projection of minimum cost is found. Let $i_{m+1} \in N(i_m)$ be the node next to i_m on the (one- or two-arc) path from i_m to \bar{j} . If i_{m+1} is a destination node, the algorithm terminates. Otherwise, the process is repeated with the sequence $(s, i_1, \dots, i_m, i_{m+1})$ replacing (s, i_1, \dots, i_m) .

Note that a fortified version of the rollout algorithm described above is possible along the lines described earlier. Also, it is possible to eliminate from the set $N_2(i_m)$ some of the 2-step neighbors of i_m that are judged less promising according to some heuristic criterion, in order to limit the number of applications of the base heuristic. This may be viewed as *selective depth lookahead*. Finally, the extension of the algorithm to look ahead more than two steps is straightforward: The 2-step-ahead neighbor set $N_2(i_m)$ is simply replaced with a suitably defined k -step-ahead neighbor set $N_k(i_m)$.

2.1.6 Interpretation in Terms of DP

Let us now reinterpret the graph-based rollout algorithm within the context of deterministic DP. The base heuristic will be viewed as a suboptimal policy, and the rollout algorithm will be viewed as a policy obtained by a process of policy improvement, provided the base heuristic is sequentially consistent.

To this end, the graph search problem is cast as a sequential decision problem, where each node corresponds to a state of a dynamic system. At each non-destination node/state i , a node j must be selected from the set of neighbors $N(i)$; then, if j is a destination, the process terminates with cost $g(j)$, and otherwise, the process is repeated with j becoming the new state. The DP algorithm calculates for every node i , the minimal cost that can be achieved starting from i , that is, the smallest value of $g(\bar{i})$ that can be obtained using paths that start from i and end at destination nodes \bar{i} . This value, denoted $J^*(i)$, is the optimal cost-to-go starting at node i . Once $J^*(i)$ is computed for all nodes i , an optimal path (i_1, i_2, \dots, i_m)

can be constructed starting from any initial node/state i_1 by successively generating nodes using the relation

$$i_{k+1} \in \arg \min_{j \in N(i_k)} J^*(j), \quad k = 1, \dots, m-1, \quad (17)$$

up to the point where a destination node i_m is encountered.³

A base heuristic \mathcal{H} defines a policy π , that is, an assignment of a successor node to any non-destination node. However, starting from a given node i , the cost of π need not be equal to $H(i)$ because if a path $(i_1, i_2, i_3, \dots, i_m)$ is generated by \mathcal{H} starting from node i_1 , it is not necessarily true that the path (i_2, i_3, \dots, i_m) is generated by the base heuristic starting from i_2 . Thus, the successor node chosen at node i_2 by policy π may be different than the one used in the calculation of $H(i_1)$. On the other hand, if \mathcal{H} is sequentially consistent, the cost of policy π starting from a node i is $H(i)$, since sequential consistency implies that the path that the base heuristic generates starting at the successor node is part of the path it generates at the predecessor node. It turns out that the cost improvement property of the rollout algorithm in the sequentially consistent case is a special case of a cost improvement property for rollout algorithms that holds for more general DP contexts, including stochastic ones.

Generally, in the DP context the rollout algorithm \mathcal{RH} is viewed as a one-step lookahead policy that uses $H(j)$ as a cost-to-go approximation from state j . In some cases, $H(j)$ is the cost of some policy (in the DP sense), such as when \mathcal{H} is sequentially consistent, as explained above. In general, however, this need not be so, in which case $H(j)$ can be viewed as a convenient cost-to-go approximation that is derived from the base heuristic. Still, the rollout algorithm \mathcal{RH} may improve on the cost of the base heuristic (e.g., when \mathcal{H} is sequentially improving, cf. [Proposition 2](#)).

3 Applications in Discrete Optimization

Finally, to provide some perspective on the rollout methodology, let us explore the connections with some important discrete optimization problems and methods. In particular, let us consider the generic discrete optimization problem of minimizing a cost function $g(x)$ over a finite set X of feasible solutions. This problem will be reformulated as a graph search problem. It may be noted that several reformulations are possible, and different choices of the underlying graph give rise to different rollout algorithms. Thus, it is important to select a reformulation that matches the type of rollout algorithm one wishes to develop. Some reformulations and corresponding rollout algorithms will be discussed in what follows, and these algorithms will be related to some general computational methods.

³It is assumed here that there are no termination/cycling difficulties of the type illustrated in the footnote following [Example 3](#).

Suppose that each solution x has N components; that is, it has the form $x = (x_1, \dots, x_N)$, where N is a positive integer. For example, in a 0-1 integer programming problem, each component x_k may correspond to a single variable that can take the values 0 or 1, or alternatively, it may correspond to a multidimensional vector involving several variables each taking the values 0 or 1. In a network optimization problem, each component x_k may correspond to a vector involving the flows of several arcs of the network. One way to reformulate the problem

$$\begin{aligned} & \text{minimize } g(x) \\ & \text{subject to } x \in X \end{aligned}$$

into the framework of the search problem is to introduce an acyclic graph involving an artificial origin node s and N subsets of nodes I_1, \dots, I_N . In particular, for each feasible solution $x \in X$ and each $k = 1, \dots, N$, the node set I_k contains a node $(x(k), k)$, where $x(k)$ consists of the first k components of x [two feasible solutions $x, x' \in X$ whose first k components are identical are mapped onto the same node $(x(k), k)$ of I_k]. Each node $(x(N), N) \in I_N$ is viewed as a destination node of the graph and has cost $g(x)$, where x is the feasible solution mapping onto $x(N)$. The origin node is connected with an arc to each node $(x(1), 1) \in I_1$. Furthermore, for every $k = 1, \dots, N-1$, each node $(x(k), k) \in I_k$ is connected with an arc to each node $(x(k+1), k+1) \in I_{k+1}$ such that the components x_1, \dots, x_k of $x(k)$ and the first k components $x(k+1)$ are identical. A few observations may be made:

- (a) Selecting one neighbor out of the set of neighbors of the origin node amounts to selecting the first component x_1 of x , while selecting one neighbor out of the set of neighbors of a node $(x(k), k) \in I_k$ amounts to selecting the $(k+1)$ st component x_{k+1} of x .
- (b) Choosing a path that starts at s and ends at a destination node $(x(N), N)$ amounts to a sequential choice of the components of x : The first component is chosen when the arc connecting s to a node $(x(1), 1) \in I_1$ is selected, and the k th component is chosen ($k = 2, \dots, N$) when the arc connecting a node $(x(k-1), k-1) \in I_{k-1}$ to a node $(x(k), k) \in I_k$ is selected. For each $k = 2, \dots, N$, the first $k-1$ components of $x(k-1)$ and $x(k)$ are identical.
- (c) Any base heuristic that starts at node s amounts to a sequential choice of the components x_k , $k = 1, \dots, N$, so that the final result (x_1, \dots, x_N) is feasible (belongs to X). Any base heuristic that starts at a non-destination node $(x(k), k) \in I_k$ amounts to a sequential choice of the components x_{k+1}, \dots, x_N , so that after they are added to the k components x_1, \dots, x_k specified by $x(k)$, the final result (x_1, \dots, x_N) is feasible.

Given a base heuristic \mathcal{H} , as described in (c) above, the k th step of the rollout algorithm \mathcal{RH} minimizes g with respect to the k th component x_k , while keeping the preceding components x_1, \dots, x_{k-1} at the values selected at the preceding steps, and using the base heuristic \mathcal{H} to supply the remaining components x_{k+1}, \dots, x_N .

Here is an example where the base heuristic is trivial, and the rollout algorithm leads to a well-known method.

Example 6 (Coordinate Descent) Assume that the set X has the (Cartesian product) form

$$\{(x_1, \dots, x_N) \mid x_k \in X_k, k = 1, \dots, N\}, \quad (18)$$

where $X_k, k = 1, \dots, N$, are some given finite sets. In principle, there is no loss of generality in this assumption since sets X_k such that the set (18) contains X can always be found, and the cost $g(x)$ can be set to a very high value for every $x \notin X$ that belongs to the set (18).

Let \bar{x} be some given feasible solution. Consider the base heuristic that operates as follows:

- (a) Starting from the origin s , it generates the solution \bar{x} .
- (b) Starting from $(x(k), k) \in I_k, k = 1, \dots, N_1$, it generates the solution that has the first k components equal to the corresponding k components of $x(k)$ and the last $N - k$ components equal to the corresponding last $N - k$ components of \bar{x} .

Then, it can be seen that the rollout algorithm is equivalent to a coordinate descent method that starts from \bar{x} and yields $(\tilde{x}_1, \dots, \tilde{x}_N)$ according to

$$\begin{aligned} \tilde{x}_1 &\in \arg \min_{x_1 \in X_1} g(x_1, \bar{x}_2, \dots, \bar{x}_N), \\ \tilde{x}_k &\in \arg \min_{x_k \in X_k} g(\tilde{x}_1, \dots, \tilde{x}_{k-1}, x_k, \bar{x}_{k+1}, \dots, \bar{x}_N), \quad k = 2, \dots, N. \end{aligned}$$

Note that a more general version of coordinate descent is obtained by using a base heuristic similarly defined by multiple solutions $\bar{x}^1, \dots, \bar{x}^m$ in place of \bar{x} . Then, the preceding equation is replaced by

$$\begin{aligned} \tilde{x}_1 &\in \arg \min_{x_1 \in X_1} \min \{g(x_1, \bar{x}_2^1, \dots, \bar{x}_N^1), \dots, g(x_1, \bar{x}_2^m, \dots, \bar{x}_N^m)\}, \\ \tilde{x}_k &\in \arg \min_{x_k \in X_k} \min \{g(\tilde{x}_1, \dots, \tilde{x}_{k-1}, x_k, \bar{x}_{k+1}^1, \dots, \bar{x}_N^1), \dots, \\ &\quad g(\tilde{x}_1, \dots, \tilde{x}_{k-1}, x_k, \bar{x}_{k+1}^m, \dots, \bar{x}_N^m)\}, \quad k = 2, \dots, N. \end{aligned}$$

The preceding example provides a baseline. It shows what can be achieved in a coordinate-based formulation of the rollout algorithm, even with a very trivial base heuristic. One can expect much better performance with more sophisticated base heuristics.

There are interesting variations of the coordinate-based reformulation of the generic discrete optimization problem into a graph search problem. In particular, the problem has been reformulated so that the components of x_1, \dots, x_N are selected in a specific order. Alternative orders are possible, and in fact an attempt to optimize the choice of order may be effected through the rollout algorithm. This can be done by introducing the index of the component of x as part of the specification of a node. In particular, the nodes in the “layer” I_k of the graph may have the form $(x(k), k, n)$ where n specifies the next component to be chosen by the rollout algorithm.

Let us also explore the relation between rollout algorithms and local search methods, which are a broad and important class of heuristics for the generic discrete optimization problem of minimizing $g(x)$ over the finite set X . A local search method uses the notion of a *neighborhood* $N(x)$ of a feasible solution $x \in X$, which is a (usually small) subset of X , containing solutions that are “close” to x in some sense.

In particular, given a solution x , the method selects among the solutions in the neighborhood $N(x)$ a successor solution x' , according to some rule. The process is then repeated with x' replacing x (or stops when some termination criterion is met). Thus, a local search method is characterized by:

- (a) The method for choosing a starting solution
- (b) The definition of the neighborhood $N(x)$ of a solution x
- (c) The rule for selecting a successor solution from within $N(x)$
- (d) The termination criterion

The definition of a neighborhood often involves intricate calculations and suboptimizations that aim to bring to consideration promising neighbors. While the definition of neighborhood is typically problem dependent, some general classes of procedures for generating neighborhoods have been developed. An example of such a class is the well-known *genetic algorithms*.

The criterion for selecting a solution from within a neighborhood is usually the cost of the solution, so that a neighbor of minimum cost is selected. Then, by assuming that each $x \in X$ belongs to its own neighborhood $N(x)$, the local search is *cost improving* and effectively stops at a *local minimum*, that is, a solution that is no worse than all other solutions within its neighborhood. Attention will be restricted to such methods, but there are important alternatives, such as in the methods of *tabu search* and *simulated annealing*, which will not be discussed here.

Consider a cost-improving local search method, as described above, and let $N(x)$ and \bar{x} be the neighborhood definition and the starting point of the method, respectively. Let us assume that there is a given limit M to the number of iterations, so that the method terminates when it reaches this limit (if it encounters a local minimum before M iterations are performed, it may be assumed that it simply repeats the local minimum, until the limit M is reached). We will provide a reformulation of the problem into the framework of the search problem of the preceding section, so that the rollout algorithm becomes identical to the local search method described above.

To this end, an acyclic graph is introduced, which consists of an origin node that corresponds to the starting solution \bar{x} , and M subsets of nodes I_1, I_2, \dots, I_M , which may be viewed as replicas of the feasible set X . In particular, for each feasible solution $x \in X$ and each $k = 1, \dots, M$, the node set I_k contains a node (x, k) . Each node $(x, M) \in I_M$ is viewed as a destination node of the graph and has cost $g(x)$. The origin node is connected with an arc to each node $(x, 1)$ such that $x \in N(\bar{x})$, while for every $k = 1, \dots, M - 1$, each node $(x, k) \in I_k$ is connected with an arc to each node $(x', k + 1) \in I_{k+1}$ such that $x' \in N(x)$.

Consider now the base heuristic that, starting from a node $(x, k) \in I_k$, generates the destination (x, N) with cost $g(x)$. Then, it can be seen that the rollout algorithm

reduces to the local search method. In particular, the rollout algorithm, given x after k steps [i.e., when at node (x, k)], it considers all x' in the neighborhood $N(x)$ and runs the base heuristic starting at x' and yielding the cost $g(x')$. It then selects $x' \in N(x)$ that yields the minimum cost. This is exactly what the local search method also does.

The preceding reformulation suggests that rollout algorithms can provide an additional dimension to local search methods, whereby intermediate infeasible solutions may be generated, and these solutions are evaluated through their projections, which are obtained through a base heuristic. Thus, while local search methods rely on a single construct, namely, neighborhoods, for selecting successive solutions, rollout algorithms bring to bear two independent constructs, neighborhoods *and* base heuristics. It should be mentioned also that rollout algorithms embody some additional important methodological ideas, namely, DP and policy iteration. For this reason, rollout algorithms admit natural extensions to stochastic control problems, for which there is no known analog of a local search method.

4 Further Reading

The main idea of rollout algorithms, obtaining an improved policy starting from some other suboptimal policy using a one-time policy improvement, has appeared in several DP application contexts. In the context of game-playing computer programs, it has been proposed by Abramson [1] and by Tesauro and Galperin [24]. The name “rollout” was coined by Tesauro in specific reference to rolling the dice in the game of backgammon. In Tesauro’s proposal, a given backgammon position is evaluated by “rolling out” many games starting from that position, using a simulator, and the results are averaged to provide a “score” for the position. The internet contains a lot of material on computer backgammon and the use of rollout, in some cases in conjunction with multistep lookahead and cost-to-go approximation.

The application of rollout algorithms to discrete optimization problems has its origin in the neuro-dynamic programming work of the author and J. Tsitsiklis [6]. The formalization as a path construction algorithm, including the notions of sequential consistency, sequential improvement, and fortified and parallel rollout, was given in the paper by Bertsekas, Tsitsiklis, and Wu [7]. The subsequent paper by Bertsekas and Castanon [5] considered its application to stochastic DP and stochastic scheduling. The analysis of the breakthrough problem ([Example 1](#)) is given in the DP book by Bertsekas [3] and is based on unpublished work by Bertsekas, Castanon, and Tsitsiklis. An analysis of the optimal policy and some suboptimal policies for this problem is given by Pearl [18]. A discussion of rollout algorithms as applied to network optimization problems may be found in the author’s network optimization book [2].

For applications of rollout algorithms, see Christodouleas [11], Duin and Voss [12], Secomandi [20–22], Ferris and Voelker [13, 14], McGovern, Moss, and Barto [16], Savagaonkar, Givan, and Chong [19], Bertsimas and Popescu [8], Guerriero and Mancini [15], Tu and Pattipati [25], Wu, Chong, and Givan [26],

Chang, Givan, and Chong [10], Meloni, Pacciarelli, and Pranzo [17], Yan, Diaconis, Rusmevichientong, and Van Roy [27], Besse and Chaib-draa [9], and Sun, Zhao, Lun, and Tomastik [23]. These works discuss a broad variety of applications and case studies and generally report positive computational experience. The survey [4] discusses rollout algorithms from a control theory point of view and explores its close connection with model predictive control (MPC).

Recommended Reading

1. B. Abramson, Expected-outcome: a general model of static evaluation. *IEEE Trans. Pattern Anal. Mach. Intell.* **12**, 182–193 (1990)
2. D.P. Bertsekas, *Network Optimization: Continuous and Discrete Models* (Athena Scientific, Belmont, 1998)
3. D.P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. I (Athena Scientific, Belmont, 2005)
4. D.P. Bertsekas, Dynamic programming and suboptimal control: a survey from ADP to MPC, in *Fundamental Issues in Control*. *Eur J. Control*, **11**(4–5), 310–334 (2005)
5. D.P. Bertsekas, D.A. Castanon, Rollout algorithms for stochastic scheduling problems. *Heuristics*, **5**, 89–108 (1999)
6. D.P. Bertsekas, J.N. Tsitsiklis, *Neuro-Dynamic Programming* (Athena Scientific, Belmont, 1996)
7. D.P. Bertsekas, J.N. Tsitsiklis, C. Wu, Rollout algorithms for combinatorial optimization. *Heuristics*, **3**, 245–262 (1997)
8. D. Bertsimas, I. Popescu, Revenue management in a dynamic network environment. *Transp. Sci.* **37**, 257–277 (2003)
9. C. Besse, B. Chaib-draa, Parallel rollout for online solution of DEC-POMDPs, in *Proceedings of 21st International FLAIRS Conference*, Coconut Grove, FL, 2008, pp. 619–624
10. H.S. Chang, R.L. Givan, E.K.P. Chong, Parallel rollout for online solution of partially observable Markov decision processes. *Discret. Event Dyn. Syst.* **14**, 309–341 (2004)
11. J.D. Christodouleas, Solution methods for multiprocessor network scheduling problems with application to railroad operations, Ph.D. thesis, Operations Research Center, Massachusetts Institute of Technology, 1997
12. C. Duin, S. Voss, The pilot method: a strategy for heuristic repetition with application to the Steiner problem in graphs. *Networks*, **34**, 181–191 (1999)
13. M.C. Ferris, M.M. Voelker, Neuro-dynamic programming for radiation treatment planning. Numerical Analysis Group Research Report NA-02/06, Oxford University Computing Laboratory, Oxford University, 2002
14. M.C. Ferris, M.M. Voelker, Fractionation in radiation treatment planning. *Math. Program. B* **102**, 387–413 (2004)
15. F. Guerriero, M. Mancini, A cooperative parallel rollout algorithm for the sequential ordering problem. *Parallel Comput.* **29**, 663–677 (2003)
16. A. McGovern, E. Moss, A. Barto, Building a basic building block scheduler using reinforcement learning and rollouts. *Mach. Learn.* **49**, 141–160 (2002)
17. C. Meloni, D. Pacciarelli, M. Pranzo, A rollout metaheuristic for job shop scheduling problems. *Ann. Oper. Res.* **131**, 215–235 (2004)
18. J. Pearl, *Heuristics* (Addison-Wesley, Reading, 1984)
19. U. Savagaonkar, R. Givan, E.K.P. Chong, Sampling techniques for zero-sum, discounted Markov games, in *Proceedings of 40th Allerton Conference on Communication, Control and Computing*, Monticello, IL, 2002
20. N. Secomandi, Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. *Comput. Oper. Res.* **27**, 1201–1225 (2000)

21. N. Secomandi, A rollout policy for the vehicle routing problem with stochastic demands. *Oper. Res.* **49**, 796–802 (2001)
22. N. Secomandi, Analysis of a rollout approach to sequencing problems with stochastic routing applications. *J. Heuristics*, **9**, 321–352 (2003)
23. T. Sun, Q. Zhao, P. Lun, R. Tomastik, Optimization of joint replacement policies for multipart systems by a rollout framework. *IEEE Trans. Autom. Sci. Eng.* **5**, 609–619 (2008)
24. G. Tesauro, G.R. Galperin, On-line policy improvement using Monte Carlo search. Presented at the 1996 neural information processing systems conference, Denver, CO, 1996; also in *Advances in Neural Information Processing Systems 9*, ed. by M. Mozer et al. (MIT, 1997)
25. F. Tu, K.R. Pattipati, Rollout strategies for sequential fault diagnosis. *IEEE Trans. Syst. Man Cybern. Part A* **33**, 86–99 (2003)
26. G. Wu, E.K.P. Chong, R.L. Givan, Congestion control using policy rollout, in *Proceedings of 2nd IEEE CDC*, Maui, HI, 2003, pp. 4825–4830
27. X. Yan, P. Diaconis, P. Rusmevichientong, B. Van Roy, Solitaire: man versus machine. *Adv. Neural Inf. Process. Syst.* **17**, 1553–1560 (2005).

Simplicial Methods for Approximating Fixed Point with Applications in Combinatorial Optimizations

Chuangyin Dang

Contents

1	Introduction	3016
2	Several Simple Triangulations	3018
2.1	The $K_2(m)$ -Triangulation of S^n	3018
2.2	The K_1 -Triangulation of R^n	3019
2.3	The D_1 -Triangulation of R^n	3019
3	Scarf's Simplicial Method for Approximating Fixed Points on the Unit Simplex	3020
4	Kuhn's Simplicial Methods for Approximating Fixed Points on the Unit Simplex	3022
4.1	Kuhn's Artificial Start Simplicial Method	3022
4.2	Kuhn's Variable Dimension Simplicial Method	3024
5	Eaves's Simplicial Homotopy Method for Approximating Fixed Points on the Unit Simplex	3025
6	van der Laan and Talman's Variable Dimension Simplicial Method for Approximating Fixed Points on the Unit Simplex	3026
7	A Variable Dimension Simplicial Method for Integer Programming	3029
8	A Homotopy-Like Simplicial Method for Integer Programming	3043
9	Conclusion	3049
	Recommended Reading	3049

Abstract

Simplicial methods were originated by Scarf for approximating fixed points of continuous mappings. They have many applications in economics and science. With regard to its geometric structure, a simplicial method can be classified as either a variable dimension simplicial method or a homotopy simplicial method. A variable dimension simplicial method works directly on the interested space,

C. Dang (✉)

Department of Systems Engineering and Engineering Management, City University of Hong Kong, Kowloon, Hong Kong
e-mail: mecdang@cityu.edu.hk

whereas a simplicial homotopy method needs to introduce an extra dimension. It is well known that integer programming is equivalent to determining whether there is an integer point in a polytope. Simplicial methods were extended to computing an integer point in a polytope. There is a significant difference between simplicial methods for approximating fixed points and simplicial methods for integer programming, though they both have the same foundation. Two most important components of simplicial methods are labeling rules and triangulations. Efficiency of simplicial methods depends critically on the underlying triangulations. Three simplest triangulations of R^n are the K_1 -triangulation, the J_1 -triangulation, and the D_1 -triangulation. This chapter presents a brief introduction to these developments of simplicial methods.

1 Introduction

Brouwer's fixed point theorem (Brouwer 1912) asserts that a continuous mapping from a convex and compact nonempty set of R^n to itself has a fixed point. This theorem and its generalizations in Kakutani [83] have many applications in economics and engineering. However, the nonconstructive proofs of these fixed point theorems limited their further applications. To overcome this shortcoming, Scarf presented in his seminal paper [144] the first method for constructively proving Brouwer's fixed point theorem on S^n (here S^n denotes the unit simplex that is a subset of R^n in which all the components of every point are nonnegative and sum up to one). Since then the substantial developments have been made on approximating fixed points.

Scarf's method subdivides the unit simplex in a large number of the so-called primitive sets. Starting at a corner of the unit simplex, the method generates a sequence of adjacent primitive sets until it meets a primitive set that yields an approximate fixed point. However, operating with the primitive sets consumes an extremely large amount of computer storage. To address this difficulty, a systematic procedure was proposed in Hansen [74] for choosing a primitive set. It was found later that the pivot steps for moving from one primitive set to an adjacent one in the procedure are identical to those of a regular triangulation of S^n given in Freudenthal [52], which was applied in Kuhn [97] to approximate fixed points. Although Scarf's method introduces the primitive sets, the subsequential developments are based on triangulations.

Two methods were proposed in Kuhn [97], one of which is an artificial start method and the other a variable dimension method. The artificial start method starts at an artificial simplex outside the unit simplex and generates a sequence of adjacent full-dimensional simplices, and the variable dimension method starts at one of the vertices of the unit simplex and generates a sequence of adjacent simplices of varying dimension. Both of Kuhn's methods terminate within a finite of iterations with

a simplex that yields an approximate fixed point. The accuracy of approximation is completely determined by grid size of the triangulation underlying a simplicial method. When the accuracy is not good enough, one could restart the method with a triangulation of smaller grid size. However, Scarf's method and Kuhn's method both have a deficiency in that they discard all the relevant information about the location of a fixed point obtained in the previous implementation.

To deal with this deficiency, a homotopy method with continuous refinement of grid size and a restart homotopy method were developed in Eaves [33] and Merrill [114], respectively. Both Eaves's and Merrill's homotopy methods can start anywhere, but they need to introduce an extra dimension. A variable dimension method without an extra dimension was invented in van der Laan and Talman [176], which directly works on S^n and can also start anywhere. After this result, several variable dimension methods were proposed in the literature such as Kojima and Yamamoto [94, 95], Wright [192], and Yamamoto [193].

It is well known that integer programming is equivalent to determining whether there is an integer point in a polytope. To compute an integer point in P , a variable dimension simplicial method and a homotopy-like simplicial method were developed in Dang [17], Dang and Maaren [18–20], and Dang and Ye [21]. These simplicial methods can be considered as an extension of Eaves's and Merrill's homotopy methods and van der Laan and Talman's $(n + 1)$ -ray variable dimension method to integer programming. However, there is a significant difference between simplicial methods for integer programming and simplicial methods for approximating fixed points. A simplicial method for approximating fixed points terminates as soon as a complete full-dimensional simplex is met, whereas a simplicial method for integer programming terminates only when it either yields an integer point in a polytope or proves no such point exists. Therefore, a simplicial method for integer programming has to go one step further, which makes it significantly different from simplicial methods for approximating fixed points.

The two most important components of simplicial methods are labeling rules and triangulations. Efficiency of simplicial methods depends critically on the underlying triangulations. The three simplest triangulations of R^n are the K_1 -triangulation in Freudenthal [52], the J_1 -triangulation in Todd [159], and the D_1 -triangulation in Dang [13]. Measures of efficiency for triangulations were defined in Todd [157], which include the number of simplices in a unit cube, the diameter of a triangulation, and the average directional density. According to these measures, the D_1 -triangulation is superior to both the K_1 -triangulation and the J_1 -triangulation.

The rest of this chapter is organized as follows: Several simple triangulations are given in Sect. 2. Scarf's simplicial method is presented in Sect. 3. Kuhn's simplicial methods are discussed in Sect. 4. Eaves's simplicial homotopy method is given in Sect. 5. van der Laan and Talman's variable dimension simplicial method is introduced in Sect. 6. A variable dimension simplicial method for integer programming is described in Sect. 7. A homotopy-like simplicial method for integer programming is presented in Sect. 8. The chapter is concluded in Sect. 9.

2 Several Simple Triangulations

Triangulations play an important role in the developments of simplicial methods. Given $k + 1$ vectors in R^n , y^0, y^1, \dots, y^k , they are affinely independent if

$$\sum_{i=0}^k \alpha_i y^i = 0 \text{ and } \sum_{i=0}^k \alpha_i = 0$$

imply that $\alpha_i = 0$ for all i . The convex hull of $k + 1$ affinely independent vectors in R^n , y^0, y^1, \dots, y^k , is called a k -dimensional simplex or a k -simplex. The vectors, y^0, y^1, \dots, y^k , are called vertices of the simplex. Let σ denote a k -simplex. A face of σ is a simplex that is the convex hull of some vertices of σ . A facet of σ is a $(k - 1)$ -dimensional facet of σ .

Definition 1 Let C be a convex subset of R^n with $\dim(C) = m$. G is a triangulation or simplicial subdivision of C if:

1. G is a collection of m -dimensional simplices.
2. C is equal to the union of all the simplices in G .
3. For any σ^1 and σ^2 in G , $\sigma^1 \cap \sigma^2$ is either empty or a common facet of both of them.
4. Every $x \in C$ has a neighborhood that meets only a finite number of simplices in G .

Many triangulations have been constructed in the literature, but only several simple triangulations are given in this section.

2.1 The $K_2(m)$ -Triangulation of S^n

Let $S^n = \{x \in R_+^{n+1} \mid \sum_{i=0}^n x_i = 1\}$, which is the n -dimensional unit simplex. Let Q be an $(n + 1) \times n$ matrix given by

$$Q = \begin{pmatrix} -1 & & & \\ 1 & -1 & & \\ & 1 & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{pmatrix},$$

and q^i denote the i th column of Q for $i = 1, 2, \dots, n$. Let $N = \{1, 2, \dots, n\}$ and $\Pi(N) = \{\pi \mid \pi = (\pi(1), \pi(2), \dots, \pi(n))\}$ is a permutation of elements of N . For any given positive integer m , let

$$K_2^0(m) = \{y \in S^n \mid my_i \text{ is an integer for } i = 0, 1, \dots, n\}.$$

Table 1 Pivot rules of the $K_2(m)$ -triangulation of S^n

i	\bar{y}	$\bar{\pi}$
0	$y + \frac{1}{m}q^{\pi(1)}$	$(\pi(2), \dots, \pi(n), \pi(1))$
$1 \leq i < n$	y	$(\pi(1), \dots, \pi(i+1), \pi(i), \dots, \pi(n))$
n	$y - \frac{1}{m}q^{\pi(n)}$	$(\pi(n), \pi(1), \dots, \pi(n-1))$

Table 2 Pivot rules of the K_1 -triangulation of R^n

i	\bar{y}	$\bar{\pi}$
0	$y + u^{\pi(1)}$	$(\pi(2), \dots, \pi(n), \pi(1))$
$1 \leq i < n$	y	$(\pi(1), \dots, \pi(i+1), \pi(i), \dots, \pi(n))$
n	$y - u^{\pi(n)}$	$(\pi(n), \pi(1), \dots, \pi(n-1))$

Given any $y \in K_2^0(m)$ and $\pi \in \Pi(N)$, let $y^0 = y$ and

$$y^k = y^{k-1} + \frac{1}{m}q^{\pi(k)}, \quad k = 1, 2, \dots, n,$$

and $\sigma = \langle y^0, y^1, \dots, y^n \rangle$ be the convex hull of y^0, y^1, \dots, y^n . If $\sigma \subseteq S^n$, it is written that $\sigma = K_2(y, \pi)$. Let $K_2(m)$ be the collection of all such $\sigma = K_2(y, \pi)$. Then, $K_2(m)$ forms a triangulation of S^n [52].

Two simplices of $K_2(m)$ are adjacent if they share a common facet. For any given simplex $\sigma = K_2(y, \pi)$ with vertices y^0, y^1, \dots, y^n , its adjacent simplex opposite to a vertex, say y^i , is given by $K_2(\bar{y}, \bar{\pi})$, where \bar{y} and $\bar{\pi}$ are generated according to the pivot rules in [Table 1](#).

2.2 The K_1 -Triangulation of R^n

For $j \in N$, let u^j denote the j th unit vector of R^n . A simplex of the K_1 -triangulation is the convex hull of $n+1$ vectors, y^0, y^1, \dots, y^n , given by $y^0 = y$ and $y^k = y^{k-1} + u^{\pi(k)}$, $k = 1, 2, \dots, n$, where y is an integer point in R^n and $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ is a permutation of elements of N . Let K_1 be the set of all such simplices. Then, K_1 forms a triangulation of R^n [52]. Since a simplex of the K_1 -triangulation is uniquely determined by y and π , it is denoted by $K_1(y, \pi)$.

Two simplices of K_1 are adjacent if they share a common facet. For any given simplex $\sigma = K_1(y, \pi)$ with vertices y^0, y^1, \dots, y^n , its adjacent simplex opposite to a vertex, say y^i , is given by $K_1(\bar{y}, \bar{\pi})$, where \bar{y} and $\bar{\pi}$ are generated according to the pivot rules in [Table 2](#).

2.3 The D_1 -Triangulation of R^n

A simplex of the D_1 -triangulation is the convex hull of $n+1$ vectors, y^0, y^1, \dots, y^n , given by: If $p = 0$, $y^0 = y$ and $y^k = y + s_{\pi(k)}u^{\pi(k)}$, $k = 1, 2, \dots, n$, and if $p > 0$, $y^0 = y + s$ and

Table 3 Pivot rules of the D_1 -triangulation of R^n

i		\bar{y}	\bar{s}	$\bar{\pi}$	\bar{p}
0	$n = 1$	$y + 2s_{\pi(1)}u^{\pi(1)}$	$s - 2s_{\pi(1)}u^{\pi(1)}$	π	p
	$n \geq 2$	y	s	π	$1-p$
	$2 \leq p \leq n-1$	y	$s - 2s_{\pi(1)}u^{\pi(1)}$	π	p
$1 \leq i$	$p = 0$	y	$s - 2s_{\pi(i)}u^{\pi(i)}$	π	p
	$i < p-1$	y	s	π^1	p
	$i = p-1$	y	s	π	$p-1$
	$p-1 < i$	$1 \leq p < n-1$	y	π^2	$p+1$
	$i = n-1$	$1 \leq p = n-1$	$y + 2s_{\pi(n)}u^{\pi(n)}$	$s - 2s_{\pi(n)}u^{\pi(n)}$	π
	$i = n$	$1 \leq p = n-1$	$y + 2s_{\pi(n-1)}u^{\pi(n-1)}$	$s - 2s_{\pi(n-1)}u^{\pi(n-1)}$	π

$$\pi^1 = (\pi(1), \dots, \pi(i+1), \pi(i), \dots, \pi(n))$$

$$\pi^2 = (\pi(1), \dots, \pi(p-1), \pi(p), \pi(p), \dots, \pi(i-1), \pi(i+1), \dots, \pi(n))$$

$$y^k = \begin{cases} y^{k-1} - s_{\pi(k)}u^{\pi(k)} & \text{if } 1 \leq k \leq p-1, \\ y + s_{\pi(k)}u^{\pi(k)} & \text{otherwise,} \end{cases}$$

$k = 1, 2, \dots, n$, where $y = (y_1, y_2, \dots, y_n)^\top$ with each component being an even integer, $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ is a permutation of elements of N , $s = (s_1, s_2, \dots, s_n)^\top$ is a sign vector with each component being either -1 or 1 , and p is an integer with $0 \leq p \leq n-1$. Let D_1 be the set of all such simplices. Then, D_1 forms a triangulation of R^n [13]. Since a simplex of the D_1 -triangulation is determined by y, π, s , and p , it is denoted by $D_1(y, \pi, s, p)$.

Two simplices of D_1 are adjacent if they share a common facet. For any given simplex $\sigma = D_1(y, \pi, s, p)$ with vertices y^0, y^1, \dots, y^n , its adjacent simplex opposite to a vertex, say y^i , is given by $D_1(\bar{y}, \bar{\pi}, \bar{s}, \bar{p})$, where $\bar{y}, \bar{\pi}, \bar{s}$, and \bar{p} are generated according to the pivot rules in Table 3.

3 Scarf's Simplicial Method for Approximating Fixed Points on the Unit Simplex

Let $N_0 = \{0, 1, \dots, n\}$. Let f be a continuous mapping from S^n to itself. Let

$$\tilde{S}^n = \{x \in R^{n+1} \mid e^\top x = 1 \text{ and } x \leq 2e\},$$

where $e = (1, 1, \dots, 1)^\top \in R^{n+1}$. Then, $\tilde{v}^i = 2e - (2n+1)v^i$ is a vertex of \tilde{S}^n with v^i being the i th unit vector of R^{n+1} for $i = 0, 1, \dots, n$.

Definition 2 (An Integer Labeling Rule) For $i = 0, 1, \dots, n$,

$$l(\tilde{v}^i) = \mod(i - 1, n + 1),$$

and for $y \in S^n$,

$$l(y) = \min\{j \mid f_j(y) \leq y_j > 0\}.$$

With this integer labeling rule, it is ready to introduce the following definitions on completeness and almost completeness.

Definition 3 (Completeness and Almost Completeness)

- A q -dimensional simplex $\sigma = \langle y^0, y^1, \dots, y^q \rangle$ of $\mathcal{K}_1(m)$ is complete if $l(y^i) \neq l(y^j)$ for any $i \neq j$.
- A q -dimensional simplex $\sigma = \langle y^0, y^1, \dots, y^q \rangle$ of $\mathcal{K}_1(m)$ is almost complete if σ carries exactly q different labels.

As a direct result of this definition, one can easily obtain

Lemma 1 $\sigma^0 = \langle \tilde{v}^1, \tilde{v}^2, \dots, \tilde{v}^n, v^0 \rangle$ is an almost complete simplex.

With these notations and results, Scarf's method can be described as follows.

Initialization: Let $k = 0$ and $y^- = \tilde{v}^1$. Go to Step 1.

Step 1: Let τ^k be the facet of σ^k opposite to y^- and σ^{k+1} the unique n -dimensional simplex that shares τ^k with σ^k . Let y^+ be the vertex of σ^{k+1} opposite to τ^k and $k = k + 1$. Go to Step 2.

Step 2: If $l(y^+) = n$, the method terminates and an approximate fixed point has been found. Otherwise, let y^- be the vertex of σ^k that is different from y^+ and carries the same label as y^+ . Go to Step 1.

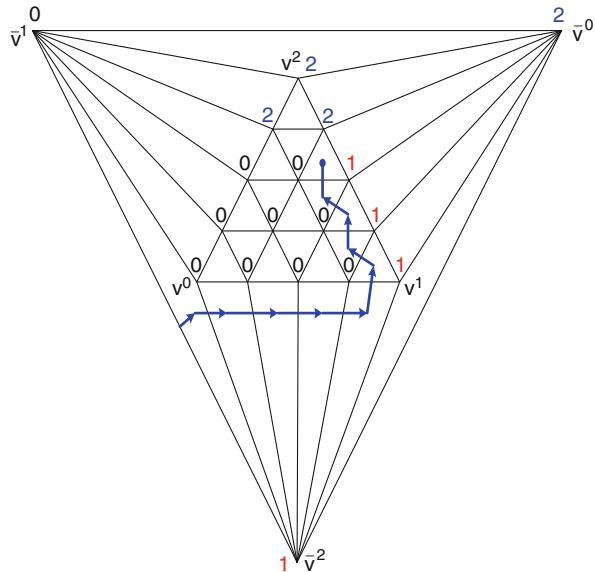
Theorem 1 (Scarf 1967) Within a finite number of iterations, the method terminates with a complete n -dimensional simplex that yields an approximate fixed point.

Example 1 (Todd 1979) Consider

$$f(x) = \begin{pmatrix} 0.2 & 0.1 & 0.3 \\ 0.3 & 0.4 & 0.3 \\ 0.5 & 0.5 & 0.4 \end{pmatrix} x.$$

f has just one fixed point, which is $x^* = (7, 11, 15)^\top / 33$. An illustration of how Scarf's method works can be found in [Fig. 1](#).

Fig. 1 An illustration of Scarf's simplicial method



4 Kuhn's Simplicial Methods for Approximating Fixed Points on the Unit Simplex

After Scarf's work, two simplicial methods were proposed in Kuhn [97]. One is an artificial start simplicial method and the other is a variable dimension simplicial method. They are presented in the following two subsections, respectively.

4.1 Kuhn's Artificial Start Simplicial Method

Let

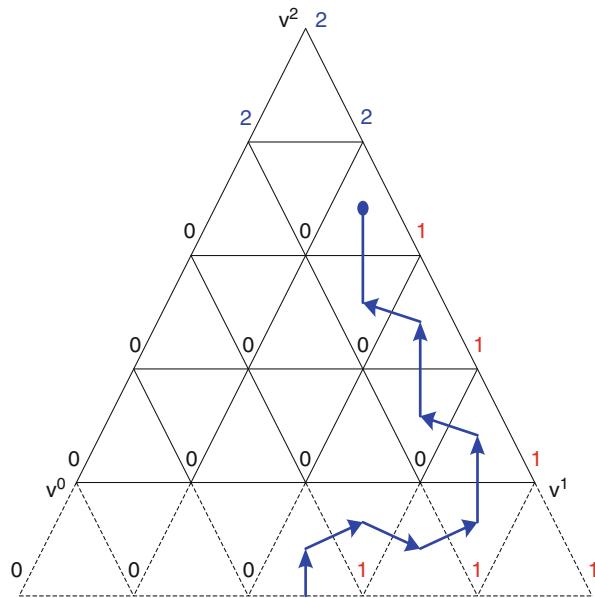
$$\tilde{S}^n = \left\{ x \in R^{n+1} \mid e^\top x = 1, x_i \geq 0, i = 0, 1, \dots, n-1, x_n \geq -\frac{1}{m} \right\}$$

and $\tilde{S}_n^n = \{x \in \tilde{S}^n \mid x_n = -\frac{1}{m}\}$.

Definition 4 (An Integer Labeling Rule) For $y \in \tilde{S}^n$,

$$l(y) = \begin{cases} \min\{j \mid f_j(y) \leq y_j > 0\} & \text{if } y \in S^n, \\ \min\{j \mid y_j = \max_{k \in N_0} y_k\} & \text{otherwise.} \end{cases}$$

Fig. 2 An illustration of Kuhn's artificial start simplicial method



With this labeling rule and [Definition 3](#), one can readily derive the following result.

Lemma 2 ([97]) *For any given triangulation of \tilde{S}^n , there is a unique complete $(n - 1)$ -dimensional simplex in \tilde{S}^n .*

With these notations and results, Kuhn's artificial start simplicial method can be described as follows.

Initialization: Let τ^0 be the unique complete $(n - 1)$ -dimensional simplex in \tilde{S}^n , σ^0 the unique n -dimensional simplex with τ^0 as a facet, y^+ the vertex of σ^0 opposite to τ^0 , and $k = 0$. Go to *Step 1*.

Step 1: If $l(y^+) = n$, the method terminates and an approximate fixed point has been found. Otherwise, let y^- be the vertex of σ^k that is different from y^+ and carries the same label as y^+ . Go to *Step 2*.

Step 2: Let τ^k be the facet of σ^k opposite to y^- and σ^{k+1} the unique n -dimensional simplex that shares τ^k with σ^k . Let y^+ be the vertex of σ^{k+1} opposite to τ^k and $k = k + 1$. Go to *Step 1*.

Theorem 2 ([97]) *Within a finite number of iterations, the artificial start simplicial method terminates with a complete n -dimensional simplex that yields an approximate fixed point.*

Example 2 Consider f the same as that in [Example 1](#). An illustration of how Kuhn's artificial start simplicial method works can be found in [Fig. 2](#).

4.2 Kuhn's Variable Dimension Simplicial Method

To avoid the introduction of an extra layer in the artificial start simplicial method, a variable dimension simplicial method was developed in Kuhn [97], which works directly on S^n and can start at any given vertex of S^n . For any given vertex v^p of S^n and $K \subset N_0$ with $p \notin K$, let

$$S^n(v^p, K) = \{x \in S^n \mid x = v^p + \sum_{j \in K} \lambda_j (v^j - v^p), \lambda_j \geq 0, j \in K\}.$$

Initialization: Let $y^+ = v^p$, $h = (h(1), h(2), \dots, h(n))$ be a given permutation of elements of $N_0 \setminus \{p\}$, $K = \emptyset$, $L = \emptyset$, and $\sigma^0 = \langle y^+ \rangle$, $q = 0$, and $k = 0$. Go to *Step 1*.

- Step 1: If $l(y^+) \notin L$, let $L = L \cup \{l(y^+)\}$ and $q = q + 1$, and go to *Step 2*. Otherwise, let y^- be the vertex of σ^k that is different from y^+ and carries the same label as y^+ and τ^{k+1} the facet of σ^k opposite to y^- , and go to *Step 3*.
- Step 2: If $q = n + 1$, the method terminates and an approximate fixed point has been found. Otherwise, let $K = K \cup \{h(q)\}$, $\tau^{k+1} = \sigma^k$, σ^{k+1} be the unique q -dimensional simplex in $S^n(v^p, K)$ with τ^{k+1} as a facet, y^+ the vertex of σ^{k+1} opposite to τ^{k+1} , and $k = k + 1$, and go to *Step 1*.
- Step 3: If $\tau^{k+1} \subseteq S^n(v^p, K \setminus \{h(q)\})$, let $K = K \setminus \{h(q)\}$ and go to *Step 4*. Otherwise, let σ^{k+1} be the unique q -dimensional simplex in $S^n(v^p, K)$ sharing with σ^k the facet τ^{k+1} , y^+ be the vertex of σ^{k+1} opposite to τ^{k+1} , and $k = k + 1$, and go to *Step 1*.

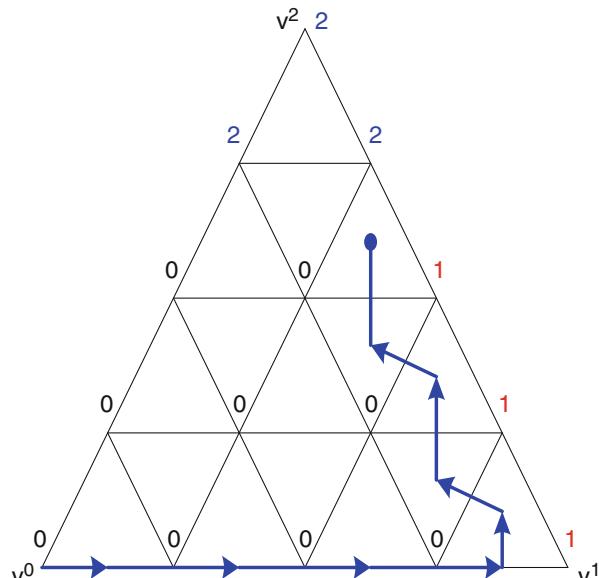


Fig. 3 An illustration of Kuhn's variable dimension simplicial method

Step 4: Let $q = q - 1$, $L = L \setminus \{h(q)\}$, $\sigma^{k+1} = \tau^{k+1}$, y^- be the vertex of σ^{k+1} with label $h(q)$, τ^{k+2} the facet of σ^{k+1} opposite to y^- , and $k = k + 1$, and go to Step 3.

Theorem 3 ([97]) Within a finite number of iterations, the variable dimension simplicial method terminates with a complete n -dimensional simplex that yields an approximate fixed point.

Example 3 Consider f the same as that in [Example 1](#). An illustration of how Kuhn's variable dimension simplicial method works can be found in [Fig. 3](#).

5 Eaves's Simplicial Homotopy Method for Approximating Fixed Points on the Unit Simplex

Scarf's and Kuhn's simplicial methods all have a deficiency that they discard all the relevant information about the location of a fixed point obtained in the previous implementation when restarting. To address this issue, a simplicial homotopy method with continuous refinement and a restart simplicial homotopy method were proposed by introducing an extra dimension in Eaves [33] and Merrill [114], respectively. Let x^0 be any given interior point of S^n . Let T be a triangulation of $S^n \times [0, 1]$ with continuous refinement of grid size.

Definition 5 (A Vector Labeling Rule) For any given point $(y, t) \in S^n \times [0, 1]$,

$$l(y, t) = \begin{cases} y - f(y) & \text{if } t > 0, \\ y - x^0 & \text{otherwise.} \end{cases}$$

With this vector labeling rule, a complete simplex can be defined as follows.

Definition 6 A simplex $\sigma = \langle y^0, y^1, \dots, y^k \rangle$ is complete if

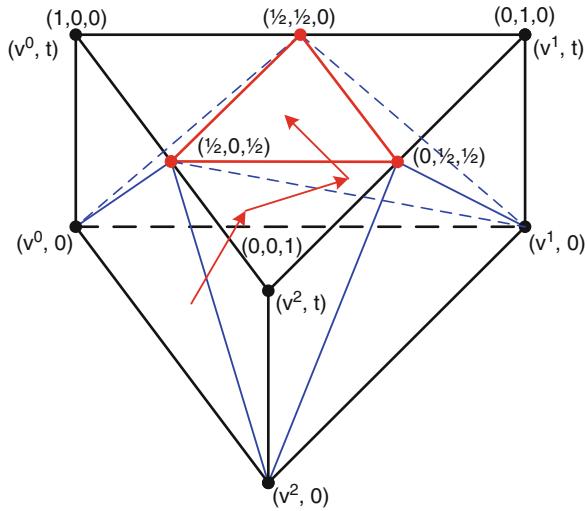
$$\sum_{i=0}^k \lambda_i \binom{1}{l(y^i)} = \binom{1}{0} \quad (1)$$

has a nonnegative solution $\lambda = (\lambda_0, \lambda_1, \dots, \lambda_k)^\top$.

Lemma 3 If $(x^0, 0)$ is an interior point of an n -simplex $\tau^0 \subseteq S^n \times \{0\}$, then τ^0 is a unique complete n -dimensional simplex in $S^n \times \{0\}$.

With these notations and results, Eaves's simplicial homotopy method of continuous refinement can be described as follows.

Fig. 4 An illustration of Eaves's simplicial homotopy method



Initialization: Let σ^0 be the unique $(n + 1)$ -dimensional simplex in $S^n \times [0, 1]$ with τ_0 as a facet, y^+ be the vertex of σ^0 opposite to τ_0 , and $k = 0$. Go to *Step 1*.

Step 1: Perform a pivoting step of linear programming on the system (Eq. 1) to determine the vertex y^- of σ^k that is replaced by y^+ . Let τ^{k+1} be the facet of σ^k opposite to y^- . If $\tau^{k+1} \subseteq S^n \times \{t\}$ with $t > 0$ yields a satisfactory approximate fixed point, the method terminates. Otherwise, go to *Step 2*.

Step 2: Let σ^{k+1} be the unique $(n + 1)$ -dimensional simplex that shares τ^{k+1} with σ^k , y^+ the vertex of σ^{k+1} opposite to τ^{k+1} , and $k = k + 1$. Go to *Step 1*.

Theorem 4 ([33]) *Under the nondegeneracy condition, Eaves's simplicial homotopy method terminates within a finite number of iterations with a satisfactory approximate fixed point.*

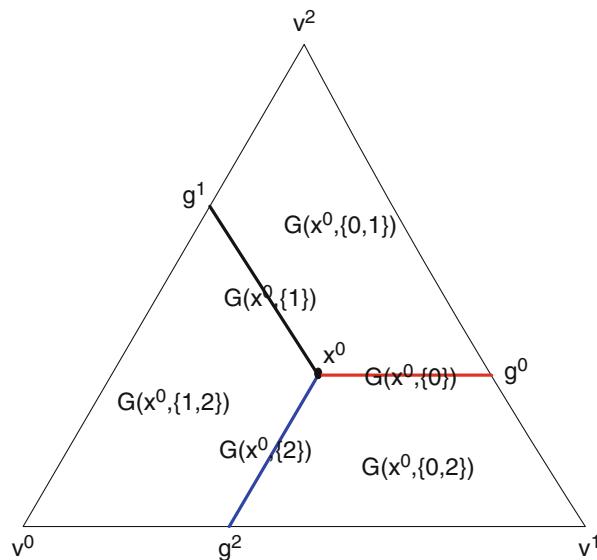
Example 4 Consider f the same as that in [Example 1](#). An illustration of how Eaves's simplicial homotopy method works can be found in [Fig. 4](#).

Merrill's simplicial homotopy method is the same as Eaves's but without the continuous refinement.

6 van der Laan and Talman's Variable Dimension Simplicial Method for Approximating Fixed Points on the Unit Simplex

Eaves's and Merrill's simplicial homotopy methods can start anywhere and are more efficient than Scarf's and Kuhn's simplicial methods, but they need to introduce an extra dimension. To get rid of this extra dimension, a variable dimension simplicial

Fig. 5 An illustration of $G(x^0, K)$



method was developed in van der Laan and Talman [176] to approximate fixed points on the unit simplex, which works directly on S^n and can also start anywhere.

Definition 7 (An Integer Labeling Rule) For $y \in S^n$,

$$l(y) = \min\{j \mid f_j(y) \leq y_j > 0\}.$$

Let Q_0 be an $(n + 1) \times (n + 1)$ matrix given by

$$Q_0 = \begin{pmatrix} -1 & & & & 1 \\ 1 & -1 & & & \\ & 1 & \ddots & & \\ & & \ddots & -1 & \\ & & & 1 & -1 \end{pmatrix}.$$

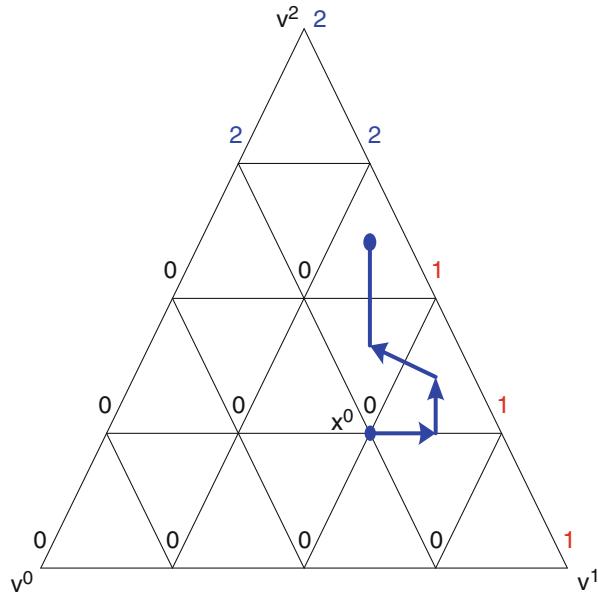
For $i \in N_0$, let g^i denote the $(i + 1)$ th column of Q_0 . Given any $x^0 \in S^n$ and $K \subset N_0$, let

$$G(x^0, K) = \{x \in S^n \mid x = x^0 + \sum_{i \in K} \lambda_i g^i, \lambda_i \geq 0, i \in K\}.$$

An illustration of $G(x^0, K)$ can be found in Fig. 5.

With these notations and results, van der Laan and Talman's variable dimension simplicial method can be described as follows.

Fig. 6 An illustration of van der Laan and Talman's variable dimension simplicial method



Initialization: Let $y^+ = x^0$, $K = \emptyset$, $\sigma^0 = \langle y^+ \rangle$, and $k = 0$. Go to *Step 1*.

- Step 1: If $l(y^+) \notin K$, let $K = K \cup \{l(y^+)\}$ and go to *Step 2*. Otherwise, let y^- be the vertex of σ^k that is different from y^+ and carries the same label as y^+ and τ^{k+1} the facet of σ^k opposite to y^- , and go to *Step 3*.
- Step 2: If $|K| = n + 1$, the method terminates and an approximate fixed point has been found. Otherwise, let $\tau^{k+1} = \sigma^k$, σ^{k+1} be the unique $|K|$ -dimensional simplex in $G(x^0, K)$ with τ^{k+1} as a facet, y^+ the vertex of σ^{k+1} opposite to τ^{k+1} , and $k = k + 1$, and go to *Step 1*.
- Step 3: If $\tau^{k+1} \subseteq G(x^0, K \setminus \{j\})$ for some $j \in K$, go to *Step 4*. Otherwise, let σ^{k+1} be the unique $|K|$ -dimensional simplex in $G(x^0, K)$ sharing with σ^k the facet τ^{k+1} , y^+ the vertex of σ^{k+1} opposite to τ^{k+1} , and $k = k + 1$, and go to *Step 1*.
- Step 4: Let $\sigma^{k+1} = \tau^{k+1}$, y^- be the vertex of σ^{k+1} with label j , τ^{k+2} the facet of σ^{k+1} opposite to y^- , and $k = k + 1$, and go to *Step 3*.

Theorem 5 [176] Within a finite number of iterations, the variable dimension simplicial method terminates with a complete n -dimensional simplex that yields an approximate fixed point.

Example 5 Consider f the same as that in [Example 1](#). An illustration of how van der Laan and Talman's variable dimension simplicial method works can be found in [Fig. 6](#).

7 A Variable Dimension Simplicial Method for Integer Programming

It is well known that integer programming is equivalent to determining whether there is an integer point in a polytope. To compute an integer point in a polytope, simplicial methods have been developed in Dang and Maaren [18–20], Dang [17] and Dang and Ye [21], which can be considered as extensions of simplicial methods for approximating fixed points. The problem is as follows: Determine whether there is an integer point in a polytope given by $P = \{x \in R^n \mid Ax \leq b\}$, where

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

is a rational matrix satisfying that each row of A has at most one positive entry and $b = (b_1, b_2, \dots, b_m)^\top$ is a rational vector. It has been shown in Lagarias [103] that

Theorem 6 *Determining whether there is an integer point in P is an NP-complete problem.*

Let $M = \{1, 2, \dots, m\}$, $N = \{1, 2, \dots, n\}$, and $N_0 = \{1, 2, \dots, n+1\}$. For $i \in M$, let a_i^\top denote the i th row of A . Then, $A = (a_1, a_2, \dots, a_m)^\top$. Without loss of generality, assume that P is bounded and full dimensional. As a result of the property of A , it follows that

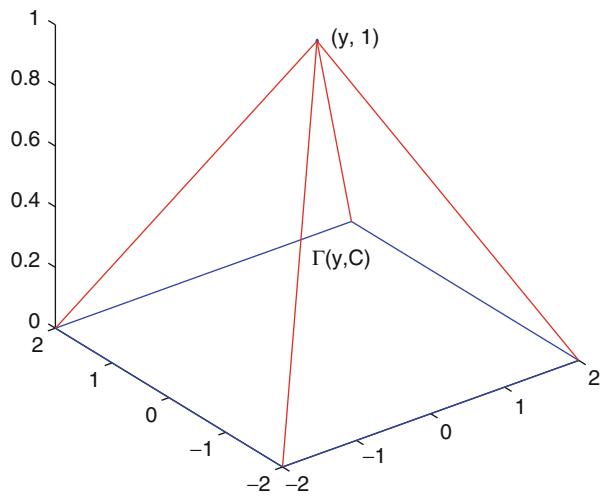
Lemma 4 *If $x^1 = (x_1^1, x_2^1, \dots, x_n^1)^\top \in P$ and $x^2 = (x_1^2, x_2^2, \dots, x_n^2)^\top \in P$, then $\bar{x} = \max(x^1, x^2) = (\max\{x_1^1, x_1^2\}, \max\{x_2^1, x_2^2\}, \dots, \max\{x_n^1, x_n^2\})^\top \in P$.*

Let $e = (1, 1, \dots, 1)^\top \in R^n$. Lemma 4 implies that $\max_{x \in P} e^\top x$ has a unique solution, which is denoted by $x^{\max} = (x_1^{\max}, x_2^{\max}, \dots, x_n^{\max})^\top$. Let $x^{\min} = (x_1^{\min}, x_2^{\min}, \dots, x_n^{\min})^\top$ with $x_j^{\min} = \min_{x \in P} x_j$, $j = 1, 2, \dots, n$. Clearly, $x^{\min} \leq x \leq x^{\max}$ for all $x \in P$. For any real number α , let $\lfloor \alpha \rfloor$ denote the greatest integer less than or equal to α and $\lceil \alpha \rceil$ the smallest integer greater than or equal to α . Let

$$D(P) = \{x \in R^n \mid x^l \leq x \leq x^u\},$$

where $x^u = \lfloor x^{\max} \rfloor = (\lfloor x_1^{\max} \rfloor, \lfloor x_2^{\max} \rfloor, \dots, \lfloor x_n^{\max} \rfloor)^\top$ and $x^l = \lceil x^{\min} \rceil = (\lceil x_1^{\min} \rceil, \lceil x_2^{\min} \rceil, \dots, \lceil x_n^{\min} \rceil)^\top$. Thus, $D(P)$ contains all integer points in P . Without loss of generality, assume that $x^l \leq x^u$ since otherwise there is no integer point in P .

Fig. 7 An illustration of $\Gamma(y, C)$



For $x \in R^n$, let

$$f(x) = \begin{cases} 0 \in R^n & \text{if } x \in P, \\ \sum_{i \in I(x)} \frac{a_i^\top x - b_i}{a_i^\top a_i} a_i & \text{otherwise,} \end{cases}$$

where $I(x) = \{i \in M \mid a_i^\top x - b_i > 0\}$.

Let $x^0 = (x_1^0, x_2^0, \dots, x_n^0)^\top$ be any given integer point of $D(P)$. For $y \in R^n$ and $C \subseteq R^n$, let $\Gamma(y, C)$ be an augmented set in R^{n+1} given by

$$\Gamma(y, C) = \{t(x, 0) + (1-t)(y, 1) \mid x \in C \text{ and } 0 \leq t \leq 1\}.$$

An illustration of $\Gamma(y, C)$ can be found in [Fig. 7](#).

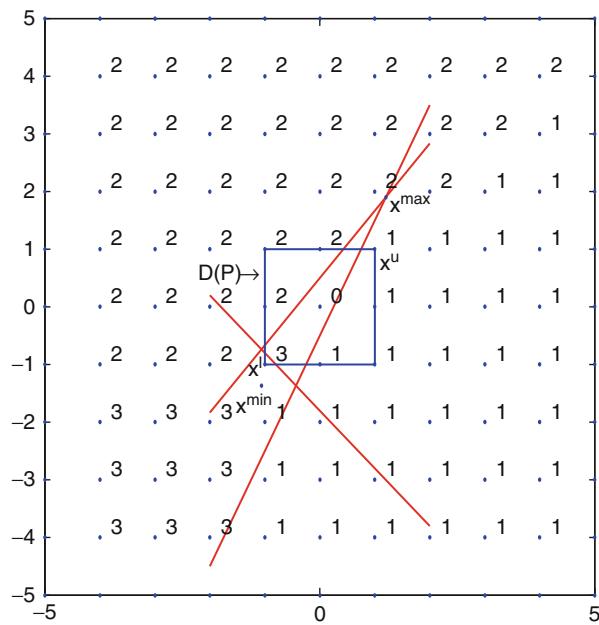
Definition 8 (An Integer Labeling Rule) For each integer point $(x, \gamma) \in \Gamma(x^0, R^n)$, assign to (x, γ) an integer label $l(x, \gamma) \in N_0 \cup \{0\}$ as follows:

1. $l(x^0, 1) = n + 1$.
2. For $(x, 0)$ with $x \in D(P)$,

$$l(x, 0) = \begin{cases} 0 & \text{if } x \in P, \\ \max\{k \mid f_k(x) = \max_{j \in N} f_j(x)\} & \text{if } f_j(x) > 0 \text{ for some } j \in N, \\ n + 1 & \text{if } f(x) \leq 0 \text{ and } f(x) \neq 0. \end{cases}$$

3. For $(x, 0)$ with $x_j > x_j^u$ for some $j \in N$,

Fig. 8 An illustration of the labeling rule on $R^2 \times \{0\}$ for Example 6



$$l(x, 0) = \max\{k \mid x_k - x_k^u = \max_{j \in N} x_j - x_j^u\}.$$

4. For $(x, 0)$ with $x \leq x^u$ and $x_j < x_j^l$ for some $j \in N$,

$$l(x, 0) = \begin{cases} n+1 & \text{if } x < x^l, \\ \max\{k \mid x_k - x_k^l = \max_{j \in N} x_j - x_j^l\} & \text{otherwise.} \end{cases}$$

Example 6 Consider

$$P = \{x = (x_1, x_2)^\top \mid 2x_1 - x_2 \leq \frac{1}{2}, -\frac{7}{6}x_1 + x_2 \leq \frac{1}{2}, -x_1 - x_2 \leq \frac{9}{5}\},$$

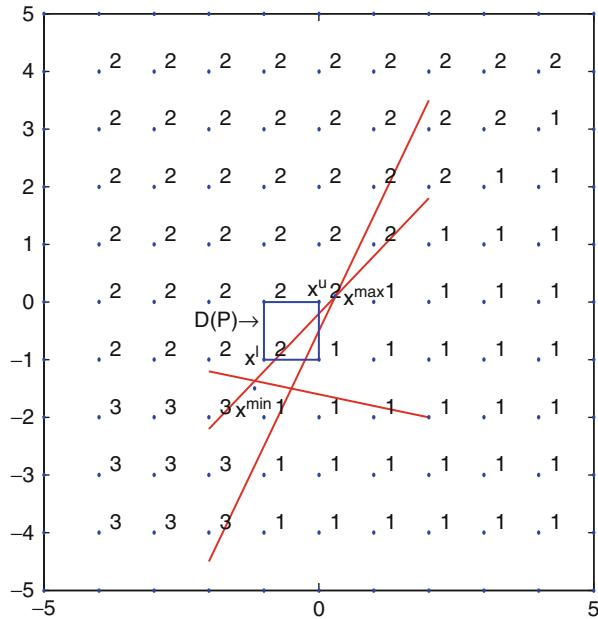
which has an integer point. **Figure 8** illustrates the labeling rule on $R^2 \times \{0\}$ for this polytope.

Example 7 Consider

$$P = \{x = (x_1, x_2)^\top \mid 2x_1 - x_2 \leq \frac{1}{2}, -x_1 + x_2 \leq -\frac{1}{5}, -\frac{1}{5}x_1 - x_2 \leq \frac{8}{5}\},$$

which has no integer point. **Figure 9** illustrates the labeling rule on $R^2 \times \{0\}$ for this polytope.

Fig. 9 An illustration of the labeling rule on $R^2 \times \{0\}$ for Example 7



Let $h(n+1) = (1, 1, \dots, 1, 0)^\top \in R^{n+1}$ and $h(j) = -u^j$, $j = 1, 2, \dots, n$. Let $G(x^0, \emptyset) = \{(x^0, 0)\}$ and $G(x^0, N_0) = \Gamma(x^0, R^n)$. For any $K \subset N_0$ with $K \neq \emptyset$, let

$$G(x^0, K) = \{(x^0, 0) + \sum_{j \in K} \lambda_j h(j) \mid 0 \leq \lambda_j, j \in K\}.$$

Clearly, $\cup_{j \in N_0} G(x^0, N_0 \setminus \{j\}) = R^n \times \{0\}$ and, for any two subsets $K^1 \subset N_0$ and $K^2 \subset N_0$, the intersection of $G(x^0, K^1)$ and $G(x^0, K^2)$, $G(x^0, K^1) \cap G(x^0, K^2)$, is a common facet of both of them. Thus, $\{G(x^0, K) \mid K \subset N_0\}$ forms a subdivision of $R^n \times \{0\}$. An illustration of $G(x^0, K)$ is given in Fig. 10.

For further developments, a cubic triangulation is needed, whose restriction on $G(x^0, K)$ is a triangulation of $G(x^0, K)$ for each $K \subseteq N_0$. For simplicity, the K_1 -triangulation in Freudenthal [52] is adopted, which is as follows.

For $j \in N_0$, let u^j denote the j th unit vector of R^{n+1} . A simplex of the K_1 -triangulation of $\Gamma(x^0, R^n)$ is the convex hull of $n+2$ vectors, y^0, y^1, \dots, y^{n+1} , given by $y^0 = y$, $y^k = y^{k-1} + u^{\pi(k)}$, $k = 1, 2, \dots, n$, and $y^{n+1} = (x^0, 1)$, where $y = (y_1, y_2, \dots, y_{n+1})^\top$ is an integer point in $R^n \times \{0\}$ and $\pi = (\pi(1), \pi(2), \dots, \pi(n), \pi(n+1))$ is a permutation of elements of N_0 with $\pi(n+1) = n+1$. Let K_1 be the set of all such simplices. Then, K_1 forms a triangulation of $\Gamma(x^0, R^n)$. Since a simplex of the K_1 -triangulation is uniquely determined by y and π , it is denoted by $K_1(y, \pi)$. An illustration of the K_1 -triangulation can be found in Fig. 11.

Fig. 10 An illustration of $G(x^0, K)$

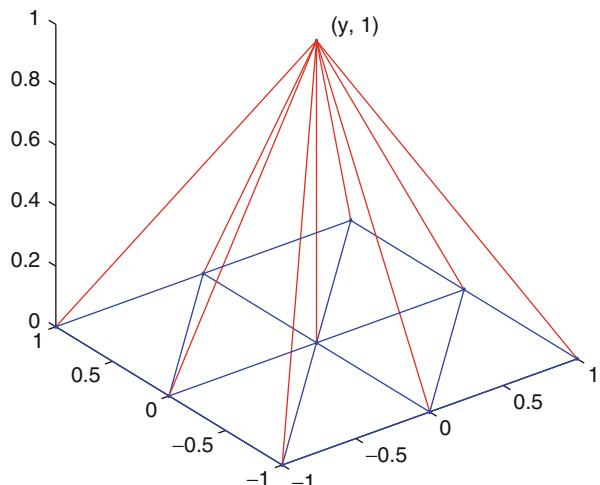
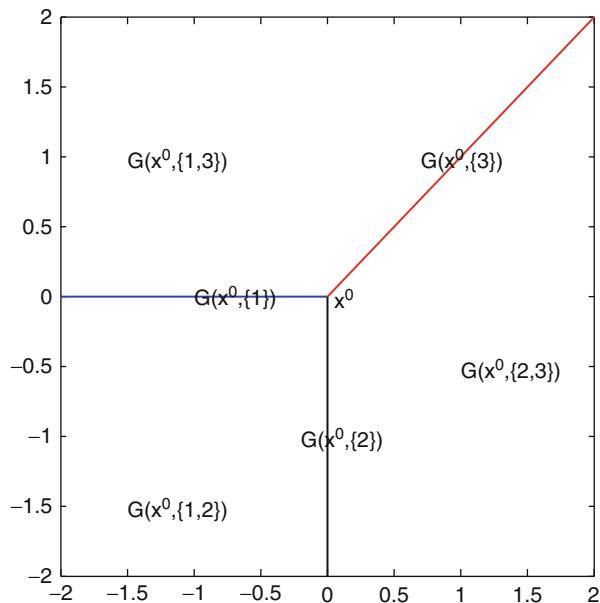


Fig. 11 An illustration of the K_1 -triangulation

Two simplices of K_1 are adjacent if they share a common facet. For any given simplex $\sigma = K_1(y, \pi)$ with vertices y^0, y^1, \dots, y^{n+1} , its adjacent simplex opposite to a vertex, say y^i , is given by $K_1(\bar{y}, \bar{\pi})$, where \bar{y} and $\bar{\pi}$ are generated according to the pivot rules in Table 4.

Let \mathcal{K}_1 be the set of faces of simplices of K_1 . A q -dimensional simplex of \mathcal{K}_1 with vertices y^0, y^1, \dots, y^q is denoted by $\langle y^0, y^1, \dots, y^q \rangle$. For $\sigma \in \mathcal{K}_1$ with $\sigma \subset R^n \times \{0\}$, let $\text{grid}(\sigma) = \max\{\|x-y\| \mid (x, 0) \in \sigma \text{ and } (y, 0) \in \sigma\}$, where $\|\cdot\|$ denotes

Table 4 Pivot rules of the K_1 -triangulation of $\Gamma(x^0, R^n)$

i	\bar{y}	$\bar{\pi}$
0	$y + u^{\pi(1)}$	$(\pi(2), \dots, \pi(n), \pi(1), \pi(n+1))$
$1 \leq i < n$	y	$(\pi(1), \dots, \pi(i+1), \pi(i), \dots, \pi(n+1))$
n	$y - u^{\pi(n)}$	$(\pi(n), \pi(1), \dots, \pi(n-1), \pi(n+1))$

the infinity norm. It is defined that $\text{mesh}(K_1) = \max\{\text{grid}(\sigma) \mid \sigma \in \mathcal{K}_1 \text{ and } \sigma \subset R^n \times \{0\}\}$. Then, $\text{mesh}(K_1) = 1$.

For $K \subset N_0$, the restriction of \mathcal{K}_1 on $G(x^0, K)$ is given by

$$\mathcal{K}_1|G(x^0, K) = \{\sigma \in \mathcal{K}_1 \mid \sigma \subset G(x^0, K) \text{ and } \dim(\sigma) = |K|\},$$

where $|\cdot|$ denotes the cardinality of a set and $\dim(\cdot)$ the dimension of a set. Obviously, $\mathcal{K}_1|G(x^0, K)$ is a triangulation of $G(x^0, K)$.

With the labeling rule and the K_1 -triangulation, a variable dimension simplicial method has been developed for computing an integer point in P , which is as follows.

Initialization: Let $K = \emptyset$, $y^0 = (x^0, 0)$, $\sigma_0 = \langle y^0 \rangle$, $y^+ = y^0$, and $k = 0$. Go to *Step 1*.

Step 1: If $l(y^+) = 0$, the method terminates, and an integer point of P has been found. If $K = N_0$ and $y^+ \geq (x^u, 0)$ with $y_{n+1}^+ = 0$, the method terminates, and P has no integer point. If $l(y^+) \in K$, let y^- be the vertex of σ_k other than y^+ and carrying integer label $l(y^+)$, and τ_{k+1} the facet of σ_k opposite to y^- , and go to *Step 2*. If $l(y^+) \notin K$, go to *Step 3*.

Step 2: If $\tau_{k+1} \subset G(x^0, K \setminus \{j\})$ for some $j \in K$, let $K = K \setminus \{j\}$ and go to *Step 4*. Otherwise, proceed as follows: Let σ_{k+1} be the unique simplex that is adjacent to σ_k and has τ_{k+1} as a facet. Let y^+ be the vertex of σ_{k+1} opposite to τ_{k+1} and $k = k + 1$. Go to *Step 1*.

Step 3: Let $K = K \cup \{l(y^+)\}$ and $\tau_{k+1} = \sigma_k$. Let σ_{k+1} be the unique $|K|$ -dimensional simplex in $G(x^0, K)$ having τ_{k+1} as a facet and y^+ the vertex of σ_{k+1} opposite to τ_{k+1} . Let $k = k + 1$, and go to *Step 1*.

Step 4: Let $\sigma_{k+1} = \tau_{k+1}$, y^- be the vertex of σ_{k+1} carrying integer label j , and τ_{k+2} the facet of σ_{k+1} opposite to y^- . Let $k = k + 1$, and go to *Step 2*.

Theorem 7 Within a finite number of iterations, the method either yields an integer point in P or proves no such point exists.

The following example illustrates how the method works when $n = 2$.

Example 8 Find an integer point in

$$P = \left\{ x = (x_1, x_2)^\top \mid -x_1 + x_2 \leq \frac{1}{2}, x_1 - x_2 \leq \frac{1}{2}, x_1 \leq \frac{4}{5}, -x_1 \leq \frac{4}{5} \right\}.$$

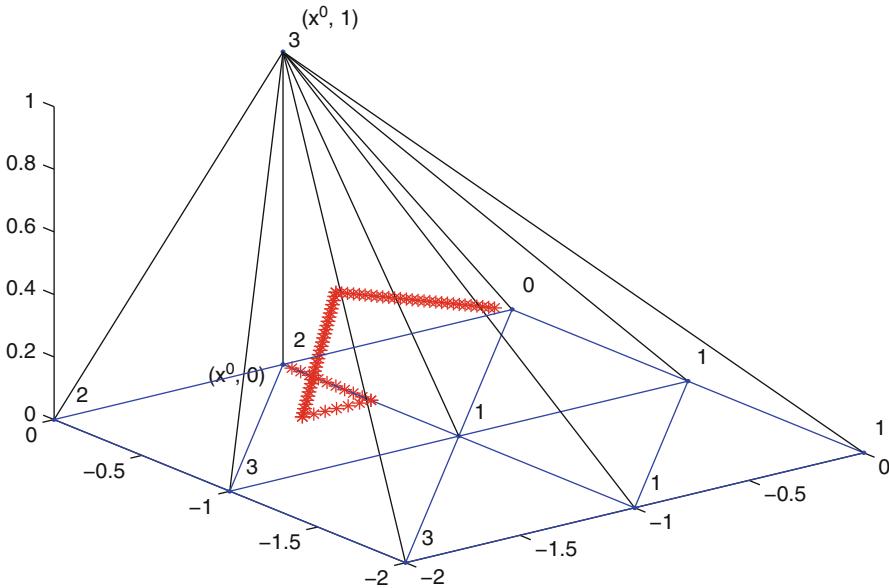


Fig. 12 An illustration of the method

Given this polytope, it is obtained that $x^u = (1, 0)^T$ and $x^l = (-1, 0)^T$. Let $x^0 = (-1, 0)^T$, $K = \emptyset$, $y^0 = (x^0, 0)$, $\sigma_0 = \langle y^0 \rangle$, and $y^+ = y^0$.

Iteration 1: $l(y^+) = 2 \notin K$. Let $K = K \cup \{2\} = \{2\}$, $\tau_1 = \sigma_0$, $y^1 = (-1, -1, 0)^T$, $\sigma_1 = \langle y^0, y^1 \rangle$, and $y^+ = y^1$.

Iteration 2: $l(y^+) = 1 \notin K$. Let $K = K \cup \{1\} = \{1, 2\}$, $\tau_2 = \sigma_1$, $y^2 = (-2, -1, 0)^T$, $\sigma_2 = \langle y^0, y^1, y^2 \rangle$, and $y^+ = y^2$.

Iteration 3: $l(y^+) = 3 \notin K$. Let $K = K \cup \{l(y^+)\} = \{1, 2, 3\} = N_0$, $\tau_3 = \sigma_2$, $y^3 = (-1, 0, 1)^T$, $\sigma_3 = \langle y^0, y^1, y^2, y^3 \rangle$, and $y^+ = y^3$.

Iteration 4: $l(y^+) = 3 = l(y^2) \in K$. Let $\tau_4 = \langle y^0, y^1, y^3 \rangle$, $y^2 = (0, 0, 0)^T$, $\sigma_4 = \langle y^0, y^1, y^2, y^3 \rangle$, and $y^+ = y^2$.

Iteration 5: $l(y^+) = 0$. An integer point of P has been found.

Figure 12 illustrates how the method moves along a simplicial path starting from $(x^0, 0)$ and ending at a feasible integer point in P .

In the remaining of this section, a proof to [Theorem 7](#) is given.

Lemma 5 *For any $x \in R^n$, $f(x) = 0$ if and only if $x \in P$.*

Proof It suffices to prove the “only if” part. Suppose that there is some $x \in R^n$ with $f(x) = 0$ and $x \notin P$. Then, $I(x) \neq \emptyset$. Thus, for any $y \in P$ (i.e., $b_i - a_i^T y \geq 0$ for all $i \in M$),

$$\begin{aligned}
0 &= (x - y)^\top f(x) = (x - y)^\top \sum_{i \in I(x)} \frac{a_i^\top x - b_i}{a_i^\top a_i} a_i = \sum_{i \in I(x)} \frac{a_i^\top x - b_i}{a_i^\top a_i} a_i^\top (x - y) \\
&= \sum_{i \in I(x)} \frac{a_i^\top x - b_i}{a_i^\top a_i} (a_i^\top x - b_i + b_i - a_i^\top y) \geq \sum_{i \in I(x)} \frac{a_i^\top x - b_i}{a_i^\top a_i} (a_i^\top x - b_i) \\
&= \sum_{i \in I(x)} \frac{(a_i^\top x - b_i)^2}{a_i^\top a_i} > 0.
\end{aligned}$$

Therefore, a contradiction occurs. This completes the proof. \square

Lemma 6 *If $f(x) \leq 0$ and $f(x) \neq 0$, then, for any $y \in P$, there is some $k \in N$ satisfying that $x_k - y_k < 0$.*

Proof Since $f(x) \neq 0$, hence, $I(x) \neq \emptyset$. Let y be an arbitrary point in P . Suppose that $x - y \geq 0$. Then, similar to that in the proof of [Lemma 5](#), one can obtain that $0 \geq (x - y)^\top f(x) > 0$. Thus, a contradiction occurs. The lemma follows immediately. \square

Definition 9

- A q -dimensional simplex $\sigma = \langle y^0, y^1, \dots, y^q \rangle$ of \mathcal{K}_1 is complete if $l(y^i) \neq l(y^j)$ for $0 \leq i < j \leq q$, and $l(y^k) \neq 0$, $k = 0, 1, \dots, q$.
- A q -dimensional simplex $\sigma = \langle y^0, y^1, \dots, y^q \rangle$ of \mathcal{K}_1 is 0-complete if $l(y^i) \neq l(y^j)$ for $0 \leq i < j \leq q$, and there is some k satisfying that $l(y^k) = 0$.
- A q -dimensional simplex $\sigma = \langle y^0, y^1, \dots, y^q \rangle$ of \mathcal{K}_1 is almost complete if labels of $q + 1$ vertices of σ consist of q different nonzero integers.

An illustration of this definition can be found in [Fig. 13](#). From [Definition 9](#), it is easy to see that an almost complete simplex has exactly two complete facets both carrying the same set of integer labels.

For $y \in R^n$ and $K \subseteq N$, let $H(y, K)$ be the “higher”-level cone originated at y along certain directions given in K , that is,

$$H(y, K) = \{y + h \in R^n \mid 0 \leq h_j, j \in K, \text{ and } h_j = 0, j \notin K\}.$$

Lemma 7 *If P has an integer point, then, for any integer point $z^0 \in P$ and any nonempty $K \subseteq N$, each integer point of $H(z^0, K) \times \{0\}$ carries either integer label 0 or an integer label in K .*

Proof Let $(x, 0)$ be an integer point of $H(z^0, K) \times \{0\}$. Consider that $x \in D(P)$. From [Lemma 6](#), it is known that $l(x, 0) \neq n + 1$ since $x \geq z^0$, which is equivalent to that $f(x) = 0$ or $f_j(x) > 0$ for some $j \in N$.

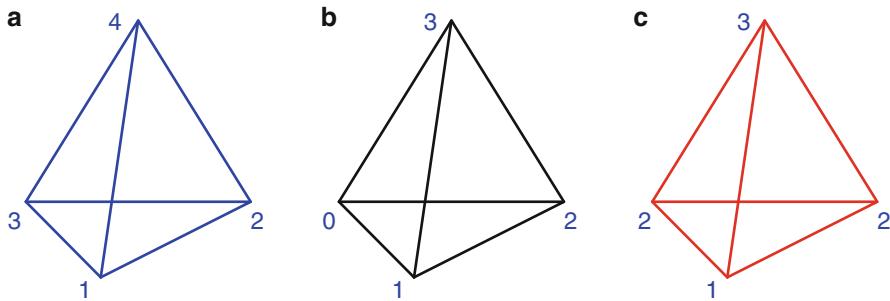


Fig. 13 An illustration of a complete simplex, a 0-complete simplex, and an almost complete simplex

Let $\lambda = x - z^0$. Then, $\lambda_j \geq 0$, $j \in K$, and $\lambda_j = 0$, $j \notin K$. Thus, for any constraint i with $a_{ij} \leq 0$ for all $j \in K$,

$$a_i^\top x = a_i^\top z^0 + a_i^\top \lambda \leq b_i + a_i^\top \lambda = b_i + \sum_{j \in K} a_{ij} \lambda_j \leq b_i.$$

This implies that, for every violated constraint $i \in I(x)$ (if exists), $a_{ij} > 0$ for some $j \in K$. Since at most one of a_{kj} , $j = 1, 2, \dots, n$, is positive for any $k \in M$; hence, $a_{ij} \leq 0$ for all $j \notin K$ and for every violated constraint i . Therefore, if $f_j(x) > 0$, one must have $j \in K$, that is, $l(x, 0) \in K$ from the labeling rule.

Consider $x \notin D(P)$. Since $x^l \leq z^0 \leq x$, hence, $x_j > x_j^u$ for some $j \in N$. From $x \in H(z^0, K)$, it is known that $x_j = z_j^0 \leq x_j^u$ for all $j \notin K$. Therefore, according to the labeling rule, $l(x, 0) \in K$. This completes the proof. \square

This lemma plays an essential role in the development. An illustration of labels of integer points in $H(z^0, K)$ can be found in Fig. 14. As a corollary of Lemma 7, it is obtained that

Corollary 1 If z^0 is an integer point of P , there is no complete n -dimensional simplex in $H(z^0, N) \times \{0\}$ carrying all integer labels in N_0 , and, for any $j \in N$ and $k \in N_0$, there is no complete $(n-1)$ -dimensional simplex in $H(z^0, N \setminus \{j\}) \times \{0\}$ carrying all integer labels in $N_0 \setminus \{k\}$.

Let

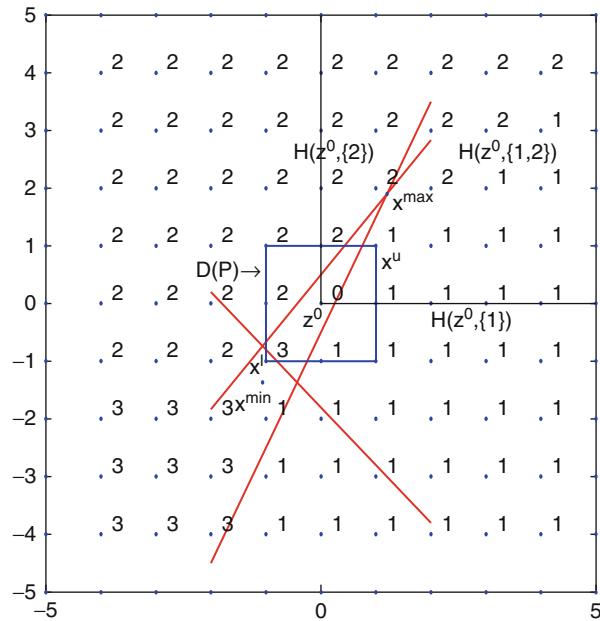
$$\Omega = \{x \in R^n \mid x^l - e \leq x \leq x^u + e\}$$

and $\partial\Omega$ denote the boundary of Ω . Clearly, $D(P) \subset \Omega$. Let $C(x^u)$ be a unit cube given by

$$C(x^u) = \{x \mid x^u \leq x \leq x^u + e\}.$$

Then, $C(x^u) = H(x^u, N) \cap \Omega$.

Fig. 14 An illustration of labels of integer points in $H(z^0, K)$



Lemma 8 $C(x^u) \times \{0\}$ contains all the complete $(n-1)$ -dimensional simplices in $\partial\Omega \times \{0\}$ carrying all integer labels in N .

Proof Let $\tau = \langle (y^1, 0), (y^2, 0), \dots, (y^n, 0) \rangle$ be a complete $(n-1)$ -dimensional simplex in $\partial\Omega \times \{0\}$ carrying all integer labels in N . Without loss of generality, it is assumed that $l(y^i, 0) = i$, $i = 1, 2, \dots, n$.

Since τ is an $(n-1)$ -dimensional simplex in the boundary of $\Omega \times \{0\}$, there must be an index $h \in N$ such that $y_h^1 = y_h^2 = \dots = y_h^n$. Suppose that

$$\tau \subset \{x \in \Omega \mid x_h = x_h^l - 1\} \times \{0\},$$

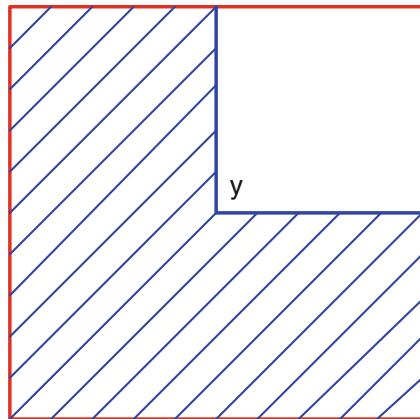
that is, this common entry hits the lower bound side of Ω . But, for any integer point $(x, 0) \in R^n \times \{0\}$, from (4) of the labeling rule, it follows that $l(x, 0) = n+1$ if $x < x^l$. Hence, $y^i \not< x^l$ for every vertex $(y^i, 0)$ of τ . In particular,

$$y_h^h - x_h^l = \max_{j \in N} y_j^h - x_j^l \geq 0.$$

This contradicts with $y_h^h = x_h^l - 1 < x_h^l$. Thus, the common entry of τ must hit the upper bound side of Ω , or

$$\tau \subset \{x \in \Omega \mid x_h = x_h^u + 1\} \times \{0\}.$$

Fig. 15 An illustration of $L(y, C)$ (shaded area)



For all $i \in N$, from $y_h^i = x_h^u + 1$, $y^i \leq x^u + e$, and

$$l(y^i, 0) = i = \max\{k \mid y_k^i - x_k^u = \max_{j \in N} y_j^i - x_j^u\},$$

it is derived that

$$y_i^i - x_i^u = \max_{j \in N} y_j^i - x_j^u = 1.$$

Therefore, as a result of $\text{mesh}(K_1) = 1$, it follows that $\tau \subset C(x^u) \times \{0\}$. This completes the proof. \square

For $y \in R^n$ and $C \subseteq R^n$, let

$$L(y, C) = \{x \in C \mid x_i \leq y_i \text{ for some } i \in N\}$$

and $\partial L(y, C)$ denote the boundary of $L(y, C)$. An illustration of $L(y, C)$ can be found in Fig. 15. As a result of Corollary 1 and Lemma 8, it is obtained that

Corollary 2 If z^0 is an integer point of P , there is no complete n -dimensional simplex in $\Gamma(x^0, \partial L(z^0, \Omega))$ carrying all integer labels in N_0 .

Lemma 9 For $K \subset N_0$ with $K \neq \emptyset$, there is no complete $(|K| - 1)$ -dimensional simplex in $G(x^0, K) \cap (\partial \Omega \times \{0\})$ carrying all integer labels in K .

Proof Suppose that $\tau = \langle (y^1, 0), (y^2, 0), \dots, (y^{|K|}, 0) \rangle$ is a complete $(|K| - 1)$ -dimensional simplex in $G(x^0, K) \cap (\partial \Omega \times \{0\})$ carrying all integer labels in K . Consider the case of $n + 1 \notin K$. Without loss of generality, it is assumed that $K = \{1, 2, \dots, k_0\}$ and $l(y^i, 0) = i$, $i = 1, 2, \dots, k_0$. Since $x^0 \leq x^u$ and $n + 1 \notin K$, there is some $h \in K$ such that

$$\tau \subset \{x \in \Omega \mid x \leq x^u \text{ and } x_h = x_h^l - 1\} \times \{0\}.$$

For any integer point $(x, 0) \in R^n \times \{0\}$, from (4) of the labeling rule, it is known that $l(x, 0) = n + 1$ if $\max_{j \in N} x_j - x_j^l < 0$. Thus, for every $i \in K$, from $x^0 \leq x^u$, $l(y^i, 0) = i$ and (4) of the labeling rule, it follows that

$$y_i^i - x_i^l = \max_{j \in N} y_j^i - x_j^l \geq 0.$$

This contradicts with $y_h^h = x_h^l - 1$.

Consider the case of $n + 1 \in K$. Since $|K| \leq n$, there is some $k \in N$ with $k \notin K$. Let $(y, 0)$ be the vertex of τ with $l(y, 0) = n + 1$. Then, from $y \in \partial\Omega$ and the labeling rule, one can get that $y = x^l - e$ since $x^l - e$ is a unique integer point in $\partial\Omega$ carrying integer label $n + 1$. However, from $(y, 0) \in G(x^0, K)$ and $k \in N$ with $k \notin K$, it holds that $y_k \geq x_k^0 \geq x_k^l$. This contradicts with $y_k = x_k^l - 1$. The lemma follows immediately. \square

[Corollaries 1, 2](#), and [Lemma 9](#) together imply the following conclusion.

Corollary 3 *All the simplices generated by the method are contained in $\Gamma(x^0, \Omega)$.*

To prove [Theorem 7](#), one needs to show first that the method does not cycle. To accomplish this task, one can rely on an undirected graph. For convenience of further discussions, several shorthand notations are introduced in the following:

1. An AC(K)S stands for an almost complete $|K|$ -dimensional simplex, which carries only integer labels in K and is contained in $G(x^0, K) \cap \Gamma(x^0, \Omega)$.
2. A C(K)S stands for a complete $|K|$ -dimensional simplex, which carries all integer labels in K and is contained in $G(x^0, K) \cap \Gamma(x^0, \Omega)$.
3. A ZC(K)S stands for a 0-complete $|K|$ -dimensional simplex, which carries all integer labels in K and is contained in $G(x^0, K) \cap \Gamma(x^0, \Omega)$.
4. A C($K, -1$)S stands for a complete $(|K| - 1)$ -dimensional simplex, which carries all integer labels in K and is contained in $\cup_{j \in K} (G(x^0, K \setminus \{j\}) \cap \Gamma(x^0, \Omega))$.
5. A C(n)S stands for a complete n -dimensional simplex in $\Gamma(x^0, \partial\Omega)$.

For $K \subset N_0$, a graph $\Xi(K)$ is defined as follows:

- Nodes of $\Xi(K)$ consist of:
 1. All AC(K)Ss
 2. All C(K)Ss
 3. All ZC(K)Ss
 4. All C($K, -1$)Ss
 - There is an edge between two nodes of $\Xi(K)$ if one is a complete facet of the other or they have a common complete facet that carries all integer labels in K .
- For $K = N_0$, a graph $\Xi(N_0)$ is defined as follows:
- Nodes of $\Xi(N_0)$ consist of:
 1. All AC(N_0)Ss
 2. All ZC(N_0)Ss
 3. All C($N_0, -1$)Ss
 4. All C(n)Ss

- There is an edge between two nodes of $\Xi(N_0)$ if one is a complete facet of the other or they have a common complete facet that carries all integer labels in N_0 .

Let Ξ be a graph formed by taking the union of the graphs $\Xi(K)$ over all the subsets K of N_0 in such a way that all the nodes given by the same complete simplex become one node of Ξ .

Two nodes of a graph are adjacent if there is an edge between them. The degree of a node of a graph is equal to the number of nodes adjacent to it.

Let us first determine the degree of each node of Ξ .

1. Consider node σ of an $AC(K)S$. From the definition of Ξ , one can obtain that node σ is only adjacent to a pair of nodes given by one of the following pairs:

- a. Two $AC(K)S$ s
- b. An $AC(K)S$ and a $C(K)S$
- c. An $AC(K)S$ and a $ZC(K)S$
- d. An $AC(K)S$ and a $C(K, -1)S$
- e. Two $C(K)S$ s
- f. A $C(K)S$ and a $ZC(K)S$
- g. A $C(K)S$ and a $C(K, -1)S$
- h. Two $ZC(K)S$ s
- i. A $ZC(K)S$ and a $C(K, -1)S$
- j. Two $C(K, -1)S$ s

Thus, node σ has degree 2 (a balanced node).

2. Consider node σ of a $C(K)S$ with $0 < |K| \leq n$. Let h denote the unique integer label that is carried by σ and does not belong to K . From the definition of Ξ , one can obtain that node σ is only adjacent to a pair of nodes given by one of the following pairs:

- a. An $AC(K)S$ and an $AC(K \cup \{h\})S$
- b. An $AC(K)S$ and a $C(K \cup \{h\})S$
- c. An $AC(K)S$ and a $ZC(K \cup \{h\})S$
- d. A $C(K)S$ and an $AC(K \cup \{h\})S$
- e. A $C(K)S$ and a $C(K \cup \{h\})S$
- f. A $C(K)S$ and a $ZC(K \cup \{h\})S$
- g. A $ZC(K)S$ and an $AC(K \cup \{h\})S$
- h. A $ZC(K)S$ and a $C(K \cup \{h\})S$
- i. A $ZC(K)S$ and a $ZC(K \cup \{h\})S$
- j. A $C(K, -1)S$ and an $AC(K \cup \{h\})S$
- k. A $C(K, -1)S$ and a $C(K \cup \{h\})S$
- l. A $C(K, -1)S$ and a $ZC(K \cup \{h\})S$

Thus, node σ has degree 2 (a balanced node).

3. Consider node σ of a $C(K)S$ with $|K| = 0$. Then, $\sigma = (x^0, 0)$. From the definition of Ξ , one can obtain that node σ is only adjacent to the node given by one of the following:

- a. An $AC(\{l(x^0, 0)\})S$
- b. A $C(\{l(x^0, 0)\})S$
- c. A $ZC(\{l(x^0, 0)\})S$

Thus, node σ has degree 1 when $|K| = 0$ (an unbalanced node).

4. Consider node σ of a $ZC(K)S$. From the definition of Ξ , one can obtain that node σ is only adjacent to the node given by one of the following:

- a. An $AC(K)S$
- b. A $C(K)S$
- c. A $ZC(K)S$
- d. A $C(K, -1)S$

Thus, node σ has degree 1 (a balanced node).

5. Consider node σ of a $C(n)S$. From the definition of Ξ , one can obtain that node σ is only adjacent to the node given by one of the following:

- a. An $AC(N_0)S$
- b. A $ZC(N_0)S$

Thus, node σ has degree 1 (an unbalanced node).

From the construction of Ξ , one can see that each node of Ξ belongs uniquely to one of these five categories. The above results show that the degree of each node of Ξ is at most two and that $(x^0, 0)$, $C(n)S$ s, and $ZC(K)S$ s are the only nodes that have degree 1. Therefore, it follows that

Lemma 10 *Each connected component of Ξ has one of the following forms:*

- *A simple circuit, in which each of nodes has degree 2.*
- *A simple path, in which each of end nodes has degree 1 and is given by one of the following:*
 1. $(x^0, 0)$
 2. A $C(n)S$
 3. A $ZC(K)S$

Clearly, the starting point of the method, $(x^0, 0)$, is a node of Ξ with degree 1. Thus, $(x^0, 0)$ is a starting node of a simple path of Ξ . Let P_{x^0} denote the simple path of Ξ starting from $(x^0, 0)$. From the method, one can see that all the simplices generated by the method are nodes of P_{x^0} . In fact, the method exactly follows P_{x^0} by moving from one node to another. Therefore, the method does not cycle.

Proof of Theorem 7

1. Assume that P has an integer point. Let z^0 be an integer point of P . Suppose that $x^0 \geq z^0$. Then, from [Lemma 7](#) and [Corollary 1](#), it is derived that all the simplices generated by the method are contained in the bounded set $\{x \in R^n \mid z^0 \leq x \leq x^0\} \times \{0\}$. Thus, within a finite number of iterations, the method terminates with a $ZC(K)S$.

Suppose that $x^0 \not\geq z^0$. Then, $x^0 \in \mathcal{L}(z^0, D(P)) \subseteq L(z^0, \Omega)$. Thus, from [Lemma 7](#), [Corollaries 1, 2](#), and [Lemma 9](#), it is derived that all the simplices generated by the method are contained in the bounded set $\Gamma(x^0, L(z^0, \Omega))$. Since $z^0 \leq x^u$, hence, within a finite number of iterations, the method terminates with a $ZC(K)S$.

2. Assume that P has no integer point. As a result of [Corollary 3](#), one can obtain that, within a finite number of iterations, the method terminates with $K = N_0$ and a vertex $(y, 0)$ such that $y \geq x^u$. This completes the proof. \square

8 A Homotopy-Like Simplicial Method for Integer Programming

Let $x^0 = (x_1^0, x_2^0, \dots, x_n^0)^\top$ be an arbitrary integer point in $D(P)$.

Definition 10 (An Integer Labeling Rule) For each integer point $(x, \gamma) \in (R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$, assign to (x, γ) an integer label $l(x, \gamma) \in N_0 \cup \{0\}$ as follows:

1. $l(x^0, 1) = n + 1$.
2. For $(x, 0)$ with $x \in D(P)$,

$$l(x, 0) = \begin{cases} 0 & \text{if } f(x) = 0, \\ \max\{k \mid f_k(x) = \max_{j \in N} f_j(x)\} & \text{if } f_j(x) > 0 \text{ for some } j \in N, \\ n + 1 & \text{if } f(x) \leq 0 \text{ and } f(x) \neq 0. \end{cases}$$

3. For $(x, 0)$ with $x \geq x^l$ and $x_j > x_j^u$ for some $j \in N$,

$$l(x, 0) = \max\{k \mid x_k - x_k^u = \max_{j \in N} x_j - x_j^u\}.$$

4. For $(x, 0)$ with $x_j < x_j^l$ for some $j \in N$, $l(x, 0) = n + 1$.

5. For $(x, -1)$ with $x \geq x^0$ and $x_j > x_j^0$ for some $j \in N$,

$$l(x, -1) = \max\{k \mid x_k - x_k^0 = \max_{j \in N} x_j - x_j^0\}.$$

6. For $(x, -1)$ with $x = x^0$ or $x_j < x_j^0$ for some $j \in N$, $l(x, -1) = n + 1$.

For further discussions, one needs a triangulation of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$ that subdivides both $R^n \times [-1, 0]$ and $\Gamma(x^0, R^n)$ into simplices in such a way that every integer point of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$ is a vertex of some simplex of the triangulation and every such vertex is an integer point of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$. Any cubic triangulation of R^n can be an underlying triangulation of the method. For simplicity, one can choose the K_1 -triangulation in Freudenthal [52], which is as follows.

For $k = 1, 2, \dots, n + 1$, let u^k be the k th unit vector of R^{n+1} . A simplex of the K_1 -triangulation of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$ is the convex hull of $n + 2$ vectors, $y^0, y^1, \dots, y^n, y^{n+1}$, given by $y^0 = y$, $y^k = y^{k-1} + u^{\pi(k)}$, $k = 1, 2, \dots, n$, and

$$y^{n+1} = \begin{cases} y^n + u^{\pi(n+1)} & \text{if } y_{n+1} = -1, \\ (x^0, 1) & \text{otherwise,} \end{cases}$$

where $y = (y_1, y_2, \dots, y_{n+1})^\top$ is an integer point in $R^n \times [-1, 0]$ and $\pi = (\pi(1), \pi(2), \dots, \pi(n+1))$ is a permutation of elements of N_0 such that $\pi(n+1) =$

Table 5 Pivot rules of the K_1 -triangulation of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$

i		\bar{y}	$\bar{\pi}$
0	$y_{n+1} = -1$	$y + u^{\pi(1)}$	$(\pi(2), \dots, \pi(n+1), \pi(1))$
	$y_{n+1} = 0$	$y + u^{\pi(1)}$	$(\pi(2), \dots, \pi(n), \pi(1), \pi(n+1))$
$1 \leq i < n$		y	$(\pi(1), \dots, \pi(i+1), \pi(i), \dots, \pi(n+1))$
n	$y_{n+1} = -1$	y	$(\pi(1), \dots, \pi(n-1), \pi(n+1), \pi(n))$
	$y_{n+1} = 0$	$y - u^{\pi(n)}$	$(\pi(n), \pi(1), \dots, \pi(n-1), \pi(n+1))$
$n+1$		$y - u^{\pi(n+1)}$	$(\pi(n+1), \pi(1), \dots, \pi(n))$

$n+1$ if $y_{n+1} = 0$. Let K_1 be the set of all such simplices. Then, K_1 is a triangulation of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$. Since a simplex of the K_1 -triangulation is uniquely determined by y and π , it is denoted as $K_1(y, \pi)$. Two simplices of K_1 are adjacent if they share a common facet. For a given simplex $\sigma = K_1(y, \pi)$ with vertices y^0, y^1, \dots, y^{n+1} , its adjacent simplex opposite to a vertex, say y^i , is given by $K_1(\bar{y}, \bar{\pi})$, where \bar{y} and $\bar{\pi}$ are generated according to the pivot rules given in [Table 5](#).

Let \mathcal{K}_1 be the set of faces of simplices of K_1 . A q -dimensional simplex of \mathcal{K}_1 with vertices y^0, y^1, \dots, y^q is denoted by $\langle y^0, y^1, \dots, y^q \rangle$. For $\sigma \in \mathcal{K}_1$ with $\sigma \subset R^n \times [-1, 0]$, let $\text{grid}(\sigma) = \max\{\|x - y\| \mid (x, \gamma_x) \in \sigma \text{ and } (y, \gamma_y) \in \sigma\}$, where $\|\cdot\|$ denotes the infinity norm. It is defined that

$$\text{mesh}(K_1) = \max\{\text{grid}(\sigma) \mid \sigma \in \mathcal{K}_1 \text{ and } \sigma \subset R^n \times [-1, 0]\}.$$

Then, $\text{mesh}(K_1) = 1$.

With the integer labeling rule in [Definition 10](#) and the K_1 -triangulation of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$, a homotopy-like simplicial method is developed to determine whether there is an integer point in P , which is as follows.

Initialization: Let $y = (x^0, -1)$ and $\pi = (1, 2, \dots, n+1)$. Then, $\sigma_0 = K_1(y, \pi) = \langle y^0, y^1, \dots, y^{n+1} \rangle$ is a unique $(n+1)$ -dimensional simplex of K_1 in $R^n \times [-1, 0]$ having τ_0 as a facet. Let y^+ be the vertex of σ_0 opposite to τ_0 and $k = 0$. Go to *Step 1*.

Step 1: Compute $l(y^+)$. If $l(y^+) = 0$, the method terminates and an integer point of P has been found. If $\sigma_k \subset \Gamma(x^0, R^n)$ and $y^+ \geq (x^u, 0)$ with $y_{n+1}^+ = 0$, the method terminates and P has no integer point. Otherwise, let y^- be the vertex of σ_k other than y^+ and carrying integer label $l(y^+)$ and τ_{k+1} the facet of σ_k opposite to y^- , and go to *Step 2*.

Step 2: Let σ_{k+1} be the unique $(n+1)$ -dimensional simplex that is adjacent to σ_k and has τ_{k+1} as a facet, y^+ the vertex of σ_{k+1} opposite to τ_{k+1} , and $k = k + 1$, and go to *Step 1*.

Theorem 8 Within a finite number of iterations, the method either yields an integer point in P or proves that no such point exists.

The following example illustrates how the method works when $n = 2$.

Example 9 Consider

$$P = \left\{ x = (x_1, x_2)^\top \middle| \begin{array}{l} -x_1 + x_2 \leq 1/2, \\ x_1 - x_2 \leq 1/2, \\ x_1 \leq 4/5, \\ -x_1 \leq 4/5 \end{array} \right\}.$$

[Figure 16](#) illustrates how the method moves along a simplicial path starting from τ_0 and ending at a feasible integer point in P .

In the following, a proof of [Theorem 8](#) is given.

For $y \in R^n$ and $K \subseteq N$, let

$$H(y, K) = \{y + x \mid 0 \leq x_i, i \in K, \text{ and } x_i = 0, i \notin K\}.$$

Let $y = (x^0, -1)$, $\pi = (1, 2, \dots, n+1)$, $y^0 = y$, $y^k = y^{k-1} + u^k$, $k = 1, 2, \dots, n+1$, $\tau_0 = \langle y^0, y^1, \dots, y^n \rangle$, and $\sigma_0 = \langle y^0, y^1, \dots, y^{n+1} \rangle$. Then, from [Definition 10](#), it holds that $l(y^0) = n+1$ and $l(y^k) = k$, $k = 1, 2, \dots, n$. Thus, τ_0 is a complete n -dimensional simplex in $R^n \times \{-1\}$.

Lemma 11 τ_0 is a unique complete n -dimensional simplex of \mathcal{K}_1 in $R^n \times \{-1\}$.

Proof Let $\tau = \langle \bar{y}^0, \bar{y}^1, \dots, \bar{y}^n \rangle$ be a complete n -dimensional simplex of \mathcal{K}_1 in $R^n \times \{-1\}$. Without loss of generality, it is assumed that $l(\bar{y}^0) = n+1$ and $l(\bar{y}^i) = i$, $i = 1, 2, \dots, n$. Then, from [Definition 10](#), it holds that $\bar{y}^i \geq (x^0, -1)$ and $\bar{y}_i^i \geq x_i^0 + 1$, $i = 1, 2, \dots, n$. Thus, as a result of $\text{mesh}(\mathcal{K}_1) = 1$, it follows that $\tau \subset H(x^0, N) \times \{-1\}$. Since $(x^0, -1)$ is a unique point of $H(x^0, N) \times \{-1\}$ carrying integer label $n+1$, hence, $\bar{y}^0 = (x^0, -1)$. Therefore, according to [Definition 10](#) and the definition of the \mathcal{K}_1 -triangulation of $(R^n \times [-1, 0]) \cup \Gamma(x^0, R^n)$, one must have that $\bar{y}^i = \bar{y}^{i-1} + u^i$, $i = 1, 2, \dots, n$. So, $\tau = \tau_0$. This completes the proof. \square

Lemma 12 There is no complete n -dimensional simplex of \mathcal{K}_1 in $\partial\Omega \times [-1, 0]$.

Proof Suppose that

$$\sigma = \langle (y^0, \gamma), (y^1, -1), \dots, (y^h, -1), (y^{h+1}, 0), \dots, (y^n, 0) \rangle$$

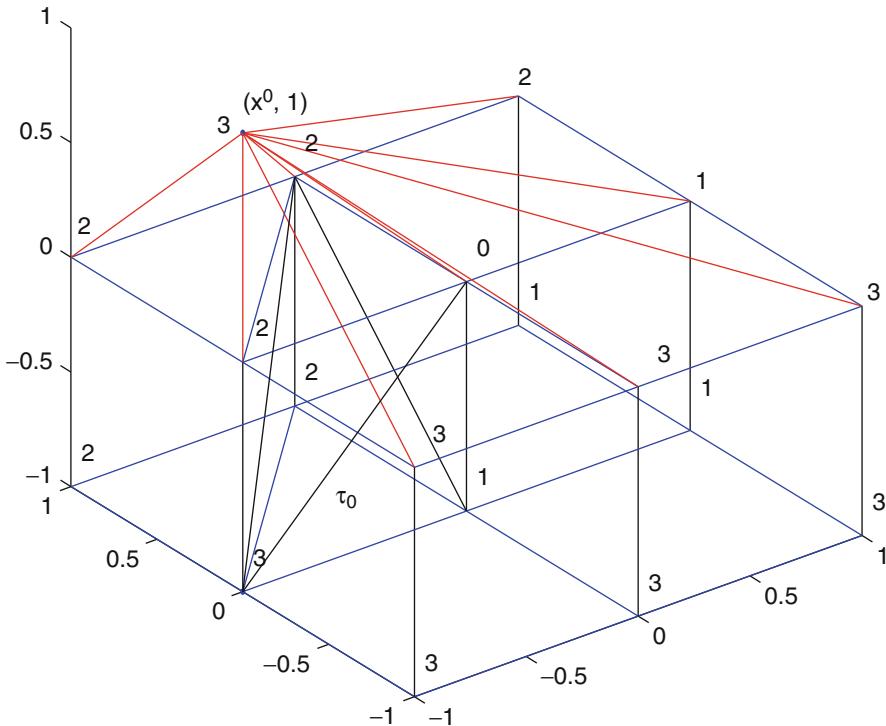


Fig. 16 An illustration of the method

with $\gamma \in \{-1, 0\}$ a complete n -dimensional simplex of \mathcal{K}_1 in $\partial\Omega \times [-1, 0]$. Without loss of generality, it is assumed that $l(y^0, \gamma) = n + 1$, $l(y^i, -1) = i$, $i = 1, 2, \dots, h$, and $l(y^i, 0) = i$, $i = h + 1, h + 2, \dots, n$. Then, from [Definition 10](#), it holds that for some $k \in N$,

$$y_k^0 \leq x_k^0 - 1 \text{ if } \gamma = -1, \text{ and } y_k^0 \leq x_k^l - 1 \text{ if } \gamma = 0,$$

and that

$$y_i^i \geq x_i^0 + 1, \quad i = 1, \dots, h,$$

$$y_i^i \geq x_i^u + 1, \quad i = h + 1, \dots, n.$$

Thus, when $\gamma = -1$, if $k \leq h$, then

$$y_k^k - y_k^0 \geq x_k^0 + 1 - x_k^0 + 1 = 2,$$

and if $k > h$, then

$$y_k^k - y_k^0 \geq x_k^u + 1 - x_k^0 + 1 \geq 2;$$

and, when $\gamma = 0$, if $k \leq h$, then

$$y_k^k - y_k^0 \geq x_k^0 + 1 - x_k^l + 1 \geq 2,$$

and if $k > h$, then

$$y_k^k - y_k^0 \geq x_k^u + 1 - x_k^l + 1 \geq 2.$$

Therefore, $\text{grid}(\sigma) \geq 2$, which contradicts with $\text{mesh}(K_1) = 1$. This completes the proof. \square

Lemma 13 $\Gamma(x^0, H(x^u, N) \cap \Omega)$ contains all the complete n -dimensional simplices in $\Gamma(x^0, \partial\Omega)$.

Proof Let $\tau = \{(y^1, 0), (y^2, 0), \dots, (y^n, 0)\}$ be a complete $(n-1)$ -dimensional simplex in $\partial\Omega \times \{0\}$ carrying all integer labels in N . Without loss of generality, it is assumed that $l(y^i, 0) = i$, $i = 1, 2, \dots, n$. Then, from [Definition 10](#), it holds that there is some $k \in N$ satisfying that $\tau \subset \{x \mid x \geq x^l \text{ and } x_k = x_k^u + 1\}$. Thus, for any $i \in N$, from $l(y^i, 0) = i$, it is derived that $y_i^i - x_i^u \geq 1$. Therefore, as a result of $\text{mesh}(K_1) = 1$, it follows that $\tau \subset (H(x^u, N) \cap \Omega) \times \{0\}$. This completes the proof. \square

[Lemmas 11, 12](#), [Corollary 1](#), [Lemma 13](#), and [Corollary 2](#) together imply the following conclusion.

Corollary 4 All the simplices generated by the method are contained in $(\Omega \times [-1, 0]) \cup \Gamma(x^0, \Omega)$.

To prove [Theorem 8](#), one needs to show first that the method does not cycle. To accomplish this task, one can rely on an undirected graph. In the following discussions,

1. A C(n)S stands for a complete n -dimensional simplex in the boundary of $(\Omega \times [-1, 0]) \cup \Gamma(x^0, \Omega)$.
2. An AC($n+1$)S stands for an almost complete $(n+1)$ -dimensional simplex in $(\Omega \times [-1, 0]) \cup \Gamma(x^0, \Omega)$.
3. A C($n+1$)S stands for a 0-complete $(n+1)$ -dimensional simplex in $(\Omega \times [-1, 0]) \cup \Gamma(x^0, \Omega)$.

Let Φ be a graph defined as follows. Nodes of Φ consist of:

1. All C(n)Ss.
2. All AC($n+1$)Ss.
3. All C($n+1$)Ss.

There is an edge between two nodes of Φ if one is a complete facet of the other or they have a common complete facet carrying all integer labels in N_0 .

Let us first determine the degree of each node of Φ :

1. Consider node σ of a $C(n)S$. From the definition of Φ , one can obtain that node σ is only adjacent to a node given by one of the following:
 - a. An $AC(n+1)S$
 - b. A $C(n+1)S$
 Thus, node σ has degree 1 (which is called an unbalanced node).
2. Consider node σ of an $AC(n+1)S$. From the definition of Φ , one can obtain that node σ is only adjacent to a pair of nodes given by one of the following pairs:
 - a. Two $C(n)S$ s
 - b. A $C(n)S$ and an $AC(n+1)S$
 - c. A $C(n)S$ and a $C(n+1)S$
 - d. Two $AC(n+1)S$ s
 - e. An $AC(n+1)S$ and a $C(n+1)S$
 - f. Two $C(n+1)S$ s
 Thus, node σ has degree 2 (which is called a balanced node).

3. Consider node σ of a $C(n+1)S$. From the definition of Φ , one can obtain that node σ is only adjacent to a node given by one of the following:
 - a. A $C(n)S$
 - b. An $AC(n+1)S$
 - c. A $C(n+1)S$

Thus, node σ has degree 1 (an unbalanced node).

From the construction of Φ , one can see that each node of Φ belongs uniquely to one of these three categories. The above results show that the degree of each node of Φ is at most two and that $C(n+1)S$ s and $C(n)S$ s are only nodes that have degree 1. Therefore, it follows that

Lemma 14 *Each connected component of Φ has one of the following forms:*

- *A simple circuit, in which each of nodes has degree 2*
- *A simple path, in which each of end nodes has degree 1 and is given by one of the following:*
 1. A $C(n)S$
 2. A $C(n+1)S$

Clearly, the starting simplex of the method, τ_0 , is a node of Φ with degree 1. Thus, τ_0 is a starting node of a simple path of Φ . Let P_{τ_0} denote the simple path of Φ starting from τ_0 . From the method, one can see that all the simplices generated by the method are nodes of P_{τ_0} . In fact, the method exactly follows P_{τ_0} by moving from one node to another. Therefore, the method does not cycle.

Proof of Theorem 8

1. Assume that P has an integer point. As a result of Corollaries 2 and 4, one can obtain that, within a finite number of iterations, the method terminates with a $C(n+1)S$.

2. Assume that P has no integer point. As a result of [Lemma 13](#) and [Corollary 4](#), it follows that, within a finite number of iterations, the method terminates with a vertex $(y, 0)$ such that $y \geq x^u$. This completes the proof. \square

9 Conclusion

In this chapter, a brief introduction to simplicial methods has been presented for approximating fixed points and computing integer points in polytopes. As a powerful mechanism for constructive proofs and numerical solutions of many important scientific and engineering problems, simplicial methods have been substantially developed in the literature. However, due to space limitation, only some representative results have been included here. The major developments of simplicial methods and their applications might be found in the recommended reading.

Acknowledgements This work was partially supported by GRF: CityU 112809 of the Government of Hong Kong SAR.

Recommended Reading

1. E.L. Allgower, K. Georg, Simplicial and continuation methods for approximating fixed points and solutions to systems of equations. *SIAM Rev.* **22**, 28–85 (1980)
2. E.L. Allgower, K. Georg, Piecewise linear methods for nonlinear equations and optimization. *J. Comput. Appl. Math.* **124**, 245–261 (2000)
3. K.J. Arrow, F.H. Hahn, *General Competitive Analysis* (Holden-Day, San Francisco 1971)
4. I. Barany, Borsuk's theorem through complementary pivoting. *Math. Program.* **18**, 84–88 (1980)
5. M.N. Broadie, B.C. Eaves, A variable rate refining triangulation. *Math. Program.* **38**, 161–202 (1987)
6. P.S. Brooks, Infinite regression in the Eaves-Saigal algorithm. *Math. Program.* **19**, 313–327 (1980)
7. L.E. Brouwer, Über Abbildung von Mannig-faltigkeiten, *Mathematische Annalen* **71**, 97–115 (1912)
8. A. Charnes, G.B. Garcia, C.E. Lemke, Constructive proofs of theorems relating to: $F(x)=y$. with applications. *Math. Program.* **12**, 328–343 (1977)
9. R.W. Cottle, Minimal triangulation of the 4-cube. *Discret. Math.* **40**, 25–29 (1982)
10. R.W. Cottle, G.B. Dantzig, Complementary pivot theory of mathematical programming. *Linear Algebra Appl.* **1**, 103–125 (1968)
11. R.W. Cottle, C.E. Lemke, Nonlinear programming, *SIAM-AMS Proceedings*, vol. 9 (AMS, Providence, 1976)
12. Y. Dai, G. van der Laan, A.J.J. Talman, Y. Yamamoto, A simplicial algorithm for the nonlinear stationary point problem on an unbounded polyhedron. *SIAM J. Optim.* **1**, 151–165 (1991)
13. C. Dang, The D_1 -triangulation of R^n for simplicial algorithms for computing solutions of nonlinear equations. *Math. Oper. Res.* **16**, 148–161 (1991)
14. C. Dang, The D_2^* -triangulation for continuous deformation algorithms to compute solutions of nonlinear equations. *SIAM J. Optim.* **3**, 784–799 (1993a)

15. C. Dang, The D_2 -triangulation for simplicial homotopy algorithms for computing solutions of nonlinear equations. *Math. Program.* **59**, 307–324 (1993b)
16. C. Dang, *Triangulations and Simplicial Methods*. Lecture Notes in Economics and Mathematical Systems, vol. 421 (Springer, Berlin/New York, 1995), 196p
17. C. Dang, An arbitrary starting homotopy-like simplicial algorithm for computing an integer point in a class of polytopes, *SIAM J. Discret. Math.* **23**, 609–633 (2009)
18. C. Dang, H. van Maaren, A simplicial approach to the determination of an integral point of a simplex. *Math. Oper. Res.* **23**, 403–415 (1998)
19. C. Dang, H. van Maaren, An arbitrary starting variable dimension algorithm for computing an integer point of a simplex. *Comput. Optim. Appl.* **14**, 133–155 (1999)
20. C. Dang, H. van Maaren, Computing an integer point of a simplex with an arbitrary starting homotopy-like simplicial algorithm, *J. Comput. Appl. Math.* **129**, 151–170 (2001)
21. C. Dang, Y. Ye, Computing an integer point in a class of polytopes. *Lect. Notes Oper. Res.* **14**, 258–263 (2011)
22. G.B. Dantzig, B.C. Eaves, *Studies in Optimization 10* (Washington, DC, American Mathematical Society, 1974)
23. G.B. Dantzig, B.C. Eaves, D. Gale, An algorithm for a piecewise linear model of trade and production with negative prices and bankruptcy. *Math. Program.* **16**, 150–169 (1979)
24. R.H. Day, S.M. Robinson, *Mathematical Topics in Economic Theory and Computation* (Academic, New York, 1972)
25. G. Debreu, *Theory of Value* (Wiley, New York, 1959)
26. T.M. Doup, *Simplicial Algorithms on the Simplex*. Lecture Notes in Economics and Mathematical Systems, vol. 318 (Springer, Berlin, 1988)
27. T.M. Doup, A.J.J. Talman, A new variable dimension algorithm to find equilibria on the product space of unit simplices. *Math. Program.* **37**, 319–355 (1987a)
28. T.M. Doup, A.J.J. Talman, The 2-ray algorithm for solving equilibrium problems on the unit simplex. *Methods Oper. Res.* **57**, 269–285 (1987b)
29. T.M. Doup, A.J.J. Talman, A continuous deformation algorithm on the product space of unit simplices. *Math. Oper. Res.* **12**, 485–521 (1987c)
30. T.M. Doup, A.H. van den Elzen, A.J.J. Talman, Simplicial algorithms for solving the nonlinear complementarity problem on the simplex, in *The Computation and Modelling of Economic Equilibria*, ed. by A.J.J. Talman, G. van Laan. Contributions to Economic Analysis, vol. 167 (North-Holland, Amsterdam, 1987a), pp. 205–230, 125–154
31. T.M. Doup, G. van der Laan, A.J.J. Talman, The $(2^{n+1}-2)$ -ray algorithm: a new simplicial algorithm to compute economic equilibria. *Math. Program.* **39**, 241–252 (1987b)
32. B.C. Eaves, An odd theorem. *Proc. Am. Math. Soc.* **26**, 509–513 (1970)
33. B.C. Eaves, On the basic theory of complementarity. *Math. Program.* **1**, 68–75 (1971a)
34. B.C. Eaves, Computing Kakutani fixed points. *SIAM J. Appl. Math.* **21**, 236–244 (1971b)
35. B.C. Eaves, The linear complementarity problem. *Manage. Sci.* **17**, 612–634 (1971c)
36. B.C. Eaves, Homotopies for the computation of fixed points. *Math. Program.* **3**, 1–22 (1972)
37. B.C. Eaves, Solving regular piecewise linear convex equations. *Math. Program. Study* **1**, 96–119 (1974a)
38. B.C. Eaves, Properly labeled simplexes, in *Studies in Optimization 10*, ed. by G.B. Dantzig, B.C. Eaves (Washington, DC, American Mathematical Society, 1974b), pp. 71–93
39. B.C. Eaves, A short course in solving equations with PL homotopies, in *Nonlinear Programming*, ed. by R.W. Cottle, C.E. Lemke, SIAM-AMS Proceedings, vol. 9 (AMS, Providence, 1976), pp. 73–143
40. B.C. Eaves, Computing stationary points. *Math. Program. Study* **7**, 1–14 (1978)
41. B.C. Eaves, Permutation congruent transformations of the Freudenthal triangulation with minimum surface density. *Math. Program.* **29**, 77–99 (1984a)
42. B.C. Eaves, *A Course in Triangulations for Solving Equations with Deformations*. Lecture Notes in Economics and Mathematical Systems, vol. 234 (Springer, Berlin, 1984b)
43. B.C. Eaves, R. Saigal, Homotopies for the computation of fixed points on unbounded regions. *Math. Program.* **3**, 225–237 (1972)

44. B.C. Eaves, H. Scarf, The solution of systems of piecewise linear equations. *Math. Oper. Res.* **1**, 1–27 (1976)
45. B.C. Eaves, J.A. Yorke, Equivalence of surface density and average directional density. *Math. Oper. Res.* **9**, 363–375 (1984)
46. B.C. Eaves, F.J. Gould, H.-O. Peitgen, M.J. Todd, *Homotopy Methods and Global Convergence*. NATO Conference Series, vol. 13 (Plenum, New York, 1983)
47. C.R. Engles, Economics equilibrium under deformation of the economy, in *Analysis and Computation of Fixed Points*, ed. by S. Robinson (Academic, New York, 1980), pp. 213–410
48. M.L. Fisher, F.J. Gould, A simplicial algorithm for the nonlinear complementarity problem. *Math. Program.* **6**, 281–300 (1974)
49. M.L. Fisher, F.J. Gould, W.J. Tolle, A new simplicial approximation algorithm with restarts: relations between convergence and labelling, in *Fixed Points: Algorithms and Applications*, ed. by S. Karamardian (Academic, New York, 1977), pp. 41–58
50. W. Forster, *Numerical Solution of Highly Nonlinear Problems* (North-Holland, Amsterdam, 1980)
51. W. Forster, Computing “all” solutions of systems of polynomial equations by simplicial fixed point algorithms, in [154], 39–58 (1987)
52. H. Freudenthal, Simplizialzerlegungen von Beschränkter Flachheit. *Ann. Math.* **43**, 580–582 (1942)
53. R.W. Freund, Variable dimension complexes part I: basic theory. *Math. Oper. Res.* **9**, 479–497 (1984a)
54. R.W. Freund, Variable dimension complexes part II: a unified approach to some combinatorial lemmas in topology. *Math. Oper. Res.* **9**, 498–509 (1984b)
55. R.W. Freund, Combinatorial theorems on the simplicope that generalize results on the simplex and cube. *Math. Oper. Res.* **11**, 169–179 (1986)
56. R.W. Freund, M.J. Todd, A constructive proof of Tucker’s combinatorial lemma. *J. Comb. Theory A* **30**, 321–325 (1981)
57. T. Fujisawa, E. Kuh, Piecewise linear theory of nonlinear networks. *SIAM J. Appl. Math.* **22**, 307–328 (1972)
58. C.B. Garcia, A fixed point theorem including the last theorem of Poincaré. *Math. Program.* **8**, 227–239 (1975)
59. C.B. Garcia, A hybrid algorithm for the computation of fixed points. *Manage. Sci.* **22**, 606–613 (1976)
60. C.B. Garcia, Computation of solutions to nonlinear equations under homotopy invariance. *Math. Oper. Res.* **2**, 25–29 (1977)
61. C.B. Garcia, F.J. Gould, A theorem on homotopy paths. *Math. Oper. Res.* **3**, 282–289 (1978)
62. C.B. Garcia, F.J. Gould, Scalar labelings for homotopy paths. *Math. Program.* **17**, 184–197 (1979)
63. C.B. Garcia, F.J. Gould, Relations between several path following algorithms and local and global Newton methods. *SIAM Rev.* **22**, 263–274 (1980)
64. C.B. Garcia, T.Y. Li, On the number of solutions to polynomial systems of equations. *SIAM Numer. Anal.* **17**, 540–546 (1980)
65. C.B. Garcia, W.I. Zangwill, Determining all solutions to certain systems of nonlinear equations. *Math. Oper. Res.* **4**, 1–14 (1979a)
66. C.B. Garcia, W.I. Zangwill, Finding all solutions to polynomial systems and other systems of equations. *Math. Program.* **16**, 159–176 (1979b)
67. C.B. Garcia, W.I. Zangwill, An approach to homotopy and degree theory. *Math. Oper. Res.* **4**, 390–405 (1979c)
68. C.B. Garcia, W.I. Zangwill, A flex simplicial algorithm, in *Numerical Solution of Highly Nonlinear Problems*, ed. by W. Forster (North-Holland, Amsterdam, 1980), pp. 71–92
69. C.B. Garcia, W.I. Zangwill, *Pathways to Solutions, Fixed Points, and Equilibria*. Series in Computational Mathematics (Prentice-Hall, Englewood Cliffs, 1981)
70. K. Georg, An application of simplicial algorithms to variational inequalities, in *Functional Differential Equations and Approximation of Fixed Points*, ed. by H.O. Peitgen. Lecture Notes in Mathematics, vol. 730 (Springer, New York/Berlin, 1979), pp. 126–135

71. F.J. Gould, J.W. Tolle, A unified approach to complementarity in optimization. *Discret. Math.* **7**, 225–271 (1974)
72. F.J. Gould, J.W. Tolle, An existence theorem for solutions to $f(x)=0$. *Math. Program.* **11**, 252–262 (1976)
73. F.J. Gould, J.W. Tolle, *Complementary Pivoting on a Pseudomanifold Structure with Applications in the Decision Sciences*. Sigma Series in Applied Mathematics, vol. 2 (Heldermann, Berlin, 1983)
74. T. Hansen, On the approximation of a competitive equilibrium. Ph.D. Thesis, Department of Economics, Yale University, New Haven (1968)
75. T. Hansen, On the approximation of Nash equilibrium points in an N-person noncooperative game. *SIAM J. Appl. Math.* **26**, 622–637 (1974)
76. M.W. Hirsch, A proof of the nonretractability of a cell onto its boundary. *Proc. Am. Math. Soc.* **14**, 364–365 (1963)
77. M.W. Hirsch, On algorithms for solving $f(x)=0$. *Commun. Pure Appl. Math.* **32**, 281–312 (1979)
78. M.W. Hirsch, S. Smale, *Differential Equations, Dynamical Systems, and Linear Algebra* (Academic, New York, 1984)
79. M. Hofkes, A simplicial algorithm to solve the nonlinear complementarity problem on $S^n \times R_+^n$. *J. Optim. Theory Appl.* **67**, 551–565 (1990)
80. T.C. Hu, S.M. Robinson, *Mathematical Programming* (Academic, New York, 1980)
81. M.M. Jeppson, A search for the fixed points of a continuous mapping, in *Mathematical Topics in Economic Theory and Computation*, ed. by R.H. Day, S.M. Robinson (Academic, New York, 1972), pp. 122–129
82. K. John, Parametric fixed point algorithms with applications to economic policy analysis. *Comput. Oper. Res.* **11**, 157–178 (1984)
83. S. Kakutani, A generalization of Brouwer's fixed point theorem. *Duke Math. J.* **8**, 457–459 (1941)
84. K. Kamiya, A.J.J. Talman, Simplicial algorithm to find zero points of a function with special structure on a simplicope. *Math. Oper. Res.* **16**, 609–626 (1991a)
85. K. Kamiya, A.J.J. Talman, Variable dimension simplicial algorithm for balanced games. Discussion paper 9025, Center Tilburg University, 1991b
86. S. Karamardian, The complementarity problem. *Math. Program.* **2**, 107–129 (1972)
87. S. Karamardian, *Fixed Points: Algorithms and Applications* (Academic, New York, 1977)
88. D. Köberl, The solution of nonlinear equations by the computation of fixed points with a modification of the Sandwich method. *Computing* **25**, 175–179 (1980)
89. M. Kojima, On the homotopic approach to systems of equations with separable mappings. *Math. Program. Study* **7**, 170–184 (1978a)
90. M. Kojima, A modification of Todd's triangulation J_3 . *Math. Program.* **15**, 223–237 (1978b)
91. M. Kojima, Studies on piecewise-linear approximations of piecewise- C^1 -mappings in fixed points and complementarity theory. *Math. Oper. Res.* **3**, 17–36 (1978c)
92. M. Kojima, H. Nishino, N. Arima, A PL homotopy for finding all the roots of a polynomial. *Math. Program.* **16**, 37–62 (1979)
93. M. Kojima, R. Saigal, On the number of solutions to a class of complementarity problems. *Math. Program.* **21**, 190–203 (1981)
94. M. Kojima, Y. Yamamoto, Variable dimension algorithms: basic theory, interpretation, and extensions of some existing methods. *Math. Program.* **24**, 177–215 (1982)
95. M. Kojima, Y. Yamamoto, A unified approach to the implementation of several restart fixed point algorithms and a new variable dimension algorithm. *Math. Program.* **28**, 288–328 (1984)
96. J.W.A.M. Kremers, A.J.J. Talman, Solving the nonlinear complementarity problem. *Methods Oper. Res.* **62**, 91–103 (1990)
97. H.W. Kuhn, Simplicial approximation of fixed points. *Proc. Natl. Acad. Sci.* **61**, 1238–1242 (1968)
98. H.W. Kuhn, Approximate search for fixed points. *Comput. Methods Optim. Probl.* **2**, 199–211 (1969)

99. H.W. Kuhn, A new proof of the fundamental theorem of algebra. *Math. Program. Study* **1**, 148–158 (1974)
100. H.W. Kuhn, Finding roots of polynomials by pivoting, in *Fixed Points: Algorithms and Applications*, ed. by S. Karamardian (Academic, New York, 1977), pp. 11–39
101. H.W. Kuhn, J.G. MacKinnon, The sandwich method for finding fixed points. *J. Optim. Theory Appl.* **17**, 189–204 (1975)
102. H.W. Kuhn, Z. Wang, S. Xu, On the cost of computing roots of polynomials. *Math. Program.* **28**, 156–163 (1984)
103. J.C. Lagarias, The computational complexity of simultaneous Diophantine approximation problems. *SIAM J. Comput.* **14**, 196–209 (1985)
104. C.W. Lee, Triangulating the d -cube. IBM Thomas J. Watson Research Centre Technical Report, Yorktown Heights, New York, 1984
105. C.E. Lemke, Bimatrix equilibrium points and mathematical programming. *Manage. Sci.* **11**, 681–689 (1965)
106. C.E. Lemke, J.T. Howson, Equilibrium points of bimatrix games. *SIAM Rev.* **12**, 413–423 (1964)
107. H.J. Luthi, A simplicial approximation of a solution for the nonlinear complementarity problem. *Math. Program.* **9**, 278–293 (1975)
108. J.G. MacKinnon, Solving urban general equilibrium problems by fixed point methods, in *Mathematical Programming*, ed. by S. Robinson, *Analysis and Computation of Fixed Points* (Academic, New York, 1980), pp. 197–212
109. O.L. Mangasarian, Equivalence of the complementarity problem to a system of nonlinear equations. *SIAM J. Appl. Math.* **31**, 89–92 (1976)
110. A. Mansur, J. Whalley, A decomposition algorithm for general equilibrium computation with application to international trade models. *Econometrica* **50**, 1547–1557 (1982)
111. P.S. Mara, Triangulations for the cube. *J. Comb. Theory A* **20**, 170–177 (1976)
112. A. Mas-Colell, *The Theory of General Economic Equilibrium*. Econometric Society Publication, vol. 9 (Cambridge University Press, Cambridge, MA, 1985)
113. N. Megiddo, On the parametric nonlinear complementarity problem. *Math. Program. Study* **7**, 142–150 (1978)
114. O.H. Merrill, Applications and extensions of an algorithm that computes fixed points of certain upper semi-continuous point to set mappings. Ph.D Thesis, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, 1972
115. M. Meyerson A.H. Wright, A new and constructive proof of the Borsuk-Ulam theorem. *Proc. Am. Math. Soc.* **73**, 134–136 (1979)
116. K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*. Sigma Series in Applied Mathematics, vol. 3 (Heldermann, Berlin, 1988)
117. A.N. Netravali, R. Saigal, Optimal quantizer design using a fixed point algorithm. *Bell Syst. Tech. J.* **55**, 1423–1435 (1976)
118. J.M. Ortega, W.C. Rheinboldt, *Iterative Solutions of Nonlinear Equations of Several Variables* (Academic, New York, 1970)
119. H.O. Peitgen, *Functional Differential Equations and Approximation of Fixed Points*. Lecture Notes in Mathematics, vol. 730 (Springer, New York/Berlin, 1979)
120. M. Prüfer, H.W. Siegberg, Complementarity pivoting and the Hopf degree theorem. *J. Math. Anal. Appl.* **84**, 133–149 (1981)
121. P.M. Reiser, A modified integer labeling for complementarity algorithms. *Math. Oper. Res.* **6**, 129–139 (1981)
122. J. Renegar, On the complexity of a piecewise linear algorithm for approximating roots for complex polynomials. *Math. Program.* **32**, 301–318 (1985a)
123. J. Renegar, On the cost of approximating all roots of a complex polynomial. *Math. Program.* **32**, 319–336 (1985b)
124. J. Renegar, Rudiments of an average case complexity theory for piecewise-linear path following algorithms. *Math. Program.* **40**, 113–163 (1988)
125. S. Robinson, *Analysis and Computation of Fixed Points* (Academic, New York, 1980)

126. R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, 1970)
127. P.H.M. Ruys, G. van der Laan, Computation of an industrial equilibrium, in *The Computation and Modelling of Economic Equilibria*, ed. by A.J.J. Talman, G. van Laan. Contributions to Economic Analysis, vol. 167 (North-Holland, Amsterdam, 1987), pp. 205–230
128. D.G. Saari, R. Saigal, Some generic properties of paths generated by fixed point algorithms, in *Analysis and Computation of Fixed Points* ed. by S. Robinson (Academic, New York, 1980), pp. 57–72 (1980)
129. D.G. Saari, C.P. Simon, Effective price mechanisms. *Econometrica* **46**, 1097–1125 (1978)
130. R. Saigal, On paths generated by fixed point algorithms. *Math. Oper. Res.* **4**, 359–380 (1976)
131. R. Saigal, Investigations into the efficiency of fixed point algorithms, in *Fixed Points: Algorithms and Applications*, ed. by S. Karamardian (Academic, New York, 1977a), pp. 203–223
132. R. Saigal, On the convergence rate of algorithms for solving equations that are based on methods of complementary pivoting. *Math. Oper. Res.* **2**, 108–124 (1977b)
133. R. Saigal, The fixed point approach to nonlinear programming. *Math. Program. Study* **10**, 142–157 (1979a)
134. R. Saigal, On piecewise linear approximations to smooth mappings. *Math. Oper. Res.* **2**, 153–161 (1979b)
135. R. Saigal, An efficient procedure for traversing large pieces in fixed point algorithms, in *Homotopy Methods and Global Convergence*, ed. by B.C. Eaves, F.J. Gould, H.-O. Peitgen, M.J. Todd. NATO Conference Series vol. 13 (Plenum, New York, 1983a), pp. 239–248
136. R. Saigal, A homotopy for solving large, sparse and structured fixed point problems. *Math. Oper. Res.* **8**, 557–578 (1983b)
137. R. Saigal, Computational complexity of a piecewise linear homotopy algorithm. *Math. Program.* **28**, 164–173 (1984)
138. R. Saigal, M.J. Todd, Efficient acceleration techniques for fixed point algorithms. *SIAM J. Numer. Anal.* **15**, 997–1007 (1978)
139. R. Saigal, D. Solow, L.A. Wolsey, A comparative study of two algorithms to compute fixed points over unbounded regions, in *Proceedings of VII-th Mathematical Programming Symposium*, Stanford, 1975
140. J.F. Sallee, A triangulation of the n-cube. *Discret. Math.* **40**, 81–86 (1982)
141. J.F. Sallee, The middle-cut triangulations of n-cube. *SIAM J. Discret. Math.* **5**, 407–419 (1984)
142. A.A. Samuel, M.J. Todd, An efficient simplicial algorithm for computing a zero of a convex union of smooth functions. *Math. Program.* **25**, 83–108 (1983)
143. D. Saupe, On accelerating PL continuation algorithms by predictor-corrector methods. *Math. Program.* **23**, 87–110 (1982)
144. H. Scarf, The approximation of fixed points of a continuous mapping. *SIAM J. Appl. Math.* **15**, 1328–1343 (1967a)
145. H. Scarf, The core of an N person game. *Econometrica* **35**, 50–69 (1967b)
146. H. Scarf, *The Computation of Economic Equilibria* (Yale University Press, New Haven, 1973)
147. S. Shamir, Two triangulations for homotopy fixed point algorithms with an arbitrary refinement factor, in *Analysis and Computation of Fixed Points*, ed. by S. Robinson (Academic, New York, 1980), pp. 25–56
148. J.B. Shoven, Applying fixed point algorithms to the analysis of tax policies, in *Fixed Points: Algorithms and Applications*, ed. by S. Karamardian (Academic, New York, 1977), pp. 403–434
149. D. Solow, Comparative computer results of a new complementary pivot algorithm for solving equality and inequality constrained optimization problems. *Math. Program.* **18**, 213–224 (1981a)
150. D. Solow, Homeomorphisms of triangulations with applications to computing fixed points. *Math. Program.* **20**, 213–224 (1981b)
151. A.J.J. Talman, *Variable Dimension Fixed Point Algorithms and Triangulations*. Mathematical Centre Tracts, vol. 128 (Mathematisch Centrum, Amsterdam, 1980)

152. A.J.J. Talman, L. Van der Heyden, Algorithms for the linear complementarity problem which allow an arbitrary starting point, in *Homotopy Methods and Global Convergence*, ed. by B.C. Eaves, F.J. Gould, H.-O. Peitgen, M.J. Todd. NATO Conference Series vol. 13 (Plenum, New York, 1983a), pp. 239–248, 267–286
153. A.J.J. Talman, G. van Laan, *The Computation and Modelling of Economic Equilibria*. Contributions to Economic Analysis, vol. 167 (North-Holland, Amsterdam, 1987)
154. A.J.J. Talman, Y. Yamamoto, A simplicial algorithm for stationary point problems on polytopes. *Math. Oper. Res.* **14**, 383–399 (1989)
155. M.J. Todd, A generalized complementary pivoting algorithm. *Math. Program.* **6**, 243–263 (1974)
156. M.J. Todd, *The Computation of Fixed Points and Applications*. Lecture Notes on Economics and Mathematical Systems, vol. 124 (Springer, Berlin, 1976a)
157. M.J. Todd, On triangulations for computing fixed points. *Math. Program.* **10**, 322–346 (1976b)
158. M.J. Todd, Orientation in complementary pivot algorithms. *Math. Oper. Res.* **1**, 54–66 (1976c)
159. M.J. Todd, Union jack triangulations, in *Fixed Points: Algorithms and Applications*, ed. by S. Karamardian (Academic, New York, 1977), pp. 315–336
160. M.J. Todd, Improving the convergence of fixed point algorithms. *Math. Program. Study* **7**, 151–179 (1978a)
161. M.J. Todd, On the Jacobian of a function at a zero computed by a fixed point algorithm. *Math. Oper. Res.* **3**, 126–132 (1978b)
162. M.J. Todd, A quadratically-convergent fixed point algorithm for economic equilibria and linearly constrained optimization. *Math. Program.* **18**, 111–126 (1980a)
163. M.J. Todd, Exploiting structure in piecewise linear homotopy algorithms for solving equations. *Math. Program.* **18**, 233–247 (1980b)
164. M.J. Todd, On the computational complexity of piecewise linear homotopy algorithms. *Math. Program.* **24**, 216–224 (1982)
165. M.J. Todd, J' : a new triangulation of R^n . *SIAM J. Algebr. Discr. Methods* **5**, 244–254 (1984)
166. M.J. Todd, ‘Fat’ triangulations, or solving certain nonconvex matrix optimization problems. *Math. Program.* **31**, 123–136 (1985)
167. H. Tuy, Pivotal methods for computing equilibrium points: unified approach and a new restart algorithm. *Math. Program.* **16**, 210–227 (1979)
168. E. van Damme, *Stability and Perfection of Nash Equilibria* (Springer, Berlin, 1987)
169. L. Van der Heyden, A refinement procedure for computing fixed points. *Math. Oper. Res.* **7**, 295–313 (1982)
170. A.H. van den Elzen, A.J.J. Talman, A procedure for finding Nash equilibria in bimatrix games. *Methods Models Oper. Res.* **35**, 27–43 (1991)
171. G. van der Laan, *Simplicial Fixed Point Algorithms*. Mathematical Centre Tracts, vol. 129 (Mathematisch Centrum, Amsterdam, 1980)
172. G. van der Laan, Simplicial approximation of unemployment equilibria. *J. Math. Econ.* **9**, 83–97 (1982)
173. G. van der Laan, The computation of general equilibrium in economies with a block diagonal pattern. *Econometrica* **53**, 659–665 (1985)
174. G. van der Laan, L.P. Seelen, Efficiency and implementation of simplicial zero point algorithms. *Math. Program.* **30**, 196–217 (1984)
175. G. van der Laan, A.J.J. Talman, A restart algorithm for computing fixed points without an extra dimension. *Math. Program.* **17**, 74–84 (1979a)
176. G. van der Laan, A.J.J. Talman, A restart algorithm without an artificial level for computing fixed points on unbounded regions, in *Functional Differential Equations and Approximation of Fixed Points*, ed. by H.O. Peitgen. Lecture Notes in Mathematics, vol. 730 (Springer, New York/Berlin, 1979b), pp. 247–2586
177. G. van der Laan, A.J.J. Talman, An improvement of fixed point algorithms by using a good triangulation. *Math. Program.* **18**, 274–285 (1980a)

178. G. van der Laan, A.J.J. Talman, A new subdivision for computing fixed points with a homotopy algorithm. *Math. Program.* **19**, 78–91 (1980b)
179. G. van der Laan, A.J.J. Talman, Variable dimension restart algorithms for approximating fixed points, in *Numerical Solution of Highly Nonlinear Problems* ed. by W. Forster (North-Holland, Amsterdam, 1980c), pp. 3–36
180. G. van der Laan, A.J.J. Talman, A class of simplicial restart fixed point algorithms without an extra dimension. *Math. Program.* **20**, 33–48 (1981a)
181. G. van der Laan, A.J.J. Talman, On the computation of fixed points in the product space of unit simplices and an application to noncooperative N-person games. *Math. Oper. Res.* **7**, 1–13 (1982)
182. G. van der Laan, A.J.J. Talman, Interpretation of the variable dimension fixed point algorithm with artificial level. *Math. Oper. Res.* **8**, 86–99 (1983a)
183. G. van der Laan, A.J.J. Talman, Note on the path following approach of equilibrium programming. *Math. Program.* **25**, 363–367 (1983b)
184. G. van der Laan, A.J.J. Talman, Simplicial algorithms for finding stationary points, a unifying description. *J. Optim. Theory Appl.* **50**, 262–281 (1986)
185. G. van der Laan, A.J.J. Talman, Adjustment processes for finding economic equilibria, in *The Computation and Modelling of Economic Equilibria*, ed. by A.J.J. Talman, G. van Laan. Contributions to Economic Analysis, vol. 167 (North-Holland, Amsterdam, 1987a), pp. 205–230, 85–124
186. G. van der Laan, A.J.J. Talman, Simplicial approximation of solutions to the nonlinear complementarity problem with lower and upper bounds. *Math. Program.* **38**, 1–15 (1987b)
187. G. van der Laan, A.J.J. Talman, Adjustment processes for finding economic equilibrium problems on the unit simplex. *Econ. Lett.* **23**, 119–123 (1987c)
188. G. van der Laan, A.J.J. Talman, L. Van der Heyden, Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. *Math. Oper. Res.* **12**, 377–397 (1987)
189. H.R. Varian, *Microeconomic Analysis* (W.W. Norton, New York, 1984)
190. A.F. Veinott, G.B. Dantzig, Integral extreme points. *SIAM Rev.* **10**, 371–372 (1968)
191. R.S. Wilmuth, A computational comparison of fixed point algorithms which used complementary pivoting, in *Fixed Points: Algorithms and Applications*, ed. by S. Karamardian (Academic, New York, 1977), pp. 249–280 (1977)
192. A.H. Wright, The octahedral algorithm, a new simplicial fixed point algorithm. *Math. Program.* **21**, 47–69 (1981)
193. Y. Yamamoto, A new variable dimension algorithm for the fixed point problem. *Math. Program.* **25**, 329–342 (1983)
194. Y. Yamamoto, A path following algorithm for stationary point problems. *J. OR Soc. Jpn.* **30**, 181–198 (1987)
195. W.I. Zangwill, An eccentric barycentric fixed point algorithm. *Math. Oper. Res.* **2**, 343–359 (1977)
196. W.I. Zangwill, C.B. Garcia, Equilibrium programming: the path following approach and dynamics. *Math. Program.* **21**, 262–289 (1981)

Small World Networks in Computational Neuroscience

Dmytro Korenkevych, Jui-Hong Chien, Jicong Zhang,
Deng-Shan Shiau, Chris Sackellares and Panos M. Pardalos

Contents

1	Introduction	3058
2	Small-World Networks: Models and Properties	3059
2.1	Characteristic Path Length and Clustering Coefficient	3060
2.2	Network Efficiency and Cost	3061
3	Brain Networks	3066
3.1	Structural Brain Networks	3068
3.2	Functional Brain Networks	3073
4	Conclusion	3082
	Cross-References	3083
	Recommended Reading	3083

D. Korenkevych (✉)

Industrial and System Engineering Department, University of Florida, Gainesville, FL, USA
e-mail: korenkevych@gmail.com; dmitriy@ufl.edu

J.-H. Chien

Biomedical Engineering Department, University of Florida, Gainesville, FL, USA
e-mail: vicquana@ufl.edu

J. Zhang

Industrial and System Engineering Department, University of Florida, Gainesville, FL, USA
e-mail: gatorjicong@gmail.com; jicong@ufl.edu

D.-S. Shiau

Research Department, Optima Neuroscience, Alachua, FL, USA
e-mail: dshiau@optimaneuro.com

C. Sackellares

Research Department, Optima Neuroscience, Gainesville, FL, USA
e-mail: csackellares@optimaneuro.com

P.M. Pardalos

Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL, USA
e-mail: pardalos@ise.ufl.edu; pardalos@ufl.edu

Abstract

Interest in network sciences has greatly increased over the last decades. Network models were recognized as a crucial tool in analyzing dynamics of complex systems. Small-world networks are apparently ubiquitous in nature and possess some special properties which make them highly suitable to model the brain. In this survey the theory of small-world networks and their applications in computational neuroscience is reviewed. First formal definitions and properties of small-world networks and how different network characteristics relate to each other are discussed. An overview of brain network models including anatomical network models and functional network models is given, and general steps of building these models are discussed. Different techniques of brain data acquisition are considered and a brief overview of computational methods of analysis these data is given. Finally, research works that have been done on neural data and reported existence of small-world networks in brain are reviewed.

1 Introduction

Over the last decades, network models have been widely appreciated as an essential part in studying complex systems [4, 16, 97]. Networks provide information on which system components interact with each other putting aside the mechanisms of these interactions. The dynamics of the system is tightly connected to the structure of underlying network. In many cases it is easy to describe the behavior of tiny system components on a microscopic level, but much more challenging to understand complex macroscopic system processes. Studying the topological properties of system networks can provide an insight about the system's high-level structure and function.

The human brain is probably one of the most complex objects in nature. Although the basic mechanisms and function principles on single neuron level are understood, not much progress has been achieved in understanding global brain processes and function. Its ability to gather and integrate information from numerous sources and generate responses is tightly connected to the complicated structure of neural connections network [18, 98]. Studying neural activity on the high level of interaction patterns between large neuron populations and segregated anatomical regions may help to understand fundamental laws of brain function.

It has been found that many complex networks in physics and biology follow common organizational principles [5, 9, 87]. These networks are densely connected on the local (neighborhood) level, but at the same time have certain number of “long-range” connections linking nodes, which otherwise would be far apart. Watts and Strogatz in their seminal paper [104] called such networks small worlds, by analogy with small-world phenomenon described by Milgram in 1960s. After this first paper numerous studies have reported small-world properties in different kinds of networks, in particular, brain networks.

Small-world networks are attractive models of the brain, since they tend to optimize the efficiency of parallel information transfer through the network, and at the same time remain fault tolerant, that is, resilient to damage of single nodes or connections [10, 46, 53, 55, 61]. The local density of brain connections is rather intuitively expected, since long wiring connections are expensive from material and energy points of view [21, 23, 25, 50]. However, computational studies showed that brain does not exhibit purely local connectivity structure, having certain number of consistent “long-range” connections between distant areas. This suggests that brain networks possess neither random nor completely regular structure. Multiple studies have shown that topology of brain networks at all scales has small-world attributes, and it is the main topic of this survey.

This chapter is organized as following. In Sect. 2 the quantitative definition of small-world networks and review of their properties are given. Two main approaches to quantify the small-world property of the network exist. These approaches were introduced by Watts et al. and Latora et al., which are fundamental in small-world networks theory, and therefore they will be reviewed in high details. In Sect. 3 brain anatomical and functional networks are reviewed, consider common approaches of constructing brain network models, and discuss some computational methods of processing real brain data. A review of studies reported small-world networks in brain is provided. In Sect. 4 the concluding remarks regarding small-world networks and their place in modern neuroscience are provided.

2 Small-World Networks: Models and Properties

The small-world phenomenon first was defined in social sciences an area where such networks naturally arise. In 1960 psychologist from Yale University Stanley Milgram has conducted series of experiments, where he tried to estimate average distance through the acquaintances between pairs of people in the world [63]. Results of his experiments were rather striking. He has found that average distance through the acquaintances between any two people is close to 6. Keeping in mind the total size of world’s population, this incredibly low number indicates that the social network of acquaintances has some very special structure. Milgram has called this finding a small-world phenomenon. Such short sequences of acquaintances between pairs of people are especially surprising, because social networks are known to have locally high-dense high-clustered structure, and most of the pairwise distances between people from different clusters are expected to be long.

Mathematically, Milgram’s findings can be formulated as low value of the average shortest path length between nodes in the dense, highly clustered network. Since in considered social network the connections are simply binary, two nodes are either connected or not; it is the topology of the network that makes it small world. Similar properties later were found in numerous social, economic, and biological networks. This gave rise to an interest in studying the properties and topology of small-world networks.

There are two main approaches in quantifying small-world network properties. One comprises evaluation of network characteristic path length and clustering coefficient, another focuses on quantifying network efficiency properties. Small-world networks have been shown to have low values of characteristic path length while maintaining high clustering coefficient. These properties can be reformulated in terms of one integrated measure of network efficiency: small-world networks have high values of network efficiency on both global and local levels. Below the formal definitions to these measures are given, and the relationships between them are discussed. Also, a review of models of network topology which give rise to small-world attributes is provided.

2.1 Characteristic Path Length and Clustering Coefficient

Consider a network $G = \{N, E\}$, where N is set of nodes, and E is a set of edges. Let d_{ij} be the length of the shortest path between nodes i and j through the connections in the network. This length computationally can be determined by variety of simple algorithms, like breadth-first search, or Dijkstra's algorithm. The characteristic path length L of the network G is defined as

$$L = \frac{1}{n(n-1)} \sum_{(i,j) \in E} d_{ij},$$

where n is a size of set N (number of nodes in the network).

Consider an arbitrary node i of the network G . Let G_i be a subgraph of G , containing nodes, adjacent to i . The local clustering coefficient C_i is defined as

$$C_i = \frac{\# \text{ of arcs in } G_i}{k_i(k_i - 1)},$$

where k_i is a number of nodes in G_i . Therefore, C_i is a fraction of arcs in subgraph G_i divided by maximum possible number of arcs, when G_i is a complete graph. The global clustering coefficient C is defined as a mean of local clustering coefficients:

$$C = \frac{1}{n} \sum_{i \in N} C_i.$$

The characteristic path length measures the typical distance between pairs of nodes, which is the global property of the network, while clustering coefficient measures the average “cliquishness,” or local connectedness of neighborhoods in the network, which is a local property. Characteristic path length and clustering coefficient are applicable to the connected binary networks. Two typical families of binary networks with simply expressed topology are regular networks or lattices and random networks. Regular networks can be thought of as networks, where nodes are placed on a grid and the neighboring nodes are connected by an edge.

Random networks, on the other hand, are the networks, where connections are thrown between nodes randomly according to uniform distribution. The topology of small-world networks lies somewhere between these two extreme cases. A simple model of binary networks exhibiting small-world topology was proposed by Watts and Strogatz [104]. The networks were generated from regular networks by random rewiring some of the edges, introducing degree of disorder into network topology. By changing the number of rewired edges, the networks can be tuned between regular and random states, passing through the range of intermediate topologies. For certain range of rewiring probability p , the resulting networks were highly clustered, like regular lattices, yet had small characteristic path lengths, like random graphs. Watts and Strogatz called these networks small worlds by analogy with the small-world phenomenon [104].

It turns out that these features are present in many real world networks. First such results were obtained by Watts and Strogatz in the same work [104], where they considered collaboration network of actors in films, the electrical power grid of the western United States, and the neural network of the nematode worm *C. elegans*. All three real networks of completely different nature exhibited small-world properties, having low characteristic path length and high clustering coefficients. This indicates that the small-world phenomenon is not merely a curiosity of social networks nor an artifact of an idealized model, but is common in networks found in nature.

2.2 Network Efficiency and Cost

Characteristic path length and clustering coefficient are useful only when studying unweighted (binary) connected networks with relatively small number of edges. Clearly, it does not cover whole variety of real networks. In most cases the information whether there is or there is not a connection between two nodes is poor and not sufficient in order to describe the network with suitable precision. Back to the network examples, considered above, in the case of films actors, the binary approximation only provides knowledge whether actors participated in some movie together, or not, and does not take into account the fact that two actors that have done ten movies together are in a much stronger relation than two actors that have acted together only once. In the case of the neural network, the number of junctions connecting a couple of neurons can vary a lot. A weighted network is more suitable to describe such system and can be built by setting the length of the connection proportional to the inverse number of junctions between nodes. The electrical power grid network is clearly a network where the geographical distances play a fundamental role.

The generalization on arbitrary networks can be made by using measure of so-called network efficiency. Network efficiency was introduced by Latora and Marchiori in [54, 55], where they proposed to express small-world properties of networks through its information transfer capabilities. The efficiency of transmitting information between two nodes in the network is inversely proportional to the distance between them (length of shortest path between given nodes). It turns out that small-world properties of the network can be formulated in terms of

the efficiency, which plays the role of characteristic path length L and clustering coefficient C . The formal definition follows.

Consider a network G as a weighted and possibly non-connected and non-sparse network. Let $(a)_{ij}$ be the adjacency matrix, containing the information about the existence of edges between nodes, and defined as for the topological network as a set of numbers $a_{ij} = 1$ for pairs of nodes (i, j) , connected by an edge, and $a_{ij} = 0$ for all other pairs. Let $(l)_{ij}$ be the matrix of weights associated with each connection, where element l_{ij} contains information about the weight of edge between nodes i and j . For the weighted network, the shortest path length d_{ij} between nodes i and j is defined as a minimum sum of weights of edges over all the possible paths in the graph from i to j . Suppose that every node in the graph sends information through the edges. One can assume that the efficiency e_{ij} of information transfer between nodes i and j is proportional to the inverse of the shortest path length from i to j , that is, $e_{ij} \sim \frac{1}{d_{ij}} \forall i \neq j$. In general this is a reasonable approximation, although in particular cases other relations can be used [55]. The average efficiency of the graph is defined as

$$E(G) = \frac{1}{N(N-1)} \sum_{i \neq j \in G} \frac{1}{d_{ij}}.$$

The value of E given by this formula may vary in the range $[0; \infty)$. The value of 0 is achieved in a completely disconnected network or in the network where lengths of all paths tend to infinity. The value of E tends to ∞ in weighted networks where lengths of shortest paths tend to 0. It is possible to normalize it by considering the extreme case of complete graph which has all $\frac{N(N-1)}{2}$ edges. In this case the information is being transferred in the most efficient way, and $d_{ij} = l_{ij} \forall i \neq j$. If one will divide actual E by E_{complete} of the complete graph, one will obtain a value from $[0; 1]$ range. Although the value 1 can be reached only when every pair of nodes in the network are connected with an edge, real networks often possess high values of normalized efficiency E [55].

The network efficiency can be measured on both local and global levels. The global meaning is defined as

$$E_{\text{global}} = \frac{E(G)}{E(G_{\text{complete}})},$$

where the normalization term $E(G_{\text{complete}})$ is an efficiency of the extreme case of complete graph. Arguably, E_{global} plays role similar to characteristic path length L , and it tells us how efficient is network in parallel information transferring [55]. In the case of disconnected graph, $L = \infty$, while E_{global} always remains finite value.

Let G_i be a subgraph of nodes, adjacent to node i , not including i itself. The local network efficiency is defined as mean of global efficiencies for subgraphs G_i over all $i \in V$:

$$E_{\text{loc}} = \frac{1}{N} \sum_{i \in G} \frac{E(G_i)}{E(G_i^{\text{complete}})},$$

where G_i^{complete} is a complete graph containing same vertices as G_i . E_{loc} is a mean of local efficiencies in the graph, and therefore it plays a role similar to the clustering coefficient. The local efficiency E_{local} tells how fault tolerant the system is, that is, how efficient is the communication between neighbors of i , when i is removed. High values of E_{local} mean that if one will randomly delete nodes from the network, the connectivity of the network will not be affected strongly.

The generalized definition of small world built in terms of the characteristics of information flow at global and local level is following: a small-world network is a network with high E_{global} and E_{loc} , that is, very efficient both in global and local communication [55]. This definition can be applied both for unweighted and for weighted networks, as well as to disconnected and/or nonsparse networks.

In order to understand the relation of two approaches to describe small-world networks, it is necessary to study the correspondence between E and L, C . Although both of measures L and E are related to the efficiency of flow transferring through the network, one should note that $1/L$ measures the efficiency of sequential system, like a system where there is only one path of information going along the network, whereas E is a measure of parallel systems, where information is being transferred through all the edges in the network [55].

In certain cases measure L works reasonably: it can be shown that $1/L$ is a reasonable approximation of E_{glob} when there are not significant differences among the distances in the graph [55]. However, it fails, like every approximation, to properly deal with all cases. Consider the limit case, where a node is isolated from the system. For neural network, this can be interpreted, for example, as the death of a neuron. In this case, $1/L$ drops to zero, since ($L = +\infty$), but the overall efficiency of the system is not affected that strongly: in fact, the brain continues to work, as all the other neurons continue to exchange information; only, the efficiency is just slightly decreased, since there is one neuron less. This slight change is properly taken into account using E_{glob} .

Similar relationship exists between clustering coefficient C and local efficiency E_{loc} . It can be shown that clustering coefficient C , in the case of undirected binary graphs, is always a reasonable approximation of E_{loc} [55]. It can be concluded therefore that there are no two different kinds of characteristics to consider when describing a network on the local and global levels. Exists just one general concept: the efficiency of information transfer [55].

Another important property of a network is its cost [55]. This measure comes into play particularly when dealing with real systems, modeled by weighted networks. Intuitively clear that the more edges there are in the network, the higher is the network's efficiency. On the other hand, in real situations there is a price to pay for number and length (weight) of edges. In particular the rewired edges that cause the fast drop of characteristic path length value L in the Watts and Strogatz model connect at no cost nodes that would otherwise be far apart. In realistic model, such connections would be expensive to create, and in certain cases this cost could overcome the gain in network efficiency, provided by adding these connections.

Therefore, it is necessary to consider weighted networks, where weights would allow us to differentiate between different edges in the network and to introduce a measure to quantify the cost of a network.

Formally, the cost of a network G is defined as

$$\text{Cost}(G) = \frac{\sum_{i \neq j \in G} a_{ij}\gamma(l_{ij})}{\sum_{i \neq j \in G} \gamma(l_{ij})}.$$

The function γ here identifies the cost needed to create a connection of a given length l_{ij} . Note that the value of total cost is normalized by dividing it by the cost of the complete graph, which contains all the possible edges. Because of such a normalization, the γ function needs only to be defined up to a multiplicative constant, and the measure $\text{Cost}(G)$ takes values from the interval $[0, 1]$, the maximum value 1 is reached for G_{complete} , that is, when all the edges are present in the graph [55]. $\text{Cost}(G)$ is nothing, but the normalized number of edges $2K/N(N - 1)$ in the case of an unweighted graph. In many cases it is reasonable to take γ function as the identity function: $\gamma(x) = x$. In fact such function is applied in case of unweighted networks and in many cases of the real networks, where the cost of a connection is proportional to its length (e.g., Euclidian distance). In such cases the definition of the cost reduces to

$$\text{Cost}(G) = \frac{\sum_{i \neq j \in G} a_{ij}l_{ij}}{\sum_{i \neq j \in G} l_{ij}}.$$

Now, an economic small-world network can be defined as a network with high local and global efficiencies and low cost [55]. In order to illustrate the use of new quantities in some abstract and practical examples, Latora and Marchiori [55] have studied several models of real and randomly generated networks, starting from the original Watts and Strogatz model, and proceeding to the models of weighted networks. These models will be briefly reviewed here, since they are good illustrations to the discussed measures of small worlds.

The Watts and Strogatz [104] network model is binary, that is, weight of every edge is equal to 1. An initial regular network was rewired with certain probability p , giving rise to a range of networks with a certain degree of randomness, depending on value of p , in particular, small-world networks. Since the total cost of a network is proportional to the number of connections, the cost of rewired networks does not change. Latora and Marchiori [55] have found that for $p = 0$ (regular network), the global efficiency E_{global} was low, while the local efficiency E_{local} possessed high values. This means that regular networks are inefficient in global information transferring, but efficient on a local level. The situation was reverted for $p = 1$, which corresponds to a completely random networks. The local efficiency E_{local} was close to 0, and the global efficiency E_{global} assumed a maximum value. The economic small-world behavior appeared for the intermediate values of p . As in case with characteristic path length L and clustering coefficient C , introduction

of several long-range connections significantly increased global efficiency E_{global} , while the local efficiency E_{local} remained almost unaffected. Moreover, the behavior of functions $E_{\text{local}}(p)/E_{\text{local}}(0)$ and $E_{\text{global}}(0)/E_{\text{global}}(0)$ was very similar to the behavior of $L(p)$ and $C(p)$.

In the second model Latora and Marchiori [55] have proposed different from reported in [104] approach to generate artificial small-world networks. Starting from a set of nodes N and empty set of edges, the edges were randomly added to the network, until the network became connected. These networks were binary, like networks in the first model, but unlike those, the cost of the given networks was not constant and was growing monotonically with increasing of number of added connections. The given procedure was giving rise to small-world networks for values of cost $\sim 0.5\text{--}0.6$, but it failed to generate economic small-world networks, which are networks with small-world properties and small number of connections.

While in previous models the long-range connections (or short cuts) have been rewired at no change in cost, which is unrealistic approach in relation to real world networks, in this third model Latora and Marchiori [55] have introduced lengths of connections in order to address this limitation. They have used the same random rewiring procedure, but after each rewiring, the length of a rewired edge has been changed from 1 to 3, and in this way the total cost of the network was increasing. The network in this model therefore was weighted, with weights of connections representing lengths of edges between nodes. The small-world behavior in these networks was still present. For small values of p , the global efficiency E_{global} grew rapidly, while local efficiency E_{local} did not change much. However, unlike the previous two models, the function $E_{\text{global}}(p)$ was not simply monotonically increasing. Contrary to it, the cost of the network was monotonically increasing with rewiring probability p . At the range of p , where generated networks possessed small-world properties, the cost of these networks was low, which means the networks were not only small worlds but also efficient. In contrast to the second model, this generalized rewiring approach was able to create economical small-world networks, by only rewiring of the edges. Therefore, the structure (topology) of the network plays a relevant role not only in its efficiency but also in economical properties.

In order to build a more realistic model, Latora and Marchiori [55] took in consideration geometrical distances between nodes. In this fourth model, they have placed the nodes of the network around the circle on the plane and they have defined the length of the connection as a Euclidian distance between corresponding nodes. For the points on the circle, this distance is equal to

$$l_{ij} = \frac{2\sin(|i - j|\pi/n)}{2\sin(\pi/n)},$$

if one will assume that the nodes are placed around the circle according to their indices i and that the length of the arc between neighbor points is equal to 1. The results of analyses of networks generated by this procedure were consistent with the results from the previous model. There was a range of values of p , where global efficiency E_{global} and local efficiency E_{local} attained high values, while the cost was relatively low, indicating economic small-world properties of the network.

As a conclusion, results derived from last two models imply that the economic small-world properties are not only an attribute of the topological abstractions, but are also present in the weighted networks, where the physical distance plays role and the rewiring of connections has a cost [55].

Latora and Marchiori [55] have applied their measures to the networks of cortico-cortical connections in macaque and cat. The databases they have considered contained only information about existence of connections between cortical areas, therefore the networks were unweighted (binary). Both the macaque and cat networks turned out to be economical small worlds. They had high global efficiency, high local efficiency, and low cost, where cost was calculated using identity cost function $\gamma(x) \equiv x$. In both networks, the global efficiency E_{global} was similar to that in random network of the same size, and the local efficiency E_{local} was much higher than that in random networks, indicating high fault tolerance of the network. In fact, in both networks, the local efficiency was greater than the global one, meaning that the fault tolerance in these systems was privileged with respect to information transmission efficiency.

Latora and Marchiori [55] studied weighted network of neural connections of *C. elegans*, the only case of a nervous system completely mapped at the level of neurons and synapses. The neural system of *C. elegans* is a system with multiple connections, that is, two neurons can be connected with more than one edges (up to 72). Latora and Marchiori [55] modeled the presence of multiple edges as weights of connections. The weight of edge between nodes i and j in the network they have defined as inverse to the number of connections between corresponding nodes. In this case the cost of the connection could not be computed anymore with identity cost evaluating function. Instead, Latora and Marchiori [55] have set the cost of the connection equal to the number of edges between corresponding pair of nodes in initial neural system.

For comparison purposes Latora and Marchiori [55] first considered obtained network as a binary one, setting the weight of the connection to 1, whenever exists at least one edge between corresponding neurons, and 0 otherwise. The unweighted *C. elegans* network, similar to the cortical networks of macaque and cat, was an economic small world. Unlike the cat and macaque networks, the values of local and global efficiencies in this case were similar to each other, showing that in *C. elegans* neural system, no privilege is given to fault tolerance with respect to efficiency.

Finally, for complete weighted *C. elegans* network, Latora and Marchiori [55] also found economic small-world properties, with almost equal values of global and local efficiencies, and low value of cost.

3 Brain Networks

Earlier studies of the brain function have focused mainly either on local or distributed properties. Modern works shifted the attention to the structure and dynamics of large-scale neuronal networks [37, 86]. In this section several characteristics of brain networks are described, in particular their small-world properties.

With a few exceptions, most part of communication between nerve cells is performed through the physical connections. Sometimes, these connections may link cells separated by long distances. Signals generated by neurons and transmitted through these connections comprise a series of action potentials or spikes. The reception of an action potential at a synaptic junction launches numerous biochemical and biophysical processes, resulting in generating an output spike, transmitted along the neuron's axon. Neurons in the cerebral cortex form a dense network with thousands of input and output connections [86]. By quantitative estimations, the human cerebral cortex contains roughly 8×10^9 neurons and 7×10^{13} connections [65]. The total length of all connections in the human brain is estimated between 10^5 and 10^7 km. However, having this tremendous number of connections, brain networks remain very sparse with a fraction of present connections of around 10^{-6} . It turns out that these sparse brain networks are not random, but have special structure patterns. Neurons tend to connect mainly to the neighbor neurons forming local groups, and thus local connectivity in neural network is much higher than it would be expected in random network [12, 86, 91].

Neural brain networks can be analyzed at different level of scale [89]. At microscopic scale, neurons form special sets of connections called local circuits. At a higher level of scale, populations of neurons communicate through the bundles of connections, forming networks within single cortical areas. Connection patterns formed by these local networks are defined by specific task each area is responsible for. Finally, at an entire brain level, the large-scale networks are formed by interconnections linking different brain areas [86].

So far neural connectivity has been discussed, that is, structural (anatomical) brain networks formed by physical connections between neurons. Another important aspect of brain function is neural activity and dynamics of brain processes. Neural activity also can be modeled by a network, the so-called functional connectivity brain network [39, 85]. The connections in functional network are defined not by physical paths in the brain but by dependencies between neural signals in different parts of the brain. If two populations of neurons exchange information, the signal patterns produced by these populations are likely to be statistically dependent, for example, correlated in time. It is commonly recognized that in resting human brain exists consistent functional network between cortical brain regions (so-called default mode network) [14, 30, 38, 73, 75, 103]. It has been proposed that functional networks form a physiological basis for information processing and mental cognition [17, 18, 20, 62]. One of the questions arising here is how do structural and functional networks relate to each other? No doubts that patterns of neural activity are defined and shaped by neural connectivity, and functional networks in large part resemble structural brain networks [45, 84]. But sometimes functional networks can tell us more about processes in the brain than just map of physical neural connections. Certain brain areas, which are not connected directly, communicate with each other through other areas, and these communications are reflected in the functional network. Therefore, functional network comprises structural network, but also contains connections between those areas which are

not connected in structural network, but link to each other dynamically, exhibiting related neural activity.

It is natural to assume that if structural networks in large part define functional network, they should have similar topological properties. And in fact this was confirmed by multiple studies, which reported small-world topology in both groups of networks. Historically, first studies on brain network properties were performed on structural networks, in particular those of mammalian brain. Further in this section studies on structural and functional brain networks will be reviewed in separate subsections, since the approaches are different in these cases.

3.1 Structural Brain Networks

As was mentioned above, structural connectivity refers to the set of physical connections linking neurons in the brain. Analysis of structural connectivity patterns can be done at different levels of spatial scale, from local circuit level with focus on the patterns of synaptic connections between individual neurons to large-scale connection patterns with pathways linking different areas of brain. Such pathways may contain thousands of individual fibers [86]. Anatomical connectivity data is available mostly for large-scale pathways in mammalian species, which has been catalogued in several surveys [29, 35, 79–81, 107]. There were also created several online databases [24, 51, 95]. Recent advances in diffusion imaging and tractography methods provided data sets of human brain cortico-cortical connections at high spatial resolution [26, 41, 42, 56, 82].

On the single neuron level, Humphries et al. [47] have shown that neural system of the medial reticular formation of the brain stem is a small world.

The neural system of a worm *C. elegans* is the only neural system which was mapped completely at the level of neurons and synapses [1]. By computing clustering coefficient and characteristic path length, Watts and Strogatz [104] have shown that this network exhibits small-world topology. Latora and Marchiori [55] have confirmed this result by analyzing more general case of weighted network, where weights of connections corresponded to the number of physical links between neurons.

A compelling small-world effect was found in macaque cortex, macaque visual cortex, and cat cortex by Sporns and Zwi [88]. They have considered binary networks built based on structural connectivity matrices. For macaque visual cortex the network consisted of $N = 30$ vertices and $K = 311$ edges; for macaque cortex, $N = 71$ and $K = 746$; and for cat cortex, $N = 52$ and $K = 820$. For all three networks they have computed clustering coefficient C and mean shortest path length L . In addition, for every case, they have constructed random and regular networks with the same number of nodes and edges and computed scaled measures C_{scl} and L_{scl} by formulas

$$C_{\text{scl}} = \frac{C_{\text{original}} - C_{\text{random}}}{C_{\text{lattice}} - C_{\text{random}}};$$

$$L_{\text{scl}} = \frac{L_{\text{original}} - L_{\text{random}}}{L_{\text{lattice}} - L_{\text{random}}}.$$

For all three networks they have obtained high values of C_{scl} , like in regular lattices, and low values of L_{scl} like in random networks. Therefore, all three networks had low average path length between any two nodes while maintained high connectivity on local level. Sporns et al. [88] compared C_{scl} and L_{scl} to those of random and regular networks of the same in- and out-degree distribution for all nodes, and again found that C_{scl} was close to regular networks and L_{scl} was close to random networks. This showed that small-world properties of the network cannot be explained by degree distribution of the nodes. It is the global topology of connections what makes it a small world.

Sporns and Zwi [88] have reported a series of experiments on simulated neural networks. They have generated three types of networks, where for given set of nodes, they introduced intra-areal and inter-areal connections according to the specific probability distributions. More precisely, they generated “all local” network, where all connections were assigned according to Gaussian probability; “local plus uniform network,” where part of the connections was generated again by Gaussian probability and another part distributed uniformly among randomly chosen nodes; and “local plus long-range” network, obtained from “all local” network by relinking part of the connections according to another Gaussian profile. Generated “all local” networks did not possess small-world properties. At the same time, both “local plus uniform network” and “local plus long-range” networks were small world, especially “small worldness” was strongly expressed in the latter ones. In both cases, introduction of a few long-range connections drastically dropped the mean shortest path length, while clustering coefficient remained almost unchanged.

Another research on mammalian structural neural networks was performed by Hilgetag et al. [44]. They have applied clustering techniques in order to find groups of densely connected areas in macaque cortex, macaque visual system, macaque somatosensory-motor system, and cat cortex. The two approaches used in order to find clusters were evolutionary algorithm (OSA) and nonparametric cluster analysis with multidimensional scaling of data matrix (NPCA). Both approaches gave consistent results, indicating that considered neural data is shaped by several dense clusters connected by few long-range connections. The evolutionary algorithm was generating partitions of the data into clusters via minimization of certain cost function. Different trials of algorithm started from different random initial partitions, but produced very close final “optimal” partitions, which indicate existence of well-expressed clusters in the data. The comparison between clusters obtained by this algorithm and NPCA showed high level of consistency, the similar analysis they have conducted on random matrices, generated by shuffling the entries of initial data connectivity matrices. In this way, the overall number of connections and degree distribution were preserved, but the topology of the network changed. The consistency between clustering analysis results by OSA and NPCA was much lesser in this case. Also, the clustering coefficients for all four considered networks were much greater in initial data than in randomly shuffled ones.

An interesting and insightful view on small-world neural networks was proposed by Sporns et al. [89]. In this study, Sporns et al. considered brain activity as multidimensional stochastic process, which can be characterized by statistical measures such as entropy, mutual information, and statistical complexity. Mainly, they have focused on measures of entropy and integration, which capture overall degree of statistical independence and dependence, respectively, and complexity, which measures the presence of functional segregation and functional integration in the system. First part of the study consisted in investigation of artificially created networks, which possess high values of certain network measures. The structural properties of such networks were studied, and apparent similarities were found with real neural networks. More detailed, the pseudo-evolutionary procedure was applied, which from initial random networks generated networks with higher value of entropy, integration, or complexity by certain rewiring or “mutations.” The network with highest value of correspondent measure was stored during the procedure. The number of nodes n and number of edges k were preserved during this procedure. Also, at each step, the network was required to be strongly connected; if mutation procedure was partitioning network into several components, such networks were discarded from the procedure. Therefore, in some sense, this procedure is similar to approximate optimization algorithms, like local search algorithms, and genetic algorithm. The solution space here is a set of possible networks of n nodes and k edges, and the objective function is either entropy, integration, or complexity. If one will assume that activity in n brain regions can be described as a Gaussian multidimensional stationary stochastic process, then the global measures of network dynamics can be derived from the network’s covariance matrix COV. The formal definitions of these measures are following. For a given system X , the entropy of the system $H(x)$ is defined as

$$H(X) = 0.5 \ln((2\pi e)^n |\text{COV}(X)|),$$

the integration of the system $I(X)$ is defined as

$$I(X) = \sum_i H(x_i) - H(x),$$

and the complexity of the system $C(X)$ is defined as

$$C(X) = H(X) - \sum_i H(x_i | X - x_i).$$

Here, $\{x_i\}$ is a set of individual components of the system X , and $|\cdot|$ denotes matrix determinant.

For all three measures the procedure of network selection converged fast for low rewiring rates, starting from random initial network. The patterns observed in the resulting networks were robust in relation to the procedure parameters and initial network characteristics n and k . Each simulation of network selection for

each network measure gave networks with different fine structure details, but with common structural motifs. In contrast, the selected networks were structurally completely different for different network measures. While all the networks had constant n and k and were strongly connected, even casual visual inspection showed that their connection patterns differed significantly. Random network apparently did not have any topological order. Networks selected by maximizing entropy $H(X)$ contained mostly connections linking nodes spread out without any evident local grouping (clustering). This structure is in consistence with network dynamics, characterized by high statistical independence of units. Networks with high integration $I(X)$ consisted of a central cluster of dense connections with a few outsiders, connected to the main cluster, and weakly connected to each other. This pattern is consistent high value of statistical dependence of the entire system. Increasing the number of connections k increased the size of main cluster, without changing the overall pattern. Finally, networks obtained by maximizing complexity $C(X)$ consisted of several clusters of densely connected nodes, with a few number of reciprocal bridges between clusters. This networks are characterized by presence of functional segregation and functional integration at the same time.

In order to investigate the relation between generated by evolutionary procedure networks and classical small-world networks, Sporns et al. [89] computed characteristic path length, clustering coefficient, and cost of these networks and compared them to random networks, which were used as a starting point for the generating procedure. The cost was estimated as a summation of weights of all connections, where weight of the connection between nodes A and B was defined to be proportional to the distance between corresponding to A and B columns in network adjacency matrix. In some sense this function of cost is similar to the one computed as a summation of Euclidian lengths of all connections, when nodes are placed around the circle according to their indices (node 1 is placed next to nodes 2 and n , node 2 is placed next to node 1 and node 3, and so on).

As expected, random networks were characterized by small value of characteristic path length L and small value of clustering coefficient C . Networks with high entropy $H(X)$ had similar characteristic path length as random networks, but even lower values of clustering coefficient C . In contrast, networks with high integration $I(X)$ had high values of characteristic path length and very high values of clustering coefficient C . Finally, networks with high values of complexity $C(X)$ possessed small values of characteristic path length and high values of clustering coefficient C and surprisingly low values of cost, even lower, than those in random networks. Therefore, networks generated by maximizing the complexity $C(X)$ were economical small worlds.

The characteristic path length, clustering coefficient, and cost of neural networks of cortical connections in macaque monkey and cat turned out to be very similar to those in generated networks with high complexity. Both network of macaque visual cortex connections and network of cat cortex connections appeared to be economic small worlds. The structure of these networks also resembled greatly the structure of artificial networks with high complexity. Finally, the complexity of both networks,

when they were modeled as dynamic systems, turned out to be high. This is due to their anatomic organization into distinct groups of areas.

Sporns et al. [89] produced several modified networks, by random rewiring of some part of connections of neural networks. In all cases resulting networks had significantly decreased value of complexity. This suggests that configurations of neural networks are close to the optimal with respect to network complexity. Therefore, the complexity is another quantifier of network small-world properties.

Human structural networks were studied by He et al. [43], where they have used diffusion MRI to measure brain cortical thickness. Cortical thickness reflects the size, density, and arrangement of neuron cells, and interregional associations in cortical thickness reflect connectivity patterns in the human brain [43, 58, 105]. The entire cerebral cortex was divided into 54 areas, and regional cortical thickness was measured in every region. The pairwise correlations in regional thickness across 124 subjects were estimated and correlation matrix was created. The values of correlation were further thresholded, and resulting binary matrix was considered as an adjacency matrix of an undirected network. The resulting network exhibited small-world properties with clustering coefficient more than twice greater than the one of random network with corresponding number of nodes and connections.

At a finer scale the human brain structure was studied by Hagmann et al. [42] via diffusion MRI tractography. Their approach, based on constructing of three-dimensional curves of maximal diffusion coherence, allowed to construct networks with 500–4,000 nodes; each node corresponded to a region of brain of millimeters scale. These regions were connected with a number of fibers, which were modeled by a weight of corresponding connection in the network. In order to examine topological properties of resulting network, Hagmann et al. have studied binary network of the same size in which two nodes were connected, if they had a connection with positive weight in the original network. As in previous study on large-scale anatomical network, this network was a small world with high clustering coefficient and low characteristic path length.

Taken together discussed studies indicate that small-world topology is an inherent property of a structural organization of neural connections at different levels of scale. Consistent findings in different mammalian species and human suggest that small-world structure comprises fundamental principles in organization and evolving of neural networks. Economical properties of small-world networks may give an explanation to this fact. As it was discussed in previous sections, small-world networks are efficient in information transfer at both global and local levels and at the same time are resilient to errors and damages. In light of this one can argue that neural networks may have developed small-world topology during evolution process. Although small-world topology is relatively general property of a network and many networks with different structural patterns may have similar small-world characteristics, it gives an understanding of how is brain organized on global level which in turn provides an insight about brain functional properties.

3.2 Functional Brain Networks

Functional connectivity refers to the patterns of temporal statistical dependencies, such as correlation, that exist between distinct groups of neurons. Such temporal dependencies are assumed to be the result of neuronal interaction along anatomical connections. There are different ways to estimate the level of dependence between neural elements. Although temporal correlation is most widely used, other measures can also be applied [22, 86]. A consistent functional network has been revealed in human brain in resting state. This network was called a default mode network [73], and later was confirmed by multiple studies [27, 28, 32, 60, 106].

There are several techniques of measuring neural activity in the brain, and resulting functional networks depend highly on which technique was used. However, the general steps in constructing functional networks are rather common [87]. First, it is necessary to define nodes of the network. Often nodes are chosen as distinct anatomical regions of brain or electrodes (sensors). Then, the measure of association between nodes should be estimated. In most cases the measure of similarity between corresponding signal time series is used (in case of anatomical regions parcellation, averaged time series in each region is computed). Often, a threshold is applied to similarity values in order to produce binary matrix of associations between each pair of nodes. This matrix is then treated as adjacency matrix of a functional connectivity network, and the network properties are further studied.

Another step in data analysis, shared by nearly all brain network studies, is the decomposition of signals into components at different frequency bands. In fact, all brain monitoring technique measurements contain significant portion of nonneural sources of fluctuations. Patient's head movements during recording session, respiratory and heart rhythms, etc. may affect significantly the data quality and the results of the study [19]. In order to localize and discard these effects, frequency analysis is applied. The signal is being decomposed into several components, and the components containing most portion of noise are being removed.

This frequency analysis can be performed by applying either Fourier transform or wavelet transform. These techniques are very common in neural data studies, and therefore each of them will be discussed in a little more details.

The Fourier transform images a signal from time space into frequency space. If the original signal depends on time, its Fourier transform depends on frequency and thus is called the frequency domain representation of the signal. In fact, the Fourier transform decomposes a signal into oscillatory functions (sines and cosines). Mathematically, for every real number ξ , a Fourier transform of a function f is defined as

$$A(\xi) = \int_{-\infty}^{+\infty} f(x)e^{-2\pi i x \xi} dx.$$

When the variable x represents time, the variable ξ represents frequency [70].

However, this classical definition of continuous Fourier transform is not very practical when dealing with real data. Real observations are always discrete, and the number of observations is finite. Therefore, continuous transform is not directly applicable. In practice, discrete Fourier transform (DFT) is used. Consider a sequence $a_t, t = (0, \dots, N - 1)$ of N variables. Its DFT is defined as a sequence:

$$A_k = \sum_{t=0}^{N-1} a_t e^{-i2\pi tk/N}, \quad k = 0, \dots, N - 1.$$

Here A_k is associated with frequency $f_k = k/N$. There was developed group of algorithms called fast Fourier transform (FFT) for fast computing Fourier transform of real sequences. These algorithms will not be discussed here; interested reader is referred to [66, 69] for further information.

To the certain degree wavelet analysis is a generalization of the Fourier analysis, where instead of sines and cosines so-called wavelet functions are used. Essentially, a wavelet is a “small wave,” that is, a function, which grows and decays in a limited time period. Strictly speaking, a wavelet is any function ψ defined over the $(-\infty, \infty)$ which satisfies following two conditions:

$$\int_{-\infty}^{+\infty} \psi(u) du = 0;$$

$$\int_{-\infty}^{+\infty} \psi^2(u) du = 1.$$

A continuous wavelet transform of a function f is defined as

$$W(\lambda, t) = \int_{-\infty}^{+\infty} \psi_{\lambda,t}(u) du,$$

where $\psi_{\lambda,t}(u) = 1/\sqrt{\lambda}\psi(\frac{u-t}{\lambda})$. Here t refers to time, and λ is associated with frequency [70].

For a discrete signal, a partial discrete wavelet transform (DWT) is defined as following. Let X be a vector of N components $X_t : t = 0, \dots, N - 1$, where the sample size N is an integer multiple of 2^{J_0} . The DWT of level J_0 of X is an orthonormal transform given by $W = WX$, where W is an N dimensional vector of DWT coefficients, and W is an $N \times N$ real matrix defining the DWT. In case when $N = 2^J$ and $J_0 = J$, we obtain a full DWT. The DWT coefficients and matrix W can be partitioned, such that $W = (W_1, \dots, W_{J_0}, V_{J_0})$, where W_j is a vector of wavelet coefficients corresponding to the scale $\tau_j = 2^{j-1}$. The vector X can be obtained from W as

$$X = W^T W = \sum_{j=1}^{J_0} W_j^T W_j + V_{J_0}^T = \sum_{j=1}^{J_0} D_j + S_{J_0},$$

which defines a multiresolution analysis (MRA) of X , that is, an additive decomposition by N -dimensional vectors $D_j = W_j^T W_j$, each of which corresponds to a particular frequency band [70]. A scale-dependent (or frequency-dependent) ANOVA can be computed based on decomposition:

$$\|X\|^2 = \sum_{j=1}^{J_0} \|W_j\|^2 + \|V_{J_0}\|^2.$$

Wavelet coefficients at all scales are also localized in time. In contrast, coefficients of DFT are not localized in time in any meaningful sense [70]. This is an advantage of wavelet analysis compared with Fourier analysis when dealing with nonstationary, non-smooth time series, which cannot be precisely represented by global transform, such as DFT.

For practical needs, like in case with discrete Fourier transform, fast computational algorithms were developed for obtaining wavelet coefficients. For further details, reader is referenced to [70].

3.2.1 Networks Derived from Electrophysiological Brain Data

Electrophysiological brain monitoring techniques (electroencephalogram (EEG) and magnetoencephalogram (MEG)) record electromagnetic potentials generated by large neuronal populations. EEG and MEG directly record signals generated by neural activity and have high temporal resolution. Their spatial resolution, however, is relatively low, and neither of these techniques allows precise reconstruction of location of sources of recorded signals. Therefore, EEG and MEG signals in most cases are not being associated with anatomical brain regions [87]. Depending on the location depth of the EEG electrodes, EEG signals may record neural activities on the scalp, cortex, or in the brain. Scalp EEG is widely used in research and clinical practice because of its portability, cost-efficiency, and easy setup.

EEG is used to study a variety of brain-related physiological phenomena including sleep [33, 36, 48], traumatic brain injury [100], brain machine interface [96], epilepsy [6, 57, 108, 109], anesthesia [49], cognition [11], Alzheimer's disease [31], and schizophrenia [67]. One of the salient characteristics of EEG data is the rhythmic oscillation. Therefore, frequency analysis is commonly used to preprocess EEG signals. It is broadly appreciated that different strengths of frequency components reveal different brain states [31, 34, 40, 68, 71]. During EEG signal analysis, filtering is usually implemented to remove artifacts or focus on the frequency bands of interest. Prior to network analysis based on EEG data, the artifacts removal procedures should be applied. The common input from artifact would likely result in the existence of spurious association among channels. In order to obtain EEG signals within a certain range of frequencies, apart from conventional fast Fourier transform-based filtering techniques, some studies implemented wavelet

transform to extract components with more specific time-frequency resolution. For example, Palva et al. [68] used Morlet wavelets [70] to filter EEG data into 36 frequency bands before constructing the oscillatory phase synchronized network.

At the network construction step, it is natural to designate EEG channels as vertices of a network and edges between vertices to associate with functional connections between corresponding groups of neurons (brain areas around electrodes). One would expect a larger correlation between two EEG channels when there is an edge between them. Most of the studies using network theoretic analysis on EEG data assume that statistical interdependencies between EEG time series reflect functional interactions between neurons of brain regions [34]. There are many statistical metrics computing the degree of association between time series. Some of the most used metrics in EEG functional connectivity studies are introduced below.

Phase-Locking Value

Phase-locking value (PLV) is a statistic which measures the frequency-specific synchronization between two neuroelectric signals [52]. This is a method focusing on the phase information of time series and is different from coherence, which gives the interrelation of both amplitude and phase between signals. The PLV is calculated by first extract the instantaneous phase information of signals through either wavelet transform [52] or Hilbert transform [64]. Both methods lead to similar result on real world EEG data [72]. When doing the wavelet transform on a signal $x(t)$, a wavelet function, $\psi(t)$, is firstly chosen. Gabor and Morlet wavelet function both have been applied on EEG data [59]. Then, the wavelet coefficient time series, $W_x(t)$, can be computed as a convolution of $x(t)$ and $W_x(t)$:

$$W_x(t) = (\psi \circ x)(t) = \int \psi(t')x(t' - t)dt' = A_x^W(t) \cdot e^{\phi^{W_x(t)}}.$$

The phase of time series, $\Phi(t)$, can be computed as follows:

$$\Phi^{X_i}(n, s) = \arctan \frac{\text{imag}(W^{X_i}(n, s))}{\text{real}(W^{X_i}(n, s))}.$$

When doing the Hilbert transform, the phase information $\Phi(t)$ of a signal $s(t)$ is

$$\Phi(t) = \arctan \frac{s(\tilde{t})}{s(t)},$$

where

$$s(\tilde{t}) = \frac{1}{\pi} \text{p.v.} \int_{-\infty}^{+\infty} \frac{s(\tau)}{t - \tau} d\tau$$

is the Hilbert transform, and p.v. denotes the Cauchy principal value. The relative phase, $\phi 1, 1(t)$, can be calculated as

$$\phi_{1,1}(t) = \phi_a(t) - \phi_b(t) = \arctan \frac{\tilde{s}_a(t)s_b(t) - s_a(t)\tilde{s}_b(t)}{s_a(t)s_b(t) + \tilde{s}_a(t)\tilde{s}_b(t)}.$$

The subscript 1, 1 signifies that the relative phase relationship $n\phi_a(t) - m\phi_b(t)$ considers only the case where $n = m = 1$. Finally, one can define the PLV as following:

$$\text{PLV} = \left| \frac{1}{N} \sum_{j=0}^{N-1} e^{\phi_{1,1}(j\delta t)} \right|.$$

PLV was used in a study by Dimitriadis et al. [34] to construct the functional connectivity networks from 30-electrode EEG data. They utilized surrogate data to generate baseline distribution of random PLVs and then determined the functional connections (edges) if there was a significantly different ($p < 0.001$) PLV of a specific pair [34]. The surrogate data is generated by permuting the order of trials of one signal for many times [59].

Synchronization Likelihood

Synchronization likelihood (SL) is a statistic measuring the nonlinear similarity between time series. SL offers an extended perspective of correlation and is not limited by linear relationship only, like, for example, coherence, which renders only the linear correlation as a function of frequency [92]. Consider a time series $X_{k,i}$ with $k = 1, \dots, M$ denoting a channel number and $i = 1, \dots, N$ denoting time index. The computation of SL first requires reconstructing $X_{k,i}$ using the method of delays [99]:

$$X_{k,i} = (x_{k,i}, x_{k,i+1}, x_{k,i+2l}, \dots, x_{k,i+(m-1)l}),$$

where l is the lag and m is embedding dimension. A measure $H_{(i,j)}$ is then defined to measure the number of channels $X_{k,i}$ and $X_{k,j}$ that are closer than a threshold $\varepsilon_{k,i}$:

$$H_{i,j} = \sum_{k=1}^M \theta(\varepsilon_{k,i} - |X_{k,i} - X_{k,j}|).$$

Finally, SL is obtained as

$$S_{k,i} = \frac{1}{2N} \sum_{j=1}^N S_{k,i,j},$$

$$S_{k,i,j} = \frac{H_{i,j} - 1}{M - 1}, \text{ when } |X_{k,i} - X_{k,j}| < \varepsilon_{k,i}, S_{k,i,j} = 0 \text{ otherwise.}$$

One can see that $S_{(k,i)}$ is a measure describing how well channel k at time i is correlated to all other $M - 1$ channels. Ponten et al. [71] used SL measure as the connectivity metric in their study on epilepsy.

Nonlinear Independence

Nonlinear independence (NIM) estimates strength of functional coupling between two time series embedded in their respective phase spaces [7, 72]. Consider time series $x(t)$ and $y(t)$. One can reconstruct the delay vectors $x_n = (x_n, \dots, x_{n-(m-1)\tau})$ and $y_n = (y_n, \dots, y_{n-(m-1)\tau})$, where m is the embedding dimension and τ is a time lag. Let $r_{n,j}$ and $s_{n,j}$, $j = 1, \dots, k$ be time indices of the k nearest neighbors of x_n and y_n correspondingly. For each x_n the mean square Euclidian distance to its k neighbors is computed,

$$R_n^{(k)}(X) = \frac{1}{k} \sum_{j=1}^k (x_n - x_{r_{n,j}})^2,$$

and the Y-conditioned meant square distance:

$$R_n^{(k)}(X|Y) = \frac{1}{k} \sum_{j=1}^k (x_n - x_{s_{n,j}})^2.$$

The NIM $N(X|Y)$ is then computed as

$$N^{(k)}(X|Y) = \frac{1}{N} \sum_{n=1}^N \frac{R_n(X) - R_n^{(k)}(X|Y)}{R_n(X)}.$$

The asymmetry of NIM ($N(X|Y) \neq N(Y|X)$) is the main advantage of this nonlinear measures since it can deliver directional information about nodes in network. In the study done by Dimitriadis et al. [33], both SL and NIM were used to quantify the extent of association between nodes. Studying the sleeping EEG data from ten healthy subjects, they have found that SL and NIM are weakly correlated and thus should be considered to reveal complementary information about the brain functional connectivity.

Phase Lag Index

Phase lag index (PLAI) quantifies the asymmetry of the distribution of instantaneous phase differences between two time series [94]. Suppose one has calculated the relative phase time series, $\phi_1, 1(t)$, then the PLAI is defined as $|\langle \text{sign}[\delta\phi(t_k)] \rangle|$. PLAI values range from 0 to 1. Values greater than 0 suggest the existence of phase locking to some extent, and value equal 0 signifies no coupling between time series. One of the benefits of PLAI measure is that it rules out the synchronization in time series caused by instantaneous volume conduction or a common source propagating spurious synchronization.

The existence of small-world structure in the cortex of a human brain has been observed through a number of studies [12, 13, 36, 90]. An aging study [40] found that the clustering coefficient and the value of characteristic path length were both lower in the elderly compared to the young subject group. This implies that the

brains of the elderly subject group appear to be closer to random networks. This study confirmed analogous findings in fMRI data [2].

Group of studies analyzed changes in small-world brain architecture caused by neural disorders [93]. Ponten et al. [71] conducted a study on network analysis using intracerebral EEG recordings from patients having drug-resistant mesial temporal lobe epilepsy. They have found an increase of clustering coefficient in the lower frequency band (1–13 Hz) and an increase in the characteristic path length in alpha and theta bands during and after a seizure, comparing to interictal recordings [71]. This implies that the functional brain network changes from a more random organization to a small-world structure. In an Alzheimer's disease (AD) study conducted by de Hann et al. [31], EEG data from patients with AD were compared with those in control subjects. All AD patients showed small-world network traits in their functional networks in all frequency bands, but only in beta band these traits were statistically significant. Palva et al. [68] conducted a study on magnetoencephalogram (MEG) and EEG data recorded while subjects were performing visual working memory (VWM) tasks. The study recorded both MEG and EEG from 13 healthy subjects and applied PLV as the measure of association between time series. The results showed that small-world network structure appeared dynamically during the VWM task execution within alpha and beta frequency bands. Moreover, the small worldness was dependent on the VWM memory load. Van Dellen et al. [102] have analyzed interictal electrocorticography (ECoG) of 27 patients with pharmacoresistant temporal lobe epilepsy (TLE). The cluster coefficient and small-world index were both negatively correlated with TLE duration in the broad frequency band (0.5–48 Hz). This may result from the accumulative damage on the brain tissue due to intermittent excessive epileptic discharges for a long time. The optimal balanced small-world structure has been impaired to form a more randomized one as the TLE duration goes on. An interesting model work by Rothkegel and Lehnertz [76] revealed co-occurrence of local wave patterns and global collective firing in a two-dimensional small-world network. Boersma et al. [15] did a study on resting state EEG in the developing young brains. They performed resting state eyes-closed EEG recording (14 channels) from 227 children (age 5–7 years). They have found evidences that the clustering coefficient in functional network derived from alpha band was increasing with age. Also, characteristic path lengths were increasing with age in all frequency bands. This suggests that a brain shifts from a random toward more ordered small-world-like configurations during the maturation. Interestingly, girls showed higher mean clustering coefficients in alpha and beta bands compared with boys. In several studies the small-world network analysis was performed on data of patients with schizophrenia. Rubinov et al. [77] recorded resting state scalp EEG from 40 subjects with a recent first episode of schizophrenia and another 40 healthy matched controls. Nonlinear interdependences were identified from bipolar derivations of EEG data, and weighted connection networks were generated. Both groups possessed networks with a small-world topology, but networks in the schizophrenia group had lower clustering coefficient and shorter characteristic path lengths, indicating more random network structure. The randomization of brain functional network (loosing ordered small-world structure) may be the reason for

cognitive and behavioral disturbances in schizophrenia patients. Another similar study [67] focused on scalp EEG data from 20 schizophrenia patients and 20 healthy controls (age and sex matched) recorded while patients performing working memory tasks. The data were analyzed using conventional coherence. Afterward, the values of coherence were further thresholded, producing binary connectivity networks. The results confirmed that schizophrenia patients possess more random neuronal network organization, especially in the alpha frequency band.

3.2.2 Networks Derived from fMRI Data

Functional magnetic resonance imaging (fMRI) measures fluctuations in the blood oxygen level-dependent (BOLD) signal, which is related to local metabolism and neuronal activity [8, 74, 83]. The BOLD signal provides only indirect measure of neural activity, since coupling between blood oxygenation and neuronal activity has complex nature and may vary across the brain regions. fMRI technique provides spatial resolution at millimeter scale, but has low temporal resolution capabilities due to physical limitations of underlying processes [87].

Group of Cambridge scientists studied brain functional network with nodes defined according to the anatomical MNI brain template [101]. This template partitions brain into 90 regions of interest. Salvador et al. [78] analyzed fMRI data of 12 healthy patients in resting state. Interregional partial correlation of fMRI signals were then estimated, assuming that time series can be considered as jointly Gaussian stationary multivariate stochastic process. The partial correlation matrices from individual patients were then averaged in order to obtain healthy group mean functional connectivity matrix. Salvador et al. found that the most significant connections were intrahemispheric and often local, involving regions in the same lobe or closely adjacent.

The obtained functional network possessed small-world topology with clustering coefficient roughly twice larger than in random network with the same number of nodes and connections, and characteristic path length similar to that in random network.

Achard et al. [3] have studied neural networks in human brain in resting state at macroscopic level based on functional magnetic resonance imaging (fMRI) data. The network connections between brain regions were defined based on temporal correlations between fMRI time series in these regions. Regional mean time series was estimated by averaging times series over all voxels in each region. Pairwise correlations between mean time series were estimated, and the network was constructed with connections based on these correlations. High correlations were assumed to indicate strong connections, whereas low connections were considered a sign of weak connectivity. Achard et al. [3] used wavelet transform in order to decompose time series into six scales, corresponding to different frequency ranges. The reason for this was that initial signal apart from neural activity contained also nonneural sources of correlations, such as cardio- and respiratory rhythms, which are mainly presented at high frequency ranges, and therefore analysis at different scales can help to emphasize the neural contribution to the signal. The pairwise correlations between wavelet coefficients of mean time series were computed for

all six scales. The correlations then were thresholded for each scale, giving six binary connectivity matrices. These matrices were then considered as adjacency matrices for the binary networks, where nodes correspond to brain regions. Value of 1 of (i, j) th element of the matrix meant that there is a connection between nodes i and j .

In this work, in order to analyze small-world properties of neural network, Achard et al. [3] used technique proposed by Watts and Strogatz [104]. They have computed characteristic path length of whole brain network L , the clustering coefficient C , and compared them to those in random networks with identical size and mean nodes degree. They have found that the functional connectivity was the most salient in the frequency range of 0.03–0.06 Hz. The threshold for the correlation matrices was chosen in such a way that the resulting networks would be sparse, while still connected, like in Watts and Strogatz model. The criteria for the threshold was chosen such that resulting networks had $k \sim \log(n)$ edges, where n is the size of the network. The results suggested that the brain neural networks possess small-world properties at all six scales of wavelets. The characteristic path length in these brain networks was similar to that of random networks, while the clustering coefficient was about two times greater.

Achard et al. [3] have studied how small-world properties of neural networks depend on threshold value R for correlation matrices. They found that small worldness decreased monotonically with R , although the small-world properties, characterized by $L \sim L_{\text{random}}$, $C > C_{\text{random}}$, were present at all scales at a range of values of R .

The degree distribution of the network was not power law, and the networks themselves were not scale-free. The best-fitting distribution was exponentially truncated power law, which is

$$P(k) = k^{\alpha-1} e^{-k/k_c},$$

where estimated parameter α was 1.80 and cutoff parameter k_c was 5. In order to examine the resilience of brain functional network to random and targeted attacks, Achard et al. [3] conducted series of simulation experiments. They randomly removed nodes one by one from the network, including all the connections, adjacent to those nodes. After each removal the size of largest cluster in the network and the characteristic path length were recalculated. In this way random attacks (or errors) on the network were modeled. In the second set of experiments, the nodes with largest degree were being selectively removed, simulating the targeted attacks on the network. The results showed that the brain networks were approximately as resilient to random attacks, as random networks, but significantly more resilient to the targeted attacks. The most damaging attacks were generally on the regions from primary visual, somatosensory cortex, motor cortex, and thalamus.

In the consequent study Achard and Bullmore [2] described effect of drugs and aging on the dynamics of neural networks. The weighted networks were created with weights of connections equal to correlation between wavelet coefficients of averaged regional time series. Authors have focused on scale 3 wavelet coefficients,

corresponding to 0.06–0.12 Hz frequency range, since in this range the findings were most salient. The correlation matrices were thresholded in order to obtain adjacency matrices of simple binary networks. Authors have studied how economical properties of networks change over a range of threshold values. Clear that the value of threshold determines the cost of the networks, and increasing (decreasing) of threshold can be considered as moving from the network with higher cost to the one with lower cost (moving from lower cost to higher cost in case of decreasing of threshold value).

As it was reported in the previous study, authors found that all brain networks obtained by describing technique showed small-world properties [2]. In particular, networks with cost from 0.05 to 0.34 interval were apparent small worlds.

Older people had reduced global and local efficiency compared with younger people. The effect of age for given sample from the study on global efficiency was statistically significant ($p = 0.006$), as it was on local efficiency ($p = 0.05$). This reduction was present for whole range of thresholds corresponding to values of cost from 0.05 to 0.34 interval. The integrals of the global and local efficiency curves were calculated in order to compare older subjects to younger ones, and again the difference was statistically significant ($p = 0.008$ for global efficiency and $p = 0.02$ for local efficiency).

Also, the drug effect, compared with placebo, in particular dopamine blockade by a dose of sulpiride, reduced global and local efficiencies for low cost networks (Cost ~ 0.1) in both young and old subjects. The drug effects were more variable in the older age group. For both global and local efficiencies, the effect of the drugs was significant ($p = 0.005$ and $p = 0.0009$ correspondingly). Also, values of integrals of global and local efficiencies over the cost range 0.05–0.34 were reduced significantly ($p = 0.03$ for global efficiency and $p = 0.02$ for local efficiency) [2].

It is known that dopamine transmission is responsible in part for modulating the frequency, phase, and spatial coherence of signal oscillations in basal ganglia and cortex. The findings of this study suggest that it also may contribute to optimizing the performance of functional brain networks. The pharmacological change of this modulating input reduces network efficiency, which is an ability to efficiently transfer information [2].

Aging is also known to be associated with loss of dopamine cells, and it would be rather expected that effects of age on economical properties of functional brain networks are similar to those of dopamine blockade. However, the effects of age on economical properties of functional networks were more salient compared to drugs, which may imply that besides of reduced dopamine transmission in aged subjects, additional factors, caused by aging, might contribute to age-related changes in network properties [2].

4 Conclusion

Number of natural and human made networks exhibit small-world topology. These networks have well-defined local clustering structure with presence of certain number of long-range connections between distant nodes. Mathematically, these

characteristics can be expressed as high value of network clustering coefficient and low value of characteristic path length. Small-world networks possess important economical properties which distinguish them from vast pool of possible network configurations. It has been shown that small-world networks are efficient in parallel information transfer. At the same time, they are more resilient to random attacks (deleting nodes at random) than random networks and more resilient to targeted attacks (deleting nodes) than scale-free networks [3].

Multiple studies reported that neural systems of different animal species as well as those of human share certain common organizational principles. These networks were shown to be small worlds at different levels of spatial scale. A group of studies suggested that brain networks tend to optimize their small-world characteristics compared to other networks with the same number of nodes and edges but different topology. Arguably, brain might have developed during evolution process to maximize its efficiency of information processing and resilience to errors and damage while minimizing the total length of neural wiring. Small-world networks were shown to have all these attributes.

One of the implications of studying small-world characteristics of human brain lies in the analysis of neural disorders impact. Neural disorders affect brain function and consequently affect functional network configuration. Changes in network topology cause changes in network quantitative characteristics. Of course, different disorders may result in different network modifications, possibly variable across patients. However, under assumption that healthy brain networks tend to optimize its small-world characteristics, any change in network topology likely will result in degrading of these characteristics. Therefore, it is possible computationally to distinguish signs of disorder by comparing its small-world characteristics to those in healthy subjects. Several works have been done in this direction [31, 68, 71, 93], and results indicate that this is a promising area of research with broad range of diagnostic implications.

Cross-References

- ▶ [Hardness and Approximation of Network Vulnerability](#)
- ▶ [Network Optimization](#)
- ▶ [Optimization Problems in Online Social Networks](#)

Recommended Reading

1. T.B. Achacoso, W.S. Yamamoto, *AY's Neuroanatomy of C. Elegans for Computation* (CRC, Boca Raton, 1992)
2. S. Achard, E. Bullmore, Efficiency and cost of economical brain functional networks. *PLoS Comput. Biol.* **3**(2), e17 (2007)
3. S. Achard, R. Salvador, B. Whitcher, J. Suckling, E. Bullmore, A resilient, low-frequency, small-world human brain functional network with highly connected association cortical hubs. *J. Neurosci.* **26**(1), 63 (2006)

4. R. Albert, A.L. Barabasi, Statistical mechanics of complex networks. *Rev. Mod. Phys.* **74**(1), 47–97 (2002)
5. L. Amaral, J. Ottino, Complex networks: augmenting the framework for the study of complex systems. *Eur. Phys. J. B Condens. Matter* **38**(2), 147–162 (2004)
6. R.G. Andrzejak, G. Widman, K. Lehnertz, P. David, C.E. Elger, Nonlinear determinism in intracranial EEG recordings allows focus localization in neocortical lesional epilepsy. *Epilepsia*, **40**(suppl 7), 171–172 (1999)
7. J. Arnhold, P. Grassberger, K. Lehnertz, C.E. Elger, A robust method for detecting interdependences: application to intracranially recorded EEG. *Phys. D Nonlinear Phenom.* **134**(4), 419–430 (1999)
8. D. Attwell, S.B. Laughlin, An energy budget for signaling in the grey matter of the brain. *J. Cereb. Blood Flow Metab.* **21**(10), 1133–1145 (2001)
9. A.L. Barabási, Z.N. Oltvai, Network biology: understanding the cell's functional organization. *Nat. Rev. Genet.* **5**(2), 101–113 (2004)
10. M. Barahona, L.M. Pecora, Synchronization in small-world systems. *Phys. Rev. Lett.* **89**(5), 54101 (2002)
11. B.D. Bartholow, On the role of conflict and control in social cognition: event-related brain potential investigations. *Psychophysiology* **47**(2), 201–212 (2010)
12. D.S. Bassett, E. Bullmore, Small-world brain networks. *Neuroscientist* **12**(6), 512 (2006)
13. D.S. Bassett, A. Meyer-Lindenberg, S. Achard, T. Duke, E. Bullmore, Adaptive reconfiguration of fractal small-world human brain functional networks. *Proc. Natl. Acad. Sci.* **103**(51), 19518 (2006)
14. B. Biswal, F.Z. Yetkin, V.M. Haughton, J.S. Hyde, Functional connectivity in the motor cortex of resting human brain using echo-planar MRI. *Magn. Reson. Med.* **34**(4), 537–541 (1995)
15. M. Boersma, D.J.A. Smit, H. de Bie, G.C.M. Van Baal, D.I. Boomsma, E.J.C. de Geus, H.A. Delemarre-van de Waal, C.J. Stam, Network analysis of resting state EEG in the developing young brain: structure comes with maturation. *Hum. Brain Mapp.* **32**, 413–425, (2011)
16. K. Borner, S. Sanyal, A. Vespignani, Network science. *Annu. Rev. Inf. Sci. Technol.* **41**(1), 537–607 (2007)
17. S.L. Bressler, Large-scale cortical networks and cognition. *Brain Res. Rev.* **20**(3), 288–304 (1995)
18. E. Bullmore, O. Sporns, Complex brain networks: graph theoretical analysis of structural and functional systems. *Nat. Rev. Neurosci.* **10**(3), 186–198 (2009)
19. E. Bullmore, J. Fadili, V. Maxim, L. Sendur, B. Whitcher, J. Suckling, M. Brammer, M. Breakspear, Wavelets and functional magnetic resonance imaging of the human brain. *NeuroImage* **23**, S234–S249 (2004)
20. G. Buzsaki, *Rhythms of the Brain* (Oxford University Press, Oxford/New York, 2006)
21. G. Buzsáki, C. Geisler, D.A. Henze, X.J. Wang, Interneuron diversity series: circuit complexity and axon wiring economy of cortical interneurons. *TRENDS in Neurosci.* **27**(4), 186–193 (2004)
22. W. Chaovallitwongse, P. Pardalos, P. Xanthopoulos, Computational Neuroscience (Springer, New York, 2010)
23. C. Cherniak, Component placement optimization in the brain. *J. Neurosci.* **14**(4), 2418 (1994)
24. M. Chicurel, Databasing the brain. *Nature* **406**(6798), 822–825 (2000)
25. D.B. Chklovskii, T. Schikorski, C.F. Stevens, Wiring optimization in cortical circuits. *Neuron* **34**(3), 341–347 (2002)
26. T.E. Conturo, N.F. Lori, T.S. Cull, E. Akbudak, A.Z. Snyder, J.S. Shimony, R.C. McKinstry, H. Burton, M.E. Raichle, Tracking neuronal fiber pathways in the living human brain. *Proc. Natl. Acad. Sci. USA* **96**(18), 10422 (1999)
27. D. Cordes, V.M. Haughton, K. Arfanakis, G.J. Wendt, P.A. Turski, C.H. Moritz, M.A. Quigley, M.E. Meyerand, Mapping functionally related regions of brain with functional connectivity MR imaging. *Am. J. Neuroradiol.* **21**(9), 1636 (2000)

28. D. Cordes, V.M. Haughton, K. Arfanakis, J.D. Carew, P.A. Turski, C.H. Moritz, M.A. Quigley, M.E. Meyerand, Frequencies contributing to functional connectivity in the cerebral cortex in “resting-state” data. *Am. J. Neuroradiol.* **22**(7), 1326 (2001)
29. F. Crick, Backwardness of human neuroanatomy. *Nature* **361**, 109–110 (1993)
30. G. Deco, V. Jirsa, A. McIntosh, Emerging concepts for the dynamical organization of resting-state activity in the brain. *Nat. Rev. Neurosci.* **12**, 43–56 (2011)
31. W. De Haan, Y.A.L. Pijnenburg, R.L.M. Strijers, Y. Van Der Made, W.M. Van Der Flier, P. Scheltens, C.J. Stam, Functional neural network analysis in frontotemporal dementia and Alzheimer’s disease using EEG and graph theory. *BMC Neurosci.* **10**(1), 101 (2009)
32. M. De Luca, S. Smith, N. De Stefano, A. Federico, P.M. Matthews, Blood oxygenation level dependent contrast resting state networks are relevant to functional activity in the neocortical sensorimotor system. *Exp. Brain Res.* **167**(4), 587–594 (2005)
33. S.I. Dimitriadis, N.A. Laskaris, Y. Del Rio-Portilla, G.C. Koudounis, Characterizing dynamic functional connectivity across sleep stages from EEG. *Brain Topogr.* **22**(2), 119–133 (2009)
34. S.I. Dimitriadis, N.A. Laskaris, V. Tsirka, M. Vourkas, S. Micheloyannis, What does delta band tell us about cognitive Processes: a mental calculation study. *Neurosci. Lett.* **483**, 11–15 (2010)
35. D.J. Felleman, D.C. Van Essen, Distributed hierarchical processing in the primate cerebral cortex. *Cereb. Cortex* **1**(1), 1 (1991)
36. R. Ferri, F. Rundo, O. Bruni, M.G. Terzano, C.J. Stam, Small-world network organization of functional connectivity of EEG slow-wave activity during sleep. *Clin. Neurophysiol.* **118**(2), 449–456 (2007)
37. S. Finger, *Origins of Neuroscience: A History of Explorations into Brain Function* (Oxford University Press, New York/Toronto, 2001)
38. M.D. Fox, M.E. Raichle, Spontaneous fluctuations in brain activity observed with functional magnetic resonance imaging. *Nat. Rev. Neurosci.* **8**(9), 700–711 (2007)
39. P. Fries, A mechanism for cognitive dynamics: neuronal communication through neuronal coherence. *Trends Cogn. Sci.* **9**(10), 474–480 (2005)
40. Z.A. Gaál, R. Boha, C.J. Stam, M. Molnár, Age-dependent features of EEG-reactivity-spectral, complexity, and network characteristics. *Neurosci. Lett.* **479**, 79–84 (2010)
41. P. Hagmann, J.P. Thiran, L. Jonasson, P. Vandergheynst, S. Clarke, P. Maeder, R. Meuli, DTI mapping of human brain connectivity: statistical fibre tracking and virtual dissection. *NeuroImage* **19**(3), 545–554 (2003)
42. P. Hagmann, M. Kurant, X. Gigandet, P. Thiran, V.J. Wedeen, R. Meuli, J.P. Thiran, Mapping human whole-brain structural networks with diffusion MRI. *PLoS One* **2**(7), 597 (2007)
43. Y. He, Z.J. Chen, A.C. Evans, Small-world anatomical networks in the human brain revealed by cortical thickness from MRI. *Cereb. Cortex* **17**(10), 2407 (2007)
44. C.C. Hilgetag, GA Burns, M.A. O’Neill, J.W. Scannell, M.P. Young, Anatomical connectivity defines the organization of clusters of cortical areas in the macaque monkey and the cat. *Philos. Trans. R. Soc. B Biol. Sci.* **355**(1393), 91 (2000)
45. C.J. Honey, R. Kötter, M. Breakspear, O. Sporns, Network structure of cerebral cortex shapes functional connectivity on multiple time scales. *Proc. Natl. Acad. Sci.* **104**(24), 10240 (2007)
46. H. Hong, M.Y. Choi, B.J. Kim, Synchronization on small-world networks. *Phys. Rev. E* **65**(2), 26139 (2002)
47. M.D. Humphries, K. Gurney, T.J. Prescott, The brainstem reticular formation is a small-world, not scale-free, network. *Proc. R. Soc. B Biol. Sci.* **273**(1585), 503 (2006)
48. H. Kattler, D.J.A.N. DIJK, A.A. Borbely, Effect of unilateral somatosensory stimulation prior to sleep on the sleep EEG in humans. *J. Sleep Res.* **3**(3), 159–164 (1994)
49. C.D. Kent, K.B. Domino Depth of anesthesia. *Curr. Opin. Anesthesiol.* **22**(6), 782 (2009)
50. V.A. Klyachko, C.F. Stevens, Connectivity optimization and the positioning of cortical areas. *Proc. Natl. Acad. Sci. USA* **100**(13), 7937 (2003)
51. R. Kötter, Neuroscience databases: tools for exploring brain structure–function relationships. *Philos. Trans. R. Soc. Lond. Ser. B Biol. Sci.* **356**(1412), 1111 (2001)

52. J.P. Lachaux, E. Rodriguez, J. Martinerie, F.J. Varela, Measuring phase synchrony in brain signals. *Hum. Brain Mapp.* **8**(4), 194–208 (1999)
53. L.F. Lago-Fernández, R. Huerta, F. Corbacho, J.A. Sigüenza, Fast response and temporal coherent oscillations in small-world networks. *Phys. Rev. Lett.* **84**(12), 2758–2761 (2000)
54. V. Latora, M. Marchiori, Efficient behavior of small-world networks. *Phys. Rev. Lett.* **87**(19), 198701 (2001)
55. V. Latora, M. Marchiori, Economic small-world behavior in weighted networks. *Eur. Phys. J. B* **32**(2), 249–263 (2003)
56. D. Le Bihan, J.F. Mangin, C. Poupon, C.A. Clark, S. Pappata, N. Molko, H. Chabriat, Diffusion tensor imaging: concepts and applications. *J. Magn. Reson. Imaging* **13**(4), 534–546 (2001)
57. K. Lehnertz, Nonlinear EEG analysis in epilepsy, in *Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE*, vol. 2 (IEEE, Piscataway, 2002), pp. 1546–1549
58. J.P. Lerch, K. Worsley, W.P. Shaw, D.K. Greenstein, R.K. Lenroot, J. Giedd, A.C. Evans, Mapping anatomical correlations across cerebral cortex (MACACC) using cortical thickness from MRI. *NeuroImage* **31**(3), 993–1003 (2006)
59. M. Le Van Quyen, J. Foucher, J.P. Lachaux, E. Rodriguez, A. Lutz, J. Martinerie, F.J. Varela, Comparison of Hilbert transform and wavelet methods for the analysis of neuronal synchrony. *J. Neurosci. Methods* **111**(2), 83–98 (2001)
60. M.J. Lowe, B.J. Mock, J.A. Sorenson, Functional connectivity in single and multislice echoplanar imaging using resting-state fluctuations. *NeuroImage* **7**(2), 119–132 (1998)
61. J. Lu, X. Yu, G. Chen, D. Cheng, Characterizing the synchronizability of small-world dynamical networks. *IEEE Trans. Circuits Syst. I Regul. Pap.* **51**(4), 787–796 (2004)
62. A.R. McIntosh, Towards a network theory of cognition. *Neural Netw.* **13**(8–9), 861–870 (2000)
63. S. Milgram, The small world problem. *Psychol. Today* **2**(1), 60–67 (1967)
64. F. Mormann, K. Lehnertz, P. David, Mean phase coherence as a measure for phase synchronization and its application to the EEG of epilepsy patients. *Phys. D Nonlinear Phenom.* **144**(3–4), 358–369 (2000)
65. J.M.J. Murre, D.P.F. Sturdy, The connectivity of the brain: multi-level quantitative analysis. *Biol. Cybern.* **73**(6), 529–545 (1995)
66. A.V. Oppenheim, R.W. Schafer, Discrete-time signal processing. Prentice Hall Signal Processing (Prentice Hall, Upper Saddle River, 2009), p. 1120
67. E. Pachou, M. Vourkas, P. Simos, D. Smit, C.J. Stam, V. Tsirka, S. Micheloyannis, Working memory in schizophrenia: an EEG study using power spectrum and Coherence analysis to estimate cortical activation and network behavior. *Brain Topogr.* **21**(2), 128–137 (2008)
68. S. Palva, S. Monto, J.M. Palva, Graph properties of synchronized cortical networks during visual working memory maintenance. *NeuroImage* **49**(4), 3257–3268 (2010)
69. D.B. Percival, A.T. Walden, *Spectral Analysis for Physical Applications: Multitaper and Conventional Univariate Techniques* (Cambridge University Press, Cambridge/New York, 1993)
70. D.B. Percival, A.T. Walden, *Wavelet Methods for Time Series Analysis* (Cambridge University Press, Cambridge, 2006)
71. S.C. Ponten, F. Bartolomei, C.J. Stam, Small-world networks and epilepsy: graph theoretical analysis of intracerebrally recorded mesial temporal lobe seizures. *Clin. Neurophysiol.* **118**(4), 918–927 (2007)
72. R. Quian Quiroga, A. Kraskov, T. Kreuz, P. Grassberger, Performance of different synchronization measures in real data: a case study on electroencephalographic signals. *Phys. Rev. E* **65**(4), 41903 (2002)
73. M. Raichle, The brain's dark energy. *Sci. Am. Mag.* **302**(3), 44–49 (2010)
74. M.E. Raichle, M.A. Mintun, Brain work and brain imaging. *Neuroscience* **29**, 449–476 (2006)

75. M.E. Raichle, A.M. MacLeod, A.Z. Snyder, W.J. Powers, D.A. Gusnard, G.L. Shulman, A default mode of brain function. *Proc. Natl. Acad. Sci. USA* **98**(2), 676 (2001)
76. A. Rothkegel, K. Lehnertz, Multistability, local pattern formation, and global collective firing in a small-world network of nonleaky integrate-and-fire neurons. *Chaos Interdiscip. J. Nonlinear Sci.* **19**, 015109 (2009)
77. M. Rubinov, S.A. Knock, C.J. Stam, S. Micheloyannis, A.W.F. Harris, L.M. Williams, M. Breakspear, Small-world properties of nonlinear brain activity in schizophrenia. *Hum. Brain Mapp.* **30**(2), 403–416 (2009)
78. R. Salvador, J. Suckling, M.R. Coleman, J.D. Pickard, D. Menon, E. Bullmore, Neurophysiological architecture of functional magnetic resonance images of human brain. *Cereb. Cortex* **15**(9), 1332 (2005)
79. J.W. Scannell, M.P. Young, The connectional organization of neural systems in the cat cerebral cortex. *Curr. Biol.* **3**(4), 191–200 (1993)
80. J.W. Scannell, C. Blakemore, M.P. Young, Analysis of connectivity in the cat cerebral cortex. *J. Neurosci.* **15**(2), 1463 (1995)
81. J.W. Scannell, G. Burns, C.C. Hilgetag, M.A. O’Neil, M.P. Young, The connectional organization of the cortico-thalamic system of the cat. *Cereb. Cortex* **9**(3), 277 (1999)
82. J.D. Schmahmann, D.N. Pandya, R. Wang, G. Dai, H.E. D’Arceuil, A.J. De Crespigny, V.J. Wedeen, Association fibre pathways of the brain: parallel observations from diffusion spectrum imaging and autoradiography. *Brain* **130**, 630–653 (2007)
83. R.G. Shulman, D.L. Rothman, K.L. Behar, F. Hyder, Energetic basis of brain activity: implications for neuroimaging. *Trends Neurosci.* **27**(8), 489–495 (2004)
84. W. Singer, Development and plasticity of cortical processing architectures. *Science* **270**(5237), 758 (1995)
85. W. Singer, Neuronal synchrony: a versatile code review for the definition of relations? *Neuron* **24**, 49–65 (1999)
86. O. Sporns, Network analysis, complexity, and brain function. *Complexity* **8**(1), 56–60 (2002)
87. O. Sporns, *Networks of the Brain* (MIT, Cambridge, 2010)
88. O. Sporns, J.D. Zwi, The small world of the cerebral cortex. *Neuroinformatics* **2**(2), 145–162 (2004)
89. O. Sporns, G. Tononi, G.M. Edelman, Theoretical neuroanatomy: relating anatomical and functional connectivity in graphs and cortical connection matrices. *Cereb. Cortex* **10**(2), 127 (2000)
90. C.J. Stam, Functional connectivity patterns of human magnetoencephalographic recordings: a ‘small-world’ network? *Neurosci. Lett.* **355**(1–2), 25–28 (2004)
91. C.J. Stam, J.C. Reijneveld, Graph theoretical analysis of complex networks in the brain. *Nonlinear Biomed. Phys.* **1**(1), 3 (2007)
92. C.J. Stam, B.W. Van Dijk, Synchronization likelihood: an unbiased measure of generalized synchronization in multivariate data sets. *Phys. D Nonlinear Phenom.* **163**(3–4), 236–251 (2002)
93. C.J. Stam, B.F. Jones, G. Nolte, M. Breakspear, P. Scheltens, Small-world networks and functional connectivity in Alzheimer’s disease. *Cereb. Cortex* **17**(1), 92 (2007)
94. C.J. Stam, G. Nolte, A. Daffertshofer, Phase lag index: assessment of functional connectivity from multi channel EEG and MEG with diminished bias from common sources. *Hum. Brain Mapp.* **28**(11), 1178–1193 (2007)
95. K.E. Stephan, L. Kamper, A. Bozkurt, G.A.P.C. Burns, M.P. Young, R. Kötter, Advanced database methodology for the Collation of Connectivity data on the Macaque brain (CoCo-Mac). *Philos. Trans. R. Soc. Lond. Ser. B Biol. Sci.* **356**(1412), 1159 (2001)
96. T. Stieglitz, B. Rubehn, C. Henle, S. Kisban, S. Herwik, P. Ruther, M. Schuettler, Brain-computer interfaces: an overview of the hardware to record neural signals from the cortex. *Prog. Brain Res.* **175**, 297–315 (2009)
97. S.H. Strogatz, Exploring complex networks. *Nature* **410**(6825), 268–276 (2001)
98. L.W. Swanson, *Brain Architecture: Understanding the Basic Plan* (Oxford University Press, Oxford/New York, 2003)

99. F. Takens, Detecting strange attractors in turbulence, in *Dynamical Systems and Turbulence, Warwick 1980* (Springer, Berlin/New York, 1981) pp. 366–381
100. V. Tsirka, P.G. Simos, A. Vakis, K. Kanatsouli, M. Vourkas, S. Erimaki, E. Pachou, C.J. Stam, S. Micheloyannis, Mild traumatic brain injury: graph-model characterization of brain networks for episodic memory. *Int. J. Psychophysiol.* **79**, 89–96 (2010)
101. N. Tzourio-Mazoyer, B. Landeau, D. Papathanassiou, F. Crivello, O. Etard, N. Delcroix, B. Mazoyer, M. Joliot, Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. *NeuroImage* **15**(1), 273–289 (2002)
102. E. Van Dellen, L. Douw, J.C. Baayen, J.J. Heimans, S.C. Ponten, W.P. Vandertop, D.N. Velis, C.J. Stam, J.C. Reijneveld, Long-term effects of temporal lobe epilepsy on local neural networks: a graph theoretical analysis of corticography recordings. *PLoS One* **4**(11), e8081 (2009)
103. F. Varela, J.P. Lachaux, E. Rodriguez, J. Martinerie, The brainweb: phase synchronization and large-scale integration. *Nat. Rev. Neurosci* **2**(4), 229–239 (2001)
104. D.J. Watts, S.H. Strogatz, Collective dynamics of ‘small-world’ networks. *Nature* **393**(6684), 440–442 (1998)
105. K.J. Worsley, J.I. Chen, J. Lerch, A.C. Evans, Comparing functional connectivity via thresholding correlations and singular value decomposition. *Philos. Trans. R. Soc. B Biol. Sci.* **360**(1457), 913 (2005)
106. J. Xiong, L.M. Parsons, J.H. Gao, P.T. Fox, Interregional connectivity to primary motor cortex revealed using MRI resting state images. *Hum. Brain Mapp.* **8**(2–3), 151–156 (1999)
107. M.P. Young, The organization of neural systems in the primate cerebral cortex. *Proc. Biol. Sci.* **252**(1333), 13–18 (1993)
108. A.S. Zandi, G.A. Dumont, M. Javidan, R. Tafreshi, An entropy-based approach to predict seizures in temporal lobe epilepsy using scalp EEG, in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE* (IEEE, Piscataway, 2009), pp. 228–231
109. H.P. Zaveri, S.M. Pincus, I.I. Goncharova, R.B. Duckrow, D.D. Spencer, S.S. Spencer, Localization-related epilepsy exhibits significant connectivity away from the seizure-onset area. *NeuroReport* **20**(9), 891 (2009)

Social Structure Detection

Kai Yang, Weili Wu, Wei Zhang, Tzuwei Hsu and Lidan Fan

Contents

1	Introduction	3090
2	Social Network Structure	3090
2.1	Link Prediction Problem on Social Network	3091
2.2	Graph Proximity Approaches	3092
2.3	Supervised Learning-Based Approaches	3097
3	Local Structure	3101
3.1	Social Community	3101
3.2	Social Community Identification	3102
4	Influence Maximization Problem	3110
4.1	Two Probabilistic Models	3111
4.2	Two Operational Models	3115
5	Compact Routing Scheme for Power-Law Graphs in Social Network	3120
5.1	Compact Routing Scheme for Power-Law Graphs	3121
5.2	The Adapted Compact Routing Scheme	3123
5.3	Analysis of Properties in Performance	3125
6	Conclusion	3129
	Cross-References	3129
	Recommended Reading	3130

Abstract

This chapter mainly studies four problems proposed in social networks: they are link prediction problem, community detection problem, influence maximization problem, and routing problem for networks that satisfy power-law distribution property. As for these problems, corresponding efficient approximation algorithms with different aspects are introduced. Except numerical results of

K. Yang (✉) • W. Wu • W. Zhang • T. Hsu • L. Fan

Department of Computer Science, The University of Texas at Dallas, Richardson, TX, USA

e-mail: yangkai2011@gmail.com; weiliwu@utdallas.edu; dc.zhangwei@gmail.com;

TzuweiHsu@utdallas.edu; ldfan28@gmail.com

experimental simulations of these algorithms, theoretical analysis for influence maximization problem and routing problem is provided.

1 Introduction

A social network is a kind of social structure that is made up of a finite set of individuals (called “nodes”) and relationships (friendship, kinship, common interest) defined on them. Social network existed before the emergence of internet. In a social network, a person can use existing contacts as potential social links to extend their relationships to more people. Recently, online social network, which appears after internet, becomes popular among the public. Most online social network such as MySpace, LinkedIn, and Facebook provide a convenient platform for people to communicate with each other, exchange their ideas or spread information, and so on. Therefore, based on the relationships established between people, communities are shown to be an important local structure, which can be used to efficiently predict new links or disseminate information to their members by already known information.

This chapter consists of four parts: social network structure, local structure, influence maximization problem, and routing scheme for power-law graphs in social networks. In Sect. 2, a link prediction problem, which is a basic computational problem underlying social network evolution, is introduced. Section 3 focuses on local structural properties of social networks, also known as community structure. Some community properties and several algorithms for community identification are introduced. Sections 4 and 5 present two hot research topics in social networks. Influence maximization problem is addressed when a new idea is supposed to have more people to believe it or a company would like to make more customers to purchase a new product with the “word-of-mouth” effect in a social network. How to choose the people and take advantage of the relationships between people to promote your idea or product is what the influence maximization problem concerns about. Routing problem mainly focuses on compact routing scheme by using the theory of unweighted random power-law graphs with fixed expected degree sequence. The method discussed has the first theoretical bound coupled to the parameter of the power-law graph model for a compact routing scheme.

2 Social Network Structure

In recent years, people have witnessed the rapid growth of interests in networks, which are pervasive in the real world. With extensive studies, many researchers point out that the real-world networks demonstrate certain surprising consistent structure properties across different fields. Particularly, studies on three major characteristics of the networks, which are small world, clustering, and scale-free, are of great popularity. And based on those structural properties, three basic topological measurements including average path length, clustering coefficient, and degree

distribution are proposed to study the network from the topological perspective. In general, the average path length is the average distance between any pair of nodes. The degree of a node is the number of edges connected to that node. Newman [51] defined the clustering coefficient measure as

$$C = \frac{3 \times \text{number of triangles in the graph}}{\text{number of connected triples}} \quad (1)$$

where a triangle is formed by a set of three vertices, among which each node is connected to both of the others. A connected triple is three vertices $u - v - w$, with both vertices u and w connected with v ($u - v - w$ and $w - v - u$ are considered as the same connected triple). According to topological features, networks fall into mainly four different classes, which are information network, social network, technological networks, and biological networks. In this section, we focus on social networks.

A social network is a set of people or groups of people with some pattern of contacts or interactions among them under certain situation. From a graph theory point of view, a social network structure is defined as this: among a set of nodes and edges, the nodes represent people or other entities embedded in an underlying social context, and the edges represent interaction, collaboration, or influence between two entities. For instance, in the academic domain, nodes represent scientists in a particular discipline, and an edge is generated between two individuals if they have coauthored papers.

On one hand, from a static perspective that all edges are considered simultaneous, a social network has many classical invariant structural properties, such as small world phenomenon, clustering, scare-free, and so on. All of these characteristics provide us important information for probing other features of networks. On the other hand, in reality, social networks are highly dynamic, that is, they grow and change over time by adding new edges between nodes, which means that new connections occur in underlying networks. In order to realistically model the real world networks, it is of great advantage to consider the structure of social networks from a dynamic perspective.

Nowadays, along with the development of online large networks and the availability of the dataset collected from social networks, people are able to use the existing information to understand social network evolution or the potential structure pattern already existing but unobserved in the whole network. In this section, we would like to introduce the link prediction problem, which is a basic computational problem concerned with social network evolution.

2.1 Link Prediction Problem on Social Network

As a subfield of social network analysis, unlike other problems such as influence maximization, community detection, and small network, link prediction problem is concerned with the problem of predicting the future existence of links among nodes in a social network.

The link prediction problem defined in [66] is as follows: Given a social network $G = (V, E)$, where V denotes the set of entities and E is the set of observed links among those groups, then the aim of the problem is to predict how likely an unobserved link $e_{ij} \notin E$ exists between an arbitrary pair of nodes (v_i, v_j) in the network. In general, the link prediction problem can be studied from three different perspectives: they are link existence prediction (Does a link exist between two nodes?), link classification (What type of the relationship between a pair of nodes?), and link regression (How does the user estimate the item?), separately.

Until recently, link prediction has obtained a wide variety of applications among areas including bibliographic domain, molecule biology, criminal investigations, recommendation systems, and so on.

The authors in [66] summarized that the techniques used in the link prediction problem for social network fall into three categories according to the types of models: node-wise similarity-based approaches, topological pattern-based approaches, and probabilistic model-based approaches. According to the resources used to predict links between nodes, methods are developed from two aspects: one is based on the attributes of nodes, and the other takes advantage of the structural properties of networks. Since it is hard to collect accurate individual attributes in real world, the structural properties make more contribution to the prediction problem.

In the following, we introduce approaches that use the topological features of social networks in link prediction problem.

2.2 Graph Proximity Approaches

The graph proximity measures for link prediction problem depend on structural features of the given network. The basic and intuitive method for predicting links is to rank all node pairs according to their graph topology measurements. That is, with regard to the input graph, a weight Score(x, y), which is a subtle predictor based on node neighborhood or path information, is assigned to a pair of nodes (x, y) ; after each pair of nodes get their own scores, these nodes are ranked in decreasing order according to the values of their associated scores, and the top-ranked pairs of nodes are predicted to have high probability to have links between them. In addition, the meanings of these scores vary according to certain contexts.

2.2.1 Original Graph Proximity Measures

In [39], Liben-Nowell et al. studied this problem through a number of proximity measures to predict the new collaborations in a coauthorship network. Given a network $G = (V, E)$, where V denotes the groups of authors, $e = (u, v) \in E$ represents the interactions(coauthorship) that take place at a particular time $t(e)$ between author u and author v .

In order to predict the interactions that will occur among those people, firstly, the authors in [39] chose four distinct time stamps $t_0 < t'_0 < t_1 < t'_1$. Assume $[t_0, t'_0]$ as

the training interval, in which the subgraph $G[t_0, t'_0]$ contains the edges that appear between the time period from t_0 to t'_0 . Meanwhile, regard $[t_1, t'_1]$ as the test interval, which is used to validate the prediction accuracy. Then, a variety of predictors modified from techniques adopted in graph theory and network analysis are applied to predict the similarity of pairs of nodes. These predictors are common neighbors, graph distance, Jaccard's coefficient, and so on, among which $\text{Score}(x, y)$ plays a vital role for predicting links that are likely to appear in the future. In this section, $\text{Score}(x, y)$ denotes the proximity or similarity between nodes x and y with respect to the network topology.

To estimate the validity of these predictors, the authors in [39] brought in two parameters, k_{training} and k_{test} , and focused on a set Core , which contains all nodes that are incident to at least k_{training} edges in $G[t_0, t'_0]$ and k_{test} edges in $G[t_1, t'_1]$. For each link predictor p , the ranked list L_p of pairs of nodes in $V \times V - E_{\text{old}}$ (the set of edges developed in training interval) is made up of predicted links, which are arranged in the order of decreasing probability to appear in the future. Denote $E_{\text{new}}^* := E_{\text{new}} \cap (\text{Core} \times \text{Core})$ and $n = |E_{\text{new}}^*|$, and take the initial n pairs of nodes in the set of $\text{Core} \times \text{Core}$; here, the size of the intersection component of these nodes with set E_{new}^* is used to measure the performance of predictor p . According to the experiment analysis of predictors on coauthorship networks from different conferences, it was suggested that the datasets applied to this kind of networks are less noisy compared to other online networks, that is, these coauthorship networks rely on their own topological features rather than other external factors. Moreover, the experiment results showed that the approaches based on the structural features outperform those ones obtained from random prediction. Meanwhile, the score proved to be better than others.

2.2.2 Weighted Graph Proximity Measures

Unlike previous methods based on structural features such as Newman's common neighbors [48], Adamic and Adar method [4], and preferential attachment [45, 46] took into account of the weight of a link, here, the weight can be viewed as the number of encounters of a user on QABB (question-answering bulletin boards) corresponding to the number of times they meet or communicate.

The authors in [46] defined new scores that combine both structure proximity and link weight as follows:

The score of weighted common neighbor:

$$\text{Score}(x, y) = \sum_{z \in N(x) \cap N(y)} \frac{w(x, z) + w(y, z)}{2}, \quad (2)$$

where $w(x, z)$ denotes the weight of the link between node x and node z .

The score of weighted Adamic and Adar method:

$$\text{Score}(x, y) = \sum_{x' \in N(x) \cap N(y)} \frac{w(x, z) + w(y, z)}{2} \times \frac{1}{\log(\sum_{z' \in N(z)} w(z', z))}. \quad (3)$$

The score of weighted preferential attachment:

$$\text{Score}(x, y) = \sum_{x' \in N(x)} w(x', x) \times \sum_{y' \in N(y)} w(y', y). \quad (4)$$

Murate and Moriyasu [46] collected the data information from the perspectives of encrypted user ID, categories, date, and time. The main idea is first, group QABB Data into two categories, that is, the early time data is used for training and the later one for testing, which aimed at making computation valid. Then, model a social network by adding links to all pairs of the answers in each question for each category. Finally, predict links that are possible to occur based on proximity measures. The validity of the predictors is measured by

$$\text{accuracy} = \frac{\text{number of correctly predicted links}}{\text{number of new links}}. \quad (5)$$

According to the experiment results in [46], it is shown that Adamic and Adar method performs better than the measure of common neighbors, and for networks whose degree distributions are almost uniform, preferential attachment performs bad. To be excited, the weighted Adamic and Adar method outperforms the original Adamic and Adar approach, weighted common neighbor measure also outperforms original common neighbors over almost all cases, and weighted preferential attachment works slightly better than original preferential attachment only when social networks are relatively dense. In general, when the weighted case is considered, the performances of link predictors are improved compared to previous pure proximity measures. The most important contribution of this measure is that it is fairly effective for open and dynamic online social networks, especially when the network is sufficiently dense.

2.2.3 Generalized Clustering Coefficient-Based Measures

The above two approaches solve the prediction problem under the condition that a network model has been given. However, when the model is not provided, how to build the model of a network? Moreover, it is shown that the methods for prediction problem have close relation with the topological structure of large-scale networks. Thus, it is plausible to generate parsimonious graph models, whose characteristics can be used to describe the significant mechanisms governing the structure of graphs.

Huang [33] showed that structural predictors summarize graph data categories with respect to graph generation model and explain link occurrences in an observed graph. Thus, those measures are of great value to predict the link appearance in the future. As for other predictors, more attention was paid on analyzing generalized

clustering coefficient, which is supposed to describe a cycle formation model. Based on the model, a new method for prediction problem was proposed.

Firstly, Huang [33] introduced relevant notations of a graph and defined the generalized clustering coefficient. Given $G = (V, E)$ a finite undirected graph with only simple edges, that is, there are no multiple edges or self-loops in the graph. $V = (1, 2, \dots, N)$ is the set of vertices of G , and $E = (e_1, e_2, \dots, e_M)$ is the set of edges of G , each e_s corresponds to a sequence of two vertices (i, j) , and the terms of link and edge can be viewed as the same one. A path of length k is denoted as $p = (v_0, v_1, \dots, v_k)$, where (v_i, v_{i+1}) is an edge of the graph for all $0 \leq i \leq k - 1$. Define a cycle of length k as a list of vertices $p = (v_0, v_1, \dots, v_k, v_0)$, where (v_i, v_{i+1}) is an edge of G for all $0 \leq i \leq k - 1$. P_{ijk} represents the set of paths of length k starting at i and ending at j , and $|P_{ijk}|$ is the number of such paths. A generalized clustering coefficient $C(k)$ of degree k was defined as

$$C(k) = \frac{\text{number of cycles of length } k \text{ in the graph}}{\text{number of paths of length } k}. \quad (6)$$

Then the algorithm applied to the cycle formation link probability model is introduced, in which the occurrence probability of a link is determined by the number of cycles (of different lengths) that will be formed by adding this link. The algorithm made the assumption that the clustering coefficient (of deferent degrees) is static.

The Algorithm [33]

In this algorithm, a cycle formation model of degree k ($k \geq 1$) is denoted as $CF(k)$, and a list of parameters c_1, \dots, c_k are adopted for corresponding link generation mechanisms $g(1), \dots, g(k)$, which are used to determine the probability that a link will appear. Here, $g(1)$ is a mechanism similar to a random link generation process. Other mechanisms $g(k)$'s ($k > 1$) are consistent with link probability governed by the paths of length k , where length one means one edge between two nodes. $c_k = \Pr((i, j) \in E | |P_{ijk}| = 1)$ is used to measure the probability that a length- k path will become a length- k cycle. Derive from an instance, an equation for c_k was generalized:

$$\Pr((i, j) \in E | |P_{ijk}| = m) = \frac{c_k^m}{(c_k^m + (1 - c_k)^m)}, \quad k > 1. \quad (7)$$

Considering the effects coming from multiple mechanisms, the above equation can be furthered as follows:

$$\begin{aligned} P_{m_2, \dots, m_k} &= \Pr((i, j) \in E | |P_{ij2} = m_2, \dots, |P_{ijk}| = m_k) \\ &= \frac{c_1 c_2^{m_2} \dots c_k^{m_k}}{c_1 c_2^{m_2} c_k^{m_k} + (1 - c_1)(1 - c_2)^{m_2}(1 - c_k)^{m_k}}, \end{aligned} \quad (8)$$

where P_{m_2, \dots, m_k} denotes the total link occurrence probability under the cycle formation model of degree k .

Then we introduce the approaches adopted to estimate the parameters of the cycle formation model based on the generalized clustering coefficients. Actually, when the parameters are figured out, to obtain the link probabilities, it only needs to apply the estimated parameters to Eq. (7). The operations used in this algorithm are iterative.

Firstly, according to the degree distribution, start the computation with a complete random model, then get the cycle formation probability c_1 . $C(k)$ is regarded as the generalized clustering coefficient and is calculated one by one. Secondly, compare $C(2)$ with c_1 , if $C(2)$ cannot be expressed by c_1 , then new formation mechanism for length-2 cycle will be generated. Then, continue to compare the observed $C(3)$ with the expected $C(3)$ under the length-2 cycle formation model, so the estimator c_3 can be derived. When it satisfies the degree of the model, the procedure terminates. The primary component of this method depends on the fact that $C(k)$ is a function of c_1 and has no relation with $c_{k'}$, here, $k' > k$. The method is formally described as follows:

1. Input information of $G = (V, E)$.
2. Compute the generalized clustering coefficients $C(2), \dots, C(k)$ through Eq. (6).
3. Compute the connecting probability under random graph with the degree distribution of G as c_1 , which is the cycle formation probability.
4. Denote c_2 as

$$c_2 = \frac{(1 - c_1)C(2)}{(c_1 - 2c_1C(2) + C(2))}.$$

5. Set $c_i = 0.5$, where $i = 3, \dots, k$.
6. For $i = 3, \dots, k$, iteratively apply the following equation:

$$c_i = \operatorname{argmin}_{c'_i} (|C(i) - f(c_1, \dots, c'_i, \dots, c_k)|).$$

7. Output the values of c_1, \dots, c_k that have been figured out.

The function which is denoted as

$$f(c_1, \dots, c'_i, \dots, c_k) = \sum_i \#(G_i) \Pr(G_i) \Pr((1, k+1) \in E | G_i) \quad (9)$$

is the total probability of link $(1, k+1)$'s occurrence conditional on a path $p = (1, 2, \dots, k+1)$. It is also the theoretical prediction of the expected clustering coefficient of degree k , denoted as $E[C(K)]$ based on the cycle formation model. For a given path of length k , G_i is supposed to be a possible graph pattern, $\#(G_i)$ denotes the number of subgraphs corresponding to this graph pattern, and $\Pr(G_i)$ is the probability for one of the subgraphs to occur, the probability for edge $(1, k+1)$ to occur under certain condition of G_i is denoted as $\Pr((1, k+1) \in E | G_i)$.

Experiment Evaluation ([33])

The experiments were carried out on Enron email dataset, which is a large email collection from a real organization over the course covering a 3.5 years period. In the experiment, the author mainly evaluated the performances of the cycle formation link probability model and the corresponding link prediction algorithm. The dataset analyzed contains 40,489 emails during May 11, 1999, to June 21, 2002. And this paper implements the link prediction analysis on the monthly email graph in 2001. The email graph is undirected and unweighted, and the edges connect senders and recipients of emails during the corresponding time periods. An edge (a, b) means that there is at least one email communication between a and b , that is, either a sends at least one email to recipients including b or b sends at least one email to recipients including a . And month t was set in 2001 to build the initial graph G_{tb} through the emails in the previous 3 months $(t - 3, t - 2, t - 1)$. This graph is regarded as the input for prediction of email links in G_t , thus, the primary goal is to predict the occurrence of links in G_t that do not exist in G_{tb} .

To evaluate the performance of link prediction, construct a receiver operating characteristics (ROC)-style curve with x-axis as the percent of total possible new links selected and y-axis as the percent of actual new links that are in the selected links, respectively. Then area under curve (AUC) measure is applied to estimate the link prediction performance. It is shown that the algorithm based on the cycle formation model performs better than others that already existed, and has great power to predict the probability of link occurrence.

2.3 Supervised Learning-Based Approaches

Facing the challenge of a wide application of the link prediction, it is necessary to construct both a powerful and universal framework. Although the work in [39] demonstrated improvement of measures for prediction problem over random predictors, they used the properties only based on network intrinsic topological structure, while there are other non-topological properties that proved to enhance the performance of methods for link prediction problem. In [32, 40, 43], the authors regarded the prediction problem as a classification modeling, in which they extracted the features both from topology and non-topology. Throughout their studies, supervised learning played a vital role, and [40] improved the predictive accuracy of supervised learning.

Firstly, we introduce the concept of supervised learning. Supervised learning is a task to infer a function from the supervised training data, which contains a big volume of samples, where each sample consists of an input object and a desired output value. Based on the data information, a supervised learning algorithm is used to analyze the training instances to produce the expected function, which will be applied to predict the output value of any input instances. In general, supervised learning contains the following steps:

1. Determine the types of training examples.
2. Collect a training set.
3. Determine the input feature representation for the learned function.
4. Determine the structure of the learned function and its corresponding learning algorithm.
5. Complete the design process.
6. Evaluate the performance accuracy of the learned function.

During supervised learning, four major issues should be considered. The first is the amount of training data available relative to the complexity of the “true” function (classifier or regression function), the second is the trade-off between bias and variance, the third is the dimensionality of the input space, and the fourth is the degree of noise in the desired output values (the supervisory targets).

2.3.1 Work of Madadhain et al. [43]

In this article, the authors studied an event-based network dataset, which consists of sets of events over time. They paid special attention to the temporal feature of the data, and two specific problems related to the event network data were extensively studied. Here, we only introduce one of them: predicting future event co-participation of entities, which aims to estimate to what extent that a given pair of individuals will co-participate in at least one event during some specific time period in the future.

To simplify the following analysis, we introduce the definition in [43], where $\mathcal{E} = \{e_1, \dots, e_m\}$ denotes a set of events and $\mathcal{V} = \{v_1, \dots, v_n\}$ represents the set of participating entities. The set of entities that participate in event e_i is named as P_i . Each event and each entity can have a set of attributes or covariates. And the covariates for e_i and v_j are denoted as y_i and x_j , respectively. The sets of all events and entity covariates are represented by \mathbf{Y} and \mathbf{X} , respectively. The time that an event e_i occurs is denoted as t_i , and i means that e_i is the i th occurred events. $v_j, v_k \in P_i$ means that v_j and v_k are co-participants in event e_i , and $v_j, v_k \in P_{t,t+\Delta t}$ means that v_j and v_k are co-participants in one or more events in the interval $[t, t + \Delta t]$. Meanwhile, name the subset of events taking place in the interval $\mathcal{E}_{t,t+\Delta t}$. Usually, a vertex corresponds to an entity in a network derived from such a dataset, and edges connect vertices that participate in the same events.

The prediction problem was considered as a data-driven classification problem, in which there are two classes consisting of co-participating and not co-participating. Firstly, probabilistic classifiers were used to offer a probability to each class according to the values of some specified features. The probability is defined as follows:

$$p(v_j, v_k \in P_{t,t+\Delta t} | f(\mathcal{E}_{1,t}, \mathcal{V}, \mathbf{X}, \mathbf{Y}) = \mathbf{w}), \quad (10)$$

where $v_j, v_k \in P_{t,t+\Delta t}$ is a binary suggestion determining whether entities v_j and v_k co-participate in any event in the time period $[t, t + \Delta t]$, f is a function used to produce a vector \mathbf{w} of feature values, $\mathcal{E}_{1,t}$ is the historical event data through to time t , and \mathbf{X}, \mathbf{Y} are the relevant entity and event covariate data, respectively. Then based on

the formulation, the problem can be regarded as learning the mapping from feature vectors to class probabilities, and it can be computed by standard “off-the-shelf” prediction algorithms.

To construct a classification model, the authors proposed foundational components for the operation of classification, which include feature selection, training sets and test sets, classification method, and evaluation metric. Then based on the model, experiments were implemented and indicated that relatively standard machine learning approaches could be used to draw predictive information from the event data, which could be regarded as a ranking machine to detect the individual pairs that have high probability to co-participate in future event.

2.3.2 Work of Lichtenwalter et al. [40]

As for previous supervised methods, the researchers were interested in a domain of the prediction problems with highly imbalanced class distributions [32], although the measures adopted work well, the results of the tests operated on the modified data cannot provide the precise information of the real world, and the performance measures in tests were not available to present the strength and weakness of the models. On the other hand, these models mainly use both the semantic and contextual information related almost exclusively with the bibliographic realm. In addition, the surprising influence of geodesic distance and the complexity of class imbalance specific to the task of link prediction were neglected in those works. Furthermore, many factors, which played significant role in influence and guide classification, had not been explored previously by supervised learning.

In [40], the authors first proposed a supervised framework, which considered the factors of the observational period in the network: generality of existing methods, variance reduction, topological causes, and degrees of imbalance. The attractive merit of the framework is its universality, which means that it can be used on any kind of networks, that is, whether a network is weighted, unweighted, directed, or undirected, the framework can be applied to it. Additionally, the framework has the capability of accepting vertex attributes even though it does not need them. Besides studies in supervised learning for prediction problems, an intuitive flow-based metric was applied to the unsupervised measures to obtain more predictive results.

Next, we would like to introduce the PropFlow method for unsupervised prediction, which was proved to be an efficient predictor. The procedure of the algorithm is given as follows.

The PropFlow method [40] corresponds to the possibility that a restricted random walker starts at v_i and terminates at v_j in k steps or fewer by using link weights as transition probabilities. The walker chooses links based on their weights, and the score in the method is applied to predict the occurrence of new links in the future. PropFlow is somewhat like rooted PageRank, but it is a more localized approach of diffusion and is insensitive to topological noise that is far from the source node. Furthermore, PropFlow simply employs a modified breadth-first search restricted to height k , by which it computes faster.

Algorithm 1 PropFlow predictor [40]

Input information of network $G = (V, E)$, node v_s and maximal length l

- 1: Add node v_s to a set denoted as *Found*
- 2: Put node v_s to another set called *NewSearch*
- 3: Add $(v_s, 1)$ into S
- 4: **For** CurrentDegree, which ranges from 0 to l, **do**
- 5: Assign the value of *NewSearch* to a set *OldSearch*
- 6: Make *NewSearch* as a empty set
- 7: **While** *OldSearch* is not empty, **do**
- 8: Bring out v_i from *OldSearch*
- 9: Find *NodeInput* through v_i in S
- 10: Assign *SumOutput* 0
- 11: **For** each v_j in v_i 's neighborhood **do**
- 12: Add weight of e_{ij} to *SumOutput*
- 13: **End for**
- 14: Set the value of *Flow* to be 0
- 15: **For** each v_j in v_i 's neighbors **do**
- 16: Assign the weight of e_{ij} to w_{ij}
- 17: Assign *Flow* the value of *NodeInput* $\times w_{ij}$ *SumOutput*
- 18: Sum $(v_j, Flow)$ and put it into S
- 19: **If** v_j is not in *Found* **then**
- 20: Add v_j into *Found*
- 21: Put v_j to *NewSearch*
- 22: **end if**
- 23: **end for**
- 24: **end while**
- 25: **end for**
- 26: **Output** score S_{sd} for all $n \leq 1$ -degree neighbors v_d of v_s

Supervised learning, which includes dataset collection, generalization, variance reduction, and sampling, was investigated. Based on the detailed study of those above operations, class imbalance is proposed. Combining the nature of the prediction problem and the benefit of supervised learning, the author implemented the supervised framework by classification including general feature extraction, ensemble of classifiers, and overcoming imbalance.

The experiments in [40] were carried out on two datasets: one is a stream of 712 million celluar from a major non-American cellular phone service provider, and the other is a stream of 19,464 multi-agent events representing condensed matter physics collaborations from 1995 to 2000. For the former dataset, a weighted, directed network (*phone*) was built, in which a node v_i represented a caller, and if v_i called v_j , a weighted and directed link e_{ij} would connect v_i with v_j , where weight was the number of calls on the link. For the latter dataset, a weighted, undirected network (*condmat*) from the collaborations was constructed, in which a node represented an author in the event, and a weighted, undirected link was used to denote the interactions between each pair of authors.

Based on those above experiments, it is shown that the general supervised framework outperformed other existing approaches, and the framework was demonstrated to be entirely general, that is, it is able to operate on any kinds of networks whether it

is weighted, unweighted, directed, or undirected. It has the capacity to accept vertex attributes though it does not need to consider them. Get more useful results reading [40].

3 Local Structure

One of the most significant characters of social networks is the local structure, also known as community structure. That is, a massive social network always contains many very dense subgraphs. Identifying these dense subgraphs or communities will help reveal the underlying network structure and analyze the common activities of individuals. In this section, we will introduce the community properties and review several algorithms for community identification.

3.1 Social Community

Several significant characters of social networks have drawn the attentions of many researchers, such as the small-world property, power-law degree distributions, and local structures, also known as communities. In this section, we will focus on this community property.

In a social network, individuals are shown as nodes, and interactions between the individuals are presented by edges, like relationships and influence. The individuals tend to form communities. A community is a group of nodes that are similar to each other and dissimilar from the rest nodes in the network. In a network, it is usually thought as a group where nodes are densely interconnected and sparsely connected to other parts of the network.

So far, several definitions for community structure have been derived. One of the most intuitive ways is to define community in terms of cliques. A clique in a graph is a subgraph in which any pair of nodes are linked. Based on this, the authors in [58] regard maximal cliques in a graph as communities. A maximal clique is a clique that is not contained by any larger clique. Abello et al. [1] generalized the definition of clique and proposed a structure named quasi-clique. A connected subgraph is considered to be a quasi-clique when it is dense enough. Filippo et al. [54] proposed two definitions for community structure. Consider a subgraph V of graph G . For any node $i \in V$, let K_i^{in} be the inside degree of node i , for example, the number of links toward nodes in V . And let K_i^{out} be the outside degree of node i . Then they defined that the subgraph V is a community in a strong sense if

$$K_i^{\text{in}} > K_i^{\text{out}}, \quad \forall i \in V. \quad (11)$$

In this strong community, each node has more connections within the community than with the rest of the graph. Also, they defined the community in a weak sense by

$$\sum_{i \in V} K_i^{\text{in}} > \sum_{i \in V} K_i^{\text{out}}. \quad (12)$$

In this weak community, the sum of all degrees within V is larger than the sum of all degrees toward the rest of the network.

Although none of the above definitions can be considered universal, they are all capable to capture the underlying properties of social networks.

3.2 Social Community Identification

Community identification is to find groups that either have an inherent or an externally specified notion of similarity among their nodes. As mentioned earlier, people in the same community are more similar than people from different communities. Which means that they might share more on information, interests, experiences, and other useful resources. So discovering the underlying community structures has direct impacts on optimizing and managing activities in a social network. Agarwal and Kempe [5] showed that identifying communities might help people study the whole massive network by communities. That is, after partitioning the graph into communities, one can start focusing on single community. Since nodes of the same community have considerable overlapping on their characters, the analysis inside one community would be more convenient and meaningful. They also claimed that by contracting each community into a node, one can simplify the original large-scale network considerably. This will help people get to know the network from a very high point of view.

Due to the above reasons, community identification has become a very important issue in the social network study. However, nowadays, social networks are usually huge massive networks consisting of millions of nodes, so in general, very little is known about the community structure of a graph; thus, one of the challenges in community identification is that the number of communities in a network and their sizes are not known beforehand. Furthermore, the communities need to be established by the community detection algorithm. Next, we will introduce algorithms within different categories and show how they deal with the challenge and generate communities effectively.

3.2.1 Hierarchical Clustering

Hierarchical clustering is the most widely used method among traditional community identification methods. The core of any hierarchical clustering method is the definition of a similarity measure between vertices. Once such a measure is set, one can compute and sort the similarity values for all pairs of nodes in the given network. Based on the direction of a community formulation procedure, the corresponding methods fall into the following two categories:

Agglomerative algorithms: start with a non-edge graph with only node set. Edges are iteratively added into this graph by decreasing order of similarity values until the original graph is formed.

Divisive algorithms: run in the opposite direction of agglomerative algorithms. They start with the original graph, and edges are deleted based on the increasing order of the similarity values.

The two categories of algorithms will finally generate a tree or dendrogram for the original graph. All the nodes of the given graph are presented by leaves. A non-leaf node on the tree denotes the community resulted from the merging of two smaller communities. The root would be the original graph.

The hierarchical clustering method does not provide any measure of when the clustering procedure should be terminated. So as for the size and number of communities, the terminating point should be manually decided according to the application scenario.

Several Similarity Measures

Structural equivalence is a property derived from sociological studies. Two nodes are of structural equivalence if they have the same set of neighbors. A measure called correlation coefficient was proposed in [63] based on structural equivalence. Suppose A_{ij} is the adjacent matrix for the given graph. Define means and variances of the columns as

$$\mu_i = \frac{1}{n} \sum_j A_{ij}, \quad \sigma_i^2 = \frac{1}{n} \sum_j (A_{ij} - \mu_i)^2; \quad (13)$$

the correlation coefficient is

$$x_{ij} = \frac{\frac{1}{n} \sum_k (A_{ik} - \mu_i) \sum_k (A_{jk} - \mu_j)}{\sigma_i \sigma_j}. \quad (14)$$

Vertices that have a high degree of structural equivalence will have high values of this similarity measure, and meanwhile, those that do not have a high degree of structural equivalence will have low values.

Another similarity measure is the number of edge (vertex)-independent paths between two nodes [64]. Intuitively, the more independent paths between two nodes, the more related they are. This measure is easy to calculate through augmenting path algorithm.

A measure called “edge betweenness” was proposed in [26]. Recall that in hierarchical clustering method, the algorithm iteratively finds the most “central” edges and adds them into communities. Different from those traditional similarity measures, “edge betweenness” describes how an edge is “between” communities and the algorithm will run in a divisive manner.

For edge e , the edge betweenness of e is defined to be the number of shortest paths between pairs of other vertices that run through e . The idea behind this definition is that if a network contains communities or groups that are only loosely connected by a few intergroup edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness.

Their algorithm is proposed as follows:

Algorithm 2 Algorithm based on edge betweenness [26]

- 1: For each edge in the graph, calculate its edge betweenness. Sort all the edges by decreasing order of the edge betweenness.
 - 2: Recalculate betweennesses for all edges (or for time efficiency, only those whose betweenness value are changed after the removal). Perform another sorting process.
 - 3: Repeat steps 2 and 3 until there is no edge left.
 - 4: Based on the removal, a hierarchical dendrogram can be generated. Then a community partition can be decided manually.
-

3.2.2 Modularity-Based Algorithm

As mentioned earlier, so far, there is no universal definition for communities. But a quantity known as modularity, derived by Newman and Girvan [50], can measure how good a community partition is.

For a given graph $G = (V, E)$, let A be its adjacent matrix; then

$$A_{vw} = \begin{cases} 1 & \text{when } (v, w) \in E, \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

Suppose all the vertices have been divided into communities. Let c_v denote the community node v belongs to. Define function δ to be

$$\delta(i, j) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases} \quad (16)$$

Then the fraction of edges that fall within communities, that is, edges that connect vertices that both lie in the same community, is

$$\frac{\sum_{vw} A_{vw} \delta(c_v, c_w)}{\sum_{vw} A_{vw}} = \frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, c_w), \quad (17)$$

where $m = \frac{1}{2} \sum_{vw} A_{vw}$ is the number of edges of graph G . This quantity is not suitable to measure the strength of community partitions for the case that all the nodes form the same community. Under this case, the quantity is maximized while there is no community structure information proposed at all. However, if the expected value of the same quantity is subtracted from it in the case that every edge is generated randomly, an effective measure can be obtained.

Set k_v to be the degree of node v :

$$k_v = \sum_w A_{vw}. \quad (18)$$

Then the probability that there is a random edge between node v and node w is

$$p_{vw} = \frac{k_v \cdot k_w}{2m}. \quad (19)$$

The modularity Q is defined to be

$$Q = \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w). \quad (20)$$

By defining modularity in this way, the case that the whole graph forms one community can be avoided since the corresponding quantity is 0. This modularity expression can be reformulated through the following two quantities:

$$e_{ij} = \frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, j) \delta(c_w, j), \quad (21)$$

which denotes the fraction of edges between community i and community j . Another quantity is

$$a_i = \frac{1}{2m} \sum_v k_v \delta(c_v, i), \quad (22)$$

which is the fraction of edges that have one endpoint falling in community i . Notice that

$$\delta(c_v, c_w) = \sum_i \delta(c_v, i) \delta(c_w, i), \quad (23)$$

then

$$\begin{aligned} Q &= \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \delta(c_v, c_w) \\ &= \frac{1}{2m} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{2m} \right] \sum_i \delta(c_v, i) \delta(c_w, i) \\ &= \sum_i \left[\frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, i) \delta(c_w, i) - \frac{1}{2m} \sum_v k_v \delta(c_v, i) \frac{1}{2m} \sum_w k_w \delta(c_w, i) \right] \\ &= \sum_i (e_{ii} - a_i^2). \end{aligned} \quad (24)$$

Newman and Girvan [50] showed that, in practice, values being greater than about 0.3 appear to indicate significant community structure. Based on modularity, many algorithms have been designed, which makes modularity currently the most widely used measurement for community identification. Next, we start introducing several of these algorithms.

Greedy Algorithms

One most intuitive algorithm is a greedy strategy derived by Newman [49]. At the beginning, each node forms a sole community. Then communities are integrated together in pairs repeatedly. Which pair to be combined is based on the increase

that they make on Q . The procedure continues until all the nodes are contained in one community. Clearly when the algorithm terminates, a dendrogram is generated, showing the order of these combinations. Then as introduced earlier, communities partition can be manually determined. Each step of the algorithm takes time $O(m + n)$ in worstcase. At most, $n - 1$ combination operations are needed to construct the complete dendrogram. So the algorithm runs in time $O((m + n)n)$. When it comes to sparse graph, it is $O(n^2)$.

Clauset et al. [16] improved the above algorithm and proposed another similar algorithm but has running time $O(md\log n)$, where d is the depth of the dendrogram describing the community structure. In their algorithm, instead of maintaining the adjacency matrix and calculating ΔQ_{ij} for each pair (i, j) , a matrix of value of ΔQ_{ij} was directly maintained and updated. Communities that have no edges between them will not make changes to the value of Q . Hence, the algorithm will only focus on communities that are linked by some edges. Additionally, another two data structures, which will further save both memory and time consumption of the algorithm, were maintained to track the largest Q_i . These two data structures are described as follows:

1. A vector array H recording the largest element of each row of matrix ΔQ_{ij} and the corresponding labels i, j
2. A vector array with elements a_i

The algorithm also starts with the state that each vertex forms a sole community. Initially

$$e_{ij} = \begin{cases} 1/2m & \text{if } (i, j) \in E, \\ 0 & \text{otherwise,} \end{cases} \quad (25)$$

and $a_i = k_i/2m$. So

$$\Delta Q_{ij} = \begin{cases} 1/2m - k_i k_j / (2m)^2 & (i, j) \in E, \\ 0 & \text{otherwise.} \end{cases} \quad (26)$$

The algorithm then proceeds as follows:

Algorithm 3 Greedy algorithm [16]

- 1: Calculate the values of ΔQ_{ij} and a_i , and construct vector array H .
 - 2: Choose the largest ΔQ_{ij} from H , merge the corresponding communities, and update the matrix ΔQ , vector array H , Q , and a_i .
 - 3: Repeat step 2 till there is only one community left.
-

Step 2 is supposed to be carried out faster according to the definition of H and a_i . If communities i and j are merged, then the i th row and column of ΔQ are removed and the j th row and column are updated. The update rules are as follows:

If community k is connected to both i and j , then

$$\Delta Q'_{jk} = \Delta Q_{ik} + \Delta Q_{jk}. \quad (27)$$

If k is only connected to i, then

$$\Delta Q'_{jk} = \Delta Q_{ik} - 2a_j a_k. \quad (28)$$

If k is only connected to j, then

$$\Delta Q'_{jk} = \Delta Q_{ik} - 2a_i a_k. \quad (29)$$

Searching Strategies

Simulated annealing [7] is an optimization technique that stochastically avoids local peaks by introducing a computational temperature T. When T is high, the searching direction is flexible. Which means that the algorithm is allowed to reach an area with an objective value worse than the current spot. When T decreases, such flexibility is weakened. And after T reaches 0, the algorithm will stop and output the current solution.

In [31], the authors applied the simulated annealing technique for modularity-based identification algorithm to maximize the modularity value. So here, the objective of the simulated annealing procedure is

$$C = -Q, \quad (30)$$

where Q is still the value of the modularity.

There are three operations during the annealing process:

1. Movements of nodes between communities
2. Merging two communities into a new community
3. Splitting one community into two new communities

These operations are performed with probability

$$p = \begin{cases} 1 & \text{if } C_f \leq C_i, \\ \exp\left(-\frac{C_f - C_i}{T}\right) & \text{if } C_f > C_i. \end{cases} \quad (31)$$

The algorithm starts with an initial value for T and runs iteratively. In each iteration, certain numbers of operation are performed based on the probability defined above. Then T is decreased for a fixed value and another iteration starts. Once T reaches 0, the algorithm will terminate and the final communities will be outputted.

3.2.3 Fast Algorithms

The real-world networks are usually huge in size and always have massive structure thus, time efficiency of the community detection algorithm is an important performance measure. The above several algorithms we introduced are usually very time-consuming while facing such complex networks. In this section, we will introduce several algorithms that have near linear running time.

In [65], the authors presented a method that allows for the discovery of communities within graphs of arbitrary size in times that scale linearly with their size. The basic idea behind the algorithm is as follows.

Firstly, partition the given graph $G = (V, E)$ into two communities. At the very beginning, two nodes A and B are selected and assumed to belong to two different communities (the selection policy will be introduced later). Suppose the two communities are presented by G_1 and G_2 . Then the whole graph is considered as an electric circuit. Edges of the graph are taken as resistors of the same resistance. A battery is placed with A and B as the poles, which will add constant voltages on them, suppose 1 on A and 0 on B. Then there will be flows flowing through all the edges. By solving Kirchhoff equations, the voltages V_i for each node i can be obtained, which lie between 0 and 1. Based on a threshold given beforehand, it will be easy to decide which community each node belongs to.

For a certain node C in the graph, suppose its neighbors are D_1, \dots, D_n . According to Kirchhoff equations, let I_i denote the current flowing from D_i to C, the following equation holds

$$\sum_{i=1}^n I_i = \sum_{i=1}^n \frac{V_{D_i} - V_C}{R} = 0, \quad (32)$$

which means that the total current flowing into C should sum up to zero. Hence the voltage of a node equals to the average of its neighbors, that is,

$$V_C = \frac{1}{n} \sum_{i=1}^n V_{D_i}. \quad (33)$$

According to the above equation, the Kirchhoff equations of G can be formulated as follows:

$$V_1 = 1, \quad (34)$$

$$V_2 = 0, \quad (35)$$

$$V_i = \frac{1}{k_i} \sum_{(i,j) \in E} V_j = \frac{1}{k_i} \sum_{j \in G} V_j a_{ij} \quad i = 3, 4, \dots, n, \quad (36)$$

where k_i is the degree of node i and a_{ij} is the element of the adjacency matrix of the graph. Equation (36) can be further written as

$$V_i = \frac{1}{k_i} \sum_{j=3}^n V_j a_{ij} + \frac{1}{k_i} a_{i1} \quad i = 3, 4, \dots, n. \quad (37)$$

By defining

$$V = (V_3, \dots, V_n)^T, \quad (38)$$

$$B = \begin{pmatrix} \frac{a_{33}}{k_3} & \dots & \frac{a_{3n}}{k_3} \\ \vdots & \ddots & \vdots \\ \frac{a_{n3}}{k_n} & \dots & \frac{a_{nn}}{k_n} \end{pmatrix}, \quad (39)$$

$$C = \left(\frac{a_{31}}{k_3}, \dots, \frac{a_{n1}}{k_n} \right)^T, \quad (40)$$

the Kirchhoff equations can be presented by a matrix form

$$V = BV + C. \quad (41)$$

The corresponding solution to the above equation is

$$V = (I - B)^{-1}C. \quad (42)$$

Furthermore, the equation can be simplified as

$$LV = D, \quad (43)$$

where

$$B = \begin{pmatrix} k_3 & -a_{34} & \dots & -a_{3n} \\ -a_{43} & k_4 & \dots & -a_{4n} \\ \dots & \dots & \dots & \dots \\ -a_{n3} & -a_{n4} & \dots & k_n \end{pmatrix}, \quad (44)$$

$$D = (a_{31}, \dots, a_{n1}). \quad (45)$$

To solve the equations, instead of applying the well-known spectral partitioning method, the authors derived a much faster technique that does not compute the eigenvectors of G . Their algorithm starts with a precomputing process which evaluates the initial values for each node. This process takes $O(V)$ time. Then iteratively, based on Eq. (33), the algorithm starts updating the voltage value of each node. The final precision is related to the number of such rounds. Therefore, the performance of the algorithm is proportional to the time it consumes, and this procedure costs $O(\sum_{i=3}^n k_i) = O(E)$ time.

After the above processes terminate, each node gets a voltage value. Then all nodes are sorted according to the voltage value. The authors applied a spectrum presentation, that is, each node is illustrated as a vertical line at the abscissa which is equal to the corresponding voltage value.

Then two challenges come up. One is that if the initial two poles are actually in the same community, obviously, the algorithm will fail. In [65], the authors gave a strategy based on the idea that two nodes that are far away from each other belong to different communities with high probability. So while determining the pole nodes, two nodes with as long distance between them as possible are considered firstly.

This is done by starting with an arbitrary node. Find a node farthest from it by breadth-first search, then find a node farthest from this second node. After certain steps, choose the farthest pair. The other challenge is how to identify the two communities based on the voltage spectrum. Their idea is to find the reasonable gap in the voltage spectrum as large as possible. Here, a reasonable gap is mentioned because the largest gap usually appears at the two endings of the voltage spectrum, which does not make clear sense.

Having shown the algorithm that partitions a network into two parts, it would be easy to discuss the condition that more than two communities are required. Wu and Huberman [65] explained this by an example of partitioning the US college football data into 13 conferences. Here totally 115 teams are involved. The algorithm runs in 13 iterations, in each of which, one community is found. At the beginning of the first iteration, two poles are selected. Then the algorithm for two community identifications is applied to generate the spectrum. Based on this spectrum, two communities are found from the two endings of the spectrum according to the preestimated community size. The process is repeated for 50 times to generate 100 such communities, which are called candidates. Then the rest task is to find 13 communities out of these 100 candidates. This is done by firstly specifying a node(team) that appear in the maximum number of candidate sets. Then select another several nodes based on the number of times they appear in the same candidate set with the specific node. Together, they form the first community. Then find the second specific node that appears in the maximum number of candidate sets, ignoring the nodes in the first community. After another 12 iterations, all the 115 teams will be partitioned into 13 conferences.

Recently, Raghavan et al. [55] proposed a localized community detection algorithm based on label propagation. In their algorithm, initially, each node is assigned a unique label. Then at every iteration of the algorithm, each node scans its neighbors, finds the label that most of them take, and adopts it. It is easy to regard the process as a label propagation through the network. Finally, nodes sharing the same label are considered to be of the same community. Clearly, this algorithm is distributed and takes almost linear time.

4 Influence Maximization Problem

A social network is like a huge container, in which thousands of individuals build up their relationships and interactions. Within a social network, ideas or information among its members spread like a cascade and the influence of the information has practical value. When it comes to marketing, for example, a salesman wants to promote the new products of his company, how should he make his marketing strategy such that the products are purchased by as many customers as possible? This kind of problem is called influence maximization problem in social networks. In order to estimate the influence between individuals and the probability of customers' acceptance, some promotion (discount) will be given to certain customers for free to maximize the sales of the products. Therefore, it can

be seen that the most important step is to select proper target customers, then the question is that by what standard a group of initial target customers will be chosen to get the best influence result. In other words, it is valuable to maximize “word-of-mouth” effect [8, 10, 27, 28, 44].

Due to the strong network effect, only taking into consideration the intrinsic value (the value that an individual purchases a product based on his own desire) is not enough. A more important values, say, network value should be taken into account. A network value is used to evaluate the positive influence of one customer on the others around him/her. The combination of the intrinsic value and network value comes up to be the standard that is used to target the initial customers.

In this section, we will introduce the influence maximization problem and its corresponding algorithms. Domingos and Richardson [20] firstly addressed this problem as a fundamental algorithmic problem. To solve this problem, we introduce two probabilistic models and two operational diffusion models in next section. Except the simplest linear function probabilistic model, the other three are all NP-hard problems. Greedy approximation algorithms with $(1 - 1/e)$ – approximation performance are given.

4.1 Two Probabilistic Models

4.1.1 A General Model

Suppose there are n potential customers in the system. Define them as X_i , which is a Boolean variable (Table 1). If customer i purchases the product, X_i is 1; otherwise, X_i is 0. X_i corresponds to the i th customer. N_i is the set of all neighbors of X_i , i.e., $N_i = \{X_{i,1}, \dots, X_{i,n_i}\} \subseteq X - X_i$, where $X = \{X_1, \dots, X_n\}$. Let $X^k(X^u)$ be the customers whose value is known(unknown), and let $N_i^k = N_i \cap X^k$ and $N_i^u = N_i \cap X^u$. Assume the product is described by a set of attributes $Y = \{Y_1, \dots, Y_m\}$. Define M_i as a variable showing the marketing action of customer i . For instance, M_i could be a Boolean variable, with $M_i = 1$ if the customer is offered a given discount, and $M_i = 0$ otherwise. Let $M = \{M_1, \dots, M_n\}$. Thus, for all $X_i \notin X^k$, there is

$$\begin{aligned}
 [20] P(X_i | X^k, Y, M) &= \sum_{C(N_i^u)} P(X_i, N_i^u | X^k, Y, M) \\
 &= \sum_{C(N_i^u)} P(X_i | N_i^u, X^k, Y, M) P(N_i^u | X^k, Y, M) \\
 &= \sum_{C(N_i^u)} P(X_i | N_i, Y, M) P(N_i^u | X^k, Y, M).
 \end{aligned} \tag{46}$$

Table 1 Symbols in the system model

Item	Description
\mathbf{X} :	$\{X_1, X_2, X_3, \dots, X_n\}$ denotes the set of customers or the purchase activity of customers
N_i :	$\{X_{i,1}, \dots, X_{i,n_i}\}$ denotes X_i 's neighbors
N_i^k :	Neighbors in N_i which are known of purchasing the product
N_i^u :	Neighbors in N_i which are unknown of purchasing the product
\mathbf{Y} :	$\{Y_1, \dots, Y_m\}$ denotes the set of product attributes
\mathbf{M} :	$\{M_1, \dots, M_n\}$ denotes the marketing plan
$C(N_i^u)$:	The set of all possible configurations of the unknown neighbors of X_i

According to [53], $P(N_i^u|X^k, Y, M)$ is approximated by its maximum entropy estimate if the marginal $P(X_j|X^k, Y, M)$ is given for $X_j \in N_i^u$. Then

$$[20] P(X_i|X^k, Y, M) = \sum_{C(N_i^u)} P(X_i|N_i, Y, M) \prod_{X_j \in N_i^u} P(X_j|X^k, Y, M). \quad (47)$$

Because in Eq. (47), $P(X_i|X^k, Y, M)$ is expressed as the function of themselves, it can be applied iteratively with an initial value. One of the initial value is $P(X_i|Y, M)$, which is the network-less probabilities. Note that the number of terms in Eq. (47) is exponential in the size of N_i^u . If this number of the unknown neighbors of X_i is small, this should not be a problem, otherwise, an approximate solution is necessary. Gibbs sampling [25] is one of the standard methods for this problem, and another one is based on an efficient k-shortest-path algorithm presented by Chakrabarti et al. [12].

If N_i and \mathbf{Y} are given, X_i should be independent of the marketing actions for other customers except its neighbors. Then a naive Bayesian model is used to present X_i as a function of N_i, Y_1, \dots, Y_m , and M_i :

$$\begin{aligned} [20] P(X_i|N_i, Y, M) &= P(X_i|N_i, Y, M_i) \\ &= \frac{P(X_i)P(N_i, Y, M_i|X_i)}{P(N_i, Y, M_i)} \\ &= \frac{P(X_i)P(N_i|X_i)P(M_i|X_i)}{P(N_i, Y, M)} \prod_{k=1}^m P(Y_k|X_i) \\ &= \frac{P(X_i|N_i)P(M_i|X_i)}{P(Y, M_i|N_i)} \prod_{k=1}^m P(Y_k|X_i). \end{aligned} \quad (48)$$

It is known that

$$P(Y, M_i|N_i) = P(Y, M_i|X_i = 1)P(X_i = 1|N_i) + P(Y, M_i|X_i = 0)P(X_i = 0|N_i).$$

The corresponding network-less probabilities are $P(X_i|Y, M) = P(X_i)P(M_i|X_i) \prod_{k=1}^m \frac{P(Y_k|X_i)}{P(Y, M_i)}$ [20].

To compute Eq. (47), it only needs to know the following probabilities if Eq. (48) is given. These probabilities are $P(X_i|N_i)$, $P(X_i)$, $P(M_i|X_i)$, and $P(Y_k|X_i)$ for all k . All of these are easily obtained except probability $P(X_i|N_i)$. The form of $P(X_i|N_i)$ depends on how the system is built. In other words, due to the mechanism by which customers are influenced by others, this probability will vary from application to application.

4.1.2 Approximate Algorithms Based on the Probability Model

For the sake of simplicity, it is assumed that \mathbf{M} is a Boolean vector. When marketing a product, suppose the cost is a constant. Use c to represent this number. Let r_0 and r_1 be the revenue from selling the product to the customer whether marketing action is performed. Let $f_i^1(\mathbf{M})$ show the result of setting M_i to 1 and leaving the rest of \mathbf{M} unchanged and $f_i^0(\mathbf{M})$, the result of setting M_i to 0 and remaining the rest of the part in \mathbf{M} the same as before. Thus, customer i 's expected lift in profit in isolation is

$$\begin{aligned} [20] \text{ELP}_i(X^k, Y, M) &= r_1 P(X_i = 1 | X^k, Y, f_i^1(\mathbf{M})) \\ &\quad - r_0 P(X_i = 1 | X^k, Y, f_i^0(\mathbf{M})) - c. \end{aligned} \quad (49)$$

Let M_0 be the null vector in which all the members are zero. The global lift in profit that after a particular choice \mathbf{M} of customers to market is as

$$\begin{aligned} [20] \text{ELP}(X^k, Y, M) &= \sum_{i=1}^n r_i P(X_i = 1 | X^k, Y, M) \\ &\quad - r_0 \sum_{i=1}^n P(X_i = 1 | X^k, Y, M) - |\mathbf{M}|c, \end{aligned} \quad (50)$$

where $r_i = r_1$ if $M_i = 1$, $r_i = r_0$ if $M_i = 0$, and the number of 1s in \mathbf{M} is $|\mathbf{M}|$. The goal is to find the assignment of values to \mathbf{M} that maximizes ELP. Trying to find the optimal \mathbf{M} is intractable. It needs to compute every possible combinations of the assignments to its members. There are three approximate methods that can finish this job. They are single-pass method, greedy search method, and hill-climbing method. Each method is much more expensive than the previous one but better solution. The algorithms are as follows:

Algorithm 4 Single pass [20]

```

1: For each i,
2:   if  $\text{ELP}(X^k, Y, f_i^1(M_0)) > 0$ 
3:      $M_i = 1;$ 
4:   else
5:      $M_i = 0;$ 
6:   end if
7: end for

```

Algorithm 5 Greedy search [20]

- 1: set $M = M_0$;
- 2: Loop through the M_i 's, until no change to M_i 's;
- 3: if $ELP(X^k, Y, f_i^l(M_0)) > ELP(X^k, Y, M)$ then
- 4: $M_i = 1$;
- 5: else
- 6: $M_i = 0$;
- 7: end if

Algorithm 6 Hill-climbing search [20]

- 1: Set $M = M_0$;
- 2: Set $M_{i_1} = 1$, where $i_1 = \operatorname{argmax}_i ELP(X^k, Y, f_i^l(M))$;
- 3: Set $M_{i_2} = 1$, where $i_2 = \operatorname{argmax}_i ELP(X^k, Y, f_i^l(f_{i_1}^l(M)))$;
- 4: Repeat until there is no i for which setting $M_i = 1$ increases ELP .

4.1.3 A Polynomial-Time Solvable Model

In last two subsections, we introduce a general probabilistic model for influence maximization problem, where the optimization problem cannot even be approximated to within a nontrivial factor. In this subsection, we introduce a similar but simpler model. Suppose the symbol is the same as listed in the last subsection, then, for all X_i , there is

$$\begin{aligned}[56] P(X_i | X - X_i, Y, M) &= P(X_i | N_i, Y, M) \\ &= \beta_i P_0(X_i | Y, M_i) + (1 - \beta_i) P_N(X_i | N_i, Y, M). \end{aligned} \quad (51)$$

$P_0(X_i | Y, M_i)$ is X_i 's internal probability of buying the new product. $P_N(X_i | N_i, Y, M)$ is the effect that X_i 's neighbors directly put on her. β_i is between 0 and 1 that measures how neighbor-reliant X_i is. In a general probabilistic model, these interactions between neighbors and X_i are modeled by a nonlinear function. In this model, a simpler linear model is used to approximate this effect instead, the probability is as follows:

$$[56] P_N(X_i = 1 | N_i, Y, M) = \sum_{X_j \in N_i} w_{ij} X_j. \quad (52)$$

w_{ij} represents the extent that customer i is affected by her/his neighbor j , with $w_{ij} \geq 0$ and $\sum_{X_j \in N_i} w_{ij} = 1$. Linear models often perform well, especially when data is sparse [19], and they provide significant advantage for computation. Thus, by combining the last two equations, there is

$$[56] P(X_i | N_i, Y, M) = \beta_i P_0(X_i | Y, M_i) + (1 - \beta_i) \sum_{X_j \in N_i} w_{ij} X_j. \quad (53)$$

Because the optimal marketing strategy for a product has not yet been introduced to the market, the state of the neighbors will not be known. Thus, a formula for computing $P(X_i = 1|Y, M)$ is listed as follows:

$$\begin{aligned}
 [56] P(X_i = 1|Y, M) &= \sum_{\tilde{N} \in C(N_i)} P(X_i = 1|\tilde{N}, Y, M)P(\tilde{N}|Y, M) \\
 &= \sum_{\tilde{N} \in C(N_i)} \beta_i P_0(X_i = 1|Y, M_i)P(\tilde{N}|Y, M) \\
 &\quad + \sum_{\tilde{N} \in C(N_i)} (1 - \beta_i) \sum_{X_j \in N_i} w_{ij}\tilde{N}_j P(\tilde{N}|Y, M) \\
 &= \beta_i P_0(X_i = 1|Y, M_i) \\
 &\quad + (1 - \beta_i) \sum_{X_j \in N_i} \sum_{(\tilde{N} \in C(N_i)) \text{ with } N_j=1} w_{ij}P(\tilde{N}|Y, M). \quad (54)
 \end{aligned}$$

Denote the set of all possible configurations of X_i 's neighbors as $C(N_i)$, and \tilde{N} is one of the state assignments. \tilde{N}_j is the value of X_j specified by \tilde{N} . Since the inner summation is over all possible values of \tilde{N} whenever $\tilde{N}_j = 1$, it is equivalent to $w_{ij}P(X_j = 1|Y, M)$, therefore,

$$\begin{aligned}
 [56] P(X_i = 1|Y, M) &= \beta_i P_0(X_i = 1|Y, M_i) \\
 &\quad + (1 - \beta_i) \sum_{X_j \in N_i} w_{ij}P(X_j = 1|Y, m). \quad (55)
 \end{aligned}$$

The equation above expresses the probabilities $P(X_i = 1|Y, M)$ as a function of themselves. It can be applied iteratively to find them, starting from a suitable initial assignment. A natural choice for initialization is to use the internal probabilities $P_0(X_i = 1|Y, M)$.

In this simple probabilistic model, both the propagation of influence and the effect of the initial targeting are linear. Thus, the influence can be maximized by solving a system of linear equations.

4.2 Two Operational Models

In the last section, we introduce two descriptive models which give a joint distribution over all vertex behaviors in a global view. In this section, operational models from mathematical sociology [29, 57] and interacting particle systems [21, 41], which show the dynamics of adoption step-by-step, are considered. Two basic diffusion models, linear threshold model and independent cascade model,

are presented. The influence maximization problem on these two models and their extension models are NP-complete, but can be approximated well. It is shown that the approximate algorithms for maximizing the spread of influence on these models can be developed in a general framework based on submodular functions. In the next subsections, the definition of submodular function and the description of these two models are given, and based on them, the influence function is shown to be submodular and the influence maximization problem is NP-hard on them.

4.2.1 Submodular Function

Consider a function $f(\cdot)$, which is submodular if it satisfies a natural “diminishing returns” property. This property states that the marginal gain from adding an element to a set A is at least as high as the marginal gain from adding the same element to a superset B . Formally, a submodular function satisfies

$$f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B) \quad (56)$$

for all elements v and all pairs of sets $A \subseteq B$.

Suppose a function f has the following four attributes:

1. Submodular
2. Takes only non-negative values
3. Monotone
4. Not decreasing: $f(A \cup \{v\}) \geq f(A)$ for all elements v and sets A

The influence maximization problem is to find a k -element set A such that $f(A)$ is maximized. This problem is NP-hard, but Nemhauser, Wolsey, and Fisher [17, 47] showed that the greedy hill-climbing algorithm approximates the optimum to within a factor of $(1 - 1/e)$ (where e is the base of the natural logarithm): start with the empty set $A(0)$, and repeatedly add an element that gives the maximum marginal gain.

Theorem 1 ([17, 34, 47]) *For a nonnegative, monotone submodular function f , let S be a set of size k obtained by selecting elements one at a time, each time choosing an element that provides the largest marginal increase in the function value. Let S^* be a set that maximizes the value of f over all k -element sets. Then $f(S) \geq (1 - 1/e)f(S^*)$; in other words, S provides a $(1 - 1/e)$ -approximation.*

4.2.2 Independent Cascade Model

Regard A_0 as the initial node set, and the process proceeds in discrete steps according to the following randomized rule. When one node u wants to activate its neighbor v , the process can succeed only with a probability $p_{u,v}$. If u succeeds at step t , then v will become active in step $t + 1$, but whether or not u succeeds, it cannot make any further attempts to activate v in subsequent steps. When there are no more nodes that can be activated, the process terminates.

Theorem 2 ([34]) *For an arbitrary instance of the independent cascade model, the resulting influence function $\sigma(\cdot)$ is submodular.*

At first sight, it is hard to compute $\sigma(A \cup \{v\}) - \sigma(A)$ for arbitrary set A and node v . The increase of the diffusion effect is very difficult to analyze directly because of the hardness of computing the size of $\sigma(A)$. Fortunately, an equivalent view of the diffusion process can be formulated to an order-independent outcome. Since between each pair of nodes, for example, node v and node w , there is a probability $p_{v,w}$ between v and w , this can be regarded as a result of flipping a coin. From the aspect of process, no matter what is the sequence of the node being activated or whether the node is activated, the result is the same. Based on this fact, it can be assumed that for each pair of neighbors (v, w) , a coin of bias $p_{v,w}$ is flipped at the beginning of the influence diffusion. Store the result for the later check whether w is activated with v already active. If the activation is successful, the edge between the two nodes becomes live, otherwise, it is blocked. So, if the initial active set A and the outcome of the coin flips are fixed, the set of all the activated nodes will be determined.

Let X be one of the outcome and A the initial node set. Denote $\sigma_X(A)$ as the total number of nodes activated when the outcome is X and the initial target set is A . Let $R(u, X)$ be the number of nodes that can be reached on live-edge paths from u , so $\bigcup_{u \in A} R(u, X) = \sigma_X(A)$.

Let A and B be two sets of nodes and $A \subseteq B$. Consider the quantity $\sigma_X(A \cup \{u\}) - \sigma_X(A)$. This is the number of elements in $R(u, X)$ that are not already in the union $\bigcup_{u \in A} R(u, X)$, it is at least as large as the number of elements in $R(u, X)$ that are not in the (larger) union $\bigcup_{u \in B} R(u, X)$. It follows that $\sigma_X(A \cup \{u\}) - \sigma_X(A) \geq \sigma_X(B \cup \{u\}) - \sigma_X(B)$, which is the definition of submodularity. Finally, there is

$$[34]\sigma(A) = \sum_{\text{outcomes } X} \text{Prob}[X]\sigma_X(A). \quad (57)$$

since the expected number of nodes activated is just the weighted average over all outcomes. And a nonnegative linear combination of submodular functions is also submodular, hence $\sigma(\cdot)$ is submodular.

Theorem 3 ([34]) *The influence maximization problem is NP-hard for the independent cascade model.*

The influence maximization problem on independent cascade model can be reduced from set cover problem. Consider an instance of the NP-complete set cover problem, define a collection of subsets T_1, T_2, \dots, T_m and a ground set $Q = \{q_1, q_2, \dots, q_n\}$. The problem aims to find out whether there are k subsets whose union is equal to Q . It is assumed that $k < n < m$. This also can be viewed as a special case of the influence maximization problem.

Given an arbitrary instance of the set cover problem, define a corresponding directed bipartite graph with $n + m$ nodes: there is a node i corresponding to each set T_i , a node j corresponding to each element q_j , and a directed edge (i, j) with activation probability $p_{i,j} = 1$ whenever $q_j \in T_i$. The set cover problem is

equivalent to deciding whether there is a set A of k nodes in this graph with $\sigma(A) \geq n + k$. Note that for the instance we have defined, activation is a deterministic process, since all probabilities are 0 or 1. Initially activating the k nodes corresponding to sets in a set cover solution results in activating all n nodes corresponding to the ground set Q , and if any set Q of k nodes has $\sigma(Q) \geq n + k$, then the set cover problem must be solvable.

4.2.3 Linear Threshold Model

In the linear threshold model, a node v is influenced by each neighbor w according to a weight $b_{v,w}$ and $\sum_{w \text{ neighbor of } v} b_{v,w} \leq 1$. The dynamic process proceeds as follows: each node v chooses a threshold θ_v uniformly at random from the interval $[0,1]$, this shows that, in order to activate node v , all the weight between v and its active neighbors must be over the threshold. If there is an initial target set A_0 and a bunch of random thresholds between each pair of nodes, the diffusion process proceeds deterministically in discrete steps: at step t , all the active nodes are still active and inactive nodes whose neighbors' overall weight larger than their thresholds become active.

$$[34] \quad \sum_{w \text{ active neighbor of } v} b_{v,w} \geq \theta_v. \quad (58)$$

Theorem 4 ([34]) *For an arbitrary instance of the linear threshold model, the resulting influence function $\sigma(\cdot)$ is submodular.*

In the last subsection, the independent cascade model was transferred into the live-edge graph and the influence function on it proved to be submodular.

Similarly, in the linear threshold model, given a graph G , assume the active node set is A_t at step t , for $t = 0, 1, 2, \dots$, and A_0 is the initial set. If node v is inactive in step t , then the probability of v to be activated by its neighbors in step $t+1$ is $\frac{\sum_{u \in A_t \setminus A_{t-1}} b_{u,v}}{1 - \sum_{u \in A_{t-1}} b_{u,v}}$.

In the live-edge model, it also starts with an initial set A_0 . At step t , if node v 's edge is among the live-edge set, then v is known, otherwise, v is unknown. Then, at step $t+1$, the chance of v being known is equal to the chance that its live edge comes from $A_t \setminus A_{t-1}$, given that its live edge has not come from any of the earlier sets. The probability is $\frac{\sum_{u \in A_t \setminus A_{t-1}} b_{u,v}}{1 - \sum_{u \in A_{t-1}} b_{u,v}}$, same as above. Thus, the two distribution processes are the same.

Once the equivalence between live-edge model and linear threshold model is proved to be true, it can be used in the last subsection to show that the influence function in linear threshold model is also submodular.

Theorem 5 ([34]) *The influence maximization problem is NP-hard for the linear threshold model.*

Consider an instance of the NP-complete vertex cover problem: given an undirected n -node graph $G = (V, E)$ and an integer k , the problem is to find whether there is a set S with k nodes in G such that every edge has at least one endpoint in S .

It is shown that this is a special case of the influence maximization problem. Given an instance of the vertex cover problem involving a graph G , define a corresponding instance of the influence maximization problem by directing all edges of G in both directions. If there is a vertex cover S of size k in G , then one can deterministically make $\sigma(A) = n$ by targeting the nodes in the set $A = S$, conversely, this is the only way to get a set A with $\sigma(A) = n$.

4.2.4 Greedy Algorithms for Operational Models

In this section, two approximate algorithms are given for the operational model. One is the basic greedy heuristic algorithm [47]; the other is CELF (cost-effective lazy forward selection) algorithm [38].

Basic Greedy Heuristic Algorithm [47]

The natural way to find the solution of a submodular function model is to start with the empty set and add new element which has the largest influence increase. Repeat the same selection process until the stop condition is satisfied. For influence maximization problem, define $d_v(A) = f(A \cup \{v\}) - f(A)$. Concrete description of the algorithm is as follows:

Algorithm 7 Greedy heuristic for submodular function model [47]

- 1: $A_0 = \text{NULL}$
 - 2: $N_0 = N$ and $t = 1$.
 - 3: Iteration t
 - 4: Select $i(t) \in N_{t-1}$
 - 5: Loop $d_{i(t)}(A_{t-1}) = \max_{i \in N_{t-1}} d_i(A_{t-1})$
 - 6: with connection settled arbitrarily. Set $d_{t-1} = d_{i(t)}(A_{t-1})$
 - 7: $A_t = A_{t-1} \cup i(t)$ and $N_t = N_{t-1} - i(t)$.
 - 8: if $t < K$ then
 - 9: $t = t + 1$.
 - 10: end if
 - 11: repeat until $t = K$. Here K is the number of the total iteration.
-

Let f_g be the solution of the greedy heuristic. There is

$$[47] f_g = f_0 + f_1 + \dots + f_K \quad (59)$$

CELF Algorithm for Submodular Function Model [38]

The basic greedy heuristic algorithm has two drawbacks. One is time-consuming for the reason that at each iteration, it needs to reevaluate each node's marginal gain; the other is that greedy heuristic algorithm only applies to uni-cost problem (the cost of choosing each node is the same). In the case of different cost problem, this algorithm works badly. It will choose the node with highest marginal gain every time without considering the cost. For example, there are two nodes, say u and v . Choosing u will improve the influence at I and cost C , while selecting v will lead to influence increase at $I + \xi$ but cost $2 * C$. Here, I and C are two constant numbers. When $\xi \rightarrow 0$, one should definitely choose u rather than v with considering the cost.

A new greedy heuristic algorithm called benefit-cost algorithm was presented in [38]. Instead of only taking the benefit into account that $\text{argmax}_f(A_{k-1} \cup \{v\}) - f(A_{k-1})$, the new algorithm computes the ratio of benefit and cost, which is

$$\frac{\text{argmax}_f(A_{k-1} \cup \{v\}) - f(A_{k-1})}{c(s)}, \quad (60)$$

where $c(s)$ is the cost to choose node v .

But in fact, this new algorithm is much worse than the original one. For example, there are two nodes n_1 and n_2 with $c(n_1) = \xi$ and $c(n_2) = B$, the total budget is B . $R(n_1) = 2 * \xi$ and $R(n_2) = B$. $R(\{n_1\}) - R(\emptyset))/c(n_1) = 2$ and $R(\{n_2\}) - R(\emptyset))/c(n_2) = 1$. Here, $R(\cdot)$ is the benefit function, and $c(\cdot)$ is the cost function. The benefit-cost greedy algorithm would pick n_1 . After selecting n_1 , n_2 cannot be satisfied anymore, and the total reward would be ξ . However, the optimal solution would pick n_2 , achieving total penalty reduction of B . As ξ approaches 0, the performance of the benefit-cost greedy algorithm becomes arbitrarily bad.

When combining the two algorithms into one algorithm, here comes the CELF algorithm [38]. It has two steps:

- Set (solution) $R(A)$: use benefit-cost greedy algorithm
Set (solution) $R(B)$: use uni-cost greedy algorithm
- $\text{argmax}(R(A), R(B))$

It is shown that although both of the two solutions can be arbitrarily bad, there is at least one of them which is not too far away from optimum, and hence CELF provides a constant factor approximation.

Theorem 6 ([38]) *Let R be a nondecreasing submodular function with $R(\emptyset) = 0$. Then $\max\{R(A), R(B)\} \geq 1/2(1 - 1/e)\text{OPT}$, where OPT is the optimal value.*

In [35], a special case of the budgeted MAX-COVER problem is proved. For arbitrary nondecreasing submodular functions, the proof is shown in [30]. This theorem stated that the best solution of the basic greedy and benefit-cost greedy (which is returned by CELF) is at most a constant factor within $1/2(1 - 1/e)$ of the optimal solution. But CELF's running time is only $O(T|V|)$ compared to the basic greedy algorithm which is $(T|V|^4)$. Here, $|V|$ is the number of nodes and T is the budget. Sviridenko [60] showed that an approximation guarantee of $(1 - 1/e)$ can be achieved even in a nonconstant environment.

5 Compact Routing Scheme for Power-Law Graphs in Social Network

In the field of complex networks, a number of different characteristics [59] have been explored for social networks. These characteristics contain: (1) small-world property: although not all nodes are neighbors of one another, most nodes can be reached from each other through a small number of hops or steps; (2) heavy-tailed

degree distributions: degree distribution of this graph approximates a power-law distribution; (3) community structure: groups of nodes in a network are more densely connected internally in a community than with the rest in the network. Based on property (2), this section mainly focuses on compact routing scheme by using the theory of unweighted random power-law graphs with fixed expected degree sequence. The method here we introduce is the first theoretical bound coupled to the parameter of the power-law graph model for a compact routing scheme.

5.1 Compact Routing Scheme for Power-Law Graphs

For a network with n nodes, a routing scheme is only allowed to have routing tables with sizes sublinear in n and message header sizes polylogarithmic in n . In general, there are two classes of compact routing schemes: the first is the *labeled* scheme, which is allowed to add labels to node addresses to encode useful information for the sake of routing, where each label has length at most polylogarithmic in n . On the other hand, the second scheme, *name-independent* scheme, does not allow the renaming of node addresses, instead they must function with all possible addresses. Both of these two kinds of compact routing schemes have been studied widely recently. Therefore, this phenomenon demonstrates the relationship between compact routing scheme and power-law graphs, which will play an important role in social network routing problem.

Power-law graphs form an essential family of networks appearing in various real-world scenarios such as some collaboration networks and the famous World Wide Web, especially in social networks, which is a rapidly expanding network in social connectivity. In a power-law graph, for some constant τ , the number of nodes with degree x is proportional to $x^{-\tau}$. In general, the range of the power-law exponent τ for many real-world networks is between 2 and 3. However, power-law graphs do not seem to belong to any of the previous well-studied network families such as trees, planar graphs, or low doubling-dimension graphs, which means that the importance of the power-law graphs should be noticed, and this property is worth paying attention.

Recently, there are some other experimental studies which concentrate on compact routing in power-law graphs and Internet-like graphs. However, they all have some drawbacks. For example, although Krioukov et al. [37] evaluated the universal routing scheme of Thorup and Zwick (TZ) [61] on random power-law graphs and they also provided experimental evidence of much better performance than the theoretical worst-case bound, however, they did not provide a theoretical bound of the TZ scheme on power-law graphs such as stretch and table size. Some other papers such as Enahescu et al. [22] and Brady and Cowen [9], even though they contributed to compact routing schemes for power-law graphs, both of them did not provide rigorous analysis. Therefore, bridging the gap in the study of compact routing schemes for power-law graphs becomes a main optimization problem.

5.1.1 Related Work

Due to the comparison with Thorup and Zwick's routing schemes and other random power-law graph models, it is necessary to provide some related work. Thorup and Zwick [61] mainly contributed two different routing schemes. The first scheme is a stretch-3 scheme with an $O(n^{1/2} \log^{3/2} n)$ -bit routing table per node and $O(\log n)$ -bit labels and headers, which is based on Cowen's earlier scheme in [18]. They both used a small subset A of nodes, called *landmarks*, and they used the landmarks to route messages. In a graph $G = (V, E)$, for every node u , define its cluster $C(u) = \{v \in V : d(v, u) < d(v, A)\}$, where $d(v, u)$ and $d(v, A)$ denote the graph distance from v to u and A , respectively. Let $l(u)$ denote the landmark in A , which is the closest one to node u (ties are arbitrarily resolved). The routing table of node u stores the port identifiers to route messages to all nodes in A and $C(u)$. If a destination v is not in $A \cup C(u)$, u routes through $l(v)$, which guarantees a stretch bound of 3 because of the definition of the cluster $C(u)$. A resampling method was used by Thorup and Zwick to achieve $|A \cup C(u)| = O(n^{1/2} \log^{1/2} n)$ for every node u .

The second scheme of Thorup and Zwick [61] is on the basis of their approximate distance oracle in [62]. For any $k \geq 2$, they designed a compact routing scheme with the attributes of $\tilde{O}(n^{1/k})$ -bit tables, $O(k \log^2 n / \log \log n)$ -bit addresses, and $O(\log^2 n / \log \log n)$ -bit headers (both the bounds on addresses and headers are for fixed-port schemes). The stretch $2k - 1$ with a stretch $4k - 5$ handshake is achieved by this scheme. In order to reduce the stretch to 3, a *handshake* is needed. The scheme used and appeared in this article is similar to the second scheme. However, there are two main differences even both the two schemes use balls and landmarks to route messages. One difference is that high-degree nodes are chosen as landmarks instead of the randomly selected nodes. By using this strategy, $|A \cup B(u)| = O(n^\gamma)$ with $\gamma = \frac{\tau-2}{2\tau-3} + \epsilon$ and $\epsilon > 0$ is achieved. Another difference is that the improved scheme will directly encode the shortest path from $l(v)$ to v in v 's address. It is short within the probability of $1 - o(1)$ because of the distance properties in random power-law graphs. The result of the modified scheme shows smaller routing table size, and the address and header size of $O(\log n \log \log n)$ is better than the second scheme and near the result of the first scheme. However, the improvement of this scheme here is only tailored to unweighted power-law graphs.

5.1.2 Preliminaries

The method adopted in this section is the random graph model for fixed expected degree sequence as defined in [6, 14, 15, 42]. The expression fixed degree random graph (**FDRG**) is used to refer to the original random graph distribution. Consequently, from the previous related paper, a definition for the random power-law graph distribution **RPLG**(n, τ) is obtained.

Definition 1 ([13]) For a constant $\tau \in \{(2, 3)\}$, the random power-law graph distribution **RPLG**(n, τ) is defined as follows. First, we let the sequence of generating

parameters $\mathbf{w} = \{w_1, w_2, \dots, w_n\}$ obey a power law, which is $w_k = (\frac{n}{k})^{1/(\tau-1)}$ for $k \in \{1, 2, \dots, n\}$. Then we insert the edge between v_i and v_j into the random graph within the probability of $\min\{w_i w_j \alpha, 1\}$, where $\alpha = \frac{1}{\sum_k w_k}$.

In the **FDRG** model, the value w_i corresponds to the expected degree of vertex v_i , and \mathbf{w} is referred to as the *expected degree sequence*. The graph is sampled here because of the generating parameter values w_i . Let D_i be the random variable denoting the degree of node v_i . In this adapted model, it can be found that the expected degree $E[D_i]$ of node v_i is smaller than or equal to the generated parameter w_i .

It is required that $n = \|V(G)\|$ is sufficiently large, satisfying the following:

$$n^{\frac{\varepsilon(2\tau-3)}{(\tau-1)}} \geq \frac{2(\tau-1)}{\tau-2} \ln n. \quad (61)$$

The results do not have any other implicit dependencies on ε . Furthermore, the **core** of a graph consists of nodes having large degrees. Let $\gamma = \frac{\tau-2}{2\tau-3} + \varepsilon$ for some $\varepsilon > 0$ and $\gamma' = \frac{1-\gamma}{\tau-1}$.

Definition 2 ([13]) For a power-law graph which has the degree sequence \mathbf{w} and a graph G with n nodes, we can define the **core** with degree threshold $n^{\gamma'}$, $\gamma' \in (0, 1)$, this kind of equations form as follows:

$$\text{core}_{\gamma'}(\mathbf{w}) := \{v_i : w_i > n^{\gamma'}\}, \quad (62)$$

$$\text{core}_{\gamma'}(G) := \{v_i : \deg_G(v_i) > n^{\gamma'}/4\}, \quad (63)$$

where $\deg_G(v_i)$ is the degree of v_i in G (the subscript G is omitted when the graph is clear from the context).

The meaning of $n^{\gamma'}$ -Core is as what $\text{core}_{\gamma'}(\mathbf{w})$ means in [42].

For each vertex u of graph G , the ball relative to the core can be defined as

$$B_G(u) := \{v \in V(G) : d(u, v) < \min_{v' \in \text{core}_{\gamma'}(G)} d(u, v')\}. \quad (64)$$

5.2 The Adapted Compact Routing Scheme

The scheme adapted here is a fixed-port scheme, and it works with arbitrary permutations of port number assignments. Let the unweighted graph $G = (V, E)$ model the network. Each node v in the network has a unique $\lceil \log_2 n \rceil$ -bit static name. Whenever v is written in a routing table, a message header, or a node address, represents $\lceil \log_2 n \rceil$ -bit static name representation. Each node v has $\deg(v)$ ports connecting it with its neighbors. Number these ports by $0, 1, \dots, \deg(v) - 1$, and

thus each port number of v requires $\lceil \log_2 \deg(v) \rceil$ bits. For every packet, the routing scheme needs to determine clearly which port the packet is to be forwarded to.

5.2.1 Routing Scheme

The routing algorithm is based on [18, 61]. A set of landmarks $A \subseteq V$ is used, but it is different from [18, 61]. $\text{Core}_{\gamma'}(G)$ is used as landmarks instead of nodes sampled at random. For each node u in G , let $l(u)$ denote u 's closest landmark, which means $l(u) := \arg \min_{v \in \text{core}_{\gamma'}(G)} d(u, v)$. The local targets of node u are defined as the elements of its *ball* $B_G(u)$. Each node u stores the ports used to route messages along the shortest paths to all landmarks and its local targets. If the target v is neither a landmark nor a local target of u , the message is routed to v 's closest landmark $l(v)$ and from there to the target v .

The scheme used is a labeled scheme. For a node u knowing $l(v)$ of any target v , the address of node v contains an encoding of $l(v)$. Moreover, for a node w on the shortest path from $l(v)$ to v ($w \neq l(v)$ and $w \neq v$), v may not be in $B_G(w)$ and thus w may not know the port to route messages to v . To resolve this issue, it is necessary to extend the address of v by encoding the shortest path from the landmark $l(v)$ to v .

Let $s = u_0, u_1, \dots, u_m = t$ denote the sequence of nodes on a shortest path from s to t . Let $SP(s, t)$ be the encoding of this shortest path as an array with m entries, where $SP(s, t)$ can be encoded with $\sum_{i=0}^{m-1} \log_2 \lceil \deg(u_i) \rceil$ bits. The precise definitions of addresses, message headers, and local routing tables are provided as follows:

Definition 3 ([13]) The address of node $u \in V$ is $\text{addr}(u) := (u, l(u), SP(l(u), u))$.

The header of a message from node s to node t is in one of the following formats:

1. header = (route, s, t), where route = local,
2. header = (route, s, addr), where route = toLandmark and addr = $\text{addr}(t)$,
3. header = (route, s, t, pos, SP), where route \in fromLandmark, direct, pos is a nonnegative integer that may be modified along the route, and $SP = SP(s, t)$ if route = direct or $SP = SP(l(t), t)$ if route = fromLandmark,
4. header = (route, s, t, SP), where route = handshake and SP is a reversed shortest path from t to s to be encoded along the path from s to t .

The local routing table for each node u forms the information about routes to the core and the information about local routes:

$$\text{tbl}(u) := \{(v, \text{port}_u(v)) : v \in \text{core}_{\gamma'}(G)\} \cup \{(v, \text{port}_u(v)) : v \in B_G(u)\}, \quad (65)$$

where $\text{port}_u(v)$ is the local port of u to route messages toward node v along some shortest path from u to v .

5.2.2 Routing Algorithm

Algorithm 8 describes the routing procedure. It includes pseudocode for the source node s to determine the method of sending a message to target t which is based on whether t is local and whether a shortest path to t is known or not. It also describes

the details that an intermediate node u determines whether to forward the message using its local routing table, or forward the message using the shortest path encoded in the header, or to switch the routing direction from the landmark $l(t)$ to the target t . The correctness of the algorithm is based on the simple observation $t \in B_G(w) \cup \text{core}_{\gamma'}(G)$.

Moreover, an additional handshake protocol described here handles the special case when t does not belong to $B_G(s)$ but s belongs to $B_G(t)$. In this situation, the basic LANDMARKBALLROUTING scheme only achieves worst case stretch 5 not stretch 3. However, t knows the reverse path from t to s . Since the graph is undirected, t can send a special **handshake** message back to s , and each node along the path encodes the reverse port number such that, in the end, s knows the shortest path from s to t . For simplicity of expression, the reasonable assumption in [2] that node u knows port q on which the message is received is used. If this assumption does not hold, this handshake protocol can be adapted accordingly as follows. In the routing table of node u , for all $v \in B_G(u) \cup \text{core}_{\gamma'}(G)$, store a $\text{rev-port}_u(v) = \text{port}_w(u)$, where w is the first node on the path from u to v . Then, when forwarding the **handshake** message from t to s , every node u on the path (including t) pretends $\text{rev-port}_u(s)$ to the SP in the header. This adds the routing table size by at most $\lceil \log_2 n \rceil$ bits per entry. In the description of this algorithm, the case of $s \in \text{core}_{\gamma'}(G)$ is also included in the case that stretch can be improved from 3 to 1.

Both the performance of [Algorithm 1](#) and the statement described above are evaluated, we will induce the following theorem from the two previous algorithms in [13].

Theorem 7 ([13]) *LANDMARKBALLROUTING along with the handshake protocol is a routing scheme with the following properties: (1) the worst-case stretch is 5 and it is without handshaking, (2) the worst-case stretch is 3 after handshaking, and (3) every routing decision takes constant time. In addition, for random graphs sampled from $RPLG(n, \tau)$, the following properties hold: (4) the expected maximum table size is $O(n^\gamma \log n)$ bits; this bound also holds with probability at least $1 - 1/n$; (5) address length and message header size are $O(\log n \log \log n)$ bits with probability $1-o(1)$; and (6) addresses and routing tables can be generated efficiently in expected time $O(n^{1+\gamma} \log n)$, and this bound also holds with probability at least $1 - 1/n$.*

5.3 Analysis of Properties in Performance

This subsection analyzes the performance of LANDMARKBALLROUTING for random power-law graphs from different parts.

5.3.1 Analysis of Properties

In stretch part, LANDMARKBALLROUTING has worst-case stretch 5. After handshaking with stretch 5, LANDMARKBALLROUTING has worst-case stretch 3,

Algorithm 8 LANDMARKBALLROUTING on node u, with source s, target t ≠ s, and header [61]

```

1: First, if u = s then
2: if t ∈ BG(s) then
3: send packet with header = (local,s,t) using ports(t) stored in tbl(s)
4: else if u knows SP(s, t) then
5: send packet and the header = (direct,s,t,0,SP(s, t)) by using port SP(s, t)[0]
6: else
7: send packet and the header = (toLandmark,s,addr(t)) using ports(l(t)) stored in tbl(s)
8: end if
9: exit
10: end if
11: Under the condition u ≠ s, we decide if u = header.t then
12: exit as the packet arrived.
13: end if
14: if header.route = toLandmark then
15: if u = header.addr.l(t) then
16: header.route ← fromLandmark; header.pos ← 0; header.SP ← header.addr.SP(l(t), t);
17: forward packet with the new header using port header.SP[0]
18: else
19: forward the packet to portu(header.addr.l(t)) stored in tbl(u)
20: end if
21: else if header.route ∈ { fromLandmark,direct} then
22: We should do the header.pos + 1, then we assign it to header.pos
23: after doing the assignment, we forward the packet by using port header.SP[header.pos]
24: else if header.route = local then
25: forward the packet using portu(header.t) stored in tbl(u)
26: end if

```

which is proved by the triangle inequality as in [18, 61]. In random power-law graphs, some properties of the adapted random power-law graph model should be addressed. Let G be a random graph sampled from **RPLG**(n, τ), the *volume* $\text{Vol}(G)$ satisfies

$$n < \text{Vol}(G) \leq \frac{\tau - 1}{\tau - 2} n. \quad (66)$$

In random power-law graphs and their cores and ball parts, the concentration results for the actual degree of a vertex and for the volume of a set of vertices in the adapted **RPLG**(n, τ) model will be shown, and the corresponding results in the original **FDRG** model are also restated. For a random graph sampled from **FDRG**(w), the random variable D_i measuring the degree of vertex v_i is around its expectation w_i as follows:

$$\Pr[D_i > w_i - c\sqrt{w_i}] \geq 1 - e^{-c^2/2} \quad (67)$$

$$\Pr[D_i < w_i + c\sqrt{w_i}] \geq 1 - e^{-\frac{c^2}{2(1+c)(3\sqrt{w_i})}}. \quad (68)$$

For a random graph sampled from **FDRG**(w), a subset of vertices S and all $0 < c \leq \sqrt{\text{Vol}(S)}$, there is

$$\Pr[|\text{vol}(S) - \text{Vol}(S)| < c\sqrt{\text{Vol}(S)}] \geq 1 - 2e^{-c^2/6}. \quad (69)$$

Let $n \geq 4^{\frac{\tau-1}{(\tau-2)^2}}$. For a random graph sampled from **RPLG**(n, τ), if $w_i \geq 32 \ln n$, for vertex v_i , the degree D_i satisfies the following condition: $\Pr[w_i/4 \leq D_i \leq 3w_i] > 1 - 2/n^4$. Let G be a random graph sampled from **RPLG**(n, τ). For a subset of vertices S satisfying $\text{Vol}(S) \geq 192 \ln n$, it holds with probability at least $1 - 2/n^3$ that $\text{Vol}(S)/8 \leq \text{vol}(S) \leq 4\text{Vol}(S)$. From previous lemma, a corollary, in which the number of edges of a random graph sampled from **RPLG**(n, τ) is at most $\text{vol}(G)/2 \leq \frac{4(\tau-1)}{\tau-2}n$ with probability at least $1 - 1/n^2$, was obtained. For any two disjoint subsets S and T with $\text{Vol}(S).\text{Vol}(T) > c.\text{Vol}(G)$, there is

$$\Pr[d(S, T) > 1] = \prod_{v_i \in S, v_j \in T} \max\{0, (1 - w_i w_j / \text{Vol}(G))\} \leq e^{-c}. \quad (70)$$

Regarding to the issue of core size, let G be a random graph sampled from **RPLG**(n, τ). The probability of the core size is at least $1 - 1/n^2$, which holds that $\text{core}_{\gamma'}(w) = \{v_i : w_i > n^{\gamma'}\} \subseteq \{v_i : \deg(v_i) > n^{\gamma'}/4\} = \text{core}_{\gamma'}(G)$. Moreover, $|\text{core}_{\gamma'}(G)| = \Theta(n^{\gamma'})$.

Now we about the ball sizes. According to the original definition of a ball by Eq. (64), let $\beta = \gamma'(\tau - 2) + \frac{(2\tau-3)e}{\tau-1}$ be a constant. Assume Eq. (61) is satisfied. For a random graph G sampled from **RPLG**(n, τ), with probability at least $1 - 3/n^2$, it holds that for all $u \in V(G)$, there is

$$\begin{aligned} |B_G(u)| &= |\{u' \in V(G) : d(u, u') < d(u, \text{core}_{\gamma'}(w))\}| \\ &= O(n^\beta), |E(B_G(u))| = O(n^\beta \log n), \end{aligned} \quad (71)$$

where $E(B_G(u))$ is the set of internal edges among vertices in $B_G(u)$.

The *existence* of every edge in random graph G is determined only when it is needed. It is treated as a probability distribution before determining which is defined in the random graph model of this analysis. According to the probability distribution *revealing* the edge, it is easy to know when the existence of an edge is determined. Under the condition that given vertex $u \in V(G)$ and a sequence of balls $(B_0 = \{u\}, B_1, B_2, \dots)$, let $V' = V \setminus \text{core}_{\gamma'}(w)$ and $B_i = \{v : d_G(u, v) \leq i\}$. Circles $C_i = B_i \setminus B_{i-1}$ for $i \geq 0$ with $B_{-1} = \emptyset$. Then, focus on the result of E_i when the condition E_i , which is the number of edges between C_i and $C_i \cup C_{i+1}$, is given. For circle C_i , the following holds with probability at least $1 - 2/n^3$: (1) If $\text{Vol}(C_i) < 192 \ln n$, then $E_i \leq 4 \cdot 192 \ln n$ and (2) if $\text{Vol}(C_i) \geq 192 \ln n$, then $E_i \leq 4\text{Vol}(C_i)$.

For table sizes and their computation issue, it is known that $\text{core}_{\gamma'}(G)$ has size $\Theta(n^{\gamma'})$ with probability at least $1 - 1/n^2$ and all balls $B_G(u)$ have size $O(n^{\gamma'})$ with probability at least $1 - 3/n^2$. Therefore, for a random graph G sampled from **RPLG**(n, τ), for all $u \in V(G)$, the expected table size is at most

$|\text{tbl}(u)| = O(n^\gamma)$, and all tables can be generated in expected time at most $O(n^{1+\gamma} \log n)$. These bounds also hold with probability at least $1 - 1/n$.

Last but not least, the address lengths should be considered seriously. Bounding the number of bits for the address of each vertex is: for a node u ; its address contains the encoding of the shortest path $\text{SP}(u, l(u))$ from u to its landmark $l(u)$. Moreover, bounding the diameter of a random power-law graph and the diameter of its core is also needed. Therefore, for a random graph sampled from **RPLG**(n, τ), with probability at least $1 - o(1)$, the diameter of its largest connected component is $\Theta(\log n)$ [14]. If it holds for all $s, t \in V(G)$, $\text{SP}(s, t)$ can be encoded with $O(\log n \log \log n)$ bits. In order to extend the core, a definition is given as follows:

Definition 4 ([13]) The extended core of a random graph from **RPLG**(n, τ) contains all nodes v_i with w_i at least $n^{1/\log \log n}$, that is, $\text{core}^+(w) = \{v_i \in V : w_i \geq n^{1/\log \log n}\}$.

From the fact that the *extended core* “contains” a dense random graph [24], let G be a random graph sampled from **RPLG**(n, τ). The diameter of the subgraph induced by $\text{core}^+(w)$ in G is $O(\log \log n)$ with probability at least $1 - 1/n$ [14]. Moreover, from [14], it is known that there exists a constant C , such that each vertex v_i with $w_i \geq \log^C n$ is at distance $O(\log \log n)$ from the extended core, with probability at least $1 - 1/n^2$. However, from the previous statement, if the probability changed with at least $1 - 1/n$, the distance between any two vertices v_i and v_j with $w_i \geq \log^C n$ and $w_j \geq \log^C n$ is $O(\log \log n)$.

5.3.2 Analysis of Approximate Distance Oracle

For social networking sites, the graph of this kind of application is *preprocessed*, and a special data structure is used for efficient *queries*. Precomputing all shortest paths by using an all-pairs shortest path algorithm and reading a shortest path from a distance table is one way to prepare for queries. However, this approach is impractical due to the constraints of time and memory. If it is supposed to efficiently preprocess a graph to allow for fast distance queries, an approximation method is needed because of general and directed graphs with n vertices, which is necessary to return the shortest distance by using $\Omega(n^2)$ space. The trade-off between approximation ratio, space, preprocessing, and query time can be addressed by approximation distance oracle, and this kind of scenario can be interpreted as a generalization of the all-pairs (approximate) shortest path problem.

To solve this kind of distance oracle for power-law graphs problem, let $\gamma = \frac{\tau-2}{2\tau+3} + \epsilon$ be a constant and suppose Eq. (61) is satisfied. There exists a preprocessing algorithm for random power-law graphs from **RPLG**(n, τ), which creates a distance oracle of expected size $O(n^{1+\gamma})$ and runs in expected time $O(n^{1+\gamma} \log n)$. This method in this chapter is modified from Thorup and Zwick’s distance oracle, which guarantees stretch 3 when $k = 2$. Unlike Thorup and Zwick’s preprocessing method, each node $v \in V$ is chosen as a landmark independently at random with probability $n^{1/2}$ for general graphs, a better balance is possible by

using high-degree nodes as landmarks for power-law graphs. In this way, fewer landmarks will be chosen, and smaller-sized balls can be obtained than original method at the same time. In preprocessing period, the first step is to compute the core for any $v \in V$ and $\deg(v) > n^{\gamma'}/4$, then for each $v \in \text{core}$, run the breadth-first search from $v \in G$. Moreover, if for each node $u \neq v$, store $d(u, v)$ and set $\text{port}_u(v)$ to be the penultimate node on the shortest path. As for each $u \in V$, compute and store $B_G(u)$ and its distances. Finally, for each $v \in B_G(u)$, set $\text{port}_u(v)$ to be the first node on the shortest path to v .

As far as the exact distance(s, t) is concerned with, the distance query result $d(s, t)$ is exact if $s \in B(t)$ or $t \in B(s)$, which runs in time $O(1)$ and achieves stretch 3.

With the rapid development of society, people have more and more opportunities to communicate with each other. From the aspect of social networks, it is of great importance to find the shortest paths for pairs of nodes. In this section, we introduced some properties that theoretically justify the importance of high-degree nodes in power-law graphs. After analyzing the adapted compact routing scheme and its algorithm for random power-law graphs, it is shown that optimizing for power-law graphs may induce better algorithm performance for problems in social network. From both real-world graphs and random power-law graphs, the optimization algorithm shows the efficiency.

Apart from compacting routing scheme in power-law graphs, some techniques in universal graphs are proposed: in [3], the first optimal compact name-independent routing scheme for arbitrary undirected graphs is proposed, and in [52], tight upper and lower bounds for the efficiency of a routing scheme and its space requirement are presented; furthermore, the bounds are improved in [24]. In addition, new compacting schemes about special graphs appear for tree graphs [36], directed graphs [11], weighted graphs [61], and so on.

6 Conclusion

Social network has shown wide range applications in real world, therefore, it is practically necessary to probe into social network structure to find its special properties, based on which many optimization problems regarding social networks in real life can be solved efficiently. In this chapter, approximation algorithms for four problems including link prediction problem, community detection problem, influence maximization problem, and routing problem are investigated. It particular, it can be seen that almost all of these algorithms are designed by using the structure features of social networks. For example, the routing schemes in Sect. 5 take advantage of power-law link distribution property of certain social networks.

Cross-References

- Optimization Problems in Online Social Networks

Recommended Reading

1. J. Abello, M.G.C. Resende, S. Sudarsky, Massive quasi-clique detection, in *LATIN*, pp. 598–612. doi:10.1007/3-540-45995-2-5134, 35
2. I. Abraham, C. Gavoille, D. Malkhi, On space-stretch trade-offs: lower bounds, in *Proceedings of the eighteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, Cambridge, 30 July–02 Aug 2006 (ACM, New York, 2006), pp. 207–216
3. I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, M. Thorup, Compact name-independent routing with minimum stretch. *16th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, Barcelona, 2004
4. L.A. Adamic, E. Adar, Friends and neighbors on the web. *Soc. Netw.* **25**(3), 211–230 (2003)
5. G. Agarwal, D. Kempe, Modularity-maximizing graph communities via mathematical programming. *Eur. Phys. J. B* **66**, 409–418 (2008)
6. W. Aiello, F.R.K. Chung, L. Lu, A random graph model for massive graphs, in *STOC*, 2000, pp. 171–180
7. R. Albert, A.L. Barabasi, Statistical mechanics of complex networks. *Rev. Mod. Phys.* **74**, 47–97 (2002)
8. F. Bass, A new product growth model for consumer durables. *Manag. Sci.* **15**, 215–227 (1969)
9. A. Brady, L. Cowen, Compact routing on power law graphs with additive stretch, in *Proceedings of the 8th Workshop on Algorithm Engineering and Experiments (ALENEX)*, Miami, 2006, pp. 119–128
10. J. Brown, P. Reinogen, Social ties and word-of-mouth referral behavior. *J. Consum. Res.* **14**(3), 350–362 (1987)
11. P. Chandra, A.D. Kshemkalyani, Compact routing in directed networks with stretch factor of two, in *High Performance Computing (HiPC)*, London, vol. 2228/2001, 2001, pp. 24–35
12. S. Charkrabarti, B. Dom, P. Indyk, Enhanced hypertext categorization using hyperlinks, in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (ACM, Seattle, 1998), pp. 307–318
13. W. Chen, C. Sommer, S. Teng, Y. Wang, Compact routing scheme and approximate distance oracle for power-law graphs. *J. ACM Trans. Algorithms* **9**(1), Article No. 4
14. F. Chung, L. Lu, The average distances in random graphs with given expected degrees. *Internet Math.* **99**, 15879–15882 (2002)
15. F.R.K. Chung, L. Lu, *Complex Graphs and Networks*. Volume 107 of CBMS Regional Conference Series in Mathematics (AMS, Providence, 2006)
16. A. Clauset, M.E.J. Newman, C. Moore, Finding community structure in very large networks. *Phys. Rev. E* **70**, 066111 (2004)
17. G. Cornuejols, M. Fisher, G. Nemhauser, Location of bank accounts to optimize float. *Manag. Sci.* **23**, 789–810 (1977)
18. L. Cowen, Compact routing with minimum stretch. *J. Algorithms* **38**(1), 170–183 (2001)
19. P. Domingos, M. Pazzani, On the optimality of the simple Bayesian classifier under zero-one loss. *Mach. Learn.* **29**, 103–130 (1997)
20. P. Domingos, M. Richardson, Mining the network value of customers, in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, 26–29 Aug 2001, pp. 57–66
21. R. Durrett, *Lecture Notes on Particle Systems and Percolation* (Wadsworth, Pacific Grove, 1988)
22. M. Enachescu, M. Wang, A. Goel, Reducing maximum stretch in compact routing, in *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, Phoenix, 13–18 Apr 2008, pp. 336–340
23. P. Erdős, A. Rényi, On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.* **5**, 17–61 (1960)

24. P. Fraigniaud, C. Gavoille, Universal routing schemes. *Distrib. Comput.* **10**(2), 65–78. doi:10.1007/s004460050025
25. S. Geman, D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.* **6**, 721–741 (1984)
26. M. Girvan, M.E.J. Newman, Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA* **99**(12), 7821–7826 (2002)
27. J. Goldenberg, B. Libai, E. Muller, Talk of the network: a complex systems look at the underlying process of word-of-mouth. *Mark. Lett.* **12**(3), 211–223 (2001)
28. J. Goldenberg, B., Libai, E., Muller, Using complex systems analysis to advance marketing theory development: modeling heterogeneity effects on new product growth through stochastic cellular automata. *Acad. Mark. Sci. Rev.* (2001)
29. M. Granovetter, Threshold models of collective behavior. *Am. J. Sociol.* **83**(6), 1420–1443 (1978)
30. C. Guestrin, A note on the budgeted maximization of submodular functions. Technical report, CMU-CALD-05-103, 2007
31. R. Guimera, L.N. Amaral, Functional cartography of complex metabolic networks. *Nature* **433**, 895–900 (2005)
32. M.A. Hasan, V. Chaoji, S. Salem, M. Zaki, Link prediction using supervised learning, in *Workshop on Link Analysis, Counter-terrorism and Security (SDM 2006 Workshop)*, Bethesda, MD, 2006
33. Z. Huang, Link prediction based on graph topology: the predictive value of the generalized clustering coefficient, in *Workshop on Link Analysis: Dynamics and Static of Large Networks, the 12th ACM SIGKDD*, Philadelphia, 2006
34. D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence through a social network, in *KDD '03 Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, pp. 137–146 (ACM, New York, 2003)
35. S. Khuller, A. Moss, J. Naor, The budgeted maximum coverage problem. *Inf. Process. Lett.* **70**, 39–45 (1999)
36. K.A. Laing, Brief announcement: name-independent compact routing in trees, in *PODC '04: Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing*, New York (ACM, 2004), pp. 382–382
37. D.V. Krioukov, K.R. Fall, X. Yang, Compact routing on internet-like graphs, in *Proceedings of IEEE INFOCOM*, Hong Kong, Mar. 2004
38. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, N. Glance, Cost-effective outbreak detection in networks, in *KDD '07 Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose (ACM, New York, 2007), pp. 420–429
39. D. Liben-Nowell, J. Kleinberg, The link prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.* **58**(7), 1019–1031 (2007)
40. R.N. Lichtenwalter, J.T. Lussier, N.V. Chawla, New perspectives and methods in link prediction, in *KDD '10 Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington (ACM, New York, 2010), pp. 243–252
41. T.M. Ligget, *Interacting Particle Systems* (Springer, New York, 1985)
42. L. Lu, Probabilistic methods in massive graphs and internet computing. Ph.D. thesis, University of California, San Diego, 2002
43. J.O. Madadhai, J. Hutchins, P. Smyth, Prediction and ranking algorithms for event-based network data. *ACM SIGKDD Explor.* **7**, 23–30 (2006)
44. V. Mahajan, E. Muller, F. Bass, New product diffusion models in marketing: a review and directions for research. *J. Mark.* **54**(1), 1–26 (1990)
45. M. Mitzenmacher, A brief history of generative models for power law and lognormal distributions. *Internet Math.* **1**(2), 226–251 (2004)
46. T. Murate, S. Moriyasu, Link prediction of social networks based on weighted proximity measures, in *2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2007)*, Silicon Valley, 2007

47. G. Nemhauser, L. Wolsey, M. Fisher, Analysis of the approximations for maximizing submodular set functions. *Math. Program.* **14**, 265–294 (1978)
48. M.E. Newman, Clustering and preferential attachment in growing networks. *Phys. Rev. Lett.* **E 64**, 025102 (2001)
49. M.E. Newman, Fast algorithm for detecting community structure in networks. *Phys. Rev. E* **69**, 066133 (2004)
50. M.E. Newman, M. Girvan, Finding and evaluating community structure in networks. *Phys. Rev. E* **69**(2), 026113 (2004)
51. M.E. Newman, S.H. Strogatz, D.J. Watts, Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E* **64**, 026118 (2001)
52. D. Peleg, E. Upfal, A trade-off between space and efficiency for routing tables. *J. ACM* **36**(3), 510–530 (1989)
53. L. Pelkowitz, A continuous relaxation labeling algorithm for Markov random fields. *IEEE Trans. Syst. Man Cybern.* **20**, 709–715 (1990)
54. F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA* **101**, 2658–2663 (2004)
55. U.N. Raghavan, R. Albert, S. Kumara, *Phys. Rev. E* **76**, 036106 (2007)
56. M. Richardson, P. Domingos, Mining knowledge-sharing sites for viral marketing, in *KDD '02 Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton (ACM, New York, 2002), pp. 61–70
57. T. Schelling, *Micromotives and Macrobbehavior* (Norton, New York 1978)
58. J. Scott, *Social Network Analysis: A Handbook*, 2nd edn. (Sage, London, 2000)
59. S.H. Strogatz, Exploring complex networks. *Nature* **410**, 268–276 (2001). Macmillan Magazines Ltd
60. M. Sviridenko, A note on maximizing a submodular set function subject to knapsack constraint. *Oper. Res. Lett.* **32**, 41–43 (2004)
61. M. Thorup, U. Zwick, Compact routing schemes, in *SPAA '01 Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, Crete Island (ACM, New York, 2001), pp. 1–10
62. M. Thorup, U. Zwick, Approximate distance oracles. *J. ACM* **52**(1), 1–24 (2005)
63. S. Wasserman, K. Faust, *Social Network Analysis* (Cambridge University Press, Cambridge, 1994)
64. D.R. White, F. Harary, *Sociol. Methodol.* **31**, 305–359 (2001)
65. F. Wu, B. Huberman, Finding communities in linear time: a physics approach. *Eur. Phys. J. B* **38**, 331 (2004)
66. E.W. Xiang, A survey on link prediction models for social network Data. Ph.D. dissertation, Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, 2008

Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work

Frederick C. Harris Jr. and Rakhi Motwani

Contents

1	Introduction	3134
2	The First Solution	3135
3	A Proposed Heuristic	3137
3.1	Background and Motivation	3137
3.2	Adding One Junction	3138
3.3	The Heuristic	3138
3.4	Results	3140
4	Problem Decomposition	3142
4.1	The Double Wedge Theorem	3143
4.2	The Steiner Hull	3144
4.3	The Steiner Hull Extension	3145
5	Winter's Sequential Algorithm	3147
5.1	Overview and Significance	3147
5.2	Winter's Algorithm	3147
5.3	Algorithm Enhancements	3148
6	A Parallel Algorithm	3149
6.1	An Introduction to Parallelism	3149
6.2	Overview and Proper Structure	3150
6.3	First Approach	3150
6.4	Current Approach	3152
7	Extraction of the Correct Answer	3152
7.1	Introduction and Overview	3152
7.2	Incompatibility Matrix	3153
7.3	Decomposition	3155
7.4	Forest Management	3155
8	Computational Results	3156
8.1	Previous Computation Times	3156
8.2	The Implementation	3157
8.3	Random Problems	3159
8.4	Grids	3161

F.C. Harris (✉) • R. Motwani

Department of Computer Science & Engineering, University of Nevada, Reno, NV, USA
e-mail: Fred.Harris@cse.unr.edu; rakhi@cse.unr.edu

9 Future Work.....	3170
9.1 Grids.....	3170
9.2 Further Parallelization.....	3171
9.3 Additional Problems.....	3172
Cross-References.....	3174
Recommended Reading.....	3174

Abstract

Given a set of N cities, construct a connected network which has minimum length. The problem is simple enough, but the catch is that you are allowed to add junctions in your network. Therefore, the problem becomes how many extra junctions should be added and where should they be placed so as to minimize the overall network length. This intriguing optimization problem is known as the Steiner minimal tree (SMT) problem, where the junctions that are added to the network are called Steiner points.

This chapter presents a brief overview of the problem, presents an approximation algorithm which performs very well, then reviews the computational algorithms implemented for this problem. The foundation of this chapter is a parallel algorithm for the generation of what Paweł Winter termed T_list and its implementation. This generation of T_list is followed by the extraction of the proper answer. When Winter developed his algorithm, the time for extraction dominated the overall computation time. After Cockayne and Hewgill's work, the time to generate T_list dominated the overall computation time. The parallel algorithms presented here were implemented in a program called PARSTEINER94, and the results show that the time to generate T_list has now been cut by an order of magnitude. So now the extraction time once again dominates the overall computation time.

This chapter then concludes with the characterization of SMTs for certain size grids. Beginning with the known characterization of the SMT for a $2 \times m$ grid, a grammar with rewrite rules is presented for characterizations of SMTs for $3 \times m$, $4 \times m$, $5 \times m$, $6 \times m$, and $7 \times m$ grids.

1 Introduction

Minimizing a network's length is one of the oldest optimization problems in mathematics, and, consequently, it has been worked on by many of the leading mathematicians in history. In the mid-seventeenth century a simple problem was posed: Find the point P that minimizes the sum of the distances from P to each of three given points in the plane. Solutions to this problem were derived independently by Fermat, Torricelli, and Cavalieri. They all deduced that either P is inside the triangle formed by the given points and that the angles at P formed by the lines joining P to the three points are all 120° or P is one of the three vertices and the angle at P formed by the lines joining P to the other two points is greater than or equal to 120° .

In the nineteenth century a mathematician at the University of Berlin, named Jakob Steiner, studied this problem and generalized it to include an arbitrarily large set of points in the plane. This generalization created a star when P was connected to all the given points in the plane and is a geometric approach to the two-dimensional center of mass problem.

In 1934 Jarník and Kössler generalized the network minimization problem even further [41]: Given n points in the plane, find the shortest possible connected network containing these points. This generalized problem, however, did not become popular until the book, *What is Mathematics*, by Courant and Robbins [16], appeared in 1941. Courant and Robbins linked the name Steiner with this form of the problem proposed by Jarník and Kössler, and it became known as the Steiner minimal tree problem. The general solution to this problem allows multiple points to be added, each of which is called a Steiner point, creating a tree instead of a star.

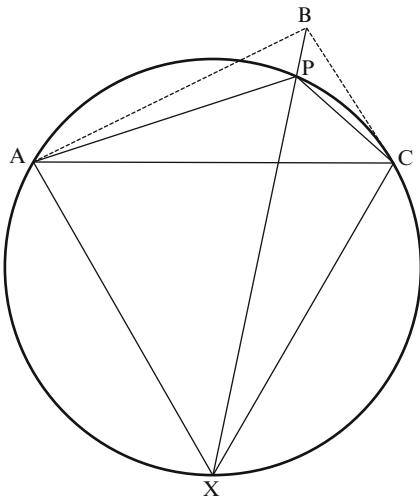
Much is known about the exact solution to the Steiner minimal tree problem. Those who wish to learn about some of the spin-off problems are invited to read the introductory article by Bern and Graham [5], the excellent survey paper on this problem by Hwang and Richards [37], or the volume in The Annals of Discrete Mathematics devoted completely to Steiner tree problems [38]. Some of the basic pieces of information about the Steiner minimal tree problem that can be gleaned from these articles are (a) the fact that all of the original n points will be of degree 1, 2, or 3, (b) the Steiner points are all of degree 3, (c) any two edges meet at an angle of at least 120° in the Steiner minimal tree, and (d) at most $n - 2$ Steiner points will be added to the network.

This chapter concentrates on the Steiner minimal tree problem, henceforth referred to as the SMT problem. Several algorithms for calculating Steiner minimal trees are presented, including the first parallel algorithm for doing so. Several implementation issues are discussed, some new results are presented, and several ideas for future work are proposed.

[Section 2](#) reviews the first fundamental algorithm for calculating SMTs. [Section 3](#) presents a proposed heuristic for SMTs. In [Section 4](#) problem decomposition for SMTs is outlined. [Section 5](#) presents Winter's sequential algorithm which has been the basis for most computerized calculation of SMTs to the present day. [Section 6](#) presents a parallel algorithm for SMTs. Extraction of the correct answer is discussed in [Section 7](#). Computational Results are presented in [Section 8](#) and Future Work and open problems are presented in [Section 9](#).

2 The First Solution

A typical problem-solving approach is to begin with the simple cases and expand to a general solution. As was seen in [Section 1](#), the trivial three point problem had already been solved in the 1600s, so all that remained was the work toward a general solution. As with many interesting problems, this is harder than it appears on the surface.

Fig. 1 $AP + CP = PX$ 

The method proposed by the mathematicians of the mid-seventeenth century for the three-point problem is illustrated in Fig. 1. This method stated that in order to calculate the Steiner point given points A , B , and C , you first construct an equilateral triangle (ACX) using the longest edge between two of the points (AC) such that the third (B) lies outside the triangle. A circle is circumscribed around the triangle, and a line is constructed from the third point (B) to the far vertex of the triangle (X). The location of the Steiner point (P) is the intersection of this line (BX) with the circle.

The next logical extension of the problem, going to four points, is attributed to Gauss. His son, who was a railroad engineer, was apparently designing the layout for tracks between four major cities in Germany and was trying to minimize the length of these tracks. It is interesting to note at this point that a general solution to the SMT problem has recently been uncovered in the archives of a school in Germany (Graham, Private Communication).

For the next 30 years after Kössler and Jarník presented the general form of the SMT problem, only heuristics were known to exist. The heuristics were typically based upon the minimum length spanning tree (MST), which is a tree that spans or connects all vertices whose sum of the edge lengths is as small as possible, and tried in various ways to join three vertices with a Steiner point. In 1968 Gilbert and Pollak [26] linked the length of the SMT to the length of an MST. It was already known that the length of an MST is an upper bound for the length of an SMT, but their conjecture stated that the length of an SMT would never be any shorter than $\frac{\sqrt{3}}{2}$ times the length of an MST. This conjecture was recently proved [17] and has led to the MST being the starting point for most of the heuristics that have been proposed in the last 20 years including a recent one that achieves some very good results [29].

In 1961 Melzak developed the first known algorithm for calculating an SMT [44]. Melzak's algorithm was geometric in nature and was based upon some simple extensions to Fig. 1. The insight that Melzak offered was the fact that you can reduce an n point problem to a set of $n - 1$ point problems. This reduction in size is accomplished by taking every pair of points, A and C in our example; calculating where the two possible points, X_1 and X_2 , would be that form an equilateral triangle with them; and creating two smaller problems, one where X_1 replaces A and C and the other where X_2 replaces A and C . Both Melzak and Cockayne pointed out however that some of these subproblems are invalid. Melzak's algorithm can then be run on the two smaller problems. This recursion, based upon replacing two points with one point, finally terminates when you reduce the problem from three to two vertices. At this termination the length of the tree will be the length of the line segment connecting the final two points. This is due to the fact that $BP + AP + CP = BP + PX$. This is straightforward to prove using the law of cosines, for when P is on the circle, $\angle APX = \angle CPX = 60^\circ$. This allows the calculation of the last Steiner point (P) and allows you to back up the recursive call stack to calculate where each Steiner point in that particular tree is located.

This reduction is important in the calculation of an SMT, but the algorithm still has exponential order, since it requires looking at every possible reduction of a pair of points to a single point. The recurrence relation for an n -point problem is stated quite simply in the following formula:

$$T(n) = 2 * \binom{n}{2} * T(n - 1).$$

This yields what is obviously a non-polynomial time algorithm. In fact Garey, Graham, and Johnson [18] have shown that the Steiner minimal tree problem is NP-Hard (NP-Complete if the distances are rounded up to discrete values).

In 1967, just a few years after Melzak's paper, Cockayne [11] clarified some of the details from Melzak's proof. This clarified algorithm proved to be the basis for the first computer program to calculate SMTs. The program was developed by Cockayne and Schiller [15] and could compute an SMT for any placement of up to seven vertices.

3 A Proposed Heuristic

3.1 Background and Motivation

By exploring a structural similarity between *stochastic Petri nets* (see [45, 49]) and *Hopfield neural nets* (see [27, 35]), Geist was able to propose and take part in the development of a new computational approach for attacking large, graph-based optimization problems. Successful applications of this mechanism include I/O subsystem performance enhancement through disk cylinder remapping [23, 24], file assignment in a distributed network to reduce disk access conflict [22], and new

computer graphics techniques for digital halftoning [21] and color quantization [20]. The mechanism is based on maximum-entropy Gibbs measures, which is described in Reynold’s dissertation [53], and provides a natural equivalence between Hopfield nets and the *simulated annealing* paradigm. This similarity allows you to select the method that best matches the problem at hand. For the SMT problem, the first author implemented the simulated annealing approach [29].

Simulated annealing [42] is a probabilistic algorithm that has been applied to many optimization problems in which the set of feasible solutions is so large that an exhaustive search for an optimum solution is out of the question. Although simulated annealing does not necessarily provide an optimum solution, it usually provides a good solution in a user-selected amount of time. Hwang and Richards [37] have shown that the optimal placement of s Steiner points to n original vertices yields a feasible solution space of the size

$$2^{-n} \binom{n}{s+2} \frac{(n-s-2)!}{s!}$$

provided that none of the original points have degree 3 in the SMT. If the degree restriction is removed, they showed that the number is even larger. The SMT problem is therefore a good candidate for this approach.

3.2 Adding One Junction

Georgakopoulos and Papadimitriou [25] have provided an $\mathcal{O}(n^2)$ solution to the *1-Steiner problem*, wherein exactly one Steiner point is added to the original set of points. Since at most $n - 2$ Steiner points are needed in an SMT solution, repeated application of the algorithm offers a “greedy” $\mathcal{O}(n^3)$ approach. Using their method, the first Steiner point is selected by partitioning the plane into oriented Dirichlet cells, which they describe in detail. Since these cells do not need to be discarded and recalculated for each addition, subsequent additions can be accomplished in linear time. Deletion of a candidate Steiner point requires regeneration of the MST, which Shamos showed can be accomplished in $\mathcal{O}(n \log n)$ time if the points are in the plane [50], followed by the cost for a first addition ($\mathcal{O}(n^2)$). This approach can be regarded as a natural starting point for simulated annealing by adding and deleting different Steiner points.

3.3 The Heuristic

The Georgakopoulos and Papadimitriou 1-Steiner algorithm and the Shamos MST algorithm are both difficult to implement. As a result, Harris chose to investigate the potential effectiveness of this annealing algorithm using a more direct, but slightly more expensive $\mathcal{O}(n^3)$ approach. As previously noted, all Steiner points have degree

```

#define EQUILIBRIUM ((accepts>=100 AND rejects>=200) OR
    (accepts+rejects > 500))
#define FROZEN ((temperature < 0.5) OR ((temperature < 1.0)
    AND (accepts==0)))

while(not(FROZEN)){
    accepts = rejects = 0;
    old_energy = energy();
    while(not(EQUILIBRIUM)){
        operation = add_or_delete();
        switch(operation){
            case ADD:
                ΔE = energy_change_from_adding_a_node();
                break;
            case DELETE:
                ΔE = energy_change_from_deleting_a_node();
                break;
        }
        if(rand(0,1) < emin{0.0,-ΔE/temperature}){
            accepts++;
            old_energy = new_energy;
        }else {
            /* put them back */
            undo_change(operation);
            rejects++;
        }
    }
    temperature = temperature*0.8;
}

```

Fig. 2 Simulated annealing algorithm

3 with edges meeting in angles of 120° . He considered all $\binom{n}{3}$ triples where the largest angle is less than 120° , computed the Steiner point for each (a simple geometric construction), selected that Steiner point giving greatest reduction, or least increase in the length of the modified tree (increases are allowed since the annealing algorithm may go uphill), and updated the MST accordingly. Again, only the first addition requires this (now $\mathcal{O}(n^3)$) step. He used the straightforward $\mathcal{O}(n^2)$ Prim's algorithm to generate the MST initially and after each deletion of a Steiner point.

The annealing algorithm can be described as a nondeterministic walk on a surface. The points on the surface correspond to the lengths of all feasible solutions, where two solutions are adjacent if they can be reached through the addition or deletion of one Steiner point. The probability of going uphill on this surface is higher when the temperature is higher but decreases as the temperature cools. The rate of this cooling typically will determine how good your solution will be. The major portion of this algorithm is presented in Fig. 2. This nondeterministic walk, starting with the MST, has led to some very exciting results.

3.4 Results

Before discussion of large problems, a simple introduction into the results from a simple six-point problem is in order. The annealing algorithm is given the coordinates for six points: (0,0), (0,1), (2,0), (2,1), (4,0), and (4,1). The first step is to calculate the MST, which has a length of 7, as shown in Fig. 3. The output of the annealing algorithm for this simple problem is shown in Fig. 4. In this case the annealing algorithm calculates the exact SMT solution which has a length of 6.616994.

Harris proposed as a measure of accuracy the percentage of the difference between the length of the MST and the exact SMT solution that the annealing algorithm achieves. This is a new measure which has not been discussed (or used) because exact solutions have not been calculated for anything but the most simple layouts of points. For the six-point problem discussed above, this percentage is 100.0 % (the exact solution is obtained).

After communicating with Cockayne, data sets were obtained for exact solutions to randomly generated 100-point problems that were developed for [14]. This allows us to use the measure of accuracy previously described. Results for some of these data sets provided by Cockayne are shown in Table 1.

An interesting aspect of the annealing algorithm that cannot be shown in the table is the comparison of execution times with Cockayne's program. Whereas Cockayne mentioned that his results had an execution cutoff of 12 h, these results were obtained in less than 1 h. The graphical output for the first line of the table,

Fig. 3 Spanning tree for 6-point problem

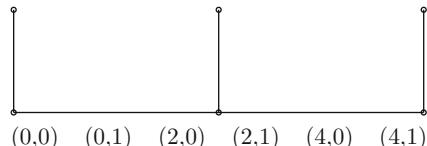


Fig. 4 6-point solution

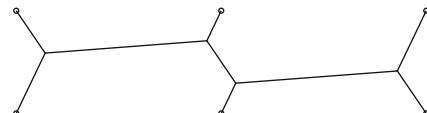


Table 1 Results from 100-point problems

Exact solution	Spanning tree	Simulated annealing	Percent covered (%)
6.255463	6.448690	6.261797	96.39
6.759661	6.935189	6.763495	98.29
6.667217	6.923836	6.675194	96.89
6.719102	6.921413	6.721283	99.01
6.759659	6.935187	6.763493	98.29
6.285690	6.484320	6.289342	98.48

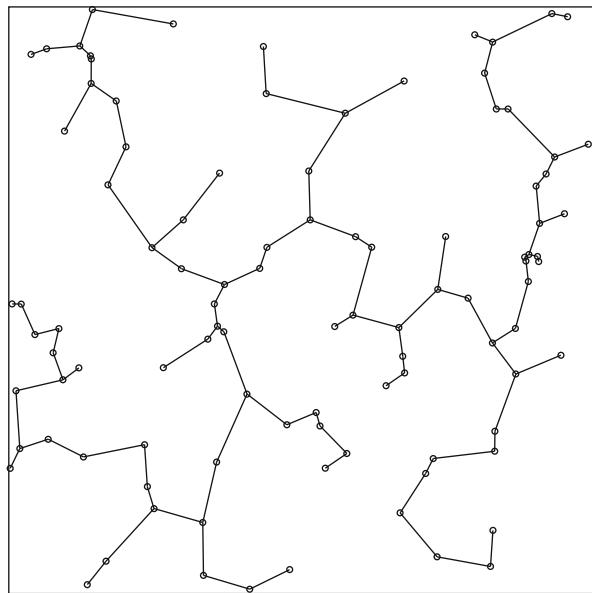


Fig. 5 Spanning tree

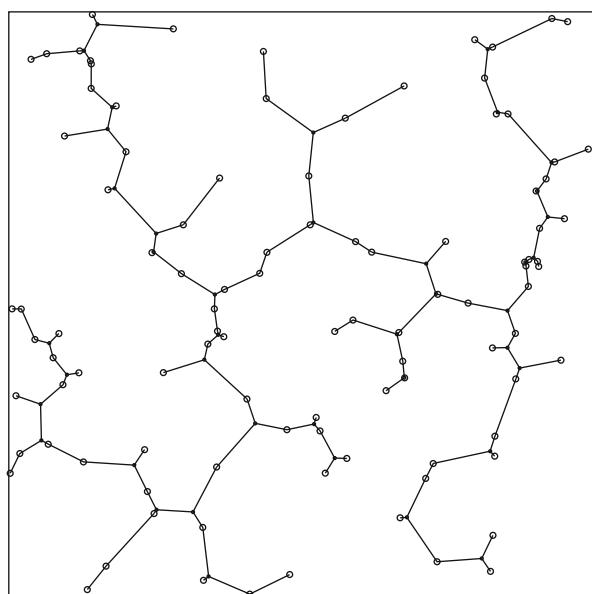


Fig. 6 Simulated annealing solution

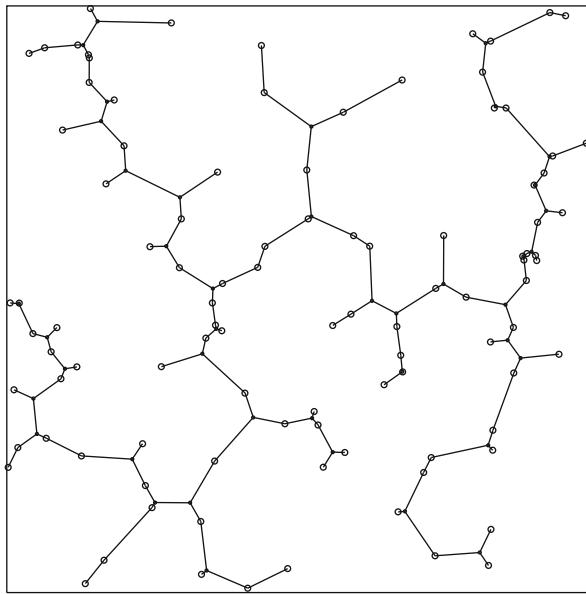


Fig. 7 Exact solution

which reaches over 96 % of the optimal value, appears as follows: The data points and the MST are shown in Fig. 5, the simulated annealing result is in Fig. 6, and the exact SMT solution is in Fig. 7. The solution presented here is obtained in less than $\frac{1}{10}$ of the time with less than 4 % of the possible range not covered. This indicates that one could hope to extend our annealing algorithm to much larger problems, perhaps as large as 1,000 points. If you were to extend this approach to larger problems, then you would definitely need to implement the Georgakopoulos–Papadimitriou 1-Steiner algorithm and the Shamos MST algorithm.

4 Problem Decomposition

After the early work by Melzak [44], many people began to work on the Steiner minimal tree problem. The first major effort was to find some kind of geometric bound for the problem. In 1968 Gilbert and Pollak [26] showed that the SMT for a set of points, S , must lie within the convex hull of S . This bound has since served as the starting point of every bounds enhancement for SMTs.

As a brief review, the convex hull is defined as follows: Given a set of points S in the plane, the convex hull is the convex polygon of the smallest area containing all the points of S . A polygon is defined to be convex if a line segment connecting any two points inside the polygon lies entirely within the polygon. An example of the convex hull for a set of 100 randomly generated points is shown in Fig. 8.

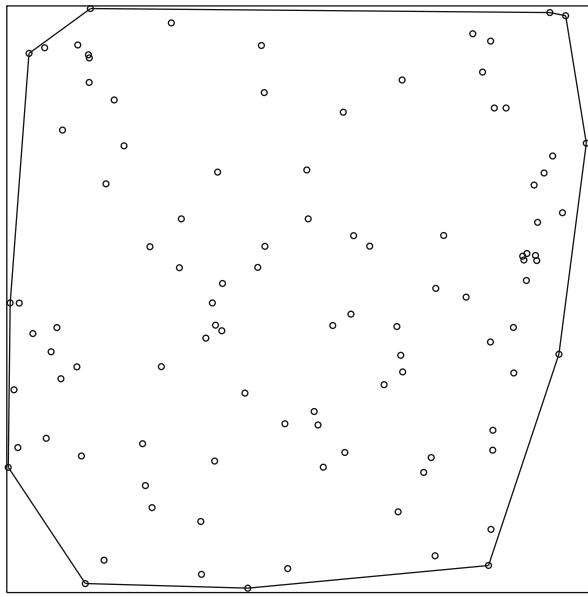


Fig. 8 The convex hull for a random set of points

Shamos in his PhD thesis [54] proposed a divide and conquer algorithm which has served as the basis for many parallel algorithms calculating the convex hull. One of the first such approaches appeared in the PhD thesis by Chow [8]. This approach was refined and made to run in optimal $\mathcal{O}(\log n)$ time by Aggarwal et al. [1], and Attalah and Goodrich [2].

The next major work on the SMT problem was in the area of problem decomposition. As with any non-polynomial algorithm, the most important theorems are those that say “If property \mathcal{P} exists, then the problem may be split into the following sub-problems.” For the Steiner minimal tree problem, property \mathcal{P} will probably be geometric in nature. Unfortunately, decomposition theorems have been few and far between for the SMT problem. In fact, at this writing there have been only three such theorems.

4.1 The Double Wedge Theorem

The first decomposition theorem, known as the Double Wedge Theorem, was proposed by Gilbert and Pollak [26]. This is illustrated in Fig. 9 and can be summarized quite simply as follows: If two lines intersect at point X and meet at 120° , they split the plane into two 120° wedges and two 60° wedges. If R_1 and R_2 denote the two 60° wedges and all the points of S are contained in $R_1 \cup R_2$, then the problem can be decomposed. There are two cases to be considered. In case 1 X

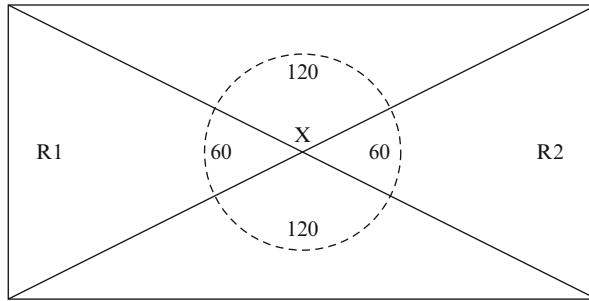


Fig. 9 An illustration of the Double Wedge

is not a point in \mathcal{S} ; therefore, the Steiner minimal tree for \mathcal{S} consists of the SMT for R_1 , the SMT for R_2 , and the shortest edge connecting the two trees. In case 2 \mathcal{X} is a point in \mathcal{S} ; therefore, the Steiner minimal tree for \mathcal{S} is the SMT for R_1 and the SMT for R_2 . Since \mathcal{X} is in both R_1 and R_2 , the two trees are connected.

4.2 The Steiner Hull

The next decomposition theorem is due to Cockayne [12] and is based upon what he termed the *Steiner hull*. The Steiner hull is defined as follows: Let P_1 be the convex hull. P_{i+1} is constructed from P_i by finding an edge (p, r) of P_i that has a vertex (q) near it such that $\angle pqr \geq 120^\circ$, and there is not a vertex inside the triangle pqr . The final polygon, P_i , that can be created in such a way is called the Steiner hull. The algorithm for this construction is shown in Fig. 10. The Steiner hull for the 100 points shown in Fig. 8 is given in Fig. 11.

After defining the Steiner hull, Cockayne showed that the SMT for \mathcal{S} must lie within the Steiner hull of \mathcal{S} . This presents us with the following decomposition: The Steiner hull can be thought of as an ordered sequence of points, $\{p_1, p_2, \dots, p_n\}$, where the hull is defined by the sequence of line segments, $\{p_1p_2, p_2p_3, \dots, p_np_1\}$. If there exists a point p_i that occurs twice in the Steiner hull, then the problem can be decomposed at point p_i . If a Steiner hull contains such a point, then the Steiner hull is referred to as *degenerate*. This decomposition is accomplished by showing that the Steiner hull splits \mathcal{S} into two contained subsets, R_1 and R_2 , where R_1 is the set of points contained in the Steiner hull from the first time p_i appears until the last time p_i appears, and R_2 is the set of points contained in the Steiner hull from the last time p_i appears until the first time p_i appears. With this decomposition it can be shown that $\mathcal{S} = R_1 \cup R_2$, and the SMT for \mathcal{S} is the union of the SMT for R_1 and the SMT for R_2 . This decomposition is illustrated in Fig. 12. Cockayne also proved that the Steiner hull decomposition includes every decomposition possible with the Double Wedge Theorem.

In their work on 100-point problems, Cockayne and Hewgill [14] mention that approximately 15 % of the randomly generated 100-point problems have degenerate

Fig. 10 The Steiner hull algorithm

```

The initial Steiner Polygon,  $P_1$ , is the Convex Hull.
Repeat
    Create Next Steiner Polygon  $P_{i+1}$  from  $P_i$  by
        1) find a set of points  $pqr \in S$  such that:
             $p$  and  $r$  are adjacent on  $P_i$ 
             $\angle pqr \geq 120^\circ$ 
             $\nexists$  a point from  $S$  in the triangle  $pqr$ 
        2) remove the edge  $pr$ .
        3) add edges  $pq$  and  $qr$ .
Until( $P_i == P_{i+1}$ )
Steiner Hull =  $P_i$ 
```

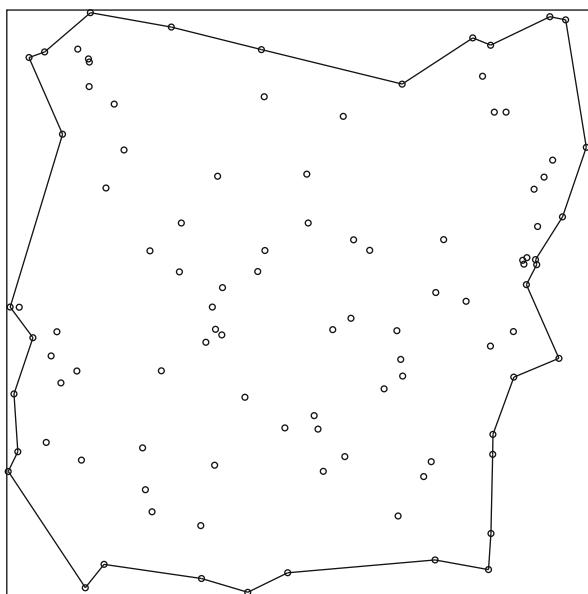


Fig. 11 The Steiner hull for a random set of 100 points

Steiner Hull's. The Steiner hull shown in Fig. 11 is not degenerate, while that in Fig. 12 is.

4.3 The Steiner Hull Extension

The final decomposition belongs to Hwang et al. [39]. They proposed an extension to the Steiner hull as defined by Cockayne. Their extension is as follows: If there exist four points a, b, c , and d on a Steiner hull such that:

Fig. 12 An illustration of the Steiner hull decomposition

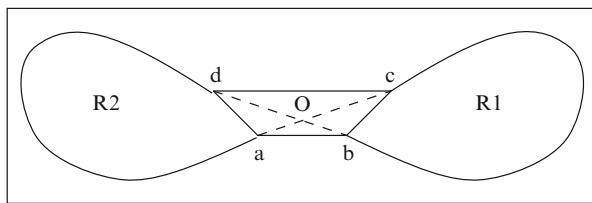
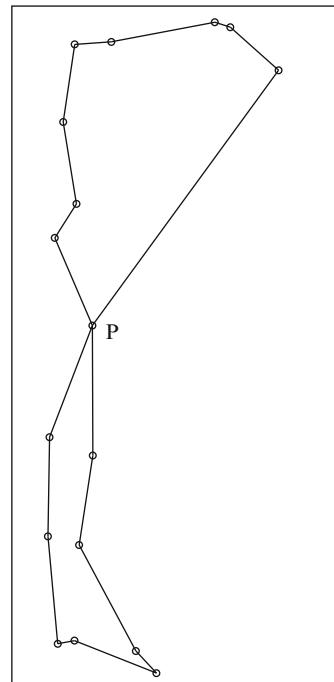


Fig. 13 An illustration of the Steiner hull extension

1. a, b, c , and d form a convex quadrilateral
2. There does not exist a point from \mathcal{S} in the quadrilateral (a, b, c, d)
3. $\angle a \geq 120^\circ$ and $\angle b \geq 120^\circ$
4. The two diagonals (ac) and (bd) meet at O , and $\angle bOa \geq \angle a + \angle b - 150^\circ$, then the SMT for \mathcal{S} is the union of the SMTs for R_1 and R_2 and the edge ab where R_1 is the set of points contained in the Steiner hull from c to b with the edge bc and R_2 is the set of points contained in the Steiner polygon from a to d with the edge ad . This decomposition is illustrated in Fig. 13.

These three decomposition theorems were combined into a parallel algorithm for decomposition presented in [28].

5 Winter's Sequential Algorithm

5.1 Overview and Significance

The development of the first working implementation of Melzak's algorithm sparked a move into the computerized arena for the calculation of SMTs. As we saw in [Section 2](#), Cockayne and Schiller [15] had implemented Melzak's algorithm and could calculate the SMT for all arrangements of 7 points. This was followed almost immediately by Boyce and Seery's program which they called STEINER72 [6]. Their work done at Bell Labs could calculate the SMT for all 10-point problems. They continued to work on the problem and in personal communication with Cockayne said they could solve 12-point problems with STEINER73. Yet even with quite a few people working on the problem, the number of points that any program could handle was still very small.

As mentioned toward the end of [Section 2](#), Melzak's algorithm yields invalid answers and invalid tree structures for quite a few combinations of points. It was not until 1981 that anyone was able to characterize a few of these invalid tree structures. These characterizations were accomplished by Paweł Winter and were based upon several geometric constructions which enable one to eliminate many of the possible combinations previously generated. He implemented these improvements in a program called GeoSteiner [60]. In his work he was able to calculate in under 30 s SMTs for problems having up to 15 vertices and stated that "with further improvements, it is reasonable to assert that point sets of up to 30 V-points could be solved in less than an hour [60]."

5.2 Winter's Algorithm

Winter's breakthrough was based upon two things: the use of extended binary trees and what he termed *pushing*. Winter proposed an extended binary tree as a means of constructing trees only once and easily identifying a full Steiner tree (FST: trees with n vertices and $n - 2$ Steiner points) on the same set of vertices readily.

Pushing came from the geometric nature of the problem and is illustrated in [Fig. 14](#). It was previously known that the Steiner point for a pair of points, a and b , would lie on the circle that circumscribed that pair and their equilateral third point. Winter set out to limit this region even further. This limitation was accomplished by placing a pair of points, a' and b' , on the circle at a and b , respectively, and attempting to push them closer and closer together. In his work Winter proposed and proved various geometric properties that would allow you to push a' toward b and b' toward a . If the two points ever crossed, then it was impossible for the current branch of the sample space tree to contain a valid answer.

Unfortunately, the description of Winter's algorithm is not as clear as one would hope, since the presence of `goto` statements rapidly makes his program difficult to understand and almost impossible to modify. Winter's goal is to build a list of

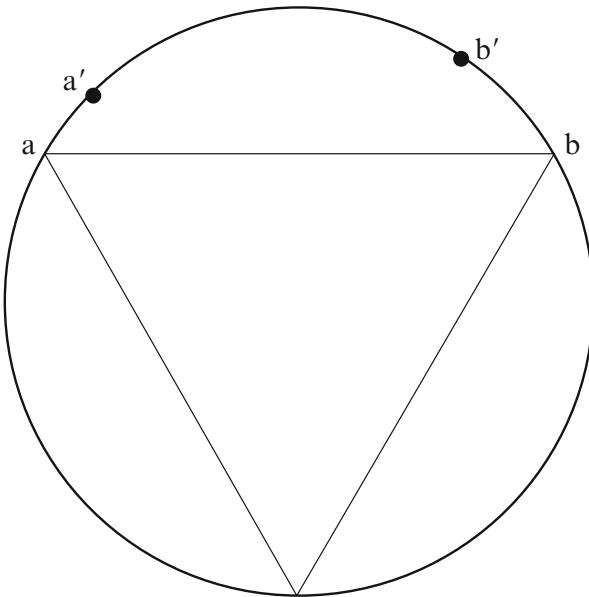


Fig. 14 An illustration of Winter’s pushing

FSTs which are candidates for inclusion in the final answer. This list, called T_list , is primed with the edges of the MST, thereby guaranteeing that the length of the SMT does not exceed the length of the MST.

The rest of the algorithm sets about to expand what Winter termed as Q_list , which is a list of partial trees that the algorithm attempts to combine until no combinations are possible. Q_list is primed with the original input points. The legality of a combination is determined in the *construct* procedure, which uses *pushing* to eliminate cases. While this combination proceeds, the algorithm also attempts to take newly created members of Q_list and create valid FSTs out of them. These FSTs are then placed onto T_list .

This algorithm was a turning point in the calculation of SMTs. It sparked renewed interest into the calculation of SMTs in general. This renewed interest has produced new algorithms such as the negative edge algorithm [57] and the luminary algorithm [36]. Winter’s algorithm has also served as the foundation for most computerized computation for calculating SMTs and is the foundation for the parallel algorithm we present in Section 6.

5.3 Algorithm Enhancements

In 1996, Winter and Zachariasen presented GEOSTEINER96 [61, 62] an enhancement to their exact algorithm that strongly improved the pruning and concatenation techniques of the GEOSTEINER algorithm just presented. This new algorithm modified the pruning tests to exploit the geometry of the problem (wedge property,

bottleneck Steiner distances) to yield effective and/or faster pruning of nonoptimal full Steiner trees (FSTs). Furthermore, efficient concatenation of FSTs was achieved by new and strong compatibility tests that utilize pairwise and subset compatibility along with very powerful preprocessing of surviving FSTs. GEOSTEINER96 has been implemented in C++ on an HP9000 workstation and solves randomly generated problem instances with 100 terminals in less than 8 min and up to 140 terminals within an hour. The hardest 100-terminal problem was solved in less than 29 min. Previously unsolved public library instances (OR-Library [3, 4]) have been solved by GEOSTEINER96 within 14 min. The authors point out that the concatenation of FSTs still remains the bottleneck of both GEOSTEINER96 and GEOSTEINER algorithms. However, the authors show that FSTs are generated 25 times faster by GEOSTEINER96 than by EDSTEINER89.

In their follow-up work [58], Winter and Zachariasen presented performance statistics for the exact SMT problem solved using the Euclidean FST generator from Winter and Zachariasen's algorithm [61, 62] and the FST concatenator of Warme's algorithm [59]. Optimal solutions have been obtained by this approach for problem instances of up to 2,000 terminals. Extensive computational experiences for randomly generated instances [100–500 terminals], public library instances (OR-Library [100–1,000 terminals] [3, 4], TSPLIB [198–7,397 terminals] [34]), and difficult instances with special structure have been shared in this work. The computational study has been conducted on an HP9000 workstation; the FST generator was implemented in C++ and the FST concatenator was implemented in C using CPLEX. Results indicate that (1) Warme's FST concatenation solved by branch-and-cut is orders of magnitude faster than backtrack search or dynamic programming based FST concatenation algorithms and (2) the Euclidean FST generator is more effective on uniformly randomly generated problem instances than for structured real-world instances.

6 A Parallel Algorithm

6.1 An Introduction to Parallelism

Parallel computation is allowing us to look at problems that have previously been impossible to calculate, as well as allowing us to calculate faster than ever before problems we have looked at for a long time. It is with this in mind that we begin to look at a parallel algorithm for the Steiner minimal tree problem.

There have been volumes written on parallel computation and parallel algorithms; therefore, we will not rehash the material that has already been so excellently covered by many others more knowledgeable on the topic, but will refer the interested readers to various books currently available. For a thorough description of parallel algorithms, and the PRAM model, the reader is referred to the book by Joseph JáJá [40], and for a more practical approach to implementation on a parallel machine, the reader is referred to the book by Vipin Kumar et al. [43], the book by Michael Quinn [51], or the book by Justin Smith [55].

6.2 Overview and Proper Structure

When attempting to construct a parallel algorithm for a problem, the sequential code for that problem is often the starting point. In examining sequential code, major levels of parallelism may become self-evident. Therefore, for this problem the first thing to do is to look at Winter's algorithm and convert it into structured code without `gos`. The initialization (step 1) does not change, and the translation of steps 2–7 appears in [Fig. 15](#).

Notice that the code in [Fig. 15](#) lies within a `for` loop. In a first attempt to parallelize anything, you typically look at loops that can be split across multiple processors. Unfortunately, upon further inspection, the loop continues while $p < q$ and, in the large if statement in the body of the loop, is the statement `q++` (line 30). This means that the number of iterations is data dependent and is not fixed at the outset. This loop cannot be easily parallelized.

Since the sequential version of the code does not lend itself to easy parallelization, the next thing to do is to back up and develop an understanding of how the algorithm works. The first thing that is obvious from the code is that you select a left subtree and then try to mate it with possible right subtrees. Upon further examination we come to the conclusion that a left tree will mate with all trees that are shorter than it and all trees of the same height that appear after it on `Q_list`, but it will never mate with any tree that is taller.

6.3 First Approach

The description of this parallel algorithm is in a master–slave perspective. This perspective was taken due to the structure of most parallel architectures at the time of its development, as well as the fact that all nodes on the `Q_list` need a sequencing number assigned to them. The master will therefore be responsible for numbering the nodes and maintaining the main `Q_list` and `T_list`.

The description from the slave's perspective is quite simple. A process is spawned off for each member of `Q_list` that is a proper left subtree (Winter's algorithm allows members of `Q_list` that are not proper left subtrees). Each new process is then given all the current nodes on `Q_list`. With this information the slave then can determine with which nodes its left subtree could mate. This mating creates new nodes that are sent back to the master, assigned a number, and added to the master's `Q_list`. The slave also attempts to create an FST out of the new `Q_list` member, and if it is successful, this FST is sent to the master to be added to the `T_list`. When a process runs out of `Q_list` nodes to check, it sends a request for more nodes to the master.

The master also has a simple job description. It has to start a process for each initial member of the `Q_list`, send them all the current members of the `Q_list`, and wait for their messages.

```

/* Step 2 */
1 for(p=0; p<q; p++){
2     AP = A(p);
3     /* Step 3 */
4     for(r=0; ((H(p) > H(r)) AND (r!=q)); r++) {
5         if((H(p) == H(r)) AND (r<p)) {
6             r = p;
7             if(Subset(V(r), AP)) {
8                 p_star = p;
9                 r_star = r;
10                for(Label = PLUS; Label <= MINUS; Label++) {
11                    /* Step 4 */
12                    AQ = A(q);
13                    if(Construct(p_star,r_star,&(E(q)))) {
14                        L(q) = p;
15                        R(q) = r;
16                        LBL(q) = Label;
17                        LF(q) = LF(p);
18                        H(q) = H(p) + 1;
19                        /* next line is different */
20                        Min(q) = max(Min(p)-1,H(r));
21                        if(Lsp(p) != 0)
22                            Lsp(q) = Lsp(p)
23                        else
24                            Lsp(q) = Lsp(r)
25                        if(Rsp(r) != 0)
26                            Rsp(q) = Rsp(r)
27                        else
28                            Rsp(q) = Rsp(p)
29                        q_star = q;
30                        q++;
31                        /* Step 5 */
32                        if(Proper_to_Add_Tree_to_Tlist(q_star)) {
33                            for_all(j in AP with Lf(R(q_star)) < j) {
34                                SRoot(t) = j;
35                                Root(t) = q_star;
36                                t++;
37                            }
38                        }
39                    }
40                    /* Step 6 */
41                    p_star = r;
42                    r_star = p;
43                }
44            }
45        }
46    }

```

Fig. 15 The main loop properly structured

This structure worked quite well for smaller problems (up to about 15 points), but for larger problems it reached a grinding halt quite rapidly. This was due to various reasons such as the fact that for each slave started the entire Q_list had to be sent. This excessive message passing quickly bogged down the network. Secondly, in their work on 100-point problems, Cockayne and Hewgill [14] made the comment that T_list has an average length of 220, but made no comment about the size of Q_list, which is the number of slaves that would be started. From our work on 100 point problems this number easily exceeds 1,000 which means that over 1,000 processes are starting, each being sent the current Q_list. From these few problems, it is quite easy to see that some major changes needed to be made in order to facilitate the calculation of SMTs for large problems.

6.4 Current Approach

The idea for a modification to this approach came from a paper by Quinn and Deo [52], on parallel algorithms for Branch-and-Bound problems. Their idea was to let the master have a list of work that needs to be done. Each slave is assigned to a processor. Each slave who requests work, is given some, and during its processing creates more work to be done. This new work is placed in the master's work list, which is sorted in some fashion. When a slave runs out of work to do, it requests more from the master. They noted that this leaves some processors idle at times (particularly when the problem was starting and stopping), but this approach provides the best utilization if all branches are independent.

This description almost perfectly matches the problem at hand. First, we will probably have a fixed number of processors which can be determined at runtime. Second, we have a list of work that needs to be done. The hard part is implementing a sorted work list in order to obtain a better utilization. This was implemented in what we term the Proc_list, which is a list of the processes that either are currently running or have not yet started. This list is primed with the information about the initial members of Q_list, and for every new node put on the Q_list, a node which contains information about the Q_list node is placed on the Proc_list in a sorted order.

The results for this approach are quite exciting, and the timings are discussed in [Section 8](#).

7 Extraction of the Correct Answer

7.1 Introduction and Overview

Once the T_list discussed in [Sect. 5](#) is formed, the next step is to extract the proper answer from it. Winter described this in step 7 of his algorithm. His description stated that unions of FSTs saved in T_list were to be formed subject to constraints described in his paper. The shortest union is the SMT for the original points.

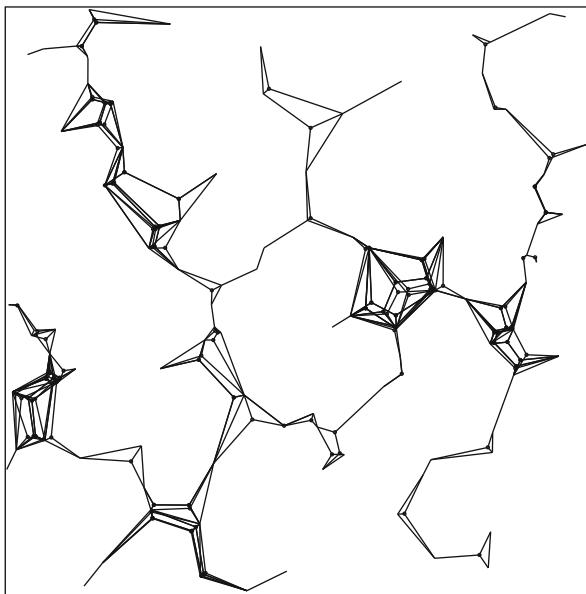


Fig. 16 T_list for a random set of points

The constraints he described were quite obvious considering the definition of an SMT. First, the answer had to cover all the original points. Second, the union of FSTs could not contain a cycle. Third, the answer is bounded in length by the length of the MST.

This led Winter to implement a simple exhaustive search algorithm over the FSTs in T_list. This approach yields a sample space of size $\mathcal{O}(2^m)$ (where m is the number of trees in T_list) that has to be searched. This exponentiality is born out in his work where he stated that for problems with more than 15 points “the computation time needed to form the union of FSTs dominates the computation time needed for the construction of the FSTs [60].” An example of the input the last step of Winter’s algorithm receives (T_list) is given in Fig. 16. The answer it extracts (the SMT) is shown in Fig. 17.

7.2 Incompatibility Matrix

Once Cockayne published the clarification of Melzak’s proof in 1967 [11] and Gilbert and Pollak published their paper giving an upper bound the SMT length in 1968 [26], many people were attracted to this problem. From this time until Winter’s work was published in 1985 [60], quite a few papers were published dealing with various aspects of the SMT problem, but the attempt to computerize the solution of the SMT problem bogged down around 12 vertices. It wasn’t until Winter’s

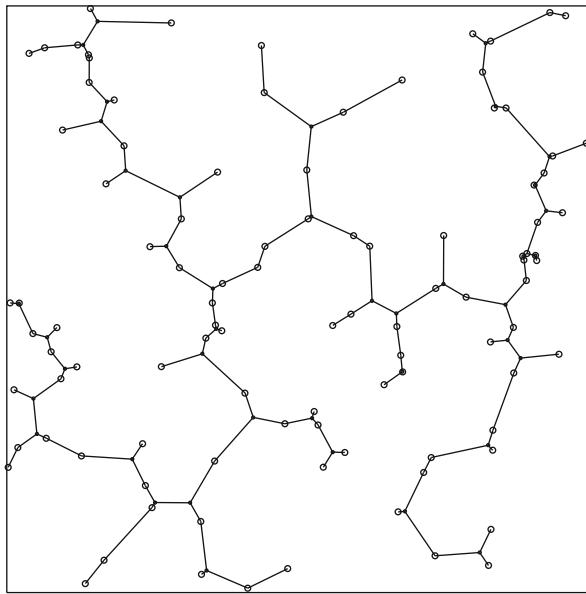


Fig. 17 SMT extracted from T_list for a random set of points

algorithm was published that the research community received the spark it needed to work on computerized computation of the SMT problem with renewed vigor. With the insight Winter provided into the problem, an attempt to computerize the solution of the SMT problem began anew.

Enhancement of this algorithm was first attempted by Cockayne and Hewgill at the University of Victoria. For this implementation Cockayne and Hewgill spent most of their work on the back end of the problem, or the extraction from T_list, and used Winter's algorithm to generate T_list. This work on the extraction focused on what they termed an *incompatibility matrix*. This matrix had one row and one column for each member of T_list. The entries in this matrix were flags corresponding to one of three possibilities: *compatible*, *incompatible*, or *don't know*. The rationale behind the construction of this matrix is the fact that it is faster to look up the value in a matrix than it is to check for the creation of cycles and improper angles during the union of FSTs.

The first value calculations for this matrix were straightforward. If two trees do not have any points in common, then we *don't know* if they are incompatible or not. If they have two or more points in common, then they form a cycle and are *incompatible*. If they have only one point in common and the angle at the intersection point is less than 120°, then they are also *incompatible*. In all other cases they are *compatible*.

This simple enhancement to the extraction process enabled Cockayne and Hewgill to solve all randomly generated problems of size up to 17 vertices in a little over 3 min [13].

7.3 Decomposition

The next focus of Cockayne and Hewgill's work was in the area of the decomposition of the problem. As was discussed earlier in Sect. 4, the best theorems for any problem, especially non-polynomial problems, are those of the form “If property \mathcal{P} exists then the problem can be decomposed.” Since the formation of unions of FSTs is exponential in nature, any theorem of this type is important.

Cockayne and Hewgill's theorem states: “Let A_1 and A_2 be subsets of A satisfying (a) $A_1 \cup A_2 = A$ (b) $|A_1 \cap A_2| = 1$ and (c) the leaf set of each FST in T_list is entirely contained in A_1 or A_2 . Then any SMT on A is the union of separate SMTs on A_1 and A_2 [13].” This means that if you break T_list into biconnected components, the SMT will be the union of the SMTs on those components.

Their next decomposition theorem allowed further improvements in the calculation of SMTs. This theorem stated that if you had a component of T_list left from the previous theorem and if the T_list members of that component form a cycle, then it might be possible to break that cycle and apply the previous algorithm again. The cycle could be broken if there existed a vertex v whose removal would change that component from one biconnected component to more than one.

With these two decomposition theorems, Cockayne and Hewgill were able to calculate the SMT for 79 of 100 randomly generated 30-point problems. The remaining 21 would not decompose into blocks of size 17 or smaller and thus would have taken too much computation time [13]. This calculation was implemented in the program they called EDSTEINER86.

7.4 Forest Management

Cockayne and Hewgill's next work focused on improvements to the *incompatibility matrix* previously described and was implemented in a program called EDSTEINER89. Their goal was to reduce the number of *don't know's* in the matrix and possibly remove some FSTs from T_list altogether.

They proposed two refinements for calculating the entry into the *incompatibility matrix* and one Tree Deletion Theorem. The Tree Deletion Theorem stated that if there exists an FST in T_list that is incompatible with all FSTs containing a certain point a , then the original FST can be deleted since at least one FST containing a will be in the SMT.

This simple change allowed Cockayne and Hewgill to calculate the SMT for 77 of 100 randomly generated 100-point problems [14]. The other 23 problems could not be calculated in less than 12 h and were therefore terminated. For those that did complete, the computation time to generate T_list had become the dominate factor in the overall computation time.

So the pendulum had swung back from the extraction of the correct answer from T_list to the generation of T_list dominating the computation time. In Sect. 8 we will look at the results of the parallel algorithm presented in Sect. 9 to see if the pendulum can be pushed back the other way one more time.

8 Computational Results

8.1 Previous Computation Times

Before presenting the results for the parallel algorithm presented in Sect. 6, it is worthwhile to review the computation times that have preceded this algorithm in the literature. The first algorithm for calculating FSTs was discussed in a paper by Cockayne [12] where he mentioned that preliminary results indicated his code could solve any problem up to 30 points that could be decomposed with the Steiner hull into regions of 6 points or less.

As we saw in Sect. 2, the next computational results were presented by Cockayne and Schiller [15]. Their program, called STEINER, was written in FORTRAN on an IBM 360/50 at the University of Victoria. STEINER could calculate the SMT for any 7-point problem in less than 5 min of CPU time. When the problem size was increased to 8, it could solve them if 7 of the vertices were on the Steiner hull. When this condition held it could calculate the SMT in under 10 min, but if this condition did not hold it would take an unreasonable amount of time.

Cockayne called STEINER a prototype for calculating SMTs and allowed Boyce and Serry of Bell Labs to obtain a copy of his code to improve the work. They improved the code, renamed it STEINER72, and were able to calculate the FST for all 9-point problems and most 10-point problems in a reasonable amount of time [6]. Boyce and Serry continued their work and developed another version of the code that they thought could solve problems of size up to 12 points, but no computation times were given.

The breakthrough we saw in Sect. 5 was by Pawel Winter. His program called GEOSTEINER [60] was written in SIMULA 67 on a UNIVAC-1100. GEOSTEINER could calculate SMTs for all randomly generated sets with 15 points in under 30 s. This improvement was put into focus when he mentioned that all previous implementations took more than an hour for nondegenerate problems of size 10 or more. In his work, Winter tried randomly generated 20-point problems but did not give results since some of them did not finish in his CPU time limit of 30 s. The only comment he made for problems bigger than size 15 was that the extraction discussed in Sect. 7 was dominating the overall computation time.

The next major program, EDSTEINER86, was developed in FORTRAN on an IBM 4381 by Cockayne and Hewgill [13]. This implementation was based upon Winter's results, but had enhancements in the extraction process. EDSTEINER86 was able to calculate the FST for 79 out of 100 randomly generated 32-point problems. For these problems the CPU time for T_list varied from 1 to 5 min, while for the 79 problems that finished the extraction time never exceeded 70 s.

Cockayne and Hewgill subsequently improved their SMT program and renamed it EDSTEINER89 [14]. This improvement was completely focused on the extraction process. EDSTEINER89 was still written in FORTRAN, but was run on a SUN 3/60 workstation. They randomly generated 200 32-point problems to solve and found that the generation of T_list dominated the computation time for problems of this size. The average time for T_list generation was 438 s, while the average time for

Table 2 SMT programs, authors, and results

Program	Author(s)	Points
STEINER	Cockayne & Schiller Univ of Victoria	7
STEINER72	Boyce & Serry ATT Bell Labs	10
STEINER73	Boyce & Serry ATT Bell Labs	12
GEOSTEINER	Winter Univ of Copenhagen	15
EDSTEINER86	Cockayne & Hewgill Univ of Victoria	30
EDSTEINER89	Cockayne & Hewgill Univ of Victoria	100
PARSTEINER94	Harris Univ of Nevada	100

forest management and extraction averaged only 43 s. They then focused on 100-point problems and set a CPU limit of 12 h. The average CPU time to generate T_list was 209 min for these problems, but only 77 finished the extraction in the CPU time limit. These programs and their results are summarized in [Table 2](#).

8.2 The Implementation

8.2.1 The Significance of the Implementation

The parallel algorithm we presented has been implemented in a program called PARSTEINER94 [28, 31]. This implementation is only the second SMT program since Winter’s GEOSTEINER in 1981 and is the first parallel code. The major reason that the number of SMT programs is so small is due to the fact that any implementation is necessarily complex.

PARSTEINER94 currently has over 13,000 lines of C code. While there is a bit of code dealing with the parallel implementation, certain sections of Winter’s algorithm have a great deal of code buried beneath the simplest statements. For example, line 13 of [Fig. 15](#) is the following:

```
if (Construct(p_star,r_star,&(E(q)))) { .
```

To implement the function `Construct()` over 4,000 lines of code were necessary, and this does not include the geometry library with functions such as `equilateral_third_point()`, `center_of_equilateral_triangle()`, `line_circle_intersect()`, and a host more.

Another important aspect of this implementation is the fact that there can now be comparisons made between the two current SMT programs. This would allow verification checks to be made between EDSTEINER89 and PARSTEINER94. This

verification is important since with any complex program it is quite probable that there are a few errors hiding in the code. This implementation would also allow other SMT problems, such as those we will discuss in [Sect. 9](#), to be explored independently, thereby broadening the knowledge base for SMTs even faster.

8.2.2 The Platform

In the design and implementation of parallel algorithms, you are faced with many decisions. One such decision is what will your target architecture be? There are times when this decision is quite easy due to the machines at hand or the size of the problem. In our case we decided not to target a specific machine, but an architectural platform called PVM [19].

PVM, which stands for Parallel Virtual Machine, is a software package available from Oak Ridge National Laboratory. This package allows a collection of parallel or serial machines to appear as a large distributed memory computational machine (MIMD model). This is implemented via two major pieces of software, a library of PVM interface routines, and a PVM demon that runs on every machine that you wish to use.

The library interface comes in two languages, C and ORTRAN. The functions in this library are the same no matter which architectural platform you are running on. This library has functions to spawn off (start) many copies of a particular program on the parallel machine, as well as functions to allow message passing to transfer data from one process to another. Application programs must be linked with this library to use PVM.

The demon process, called *pvmd* in the user's guide, can be considered the back end of PVM. As with any back end, such as the back end of a compiler, when it is ported to a new machine, the front end can interface to it without change. The back end of PVM has been ported to a variety of machines, such as a few versions of Crays, various Unix machines such as Sun workstations, HP machines, Data General workstations, and DEC Alpha machines. It has also been ported to a variety of true parallel machines such as the iPSC/2, iPSC/860, CM2, CM5, BBN Butterfly, and the Intel Paragon.

With this information it is easy to see why PVM was picked as the target platform. Once a piece of code is implemented under PVM, it can be recompiled on the goal machine, linked with the PVM interface library on that machine, and run without modification. In our case we designed PARSTEINER94 on a network of SUN workstations, but, as just discussed, moving to a large parallel machine should be trivial.

8.2.3 Errors Encountered

When attempting to implement any large program from another person's description, you often reach a point where you don't understand something. At first you always question yourself, but as you gain an understanding of the problem you learn that there are times when the description you were given is wrong. Such was the case with the SMT problem. Therefore, to help some of those that may come along and

attempt to implement this problem after us, we recommend that you look at the list of errors we found while implementing Winter’s algorithm [28].

8.3 Random Problems

8.3.1 Hundred-Point Random Problems

From the literature it is obvious that the current standard for calculating SMTs has been established by Cockayne and Hewgill. Their work on SMTs has pushed the boundary of computation out from the 15-point problems of Winter to being able to calculate SMTs for a large percentage of 100-point problems.

Cockayne and Hewgill, in their investigation of the effectiveness of EDSTEINER89, randomly generated 100 problems with 100 points inside the unit square. They set up a CPU limit of 12 h, and 77 of 100 problems finished within that limit. They described the average execution times as follows: T-list construction averaged 209 min, forest management averaged 27 min, and extraction averaged 10.8 min.

While preparing the code for this project, Cockayne and Hewgill were kind enough to supply us with 40 of the problems generated for [14] along with their execution times. These data sets were given as input to the parallel code PARSTEINER94, and the calculation was timed. The wall clock time necessary to generate T-list for the two programs appears in Table 3. For all 40 cases, the average time to generate T-list was less than 20 min. This is exciting because we have been able to generate T-list properly while cutting an order of magnitude off the time.

These results are quite promising for various reasons. First, the parallel implementation presented in this work is quite scalable and therefore could be run with many more processors, thereby enhancing the speedup provided. Second, with the PVM platform used, we can in the future port this work to a real parallel MIMD machine, which will have much less communication overhead, or to a shared memory machine, where the communication could all but be eliminated, and expect the speedup to improve much more.

It is also worth noting that proper implementation of the cycle breaking which Cockayne and Hewgill presented in [13] is important if extraction of the proper answer is to be accomplished. In their work, Cockayne and Hewgill mentioned that 58 % of the problems they generated were solvable without the cycle breaking being implemented, which is approximately what we have found with the data sets they provided. An example of such a T-list that would need cycles broken (possibly multiple times) is provided in Fig. 18.

8.3.2 Larger Random Problems

Once the 100-point problems supplied by Cockayne and Hewgill had been successfully completed, the next step was to try a few larger problems. This was done with the hope of gaining an insight into the changes that would be brought about from the addition of more data points.

Table 3 Comparison of T_list times

Test case	PARSTEINER94	EDSTEINER89
1	650	8,597
2	1,031	13,466
3	1,047	15,872
4	1,687	17,061
5	874	13,258
6	1,033	15,226
7	1,164	12,976
8	1,109	16,697
9	975	15,354
10	554	8,650
11	660	9,894
12	946	13,057
13	858	13,687
14	978	17,132
15	819	11,333
16	752	12,766
17	896	13,815
18	788	10,508
19	618	10,550
20	724	11,193
21	983	11,357
22	889	12,999
23	1,449	15,028
24	890	14,417
25	912	17,562
26	1,125	12,395
27	943	15,721
28	583	10,014
29	1,527	18,656
30	681	10,033
31	873	16,401
32	791	10,217
33	1,132	18,635
34	1,097	18,305
35	1,198	19,657
36	803	11,174
37	923	15,256
38	824	12,920
39	826	12,538
40	972	15,570
Avg.	939	13,748

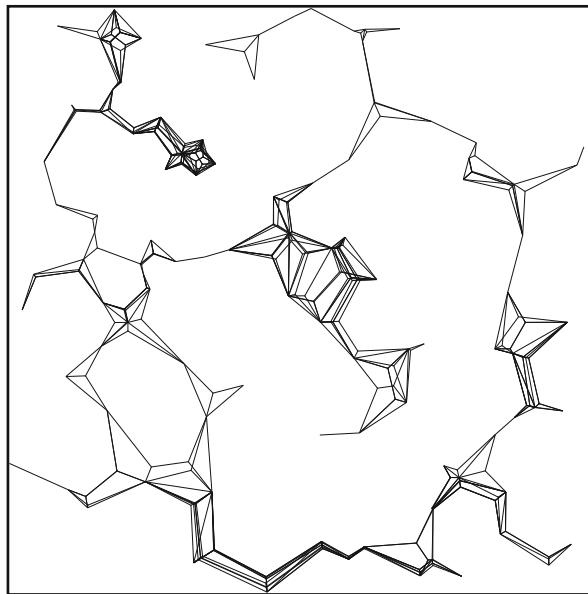


Fig. 18 T_list with more than 1 cycle

For this attempt we generated several random sets of 110 points each. The length of T_list increased by approximately 38 %, from an average of 210 trees to an average of 292 trees. The time to compute T_list also increased drastically, going from an average of 15 min to an average of more than 40 min.

The interesting thing that jumped out the most was the increase in the number of large biconnected components. Since the extraction process must do a complete search of all possibilities, the larger the component, the longer it will take. This is a classic example of an exponential problem, where when the problem size increases by 1, the time doubles. With this increased component size, none of the random problems generated finished inside a 12 h cut off time.

This rapid growth puts into perspective the importance of the work previously done by Cockayne and Hewgill. Continuation of their work with incompatibility matrices as well as decomposition of T_list components appears at this point to be very important for the future of SMT calculations.

8.4 Grids

The problem of determining SMTs for grids was mentioned to the author by Ron Graham. In this context we are thinking of a grid as a regular lattice of unit squares. The literature has little of information regarding SMTs on grids, and most of the information that is given is conjectured and not proven. In Sect. 8.4.1 we will

look at what is known about SMTs on grids. In the following subsections, we will introduce new results for grids up through $7 \times m$ in size. These results presented are computational results from PARSTEINER94 [28, 30, 31] which was discussed previously.

8.4.1 $2 \times m$ and Square Grids

The first proof for anything besides a 2×2 grid came in a paper by Chung and Graham [10] in which they proved the optimality of their characterization of SMTs for $2 \times m$ grids. The only other major work was presented in a paper by Chung, Gardner, and Graham [9]. They argued the optimality of the SMT on 2×2 , 3×3 , and 4×4 grids and gave conjectures and constructions for those conjectures for SMTs on all other square lattices.

In their work Chung, Gardner, and Graham specified three building blocks from which all SMTs on square $(n \times n)$ lattices were constructed. The first, labeled \mathcal{I} , is just a K_2 or a path on two vertices. This building block is given in Fig. 19a. The second, labeled \mathcal{Y} , is a full Steiner tree (FST) (n vertices and $n - 2$ Steiner points) on 3 vertices of the unit square. This building block is given in Fig. 19b. The third, labeled \mathcal{X} , is an FST on all 4 vertices of the unit square. This building block is given in Fig. 19c. For the generalizations we are going to make here, we need to introduce one more building block, which we will label \mathcal{S} . This building block is an FST on a 3×2 grid and appears in Fig. 19d.

SMTs for grids of size $2 \times m$ have two basic structures. The first is an FST on all the vertices in the $2 \times m$ grid. An example of this for a 2×3 grid is given in Fig. 19d. The other structure is constructed from the building blocks previously described. We hope that these building blocks, when put in conjunction with the generalizations for $3 \times m$, $4 \times m$, $5 \times m$, $6 \times m$, and $7 \times m$ will provide the foundation for a generalization of $m \times n$ grids in the future.

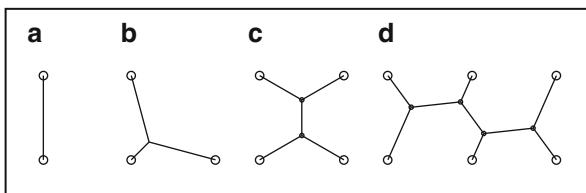


Fig. 19 Building blocks

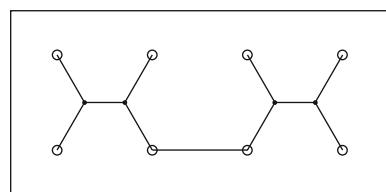


Fig. 20 SMT for a 2×4 grid

In their work on ladders ($2 \times m$ grids) Chung and Graham established and proved the optimality of their characterization for $2 \times m$ grids. Before giving their characterization, a brief review of the first few $2 \times m$ SMTs is in order. The SMT for a 2×2 grid is shown in Fig. 19c, the SMT for a 2×3 grid is shown in Fig. 19d, and the SMT for a 2×4 grid is given in Fig. 20.

Chung and Graham [10] proved that SMTs for ladders fell into one of two categories. If the length of the ladder was odd, then the SMT was the FST on the vertices of the ladder. The SMT for the 2×3 grid in Fig. 19d is an example of this. If the length of the ladder was even, the SMT was made up of a series of $(\frac{m}{2} - 1)$ $\mathcal{X}\mathcal{I}$ s followed by one last \mathcal{X} . The SMT for the 2×4 grid in Fig. 20 is an example of this.

8.4.2 $3 \times m$ Grids

The SMT for $3 \times m$ grids has a very easy characterization which can be seen once the initial cases have been presented. The SMT for the 3×2 grid is presented in Fig. 19d. The SMT for the 3×3 grid is presented in Fig. 21.

From here we can characterize all $3 \times m$ grids. Except for the 3×2 grid, which is an \mathcal{S} building block, there will be only two basic building blocks present, \mathcal{X} 's and \mathcal{I} 's. There will be exactly two \mathcal{I} 's and $(m - 1)\mathcal{X}$'s. The two \mathcal{I} 's will appear on each end of the grid. The \mathcal{X} 's will appear in a staggered checkerboard pattern, one on each column of the grid the same way that the two \mathcal{X} 's are staggered in the 3×3 grid. The 3×5 grid is a good example of this and is shown in Fig. 22.

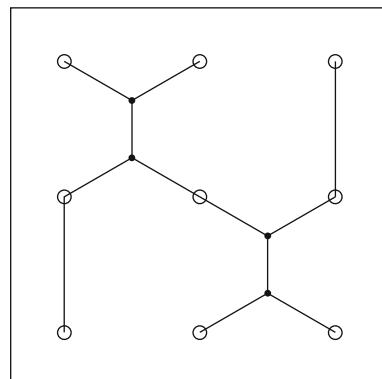


Fig. 21 SMT for a 3×3 grid

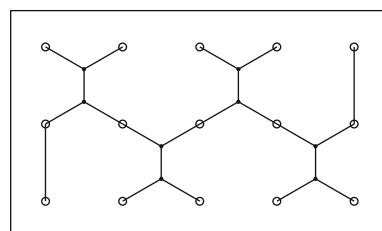


Fig. 22 SMT for a 3×5 grid

8.4.3 $4 \times m$ Grids

The foundation for the $4 \times m$ grids has already been laid. In their most recent work, Cockayne and Hewgill presented some results on square lattice problems [14]. They looked at $4 \times m$ grids for $m = 2$ to $m = 6$. They also looked at the SMTs for these problems when various lattice points in that grid were missing. What they did not do, however, was characterize the structure of the SMTs for all $4 \times m$ grids.

The 4×2 grid is given in Fig. 20. From the work of Chung et al. [9], we know that the SMT for a 4×4 grid is a checkerboard pattern of 5 \mathcal{X} 's. This layout gives us the first two patterns we will need to describe the $4 \times m$ generalization. The first pattern, which we will call pattern \mathcal{A} , is the same as the 3×4 grid without the two \mathcal{I} 's on the ends. This pattern is given in Fig. 23. The second pattern, denoted as pattern \mathcal{B} , is the 2×4 grid in Fig. 20 without the connecting \mathcal{I} . This is shown in Fig. 24.

Before the final characterization can be made, two more patterns are needed. The first one, called pattern \mathcal{C} , is a 4×3 grid where the pattern is made up of two non-connected 2×3 SMTs, shown in Fig. 25. The next pattern, denoted by pattern \mathcal{D} ,

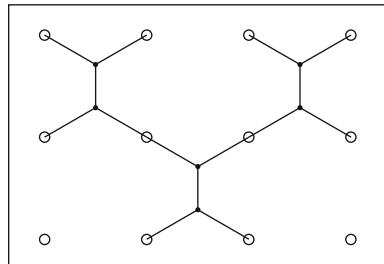


Fig. 23 $4 \times m$ pattern \mathcal{A}

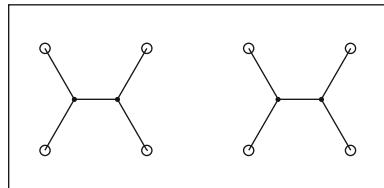


Fig. 24 $4 \times m$ pattern \mathcal{B}

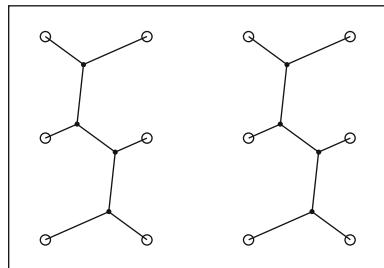
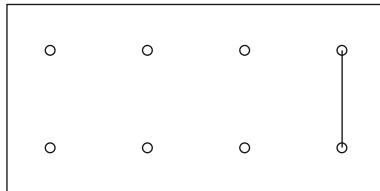
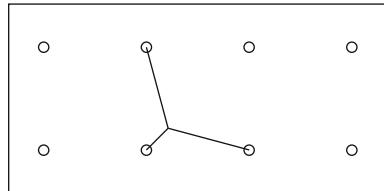


Fig. 25 $4 \times m$ pattern \mathcal{C}

Fig. 26 $4 \times m$ pattern \mathcal{D} **Fig. 27** $4 \times m$ pattern \mathcal{E} **Table 4** Rewrite rules for $4 \times m$ grids

1	$B \rightarrow C$
2	$C \rightarrow BD\mathcal{B}$
3	$DC \rightarrow EAB$

Table 5 String representations for $4 \times m$ grids

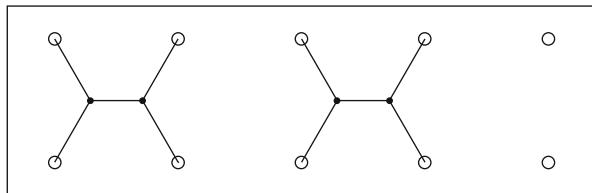
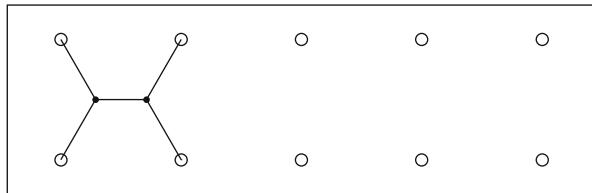
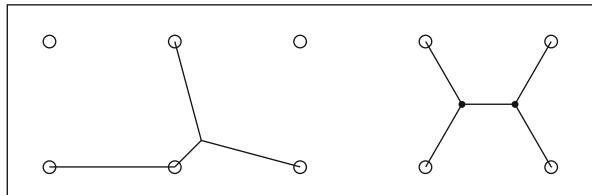
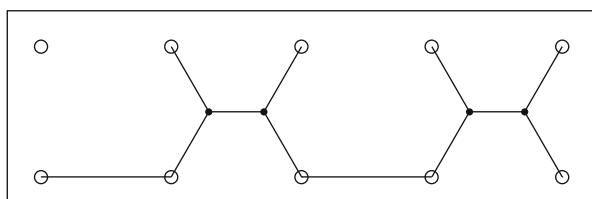
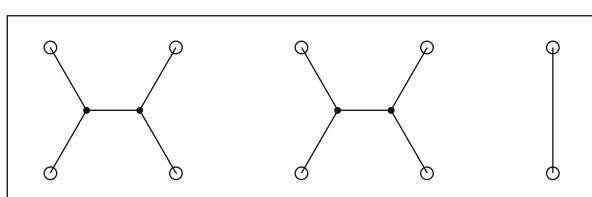
m	5	6	7	8
String	AC	$ABDB$	$ABDC$	$ABEA$
m	9	10	11	
String	$ABEAC$	$ABEABDB$	$ABEABDC$	

is quite simply a \mathcal{Y} centered in a 2×4 grid. This is shown in Fig. 26. The final pattern, denoted by \mathcal{E} , is just an \mathcal{I} on the right side of a 2×4 grid. This is shown in Fig. 27.

Now we can begin the characterization. The easiest way to present the characterization is with some simple string rewriting rules. Since the 4×2 , 4×3 , and 4×4 patterns have already been given, the rules will begin with a 4×5 grid. This grid has the string AC . The first rule is that whenever there is a C on the right end of your string, replace it with $BD\mathcal{B}$. Therefore, a 4×6 grid is $ABDB$. The next rule is that whenever there is a B on the right end of your string, replace it with a C . The final rule is whenever there is a DC on the right end of your string, replace it with an EAB . These rules are summarized in Table 4. A listing of the strings for m from 5 to 11 is given in Table 5.

8.4.4 $5 \times m$ Grids

For the $5 \times m$ grids, there are 5 building blocks (and their mirror images which are denoted with an ') that are used to generate any $5 \times m$ grid. These building blocks appear in Figs. 28–32.

**Fig. 28** $5 \times m$ pattern \mathcal{A} **Fig. 29** $5 \times m$ pattern \mathcal{B} **Fig. 30** $5 \times m$ pattern \mathcal{C} **Fig. 31** $5 \times m$ pattern \mathcal{D} **Fig. 32** $5 \times m$ pattern \mathcal{E}

With the building blocks in place, the characterization of $5 \times m$ grids is quite easy using grammar rewrite rules. The rules used for rewriting strings representing a $5 \times m$ grid are given in [Table 6](#). The SMTs for 5×2 , 5×3 , and 5×4 have already been given. For a 5×5 grid the SMT is made up of the following string: $\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{D}$. As a reminder, the \mathcal{A}' signifies the mirror of building block \mathcal{A} . A listing of the strings for m from 5 to 11 is given in [Table 7](#).

8.4.5 $6 \times m$ Grids

For the $6 \times m$ grids, there are five building blocks that are used to generate any $6 \times m$ grid. These building blocks appear in [Figs. 33–37](#).

The solution for $6 \times m$ grids can now be characterized by using grammar rewrite rules. The rules used for rewriting strings representing a $6 \times m$ grid are given in [Table 8](#). The basis for this rewrite system is the SMT for the 6×3 grid which is \mathcal{AC} .

Table 6 Rewrite rules for $5 \times m$ grids

1	$\mathcal{C} \rightarrow \mathcal{B}'\mathcal{D}'$
2	$\mathcal{D} \rightarrow \mathcal{A}'\mathcal{E}$
3	$\mathcal{E} \rightarrow \mathcal{AC}$
4	$\mathcal{C}' \rightarrow \mathcal{BD}$
5	$\mathcal{D}' \rightarrow \mathcal{AE}'$
6	$\mathcal{E}' \rightarrow \mathcal{AC}'$

Table 7 String representations for $5 \times m$ grids

m	5	6	7	8
String	$\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{D}$	$\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{E}$	$\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{C}$	$\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{D}'$
m	9	10	11	
String	$\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{A}\mathcal{E}'$	$\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{A}\mathcal{A}'\mathcal{C}'$	$\mathcal{E}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{A}\mathcal{A}'\mathcal{B}\mathcal{D}$	

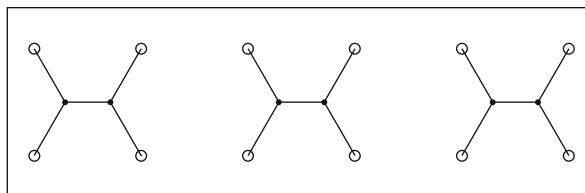


Fig. 33 $6 \times m$ pattern \mathcal{A}

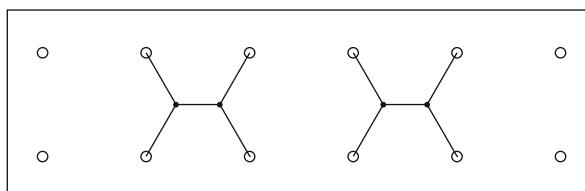
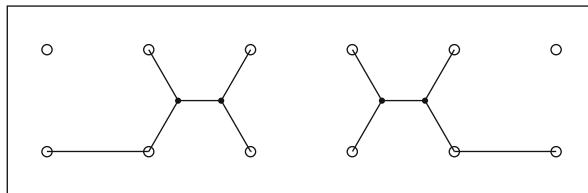
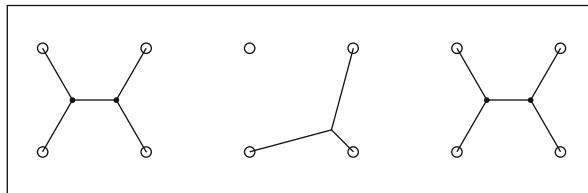
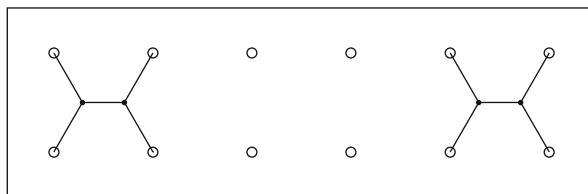


Fig. 34 $6 \times m$ pattern \mathcal{B}

**Fig. 35** $6 \times m$ pattern \mathcal{C} **Fig. 36** $6 \times m$ pattern \mathcal{D} **Fig. 37** $6 \times m$ pattern \mathcal{E} **Table 8** Rewrite rules for $6 \times m$ grids

1	$\mathcal{C} \rightarrow \mathcal{BD}$
2	$\mathcal{D} \rightarrow \mathcal{EC}$

Table 9 String representations for $6 \times m$ grids

$m =$	6	7	8
String	$\mathcal{ABE}\mathcal{BD}$	$\mathcal{ABE}\mathcal{BEC}$	$\mathcal{ABE}\mathcal{BEBD}$
$m =$	9	10	11
String	$\mathcal{ABE}\mathcal{BEBEC}$	$\mathcal{ABE}\mathcal{BEBEBD}$	$\mathcal{ABE}\mathcal{BEBEBEC}$

It is also nice to see that for the $6 \times m$ grids, there is a simple regular expression which can characterize what the string will be. That regular expression has the form $\mathcal{A}(\mathcal{BE})^*(\mathcal{C}|\mathcal{BD})$, which means that the \mathcal{BE} part can be repeated 0 or more times and the end can be either \mathcal{C} or \mathcal{BD} . A listing of the strings for m from 6 to 11 is given in Table 9.

8.4.6 $7 \times m$ Grids

For the $7 \times m$ grids, there are six building blocks that are used to generate any $7 \times m$ grid. These building blocks appear in Figs. 38–43.

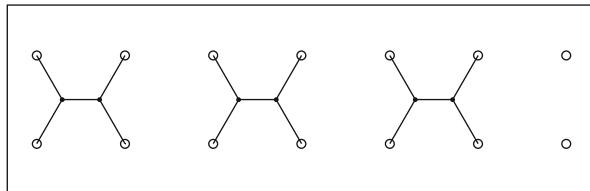


Fig. 38 $7 \times m$ pattern \mathcal{A}

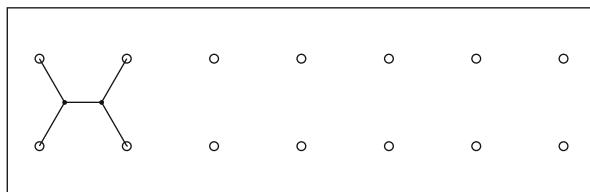


Fig. 39 $7 \times m$ pattern \mathcal{B}

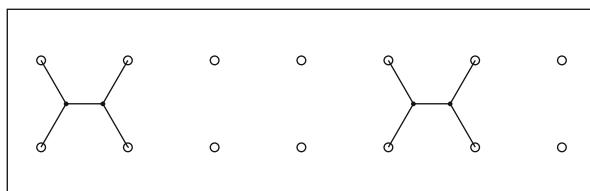


Fig. 40 $7 \times m$ pattern \mathcal{C}

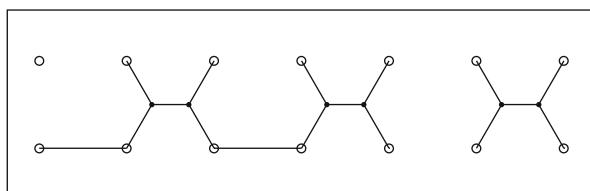
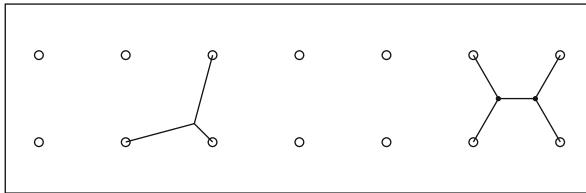
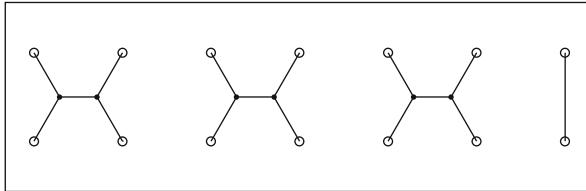


Fig. 41 $7 \times m$ pattern \mathcal{D}

**Fig. 42** $7 \times m$ pattern \mathcal{E} **Fig. 43** $7 \times m$ pattern \mathcal{F} **Table 10** Rewrite rules for $7 \times m$ grids

1	$\mathcal{E}'\mathcal{F}' \rightarrow \mathcal{B}\mathcal{A}'\mathcal{F}$
2	$\mathcal{F} \rightarrow \mathcal{C}\mathcal{D}$
3	$\mathcal{C}\mathcal{D} \rightarrow \mathcal{A}\mathcal{E}\mathcal{F}$
4	$\mathcal{E}\mathcal{F} \rightarrow \mathcal{B}'\mathcal{A}\mathcal{F}'$
5	$\mathcal{F}' \rightarrow \mathcal{C}'\mathcal{D}'$
6	$\mathcal{C}'\mathcal{D}' \rightarrow \mathcal{A}'\mathcal{E}'\mathcal{F}'$

Table 11 String representations for $7 \times m$ grids

m	6	7	8	9
String	$\mathcal{F}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{F}$	$\mathcal{F}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{C}\mathcal{D}$	$\mathcal{F}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{E}\mathcal{F}$	$\mathcal{F}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{A}\mathcal{F}'$
m	10	11	12	
String	$\mathcal{F}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{A}\mathcal{C}'\mathcal{D}'$	$\mathcal{F}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{A}\mathcal{A}'\mathcal{E}'\mathcal{F}'$	$\mathcal{F}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{A}\mathcal{B}'\mathcal{A}\mathcal{A}'\mathcal{B}\mathcal{A}'\mathcal{F}$	

The grammar rewrite rules for strings representing a $7 \times m$ grid are given in [Table 10](#). The basis for this rewrite system is the SMT for the 7×5 grid which is $\mathcal{F}\mathcal{A}'\mathcal{E}'\mathcal{F}'$. A listing of the strings for m from 6 to 11 is given in [Table 11](#).

9 Future Work

9.1 Grids

In this work we reviewed what is known about SMTs on grids and then presented results from PARSTEINER94 [28, 31] which characterize SMTs for $3 \times m$ to $7 \times m$ grids. The next obvious question is the following: What is the characterization for an $8 \times m$ grid or an $n \times m$ grid? Well, this is where things start getting nasty. Even though PARSTEINER94 cuts the computation time of the previous best program

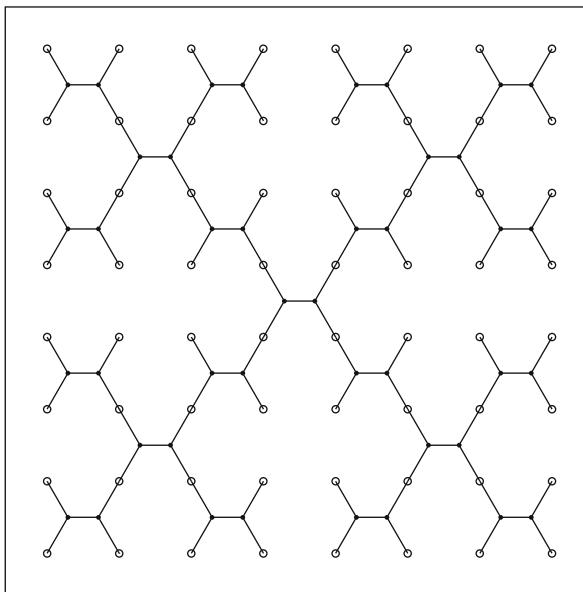


Fig. 44 8×8

for SMTs by an order of magnitude, the computation time for an NP-Hard problem blows up sooner or later, and $8 \times m$ is where we run into the computation wall.

We have been able to make small chips into this wall though and have some results for $8 \times m$ grids. The pattern for this seems to be based upon repeated use of the 8×8 grid which is shown in Fig. 44. This grid solution seems to be combined with smaller $8 \times$ solutions in order to build larger solutions. However, until better computational approaches are developed, further characterizations of SMTs on grids will be very hard and tedious.

9.2 Further Parallelization

9.2.1 Algorithm Enhancements

There remains a great deal of work that can be done on the Steiner minimal tree problem in the parallel arena. The first thing to consider is whether there are other ways to approach the parallel generation of T_list that would be more efficient. Improvement in this area would push the computation pendulum even further away from T_list generation and toward SMT extraction.

The next thing to consider is the entire extraction process. The initial generation of the *incompatibility matrix* has the appearance of easy parallelization. The forest management technique introduced by Cockayne and Hewgill could also be put into a parallel framework, thereby speeding up the preparation for extraction quite a bit.

With this initialization out of the way, decomposition could then be considered. The best possible enhancement here might be the addition of thresholds. As with most parallel algorithms, for any problem smaller than a particular size, it is usually faster to solve it sequentially. These thresholds could come into play in determining whether to call a further decomposition, such as the cycle decomposition introduced by Cockayne and Hewgill that was discussed in Sect. 7.

The final option for parallelization is one that may yield the best results and that is in the extraction itself. Extraction is basically a branch-and-bound process, using the *incompatibility matrix*. This branch and bound is primed with the length of the MST as the initial bound and continues until all possible combinations have been considered. The easiest implementation here would probably be the idea presented in the paper by Quinn and Deo [52] that served as the basis for the parallel algorithm in Sect. 6.

9.2.2 GPU Implementation

With games and visualization driving the evolution of graphics processors, the fixed functionality of the rendering pipeline once offered has been steadily replaced by the introduction of programmable pipeline components called shaders. These shaders not only allow the GPU to be used for more elaborate graphical effects but also allow it to be used for more general purpose computations. By storing general data as texture data, user-programmed vertex and fragment shaders can transform the GPU into a highly data parallel multiprocessor [48].

In 2007, Nvidia released CUDA [46], a programming language which allows for direct GPGPU programming in a C-like environment. Modern GPUs offer 512 processing cores [47], which is far more than any CPU currently provides. Many researchers have taken advantage of the environment provided by CUDA to easily map their parallel algorithms to the GPU.

Of note is the work being done by Joshua Hegie [33]. In his thesis, Hegie has mapped out an implementation of Winter's work onto the GPU. Preliminary results are very promising, and in the future work, he maps out a methodology for the use of multiple GPUs which will open the door for much larger problems at a reasonable computation time.

9.3 Additional Problems

9.3.1 1-Reliable Steiner Tree Problem

If we would like to be able to sustain a single failure of any vertex, without interrupting communication among remaining vertices, the minimum length network problem takes on a decidedly different structure. For example, in any FST all of the original vertices are of degree 1, and hence, any one can be disconnected from the network by a single failure of the adjacent Steiner point.

We would clearly like a minimum length 2-connected network. The answer can be the minimum length Hamiltonian cycle (consider the vertices of the unit square), but it does not need to be, as shown in the Θ graph given in Fig. 45.

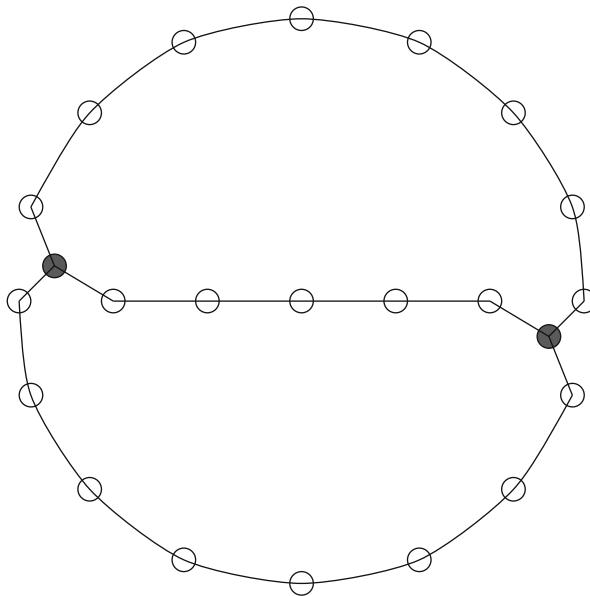


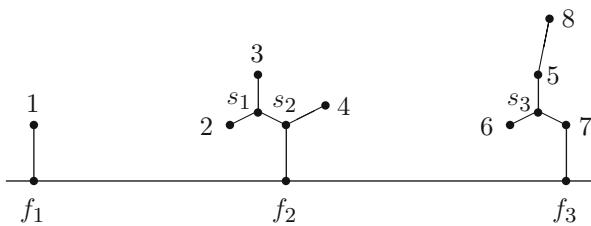
Fig. 45 Theta graph

Here we can add Steiner points near the vertices of degree 3 and reduce the network length without sacrificing 2-connectivity. This is not just a single graph, but is a member of a family of graphs that look like ladders, where the Θ graph has only one internal rung. We hope to extend earlier work providing constructions on 2-connected graphs [32] to allow effective application of an annealing algorithm that could walk through graphs within the 2-connected class.

9.3.2 Augmenting Existing Plane Networks

In practical applications, it frequently happens that new points must be joined to an existing Steiner minimal tree. Although a new and larger SMT can, in principle, be constructed which connects both the new and the existing points, this is typically impractical, e.g., in cases where a fiber optic network has already been constructed. Thus, the only acceptable approach is to add the new points to the network as cheaply as possible. Cockayne has presented this problem which we can state as follows:

Augmented Steiner Network: Given a connected plane graph $G = (V, E)$ (i.e., an embedding of a connected planar graph in E^2) and a set V' of points in the plane which are not on edges of G , construct a connected plane supergraph $G'' = (V'', E'')$, such that V'' contains $V \cup V'$, E'' contains E , and the sum of the Euclidean lengths of the set of edges in $E'' - E$ is a minimum. In constructing the plane graph G'' , it is permitted to add an edge connecting a point in V' to an

Fig. 46 An optimal forest

interior point of an edge in G . It is also permitted to add Steiner points. Thus, strictly speaking, G'' does not need to be a supergraph of G .

The Augmented Steiner Network Problem clearly has applications in such diverse areas as canal systems, rail systems, housing subdivisions, irrigation networks, and computer networks. For example, given a (plane) fiber optic computer network $G = (V, E)$ and a new set V' of nodes to be added to the network, the problem is to construct a set F' of fiber optic links with minimum total length that connects V' to G . The set F' of new links is easily seen to form a forest in the plane, because the minimum total length requirement ensures that there cannot be cycles in F' .

As an example, consider the situation in Fig. 46 where G consists of a single, long edge and $V' = v_1, \dots, v_8$. The optimal forest F' consists of three trees joining G at f_1, f_2 , and f_3 . It is necessary that extra Steiner points s_1, s_2 , and s_3 be added so that F has minimum length.

While we are aware of several algorithms for solving special cases of the Augmented Existing Plane Network Problem, such as those by Chen [7] and Trietsch [56] or the special case where the graph G consists of a single vertex, in which case the problem is equivalent to the classical Steiner minimal tree problem, we are not aware of any algorithms or computer programs available for exact solutions to the general form of this problem. Here, “exact” means provably optimal except for roundoff error and machine representation of real numbers. Non-exact (i.e., heuristic) solutions are suboptimal although they may often be found considerably faster.

Cross-References

- [Gradient-Constrained Minimum Interconnection Networks](#)
 - [Steiner Minimum Trees in \$E^3\$: Theory, Algorithms, and Applications](#)
-

Recommended Reading

1. A. Aggarwal, B. Chazelle, L. Guibas, C. O’Dunlaing, C. Yap, Parallel computational geometry. *Algorithmica* 3(3), 293–327 (1988)
2. M.J. Atallah, M.T. Goodrich, Parallel algorithms for some functions of two convex polygons. *Algorithmica* 3(4), 535–548 (1988)

3. J.E. Beasley, Or-library: distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41**(11), 1069–1072 (1990)
4. J.E. Beasley, Or-library. <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>. Last Accessed 29 Dec 2010
5. M.W. Bern, R.L. Graham, The shortest-network problem. *Sci. Am.* **260**(1), 84–89 (1989)
6. W.M. Boyce, J.R. Seery, STEINER 72 – an improved version of Cockayne and Schiller's program STEINER for the minimal network problem. Technical Report 35, Bell Labs., Department of Computer Science, 1975
7. G.X. Chen, The shortest path between two points with a (linear) constraint [in Chinese]. *Knowl. Appl. Math.* **4**, 1–8 (1980)
8. A. Chow, Parallel Algorithms for Geometric Problems. PhD thesis, University of Illinois, Urbana-Champaign, IL, 1980
9. F.R.K. Chung, M. Gardner, R.L. Graham, Steiner trees on a checkerboard. *Math. Mag.* **62**, 83–96 (1989)
10. F.R.K. Chung, R.L. Graham, in *Steiner Trees for Ladders*, ed. by B. Alspach, P. Hell, D.J. Miller, *Annals of Discrete Mathematics*, vol. 2 (Elsevier Science Publishers B.V., The Netherlands, 1978), pp. 173–200
11. E.J. Cockayne, On the Steiner problem. *Can. Math. Bull.* **10**(3), 431–450 (1967)
12. E.J. Cockayne, On the efficiency of the algorithm for Steiner minimal trees. *SIAM J. Appl. Math.* **18**(1), 150–159 (1970)
13. E.J. Cockayne, D.E. Hewgill, Exact computation of Steiner minimal trees in the plane. *Info. Process. Lett.* **22**(3), 151–156 (1986)
14. E.J. Cockayne, D.E. Hewgill, Improved computation of plane Steiner minimal trees. *Algorithmica* **7**(2/3), 219–229 (1992)
15. E.J. Cockayne, D.G. Schiller, in *Computation of Steiner Minimal Trees*, ed. by D.J.A. Welsh, D.R. Woodall, *Combinatorics*, pp. 52–71, Maitland House, Warrior Square, Southend-on-Sea, Essex SS1 2J4, 1972. Mathematical Institute, Oxford, Inst. Math. Appl.
16. R. Courant, H. Robbins, *What Is Mathematics? An Elementary Approach to Ideas and Methods* (Oxford University Press, London, 1941)
17. D.Z. Du, F.H. Hwang, A proof of the Gilbert-Pollak conjecture on the Steiner ratio. *Algorithmica* **7**(2/3), 121–135 (1992)
18. M.R. Garey, R.L. Graham, D.S. Johnson, The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.* **32**(4), 835–859 (1977)
19. A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing* (MIT Press, Cambridge, MA, 1994)
20. R. Geist, R. Reynolds, C. Dove, Context sensitive color quantization. Technical Report 91–120, Dept. of Comp. Sci., Clemson Univ., Clemson, SC 29634, July 1991
21. R. Geist, R. Reynolds, D. Suggs, A markovian framework for digital halftoning. *ACM Trans. Graph.* **12**(2), 136–159 (1993)
22. R. Geist, D. Suggs, Neural networks for the design of distributed, fault-tolerant, computing environments, in *Proc. 11th IEEE Symp. on Reliable Distributed Systems (SRDS)*, Houston, Texas, October 1992, pp. 189–195
23. R. Geist, D. Suggs, R. Reynolds, Minimizing mean seek distance in mirrored disk systems by cylinder remapping, in *Proc. 16th IFIP Int. Symp. on Computer Performance Modeling, Measurement, and Evaluation (PERFORMANCE '93)*, Rome, Italy, September 1993, pp. 91–108
24. R. Geist, D. Suggs, R. Reynolds, S. Divatia, F. Harris, E. Foster, P. Kolte, Disk performance enhancement through Markov-based cylinder remapping, in *Proc. of the ACM Southeastern Regional Conf.*, ed. by C.M. Pancake, D.S. Reeves, Raleigh, North Carolina, April 1992, pp. 23–28. The Association for Computing Machinery, Inc.
25. G. Georgakopoulos, C. Papadimitriou, A 1-steiner tree problem. *J. Algorithm* **8**(1), 122–130 (1987)
26. E.N. Gilbert, H.O. Pollak, Steiner minimal trees. *SIAM J. Appl. Math.* **16**(1), 1–29 (1968)

27. S. Grossberg, Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Network* **1**, 17–61 (1988)
28. F.C. Harris, Jr, *Parallel Computation of Steiner Minimal Trees*. PhD thesis, Clemson, University, Clemson, SC 29634, May 1994
29. F.C. Harris, Jr, A stochastic optimization algorithm for steiner minimal trees. *Congr. Numer.* **105**, 54–64 (1994)
30. F.C. Harris, Jr, An introduction to steiner minimal trees on grids. *Congr. Numer.* **111**, 3–17 (1995)
31. F.C. Harris, Jr, Parallel computation of steiner minimal trees, in *Proc. of the 7th SIAM Conf. on Parallel Process. for Sci. Comput.*, ed. by David H. Bailey, Petter E. Bjørstad, John R. Gilbert, Michael V. Maccagni, Robert S. Schreiber, Horst D. Simon, Virgia J. Torczan, Layne T. Watson, San Francisco, California, February 1995. SIAM, pp. 267–272
32. S. Hedetniemi, Characterizations and constructions of minimally 2-connected graphs and minimally strong digraphs, in *Proc. 2nd Louisiana Conf. on Combinatorics, Graph Theory, and Computing*, Louisiana State University, Baton Rouge, Louisiana, March 1971, pages 257–282
33. J. Hegie, Steiner minimal trees on the gpu. Master's thesis, University of Nevada, Reno, 2012
34. Universitat Heidelberg, Tsplib. <http://comopt.ifii.uni-heidelberg.de/software/TSPLIB95/>. Last Accessed 29 Dec 2010
35. J.J. Hopfield, Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.* **81**, 3088–3092 (1984)
36. F.K. Hwang, J.F. Weng, The shortest network under a given topology. *J. Algorithm* **13**(3), 468–488 (1992)
37. F.K. Hwang, D.S. Richards, Steiner tree problems. *Networks* **22**(1), 55–89 (1992)
38. F.K. Hwang, D.S. Richards, P. Winter, *The Steiner Tree Problem*, vol. **53** of *Ann. Discrete Math.* (North-Holland, Amsterdam, 1992)
39. F.K. Hwang, G.D. Song, G.Y. Ting, D.Z. Du, A decomposition theorem on Euclidian Steiner minimal trees. *Disc. Comput. Geom.* **3**(4), 367–382 (1988)
40. J. JáJá, *An Introduction to Parallel Algorithms* (Addison-Wesley, Reading, MA, 1992)
41. V. Jarník, O. Kössler, O minimálních grátech obsahujících n daných bodu [in Czech]. *Casopis Pesk. Mat. Fys.* **63**, 223–235 (1934)
42. S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing. *Science* **220**(13), 671–680 (1983)
43. V. Kumar, A. Grama, A. Gupta, G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms* (The Benjamin/Cummings Publishing, Redwood City, 1994)
44. Z.A. Melzak, On the problem of Steiner. *Can. Math. Bull.* **4**(2), 143–150 (1961)
45. M.K. Molloy, Performance analysis using stochastic petri nets. *IEEE Trans. Comput.* **C-31**(9), 913–917 (1982)
46. Nvidia, Cuda zone. http://www.nvidia.com/object/cuda_home_new.html. Last Accessed 29 Dec 2010
47. Nvidia, Geforce gtx 580. <http://www.nvidia.com/object/product-geforce-gtx-580-us.html>. Last Accessed 29 Dec 2010
48. J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A.E. Lefohn, T.J. Purcell, A survey of general-purpose computation on graphics hardware. *Comput. Graph. Forum* **26**(1), 80–113 (2007)
49. J.L. Peterson, *Petri Net Theory and the Modeling of Systems* (Prentice-Hall, Englewood Cliffs, 1981)
50. F.P. Preparata, M.I. Shamos, *Computational Geometry: An Introduction* (Springer, New York, 1988)
51. M.J. Quinn, *Parallel Computing: Theory and Practice* (McGraw-Hill, New York, 1994)
52. M.J. Quinn, N. Deo, An upper bound for the speedup of parallel best-bound branch-and-bound algorithms. *BIT* **26**(1), 35–43 (1986)
53. W.R. Reynolds, *A Markov Random Field Approach to Large Combinatorial Optimization Problems*. PhD thesis, Clemson, University, Clemson, SC 29634, August 1993

54. M.I. Shamos, *Computational Geometry*. PhD thesis, Department of Computer Science, Yale University, New Haven, 1978
55. J.R. Smith, *The Design and Analysis of Parallel Algorithms* (Oxford University Press, New York, 1993)
56. D. Trietsch, Augmenting Euclidean networks – the Steiner case. SIAM J. Appl. Math. **45**, 855–860 (1985)
57. D. Trietsch, F.K. Hwang, An improved algorithm for Steiner trees. SIAM J. Appl. Math. **50**, 244–263 (1990)
58. D.M. Warme, P. Winter, M. Zachariasen, Exact algorithms for plane steiner tree problems: a computational study, in *Advances in Steiner Trees*, ed. by D.-Z. Du, J.M. Smith, J.H. Rubinstein (Kluwer Academic, Boston, 2000), pp. 81–116
59. D.M. Warme, A new exact algorithm for rectilinear steiner trees, in *16th International Symposium on Mathematical Programming*. American Mathematical Society, Lausanne, Switzerland, 1997, pp. 357–395
60. P. Winter, An algorithm for the Steiner problem in the Euclidian plane. Networks **15**(3), 323–345 (1985)
61. P. Winter, M. Zachariasen, Large euclidean steiner minimum trees in an hour. Technical Report 96/34, DIKU, Department of Computer Science, University of Copenhagen, 1996
62. P. Winter, M. Zachariasen, Euclidean Steiner minimum trees: an improved exact algorithm. Networks **30**, 149–166 (1997)

Steiner Minimum Trees in E^3 : Theory, Algorithms, and Applications

J. MacGregor Smith

Contents

1	Introduction	3180
2	Background	3182
2.1	Assumptions	3183
2.2	Notation	3183
2.3	Steiner Properties in E^2	3183
2.4	E^2 Heuristic	3185
2.5	Results from the Unit Sphere	3188
2.6	Steiner Points in a Spherical Δ	3189
2.7	Heuristic on Φ	3190
3	E^3 Problem	3191
3.1	$N = 3$ Case	3191
3.2	$N = 4$ Case	3193
3.3	Primal Problem	3193
3.4	Complexity of the Primal Problem	3194
3.5	Dual Problem	3195
3.6	$N = 5$ Case	3198
3.7	$N = 6$ Case	3199
3.8	\mathcal{R} -Sausages	3202
3.9	\mathcal{R} -Sausage Properties	3202
3.10	Sausage Experiments	3204
4	Heuristics	3206
4.1	CG Approach	3207
4.2	Ribbon Decomposition Problem	3207
4.3	Heuristic Algorithm	3209
4.4	General Algorithm Description	3209
4.5	Heuristic Example	3212
4.6	Complexity Analysis	3214
4.7	Experiments	3216

J.M. Smith

Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst,
MA, USA
e-mail: jmsmith@ecs.umass.edu

5 Applications.....	3217
5.1 Minimum Energy Configurations (MECs).....	3218
5.2 Lower Bounds.....	3221
6 Fundamentals of Protein Structures.....	3221
6.1 Steiner-Algorithm Approach.....	3222
6.2 Primary Structure of Amino Acids.....	3222
6.3 Perturbed Amino Acids.....	3227
7 Secondary Structure.....	3230
7.1 Collagen.....	3233
7.2 Silk.....	3235
7.3 Myosin.....	3236
7.4 HIV-1 Protease.....	3239
7.5 Steiner Ratio Versus Torsion Energy.....	3242
7.6 Theoretical Versus Empirical ρ Values.....	3246
7.7 Protein Fold Recognition Using Steiner Ratio.....	3248
8 Overall Protein Results.....	3254
8.1 Other Applications.....	3255
9 Conclusion.....	3255
Cross-References.....	3255
Recommended Reading.....	3256

Abstract

This chapter summarizes some of the most important properties, algorithms, and heuristic properties for the solution of Euclidean Steiner minimal trees (ESMTs) in E^3 . Starting from the problem in E^2 , the basic primal and dual approaches to the problem in E^3 are presented. Given the \mathcal{NP} -hardness of the problem, the challenge of solving this combinatorial optimization problem is discussed not only with the solution of small order point sets but point sets with special structure. Heuristic solutions are examined, and the complexity of computing these solutions is discussed. Also of some interest are the applications of ESMTs to computational biology through the secondary structure of the protein-folding problem. Detailed studies of the application of ESMTs to the secondary structure of the protein-folding problem are described. Finally, the application of this fascinating problem to other areas of science and engineering is described.

1 Introduction

If someone is going to design a new electronic circuit at the molecular/atomic level where it was known beforehand the basic number of atomic elements of the circuit and they wished to arrange them so that they achieved an optimal configuration that is both compact, stable, and integrated electrically, *how should they configure the atoms?* If one assumes that the total cost of putting together the molecular structure is proportional to distance, i.e., the potential energy in the system, then one would want to minimize the overall interconnecting distance between the atoms. This is fundamentally the Steiner problem in E^3 . **Figure 1** demonstrates a geometric construction for the Steiner minimal tree of $N = 6$ vertices in E^3 .

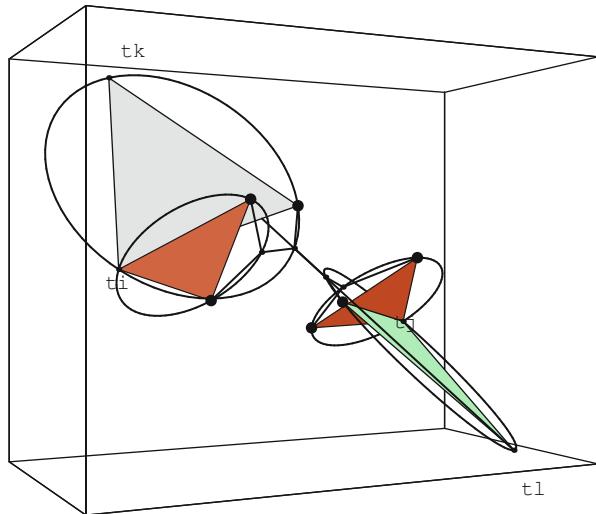


Fig. 1 $N = 6$ 3-sausage dual construction

The Steiner problem seeks to minimize the total length of a network, given a fixed set of vertices V that must be in the network and another set S from which vertices may be added [9, 13, 20, 21, 23, 42, 47, 62, 86]. The cardinality of S is not known beforehand which makes the problem very difficult, and the focus of this chapter is on a better understanding of the theory, algorithms, and potential applications of Steiner trees in E^3 . The Euclidean minimal spanning tree (EMST) is the optimal solution to the problem for which no vertices may be added from the set S [16, 81]. The Steiner ratio is the ratio of the length of the Steiner tree (or a heuristic approximation) to that of the EMST of the given set. The smaller this ratio is, the more advantage there is in finding the Steiner tree, since it indicates the configuration of vertices which achieves the minimum possible value. Thus, studying the problem of finding sets of vertices for which this ratio is a minimum is of central interest. The algorithmic process is designed to examine configurations which have a Steiner ratio close to the minimum. Configurations which have a ratio equal to the conjectured minimum for the same number of vertices are called *optimal configurations*.

Research results developed in 1992 [70] revealed some surprising insights into the properties of Steiner trees and their influence on the design of algorithms for computing them, and all of these will be explored in this chapter. The decomposition and recomposition processes underlying the algorithmic approach also have some impact on the potential applications of these results, and these will be briefly discussed.

Lastly, the problem of computing Steiner minimal trees is viewed from two fundamental questions or issues:

Issue I: *Given a set of vertices V in E^3 , how should they be decomposed into subsets so that optimal and suboptimal solutions for each subset may be combined to yield an effective solution to the Steiner minimal tree for the whole set?*

Issue II: *On the other hand, how should a set of vertices V in E^3 be arranged or comprised to yield the best configuration possible?*

The first issue arises where the vertex sets either have some fixed special structure or are generated randomly from some probability distribution. The second issue arises from the need to determine how the optimal configurations may be identified and combined to yield the best possible configurations for large vertex sets. This is important for potential applications as will be shown. Both issues overlap, and their interplay integrates much of what follows.

One of the truly fascinating features of Steiner trees in E^3 is how all the algebraic, geometric, and optimization issues are intertwined. Because of the complexity of these networks, algebraic, computational geometry, and numerical optimization concepts and tools are often necessary for understanding, constructing, and analyzing these networks. It is hoped that this chapter, in some small way, may illuminate the understanding of how the methodological issues underlying Steiner trees are woven together.

Chapter Outline

In Sect. 2, a detailed background is provided of known results in E^2 as well as for the unit sphere. In Sect. 3, the optimization and geometric constructions possible for $N = 3, 4, 5$, and 6 vertices are examined in detail, since a geometric understanding of the problem for small vertex sets gives one a good intuitive grasp of not only the complexity of computing Steiner trees in E^3 but also some insights as to ways of computing them. Section 4 describes one heuristic for computing Steiner trees in E^3 along with computational results. In Sects. 5–8, potential applications of Steiner trees for minimizing energy configurations are provided, e.g., protein modeling and computational biology issues, along with a small sample of experimental results and open research questions, and, finally, a brief summary and conclusions is provided in Sect. 9.

2 Background

It is well known that the complexity of computing Steiner minimal trees in the plane is \mathcal{NP} -hard [39, 40]. Also, because the Euclidean version is not known to be in \mathcal{NP} , then the complexity of computing optimal Steiner minimal trees in d-space $d \geq 3$ is demonstrably more difficult since there is no inherent combinatorial structure present in the problem [68].

2.1 Assumptions

- A set of finite vertices (possibly infinite) $V = \{v_1, v_2, \dots, v_\infty\}$ is given or randomly generated within a bounded region of the Euclidean plane or space E^2, E^3 with Cartesian coordinates (x_i, y_i, z_i) for $i = 1, 2, \dots, \infty$. The bounded region is often necessary for the probabilistic analysis of the algorithms. In the Steiner network problems described later in Sect. 5, the set V will be augmented with points from the set S so that the entire vertex set for the network design problem may contain $Z = V \cup S$ vertices. For the sake of clarity, “points” will refer to the additional vertices from the Steiner set S , while “vertices” will refer to those from the given set V .
- When necessary, it is assumed that total cost of the network is a linear function of the flows in the network unless specified otherwise. In many problems to be described, one assumes that the cost of an edge is proportional to its length. In Sect. 5 on applications, the relationship of distance to potential energy minimization is shown.
- For the purposes of this chapter, no obstacles impede or restrict the location of nodes and arcs in the network, although for many practical applications, such restrictions are very important.

2.2 Notation

The following are additional useful notation:

$ESMT(V)$:= Euclidean Steiner minimal tree of a given vertex set V

$EMST(V)$:= Euclidean minimum spanning tree of a given vertex set V

ρ_d := The minimal Steiner ratio of all vertex sets V in dimension d , i.e.,

$\rho = \inf_{V \in E^d} \rho(V)$ where $\rho(V) = \frac{ESMT(V)}{EMST(V)}$

$|S|$:= Cardinality of the number of vertices in the Steiner tree

M := Number of Steiner points from set S

N := Number of given vertices from set V

FST := Full Steiner tree with $|S| = n - 2$

As demonstrated, FSTs are more prevalent in E^3 than they are in E^2 . In E^2 , there are more degenerate situations, even where optimal solutions to vertex sets may occur (see Fig. 2). In Fig. 2, for the lattice structure in E^2 , an FST would have $S = 9$ vertices, whereas the optimal solution here has only $S = 5$.

2.3 Steiner Properties in E^2

There are certain elemental facts in the planar Steiner-tree problem which are applicable. They are:

- $|S| \leq N - 2$ [41]. This is an important upper bound on the cardinality of Steiner points, although the minimum number of Steiner points for a given vertex set is unknown.

- In the plane, the ESMT is a union of disjoint FSTs. Even for lattice configurations with special structure, computation of the optimal configurations is non-trivial for large n . For the best algorithm for computing Steiner trees in E^2 the reader is referred to Warne, Winter and Zachariasen's work [84].

- There are many interesting special structures besides the equilateral decomposition such as checkerboards and ladders, see [12, 17, 18, 35] for additional details and examples.

- On the right, we have the E^3 solution for the unit cube. There are a total of eight given vertices and six Steiner points found for this optimal solution. Notice that all of the Steiner points lie in a plane in space. This actually turns out to be an important property of Steiner trees in modeling proteins as described in §6 of this Chapter.

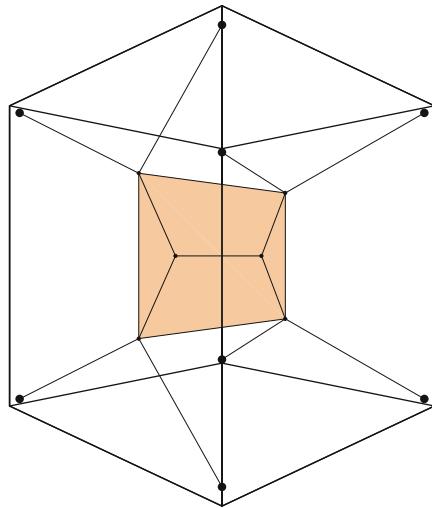
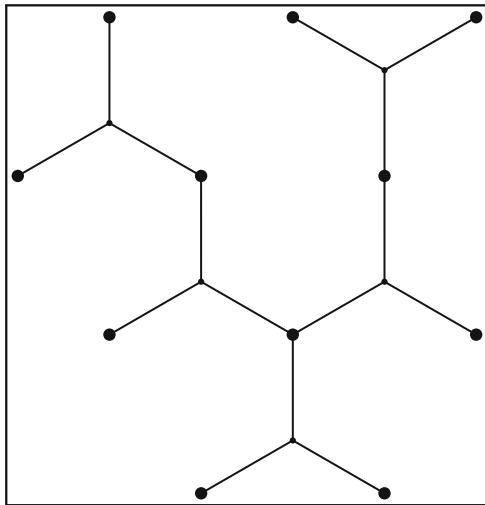


Fig. 2 E^2 lattice, $N = 11$; E^3 unit cube, $N = 8$

- $\rho_2 = \sqrt{3}/2 \forall V$ [33, 41]. This optimal ratio exists for a single equilateral triangle and certain collections of equilateral triangles and lattices. Recently, Ivanov and Tuzhilin have shown that the Gilbert-Pollak conjecture is still open [48].

For two dimensions and three given vertices, the mathematical programming problem in E^2 is a classical nonlinear programming problem to find the coordinates of a single Steiner point S with coordinates (x, y) such that

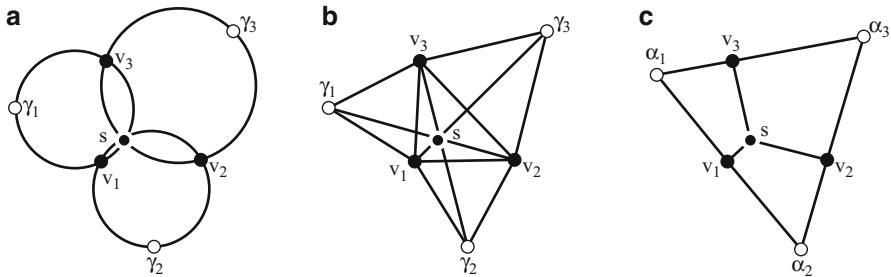


Fig. 3 Alternative E^2 constructions of a Steiner point in a triangle (v_1, v_2, v_3)

$$\text{Min } Z = \sum_{i=1}^3 [(x - x_i)^2 + (y - y_i)^2]^{\frac{1}{2}}$$

In E^2 , (see Fig. 3) geometric constructions are possible for constructing optimal Steiner trees for small vertex sets.

Figure 3 indicates three alternative methods in the plane for constructing the Steiner point in a triangle. Methods (a) and (b) are similar since by reflection of equilateral triangles on the sides of the original triangle, one creates three new vertices $\gamma_i, i = 1, 2, 3$ [24]. In method (a), one passes a circle through the reflected vertex γ_i and the base of the edge which generated it, while in method (b), one joins the reflected vertex γ_i with the vertex of the given triangle opposite it. Method (c) is carried out by solving a geometric dual problem which entails constructing the circumscribing triangle of the given triangle with maximum altitude [50, 53].

Some of these planar algorithmic and geometric concepts carry over to E^3 , although the construction problems become measurably more difficult since an optimization step becomes necessary. Heuristic approaches in E^2 also are extendable to E^3 , and one heuristic approach which will be briefly described in this chapter is directly applicable in E^3 with a slight twist.

2.4 E^2 Heuristic

One heuristic for E^2 is based on computational geometry \mathcal{CG} data structures of the Voronoi diagram $VOR(V)$ and the Delaunay triangulation $DT(V)$ [5, 73]. There are also other related data structures such as Gabriel graphs and relative neighborhood graphs that could be utilized in the design of a heuristic, and these are discussed in [72]. The $DT(V)$ provides a combinatorial framework for the construction of the ESMT solution, while the $VOR(V)$ provides locus information for concatenating local optimal solutions for small vertex set clusters. The data structures of the $VOR(V)$ and $DT(V)$ are now defined, as they will become important in higher dimensions.

- The polygon $VOR(v_i)$ is convex and its interior is the locus of vertices closer to v_i than to any other V -vertex (Figure 4).
- The union of these polygonal boundaries forms the *Voronoi diagram* for V (Figure 4), denoted by $VOR(V)$.

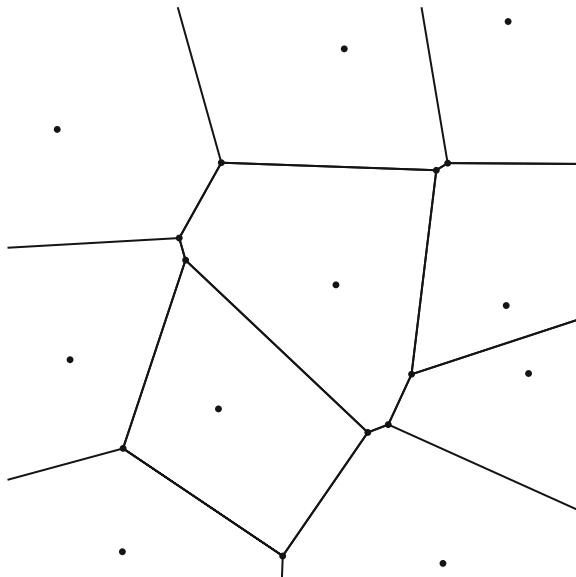


Fig. 4 Voronoi diagram for $N = 10$

Let v_i and v_j denote two distinct vertices. Furthermore, let $H_2(v_i, v_j)$ denote the set of vertices not farther from v_i than from v_j . $H_2(v_i, v_j)$ is a half plane. Let a Voronoi polygon around a given vertex v_i be defined as

$$VOR(v_i) = \bigcap_{v_j \in V \setminus \{v_i\}} H_2(v_i, v_j)$$

Further, let $P(v_i)$ denote the boundary of $VOR(v_i)$. The boundary edges are called *Voronoi edges*. Vertices where Voronoi edges meet are called *Voronoi points*. Voronoi points are similar to Steiner points in that they are extra vertices with degree 3, but their similarity to Steiner points ends there, at least for the Euclidean case. Voronoi diagrams form the foundation of many CG algorithms in geometric TND. For a detailed survey on Voronoi diagrams, the reader is referred to Boots [10] and Aurenhammer [4] and also [36, 58].

One of the most important properties of the $DT(V)$ is that the EMST is a subgraph of the $DT(V)$. Thus, once the $DT(V)$ is constructed, one has an $O(N)$ algorithm for constructing the EMST. Since the EMST provides an upper bound on the ESMT, then using the $DT(V)$ and $VOR(V)$ results in a performance-guaranteed heuristic. In fact, the algorithm provides a $2/\sqrt{3} \approx 1.1547$ performance ratio guarantee.

An $O(N \log N)$ geometric heuristic based on the concatenation of appropriately chosen ESMTs for clusters of 2, 3, and 4 V -vertices was given by Smith et al. [73].

- The straight-line dual of $VOR(V)$ obtained by connecting a pair v_i and v_j of vertices if and only if $P(v_i)$ and $P(v_j)$ share a common side yields in fact the $DT(V)$.
- Hence, the $DT(V)$ can be determined in $\Theta(N \log N)$ time. The optimality of this algorithm follows from the fact that sorting of n numbers is linear-time transformable to the problem of finding a triangulation of V -vertices (Figure 5).

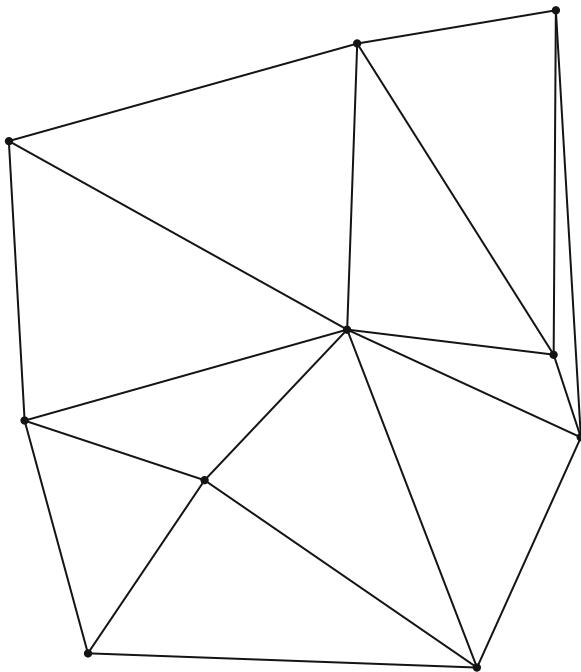


Fig. 5 Delaunay triangulation for $N = 10$

The identification of 2, 3, and 4 V -vertex sets is based on the information available from the Delaunay triangulation $DT(V)$ and the Voronoi diagram $VOR(V)$ along with the EMST for V . The Delaunay and Voronoi diagrams act as data structures for the Steiner constructions. More specifically, only the following cluster subsets of size $k = 2, 3, 4$ are considered during the concatenation phase:

- $k = 2$: pairs of vertices connected by an edge of the EMST
- $k = 3$: triples of vertices on a common triangle of $DT(V)$ and with two of its edges in the EMST
- $k = 4$: quadruples of vertices based on minimizing the Voronoi edge distance of two adjacent edge-sharing triangles of the $DT(V)$ which are also connected by EMST edges

The heuristic, in fact, first attempts to construct the $k = 4$ clusters based on the Voronoi information, then the $k = 3$, and finally the $k = 2$ edges.

The efficacy of the heuristic is based upon the data structure provided by the $DT(V)$ and $VOR(V)$, since the Delaunay tends to create “equilateral” triangles in its decomposition, which closely approximate the “best” solution in E^2 , i.e., $\rho_2 = \sqrt{3}/2$. This essential idea of decomposing the vertex set around the “best” configuration possible is important in higher dimensions and especially in E^3 as will be demonstrated.

2.5 Results from the Unit Sphere

The *unit sphere* Φ [29] provides a logical transition between E^2 and E^3 . Computing Steiner trees even for triangles on the sphere becomes more difficult because the distance metric is more complex and the Steiner vertices are affected by the curvature and wraparound effects of the spherical surface.

The geodesic minimum spanning tree (GMST) problem and the geodesic Steiner minimal tree (GSMT) problem on Φ will both be examined. The problems are computationally complex, and the focus of concern will be the development of algorithms and heuristics for these problems from a \mathcal{CG} vertex of view [61]. For the problem on the sphere, locations of the given vertices are defined by their latitude and longitude, namely, (ϕ_i, θ_i) for $i = 1, 2, \dots, n$. In order to discuss the minimal networks on the sphere, one needs a metric for length. The standard metric, which corresponds to the L_2 metric in Euclidean space, is the great circle distance. A *great circle* is any circle having a radius equal to that of the sphere. The distance between a pair of vertices on the sphere is the length (in the L_2 sense) of the lesser arc of a great circle between the vertices.

[Figure 6](#) nicely illustrates the minimization problem and some of the natural difficulties of finding vertices inside spherical triangles. The problem is to identify the additional vertex $s_i(\phi_i, \theta_i)$, if it exists, where the sum of the geodesic distances is minimized [2, 31, 52, 53, 84].

The following unconstrained optimization problem is presented for three given vertices Φ :

$$\text{Minimize } Z(X) = \sum_{j=1}^3 D(X, v_j) \quad (1)$$

where $X = (x_1, x_2)$ are the latitude and longitude coordinates of the Steiner point and $D(X, v_j)$ is the shortest geodesic or great circle distance on the surface of the sphere between the Steiner point X and an existing vertex j located at $v_j = (\phi_j, \theta_j)$. In fact, the geodesic distance is given by Donnay [30]:

$$D(X, v_j) = \cos^{-1} [\cos x_1 \cos \phi_j \cos(x_2 - \theta_j) + \sin x_1 \phi_j] \quad (2)$$

There are a number of issues related to the geometry of the sphere that complicate attempts to directly compute *GMST* and *GSMT* solutions on Φ .

- *First and most obviously, the sphere embodies degeneracies usually associated with antipodes – e.g., the multiplicity of geodesics between antipodal vertices. Such events require special tests and conventions to handle.*
- *Second, the sphere is bounded – i.e., the wraparound that exists complicates the neighbor relation. For example, a triangle of vertices on the sphere defines two spherical triangles, usually a major and a minor one. Without information about the rest of the spanning network, it is unclear in which of these to search for a Steiner point. The Delaunay triangulation and Voronoi diagrams, however, provide this oriented proximity information, since by first computing these data*

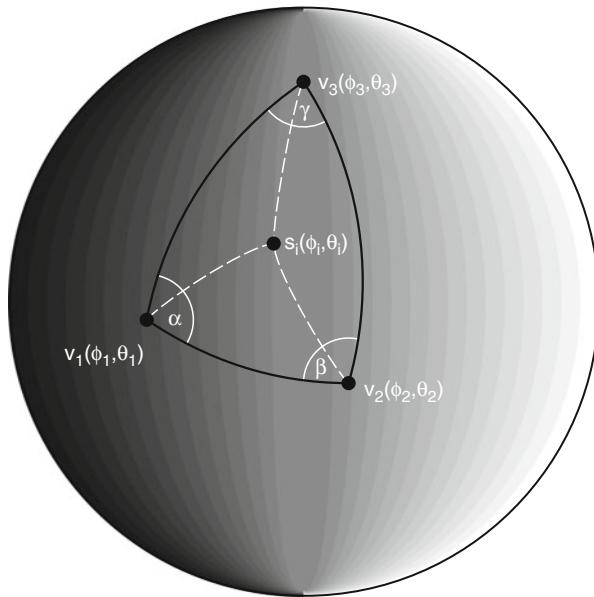


Fig. 6 Steiner minimal tree on Φ

structures, the task of constructing spanning trees is simplified, and ultimately the basis for the GSMT is formed.

- Third, the sphere lacks a natural ordering by means of which positional relations such as left-of and right-of are defined. The use of stereographic projection imparts such an ordering during the Delaunay construction process.

2.6 Steiner Points in a Spherical Δ

Some of the known properties of Steiner points in spherical triangles that have already been developed and form the foundation for the heuristic search for Steiner trees on Φ are now presented.

Theorem 1 ([19, 43]) *Let v_1, v_2, v_3 be three distinct vertices on the surface of a sphere, not all of which are on the same great circle, then any vertex X which minimizes the sum of the minor great circle arcs as in (1):*

$$\sum_{j=1}^3 D(X, v_j) \in \Delta v_1 v_2 v_3$$

is a Steiner point.

Theorem 2 ([19]) Either there is a unique point X which minimizes

$$\sum_{j=1}^3 D(X, v_j)$$

or X coincides with one of the given vertices v_1, v_2, v_3 or if

$$\angle v_1 v_2 v_3 \text{ or } \triangle v_1 v_2 v_3 < 2\pi/3$$

then the minimum vertex of $v_1 v_2 v_3 \neq v_2$.

Theorems 1 and 2 provide the analogues for locating Steiner points on Φ as they occur in E^2 . Also, under an azimuthal equidistant projection, with a Steiner vertex as a pole, an optimum Steiner point for three given vertices on Φ is the optimum for the corresponding problem in the plane E^2 [52]. In fact, it is this local projection property for three and four vertices on Φ that will be explored in the development of a heuristic for the general Steiner problem on Φ .

Finally, there are numerical difficulties with directly solving for the Steiner points on Φ as a continuous nonlinear optimization problem as can be surmised from the trigonometric relations of the distance function [53]. In fact, in problems where the given fixed vertices cannot be covered by a disc of radius $\frac{\pi}{4}$, local minima, maxima, and flat vertices must be dealt with [31]. As an alternative to solving for the GSMT as a continuous nonlinear optimization problem, the heuristic based on a computational geometry approach is briefly summarized.

2.7 Heuristic on Φ

As in E^2 , the approach utilizes the $DT(V)$ and $VOR(V)$ as data structures. Stereographic projection is utilized to project V onto the plane where the $DT(V)$ and $VOR(V)$ are constructed and then are mapped back to the unit sphere. See [29] for more details. The heuristic for the GSMT works by utilizing the simplicial decomposition of the $DT(V)$ and the GMST to decompose the vertex set into local optimal solutions of the GSMT then concatenates these local optimal solution into a suboptimal solution for \mathcal{V} .

The concatenation part of the heuristic is largely based on the ideas for solving the ESMT in E^2 [73], although there are some unique problems associated with solving for the local three and four vertex cases on Φ . As in the planar solution, only 2, 3, and 4-vertex cluster components are identified, since experimentally, they have the most significant reduction in GSMT solutions.

Thus, to briefly summarize the algorithmic approach, the Delaunay triangulation is utilized to partition the vertex set and help avoid degeneracies, the Voronoi diagram is then employed to provide locus information, and finally, stereographic

projection is employed to provide a natural ordering of the vertices for solutions to the *GMST* and *GSMT* problems.

All in all, the overall worst case running time of the algorithm is bounded by the time complexity of constructing the Delaunay triangulation which is $O(N \log N)$. All the remaining major steps and sub-steps of the algorithm for the *GMST* and the suboptimal solution for the *GSMT* are either $O(1)$ or in the worst case $O(N)$. Storage is also $O(N)$.

3 E^3 Problem

Now as one transitions to E^3 , most properties and features of Steiner trees in the plane will carry over, some will be different, and some new properties will emerge. In particular, the maximum number of Steiner points in E^3 is also $N - 2$. For a collection of other properties, see [68]. Not all optimal configurations require FSTs as will be demonstrated, since some of the given vertices can act as degenerate Steiner points. Things that one should be especially prepared for are the dynamic nature of ρ_3 , the recurring sets of planar Steiner solutions found in E^3 , and the elongated optimal nature of some of the Steiner point sets. These features will occur over and over again not only in this section but in [Sects. 4](#) and [5](#).

To start, the three vertex cases in E^3 are presented.

3.1 $N = 3$ Case

Take, for example, the following three regular vertices:

$$v_i : 1. \ 1. \ 1.; \ v_j : 1. \ -1. \ -1.; \ v_k : -1. \ 1. \ -1.$$

An assumption about these vertices is that the largest interior angle is less than $\frac{2\pi}{3}$ otherwise the Steiner point would be degenerate with one of the three given vertices. For three vertices in E^3 , they define a plane which thus reduces the three vertex problems to the planar version of the Steiner minimal tree problem. By the fundamental properties of constructing a Steiner vertex in the plane, (see [Fig. 3](#)) any reflected equilateral vertex and corresponding Melzak circle define the length of the SMT and the location of the Steiner point [41, 55].

Given that the three vertices lie in a plane, the arbitrary oriented plane is found in E^3 , and then one utilizes one of the previous methods in E^2 to construct an ESMT solution. The optimal length of this solution is defined as *SMT**. As a consequence of the Melzak circle construction in the arbitrary oriented plane, the equilateral vertex will be located farthest from the third given vertex v_k .

To understand why this occurs, as the Melzak circle is extended from say vertices v_i and v_j , the equilateral vertex on the circle will trace a sphere in space (see [Fig. 7](#)). The distance from the equilateral vertex on the Melzak circle to the opposite vertex v_k represents the length of the Steiner tree and is sometimes known as the Simpson line. As the Melzak circle rotates in space, the Simpson line and the length of the

- That it is the furthest vertex is an important property in looking at the $N \geq 4$ cases which will now be examined.

- The construction example for the 3 given vertices is nicely illustrated in Figure 7 showing the equilateral vertex, Melzak circle, and SMT length connecting t_{ij} and v_k .

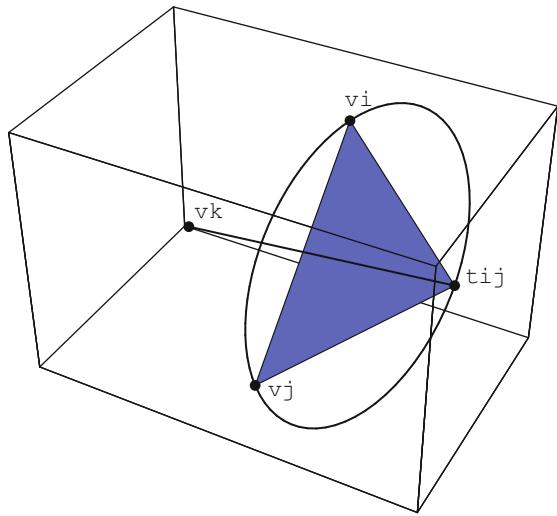


Fig. 7 Triangle in E^3

Steiner tree will vary in length. Further, as a consequence of this rotation process, the length of the Steiner tree will be longest when the circle lies in the arbitrary oriented plane, because if it did not lie in this space plane, then the planar construction would yield a Steiner point without the 120° angle property, a contradiction to the optimal solution.

Thus, one alternative to solve the problem is to find the orientation of the equilateral vertex and Melzak circle, by establishing the following quadratic optimization problem, where the location of t_{ij} with coordinates x_{ij} , y_{ij} , and z_{ij} is sought:

$$\text{Maximize } \Phi = \|t_{ij} - v_k\| \quad (3)$$

s.t.

$$\left[(x_{ij} - x_i)^2 + (y_{ij} - y_i)^2 + (z_{ij} - z_i)^2 \right]^{\frac{1}{2}} = e_{ij} \quad (4)$$

$$\left[(x_{ij} - x_j)^2 + (y_{ij} - y_j)^2 + (z_{ij} - z_j)^2 \right]^{\frac{1}{2}} = e_{ij} \quad (5)$$

The solution to this optimization problem will yield a chord length Φ^* , with t_{ij} and the Melzak circle lying in the same plane, otherwise $\Phi^* < SMT^*$, a contradiction to the optimality of SMT^* , thus $\Phi^* = SMT^*$, and t_{ij} is the furthest from v_k .

In the example construction, the length of the Simpson line is $2\sqrt{6}$, and since the length of each edge of the equilateral triangle is $\sqrt{8}$, the optimal Steiner solution of $\rho_3(V) = \sqrt{3}/2$ results.

3.2 $N = 4$ Case

Starting with an equilateral tetrahedron with the following four vertices, a description of the primal approach to constructing a solution for the $N = 4$ case will be presented followed by a dual formulation which is based on some of the distance optimization ideas found for the $N = 3$ case.

$$\begin{aligned} v_i : & \quad 1. \quad 1. \quad 1.; & v_j : & \quad 1. \quad -1. \quad -1. \\ v_k : & -1. \quad 1. \quad -1.; & v_\ell : & -1. \quad -1. \quad 1. \end{aligned}$$

There are three alternative topologies for a SMT in a tetrahedron. It will be useful to consider the three topologies in relation to the edge-pair orientations of the tetrahedron.

The three alternative topologies for the edge pairs are given in relation to the direction in which the (s_i, s_j) edge length exists:

$$(v_i, v_j) \cap (v_k, v_l); \quad (v_i, v_k) \cap (v_j, v_l); \quad (v_i, v_l) \cap (v_j, v_k)$$

3.3 Primal Problem

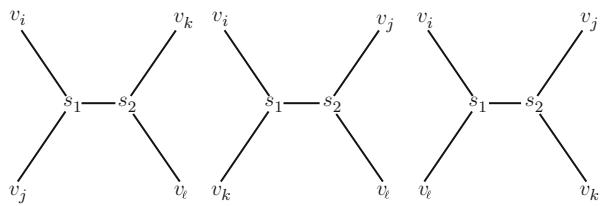
Because an equilateral tetrahedron is being examined, the first topology on the left of Fig. 8 with vertices as given will be conveniently assumed. Since the Steiner link geometry depends upon the topology of how the given vertices and Steiner points interact, establishing the topology is essential in formulating the optimization problem for constructing the Steiner tree (Fig. 9).

If one wishes to find the SMT geometry directly along with the coordinates of the Steiner points, one can minimize the following unconstrained nonlinear programming problem:

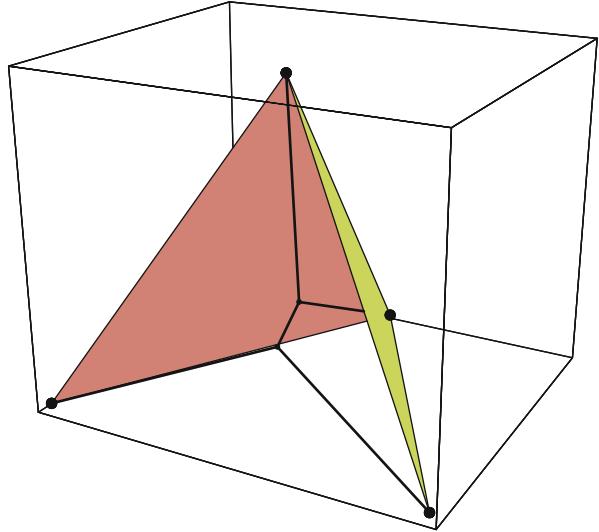
$$\text{Min } Z = \sum_{i \in V \cup S} \sum_{j \in S} [(x_i - s_{jx})^2 + (y_i - s_{jy})^2 + (z_i - s_{jz})^2]^{\frac{1}{2}} \quad (6)$$

where the indices in the summation are dependent on the three alternative topologies possible for the Steiner tree. In particular, writing out the primal problem, the following formulation is obtained (see Fig. 10 for how the topology and the constraint equations are interrelated):

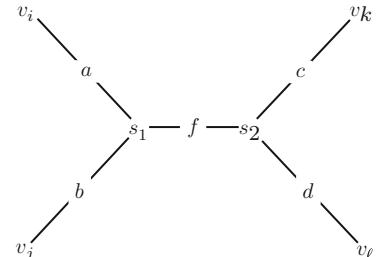
Directly solving the primal problem is carried out through such an algorithm as Warren Smith's [68] where a branch-and-bound code plays off iteratively searching the alternative FST topologies and numerical estimates of the Steiner points within the FSTs. Interestingly enough, Warren Smith's algorithm does not use a lower bounding schema which impedes its efficiency in pruning the enumeration tree. An alternative to the primal algorithmic approach is through the following dual quadratic programming formulation which must be generated for each topology as in the primal.

Fig. 8 Vertex set

- It was postulated by Gilbert and Pollack in 1968, [41] that the simplex in d -dimensional space would yield the optimal Steiner tree ratio.
- In 1996, the Gilbert-Pollack conjecture was shown to be false [34, 35].

**Fig. 9** Tetrahedron $\rho_3(V) = 0.81305$

$$\begin{aligned}
 a &= [(x_i - s_{ix})^2 + (y_i - s_{iy})^2 + (z_i - s_{iz})^2]^{\frac{1}{2}} \\
 b &= [(x_j - s_{ix})^2 + (y_j - s_{iy})^2 + (z_j - s_{iz})^2]^{\frac{1}{2}} \\
 c &= [(x_k - s_{ix})^2 + (y_k - s_{iy})^2 + (z_k - s_{iz})^2]^{\frac{1}{2}} \\
 d &= [(x_\ell - s_{jx})^2 + (y_\ell - s_{jy})^2 + (z_\ell - s_{jz})^2]^{\frac{1}{2}} \\
 f &= [(s_{ix} - s_{jx})^2 + (s_{iy} - s_{jy})^2 + (s_{iz} - s_{jz})^2]^{\frac{1}{2}}
 \end{aligned}$$

**Fig. 10** Primal tetrahedron topology

3.4 Complexity of the Primal Problem

There is a property which states that given four regular sites in 3-space such that their SMT is a full topology with $N - 2$ Steiner vertices that no *ruler and compass* geometric solution is possible [68]:

Theorem 3 *The coordinates of the two Steiner points may be expressed as the root of an essentially general (e.g., generally irreducible) eighth-degree polynomial. Therefore, the points are not obtainable by any straightedge and compass construction nor by solving any sequence of equations of degree ≤ 7 .*

While there may be no *exact* solution, the theorem does not say anything about the development of geometric constructions based on numerical methods or approximations to the optimal solution which is the focus of this chapter. It is felt that unless one views the construction of Steiner trees from this geometric construction process, one does not have a true understanding of both the complexity and the beauty of these Steiner constructions in E^3 .

3.5 Dual Problem

In this dual formulation related to the geometric construction in Fig. 7, the location of Melzak circle vertices t_{ij} and $t_{k\ell}$ that are furthest apart from each other is sought [68]. The argument for using this optimization formulation to find the furthest Melzak circle vertices follows a similar one as in the $N = 3$ case [68]. Finding these Melzak circle vertices and the resulting line segment between them as depicted in Fig. 11 yields a lower bound on the length of the Steiner problem for the particular topology.

The dual formulation is edge-based, and the Steiner points are found after the optimization of the edges, while the Primal is vertex-based, and one directly seeks the location of the Steiner vertices. The following optimization problem with the constraints ensuring the equilateral edges are satisfied is presented:

$$\text{Maximize } \Phi = \|t_{ij} - t_{k\ell}\| \quad (7)$$

subject to:

$$[(x_{ij} - x_i)^2 + (y_{ij} - y_i)^2 + (z_{ij} - z_i)^2]^{\frac{1}{2}} = e_{ij} \quad (8)$$

$$[(x_{ij} - x_j)^2 + (y_{ij} - y_j)^2 + (z_{ij} - z_j)^2]^{\frac{1}{2}} = e_{ij} \quad (9)$$

$$[(x_{k\ell} - x_k)^2 + (y_{k\ell} - y_k)^2 + (z_{k\ell} - z_k)^2]^{\frac{1}{2}} = e_{k\ell} \quad (10)$$

$$[(x_{k\ell} - x_\ell)^2 + (y_{k\ell} - y_\ell)^2 + (z_{k\ell} - z_\ell)^2]^{\frac{1}{2}} = e_{k\ell} \quad (11)$$

The SMT is equal to the distance between t_{ij} and $t_{k\ell}$ and, in fact, $SMT = 2\sqrt{6} + 2$ and the Steiner ratio $\rho_3(n) = \frac{2\sqrt{6}+2}{3\sqrt{8}} \approx 0.81305$ which is the optimal Steiner ratio of a regular tetrahedron [70]. In the construction Fig. 12, the twist angle between the two Melzak circles is 90° which is due to the mutual orthogonality of the edge pairs (v_i, v_j) and (v_k, v_ℓ) .

This quadratic optimization formulation does not admit to a polynomial time algorithm for its solution and, thus, remains a very difficult optimization problem

Fig. 11 Dual Simpson line construction

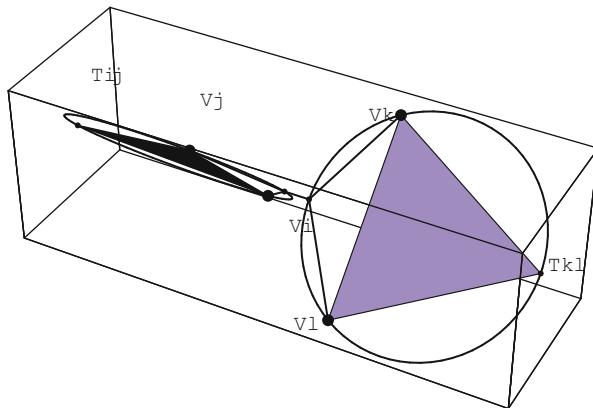
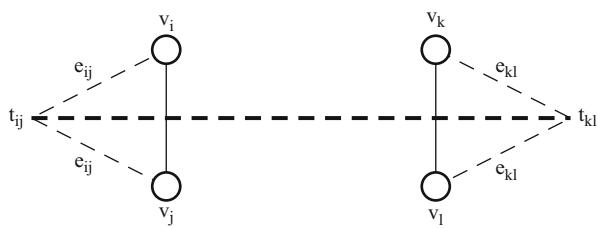


Fig. 12 $N = 4$ equilateral triangles and Melzak circles

even for four vertices. Even though the problem is convex, the objective function is a maximization one, so there may not be a unique optimum.

The beauty of solving this dual optimization problem is that one clearly sees the underlying geometry for constructing the Steiner tree in E^3 . Solving the quadratic optimization problem will work for any irregular tetrahedra, not just equilateral ones. In the dual problem, one can establish a relaxed version of the general problem if one uses a distance-squared formulation. The Lagrangian of the distance-squared formulation of the dual is

$$L(\bar{x}, \bar{\lambda}) = f(\bar{x}) + \bar{\lambda} h(\bar{x})$$

If the above is then differentiated, $\partial f / \partial x_{ij}, \partial f / \partial \lambda_i$ generating 10 nonlinear equations in the 10 independent variables and setting them equal to 0:

$$\frac{\partial f}{\partial x_{ij}} : 2(x_{ij} - x_{k\ell}) + 2\lambda_1(x_{ij} - x_i) + 2\lambda_2(x_{ij} - x_j) = 0$$

$$\frac{\partial f}{\partial x_{k\ell}} : -2(x_{ij} - x_{k\ell}) + 2\lambda_3(x_{k\ell} - x_k) + 2\lambda_4(x_{k\ell} - x_l) = 0$$

$$\begin{aligned}
\frac{\partial f}{\partial y_{ij}} : & 2(y_{ij} - y_{k\ell}) + 2\lambda_1(y_{ij} - y_i) + 2\lambda_2(y_{ij} - y_j) = 0 \\
\frac{\partial f}{\partial y_{k\ell}} : & -2(y_{ij} - y_{k\ell}) + 2\lambda_3(y_{k\ell} - y_k) + 2\lambda_4(y_{k\ell} - y_l) = 0 \\
\frac{\partial f}{\partial z_{ij}} : & 2(z_{ij} - z_{k\ell}) + 2\lambda_1(z_{ij} - z_i) + 2\lambda_2(z_{ij} - z_j) = 0 \\
\frac{\partial f}{\partial z_{k\ell}} : & -2(z_{ij} - z_{k\ell}) + 2\lambda_3(z_{k\ell} - z_k) + 2\lambda_4(z_{k\ell} - z_l) = 0 \\
\frac{\partial f}{\partial \lambda_1} : & -e_{ij}^2 + (x_{ij} - x_i)^2 + (y_{ij} - y_i)^2 + (z_{ij} - z_i)^2 = 0 \\
\frac{\partial f}{\partial \lambda_2} : & -e_{ij}^2 + (x_{ij} - x_j)^2 + (y_{ij} - y_j)^2 + (z_{ij} - z_j)^2 = 0 \\
\frac{\partial f}{\partial \lambda_3} : & -e_{k\ell}^2 + (x_{k\ell} - x_k)^2 + (y_{k\ell} - y_k)^2 + (z_{k\ell} - z_k)^2 = 0 \\
\frac{\partial f}{\partial \lambda_4} : & -e_{k\ell}^2 + (x_{k\ell} - x_l)^2 + (y_{k\ell} - y_l)^2 + (z_{k\ell} - z_l)^2 = 0
\end{aligned}$$

In the above, the first six equations represent the dual feasibility conditions, while the latter four represent the primal feasibility conditions. Including the complementary slackness conditions together in the Lagrangian objective function $\bar{h}(x) = 0$, one sees that indeed the dual does solve the primal since maximizing the distance between two extreme vertices is equivalent to the primal objective function.

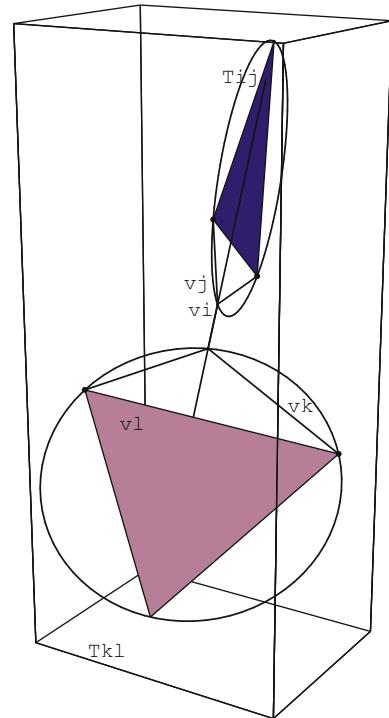
As an example of the irregular four-vertex case, the optimal length of a general tetrahedron is given as follows (this is a slight perturbation of the equilateral tetrahedra):

$$\begin{aligned}
v_i : & 1.25 \ 1. \ 1. \ v_j : 1. \ -1.75 \ -1. \ v_k : -1. \ 1. \ -2. \\
& v_\ell : -1. \ -1. \ 1.
\end{aligned}$$

Carrying out the modified algorithm with the nonlinear programming approach embedded in *Mathematica*, which is based on a Newton-type procedure for solving a set of nonlinear equations, the coordinates were found to be

$$\begin{aligned}
t_{ij} &= (-0.200706, 0.366419, 3.56057); \\
t_{k\ell} &= (0.374448, -1.17195, -4.44045)
\end{aligned}$$

The Lagrange multipliers for this irregular configuration are $\lambda_1 = -1.36566$, $\lambda_2 = -1.75906$, $\lambda_3 = -1.53361$, and $\lambda_4 = -1.11645$. The chordal distance between the extreme vertices was found to yield an $SMT = 8.16789$ with a Steiner ratio $\rho_3(V) = 0.861507$ (Fig. 13).

Fig. 13 Dual construction $N = 4$ 

3.6 $N = 5$ Case

As the next logical vertex set, the case for $N = 5$ vertices is considered. Two adjacent equilateral tetrahedra form the foundation of this vertex set. In general, for the $N = 5$ vertex case, there are a total of 15 possible FST topologies, so that the solution procedure becomes slightly more complex with the additional vertex. Let us consider the following regular vertex set:

$$\begin{aligned} v_i : & \quad 1. \quad 1. \quad 1.; \quad v_j : \quad 1. \quad -1. \quad -1.; \quad v_c : \quad -1. \quad 1. \quad -1. \\ v_k : & \quad -1. \quad -1. \quad 1.; \quad v_\ell : \quad -5/3, \quad -5/3, \quad -5/3 \end{aligned}$$

One of the topologies appears in Fig. 14, and this is the one used in the example discussion.

For the above topology, the Melzak circles are constructed iteratively, pairing up edges (v_i, v_j) and finding t_{ij} , pairing up edges (v_k, v_ℓ) and finding $t_{k,\ell}$, then taking t_{ij} and $t_{k,\ell}$ to find the vertex t_{ijkl} and the resulting maximum distance between t_{ijkl} and v_c the central vertex.

The dual formulation is again dependent upon specifying the topology of the vertices in the network. The dual formulation for $N = 5$ requires the solution of the following quadratic optimization problem (*n.b.* the squared distance function is used to simplify somewhat the optimization problem):

- Figure 14 illustrates the mapping between the example configuration and the optimization problem.
- The difficulty of this optimization problem stems from the need to simultaneously locate t_{ijkl}, t_{ij}, t_{kl} .

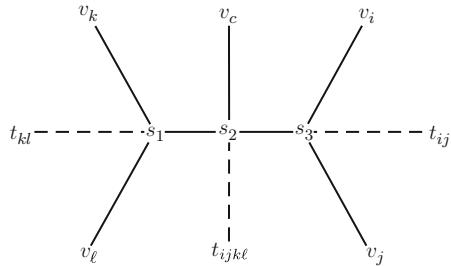


Fig. 14 Dual construction for vertex set $N = 5$

$$\text{Maximize } \Phi = \|t_{ijkl} - v_c\| \quad (12)$$

subject to:

$$(x_{ij} - x_i)^2 + (y_{ij} - y_i)^2 + (z_{ij} - z_i)^2 = e_{ij}^2 \quad (13)$$

$$(x_{ij} - x_j)^2 + (y_{ij} - y_j)^2 + (z_{ij} - z_j)^2 = e_{ij}^2 \quad (14)$$

$$(x_{kl} - x_k)^2 + (y_{kl} - y_k)^2 + (z_{kl} - z_k)^2 = e_{kl}^2 \quad (15)$$

$$(x_{kl} - x_\ell)^2 + (y_{kl} - y_\ell)^2 + (z_{kl} - z_\ell)^2 = e_{kl}^2 \quad (16)$$

$$(x_{ijkl} - x_{ij})^2 + (y_{ijkl} - y_{ij})^2 + (z_{ijkl} - z_{ij})^2 - \\ (x_{k\ell} - x_{ijk\ell})^2 + (y_{k\ell} - y_{ijk\ell})^2 + (z_{k\ell} - z_{ijk\ell})^2 = 0 \quad (17)$$

$$(x_{ijkl} - x_{ij})^2 + (y_{ijkl} - y_{ij})^2 + (z_{ijkl} - z_{ij})^2 - \\ (x_{k\ell} - x_{ij})^2 + (y_{k\ell} - y_{ij})^2 + (z_{k\ell} - z_{ij})^2 = 0 \quad (18)$$

Figure 15 illustrates the resulting optimal solution for the $N = 5$ vertex case. In order to solve this difficult nonlinear programming problem, a sequential quadratic programming (SQP) algorithm available in the IMSL Library was utilized. For the $N = 5$ equilateral configuration, the $\rho_3(V) = 0.815469$ which is larger than the $N = 4$ case. In what follows, it will be shown that for a certain configuration of $N = 6$ equilateral vertices, one can reduce this $\rho_3(V)$ value below the $N = 5$ vertex case.

3.7 $N = 6$ Case

Now consider three equilateral tetrahedra with the following vertex set coordinates and corresponding topology:

Figure 16 represents one of the 120 possible topologies interconnecting the set of six given vertices and the Steiner points. This particular topological arrangement,

- It is clear from Figure 15 that the edge segment between $tijkl$ and v_c is the lower bound length that is sought.

- What is the difficulty here is in the simultaneous location of the vertices tij , tkl , $tijk\ell$.

- An important feature of the dual construction is the resulting conjunction of the planes of Melzak Circles and the intersecting Simpson lines in E^3 .

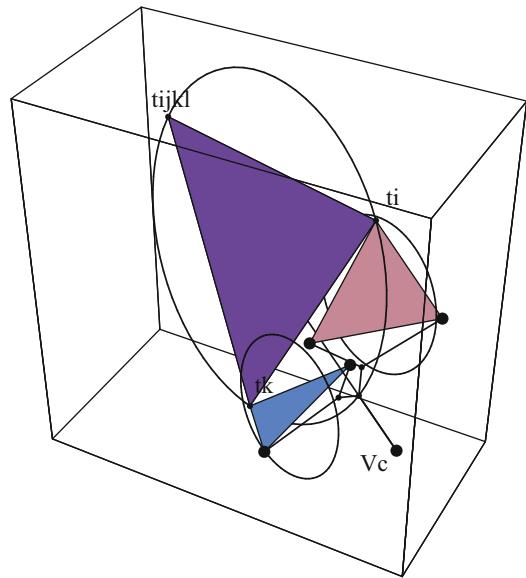


Fig. 15 $N = 5$ dual construction

$$v_i; 1, 1, 1.; \quad v_j; 1, -1, -1.; \quad v_c; -1, 1, -1.$$

$$v_d; -1, -1, 1.; \quad v_k; -5/3, -5/3, -5/3; \quad v_\ell; -3.444, -1/9, -1/9$$

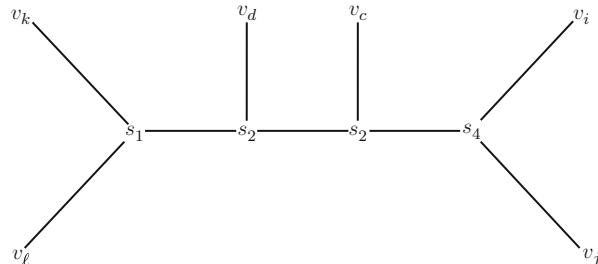


Fig. 16 Vertex set $N = 6$

however, represents a special property of the vertex set, referred to as a “path topology” which will be defined formally later. This “path topology” is very important for vertex sets $N \geq 6$ as will be demonstrated.

3.7.1 $N = 6$ Dual Construction

The detailed dual optimization formulation for $N = 6$ is similar to the $N = 5$ case and will not be repeated here for the sake of brevity. It again requires the construction of Melzak circles for adjacent pairs of vertices and further Melzak circles built upon these pairs. The quadratic optimization formulation was also solved with the SQP algorithm described earlier for the $N = 5$ case.

- Figure 1 (previously shown at the beginning of the Chapter) and Figure 17 reveal the rather interesting solutions for the $N = 6$ case.

- Again, the planes of Melzak Circles and intersecting Simpson lines are a unique trademark of what happens in the ESMT solutions in E^3 ,

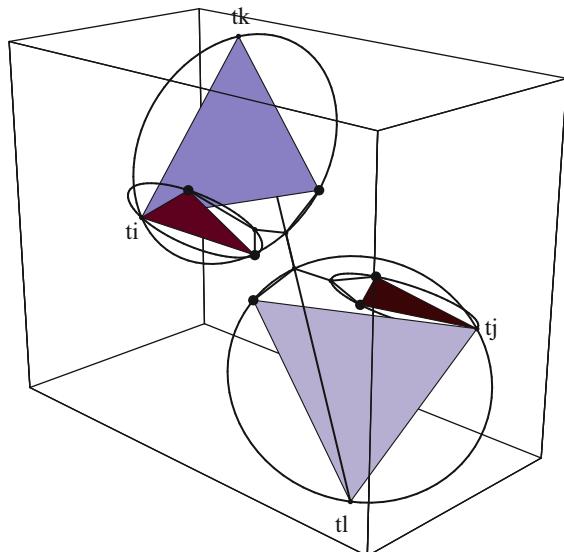


Fig. 17 $N = 6$ optimal dual construction $\rho_3(V) = 0.80865$

For this special configuration of three tetrahedra, it is curious why it should decrease the Steiner ratio after two tetrahedra raised it above the simplex, but, in one sense, this is related to the planar case, since the $N = 5$ optimal configuration in E^2 is the degenerate $N = 4$ tetrahedron with the fifth vertex and ϵ -distance removed from one of the four given vertices.

3.7.2 Conjectured Optimal Construction in E^3

While for $N = 6$ vertices one can construct an optimal solution using the past dual formulations, there are other properties of this vertex set which have even further significance for Steiner trees in E^3 .

In order to add additional vertices to the $N = 6$ base construction, the following construction technique referred to as the “buckyball” rule is presented [70]:

Rule: Successively add vertices so that the N th vertex added is always equidistant to the min ($3, N - 1$) most recently added vertices.

This rule affords a way and a corresponding data structure to configure chains of tetrahedra built upon the $N = 6$ basic equilateral structure. In fact, as vertices are added to this basic structure, one creates a ribbon sausage of vertices which dramatically decreases the ρ_3 value. The names of the tetrahedra structures generated are based on the fact that a single carbon atom forms a tetrahedron and more complex carbon molecules are simply built upon this single carbon atom base via the buckyball rule.

This monotonically decreasing property of these structures is truly remarkable. One possible explanation of why six vertices are needed is that the convex hull of the

three tetrahedra becomes the foundation of the triple helix geometry which forms the basis of the conjectured optimal configuration.

3.8 \mathcal{R} -Sausages

The search for the conjectured optimal configuration for minimizing ρ_3 in E^3 , Issue #2 raised at the beginning of the chapter, began by looking at two and three tetrahedra as well as gluing various types and combinations of Platonic solids (cubes, octahedron, etc.) together [70].

As an aside, Fejes Tóth conjectured in 1975 [80, 85] that the optimal configuration of spheres in $d \geq 3$ for the penny-packing problem was a set of unit balls on a line. This is the famous “sausage” conjecture. The conjectured configuration for the Steiner problem in E^3 is slightly different, since the unit balls with each tetrahedron are slightly offset from one another and not *in a line*. To distinguish the problem and conjecture from that of the sausage conjecture, since it is really different, the conjectured optimal configuration will be called a “ribbon/sausage” or \mathcal{R} -sausage for short.

The most interesting property of this \mathcal{R} -sausage is that it appears to be the optimal configuration of vertices in E^3 , it is thus equivalent to the best configuration, the equilateral triangle in E^2 which as was shown earlier is critical in any heuristic algorithm for vertex set decomposition.

Conjecture The \mathcal{R} -sausage achieves ρ_3 , and

$$\rho_3 = \sqrt{\frac{283}{700} - \frac{3\sqrt{21}}{700} + \frac{9\sqrt{11-\sqrt{21}}\sqrt{2}}{140}} \approx 0.784190373377122247108395477.$$

Of some note, Diaconis and Graham recently offered a prize of \$1000 to anyone who can prove or disprove this conjecture [25].

3.9 \mathcal{R} -Sausage Properties

For a known ribbon topology, the following properties within the \mathcal{R} -sausage exist, namely,

Path Topology As can be seen in Fig. 18, the \mathcal{R} -sausage has a unique path topology which relates the given vertex set V with the Steiner points in S . For the \mathcal{R} -sausage, there are $(N - 2)$ Steiner points, Steiner point i being connected to Steiner point $i + 1$ for $i = 1 \dots N - 3$. Also sausage vertex i is attached to Steiner point $i - 1$ for $i = 2 \dots N - 1$, and also sausage vertex 1 is attached to Steiner point 1, and sausage vertex n is attached to Steiner point $N - 2$. This path topology is an important indicator of the fundamental structure of a particular set of vertices.

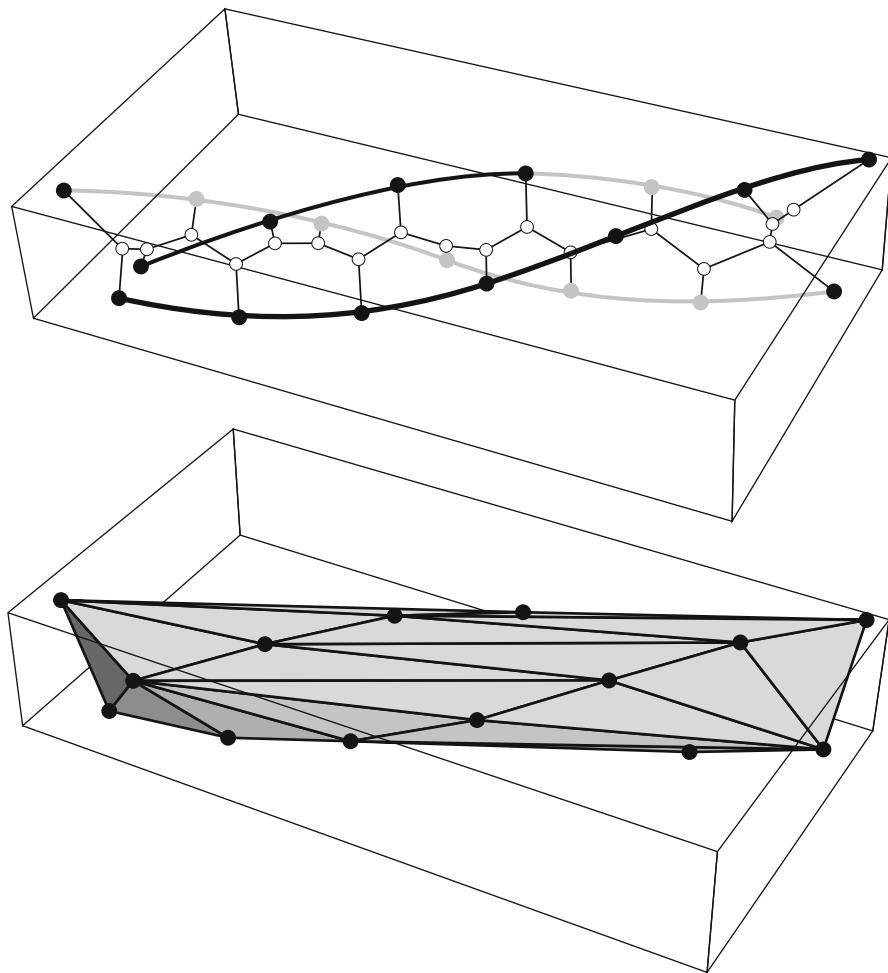


Fig. 18 $N=20$ helix geometry and path topology

Monotonically Decreasing [70] The Steiner ratio is monotonically decreasing, as the number of vertices in V increases in the \mathcal{R} -sausage. This is exemplified in [Table 1](#).

This dynamic nature of the Steiner ratio implies that the longer the \mathcal{R} -sausage the better. This would seem to imply some importance to the computational biological applications examined in [Sect. 6](#).

FSTs In particular, the maximum number of Steiner points in E^3 is also $N-2$. For a collection of other properties, see [68]. Not all optimal configurations require FSTs as will be shown in [Sects. 4](#) and [5](#), since some of the given vertices act as degenerate Steiner points.

Angles All the angles at the Steiner junctions are 120° . This is the same property as in the plane. When the applications to biochemical proteins are discussed, this

Table 1 Steiner ratios for N-vertex \mathcal{R} -sausages

n	rho	name
4	0.813052529585	(regular tetrahedron)
5	0.815469669674	(triangular bipyramid)
6	0.808064936179	("propane")
7	0.802859896946	(1 of 2 "chain butanes")
8	0.800899342742	pentane
9	0.798704227344	hexane
10	0.797013231353	heptane
11	0.795785747249	octane
12	0.794720989050	nonane
13	0.793838038891	decane
14	<= 0.7934	
15	<= 0.7926	
infinity	<= 0.784190373377122	R-sausage

angular requirement will not hold in all cases since the mass of the atoms and the electrodynamic forces interrelating them are not uniformly equivalent.

Steiner point Degree All the Steiner points have three arcs incident to each vertex or $\delta(s_j) = 3, \forall j$. All given vertices have $\delta(v_i) = 1, \forall i$. Figure 18 illustrates the convex hull and Steiner tree for $n = 20$ vertices. The diagram clearly indicates the triple helix construction. Notice that all vertices from V lie on the convex hull of the \mathcal{R} -sausage, while all the Steiner points lie in the interior.

Helical Axis There is a well-defined axis of rotation about which both the V and S vertices rotate.

Figures 19 and 20 illustrate two end views of the vertex set for $n = 75$, the former very close to the start of the \mathcal{R} -sausage while the latter from a distance. The chords across this end view in Fig. 20 represent the Steiner points and line segments at both ends of this finite \mathcal{R} -sausage, yet for an infinite \mathcal{R} -sausage, these would not exist, one would have two concentric convex hulls or 3d onions as they are called in Computational Geometry.

Periodicity Curiously enough, it has been shown that the \mathcal{R} -sausage is not periodic. If one selects a single vertex as a starting vertex for the \mathcal{R} -sausage, then after 28 tetrahedra or 11 chords along the helix, because of the screw symmetry of the \mathcal{R} -sausage, it should reappear after a 360° revolution, but it reappears slightly off at $354^\circ 20'$. This was demonstrated by Coxeter (p. 412) [24].

3.10 Sausage Experiments

Many hundreds of thousands of computational experiments were tried to see if there was a better optimal configuration of vertices which has the smallest ρ_3 , and the remarkable result so far is that nothing surpasses the \mathcal{R} -sausage for minimizing ρ_3 [70]. While one cannot conclusively prove it is the optimal configuration at this vertex in time, at least it yields a very strong upper bound on the Steiner ratio. This

- Figure 19 illustrates that for vertices propagating out along the R-Sausage, they appear in clusters of essentially 7 vertices.
- Since there are a total of $n = 75$ vertices, two clusters have 6 vertices, while there are nine clusters of 7 vertices.

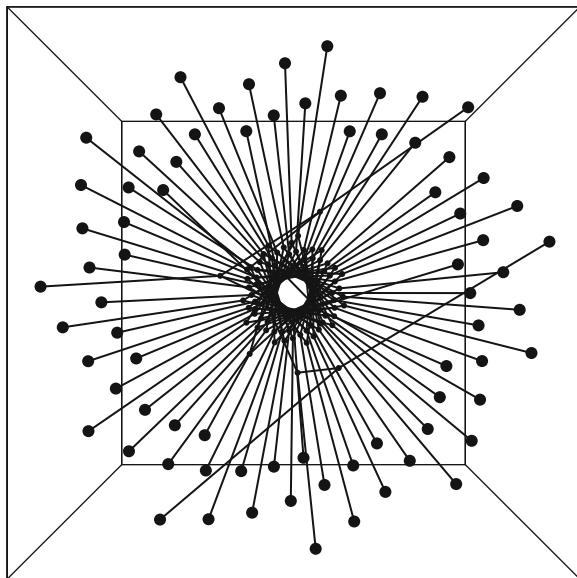


Fig. 19 \mathcal{R} -sausage screw symmetry

- Figure 20 is another view of the same R-Sausage from a further distance where it is clear that all the given vertices V lie on the convex hull of V and all the Steiner points S lie in an interior convex hull.
- This view also reconfirms the distance symmetries which are present in the R-Sausage along with the existence of a cylindrical Steiner convex hull inside the given vertex set V .

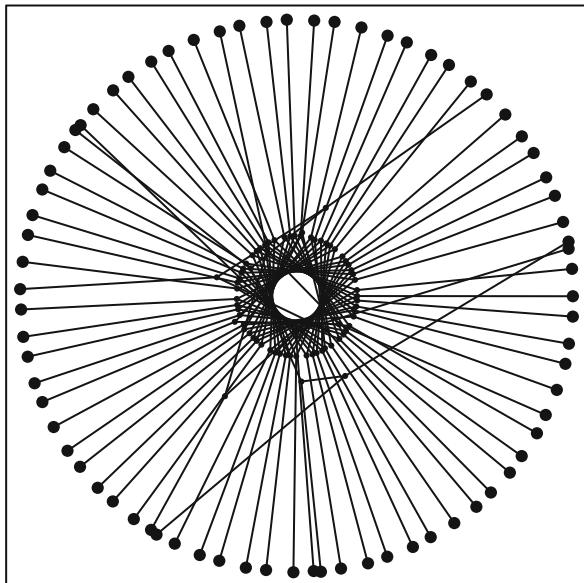


Fig. 20 End view of \mathcal{R} -sausage

- Figure 21 illustrates a Steiner tree construction where there are subsets of small vertex clusters $N = 2, 3, 4, 5, 6$ which comprise the Steiner tree solution.
- Thus, the discussion of the properties of Steiner trees in §3 carry over to the discussion of an effective heuristic algorithm.

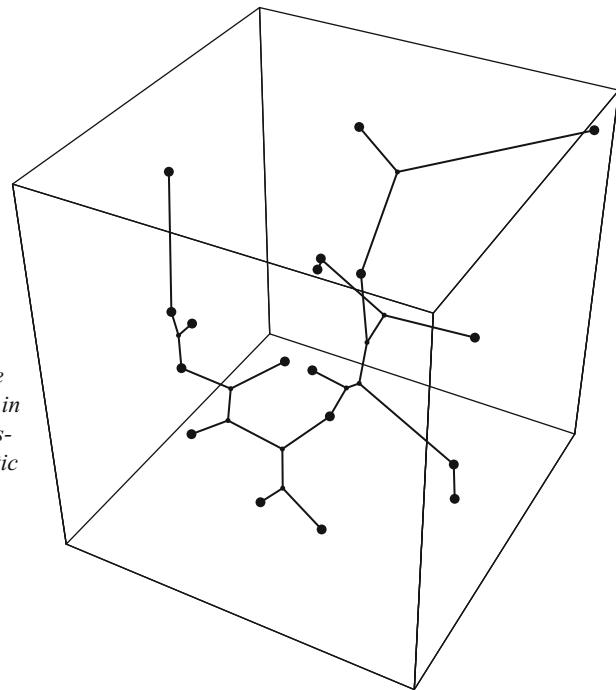


Fig. 21 Steiner construction $N = 18$

is important in the next section of this chapter for the development of a methodology to decompose an arbitrary vertex set and eventually develop a heuristic algorithm. As was the case in E^2 and on the unit sphere, decomposing according to the optimal vertex set configuration should yield an effective heuristic.

4 Heuristics

In general, when faced with computing a Steiner tree for a large randomly generated vertex set in E^3 with no special structure, optimal solutions will probably not be possible. Thus, some type of decomposition procedure is necessary to make solutions tractable. Of course the most natural decomposition tool is the \mathcal{R} -sausage itself, since as was the case in E^2 and the unit sphere, the equivalent optimal configuration vertex set should be a good decomposition strategy. Once the subsets of vertices are carved out of the larger set, then fast computation of these small subsets is important to the overall Steiner construction for the entire vertex set. If this construction can be done in better time than with an exponential worst-case time algorithm, then the overall running time and performance of a heuristic will be greatly enhanced.

4.1 CG Approach

The general strategy followed here in constructing E^3 Steiner trees is a locus-based strategy and is founded on past approaches for E^2 [73] and on the unit sphere [29]. These past approaches rely on computational geometry \mathcal{CG} data structures from which the local optimal solutions are formed. The crucial concepts needed in E^3 are the following [36]:

Voronoi Polyhedra: Given a set of vertices V in E^3 , the *Voronoi cell* of a vertex v_i is the set of vertices in E^3 closer to v_i than to any other given vertex v_j , $j \in V$. This set can also be described as an intersection of half-spaces.

$$VOR(v_i) = \cap_{i \neq j}^N H_3(v_i, v_j)$$

Voronoi diagram: Is the collection of all nonempty Voronoi cells. The intersection of two adjacent Voronoi polyhedra is a *Voronoi plane*, and the Voronoi planes intersect in Voronoi edges and eventually Voronoi vertices.

Delaunay tetrahedralization: Is also a cluster of simplicial polyhedra in E^3 and is the mathematical dual of the Voronoi diagram. In addition, the facets of the Delaunay tetrahedralization can be characterized by the property: *For T a proper triangulation, let $\text{maxrad}(\Phi)$ be the maximum radius of any minimum containment sphere of any simplex in T , then overall proper triangulations T in E^3 , the Delaunay triangulation minimizes $\text{maxrad}(\Phi)$* [36].

The Delaunay tetrahedralization is utilized as the basis for the algorithms since the equilateral tetrahedra are crucial for the optimal Steiner point set configurations as discussed in Sect. 3. Beasley and Goffinet's use of the Delaunay triangulation in their heuristic for the SMT problem in the E^2 yields the best results so far for the plane [6] and reinforces the approach for E^3 . There are other methods of tetrahedralization possible, yet for the purposes of computing Steiner trees, the more equilateral the tetrahedralization, the better in light of the \mathcal{R} -sausage properties.

4.2 Ribbon Decomposition Problem

Given the fact that the infinite ribbon or triple helix is conjectured to be the optimal configuration, one needs to utilize these ribbons to effectively decompose a vertex set so that an overall solution to the problem can be achieved. Even if it turns out that the infinite ribbon is not optimal, it still provides a valid decomposition strategy for a heuristic to the problem. In this section, vertex sets which are comprised of ribbons of regular tetrahedra joined together are examined. Although this is still not the most general case, one needs to illustrate how the tetrahedra join together, and it is clearer to illustrate the algorithmic concepts with equilateral tetrahedra.

Keeping in mind how the buckyball rule generates ribbons, it is instructive to consider junctions of these ribbons when this rule is not followed. If the next vertex in generating a ribbon is chosen to be equidistant from three vertices that are not the

ones most recently added, then one achieves a *two-way* junction. For example, if the ribbon has 10 vertices, then choosing a vertex equidistant from 7, 9, and 10 would break the \mathcal{R} -sausage structure. If in addition the 12th vertex were chosen to be equidistant from 7, 8, and 10, one would get a *three-way* junction.¹ This construction can be generalized to non-equilateral tetrahedra as well.

Some additional notation and definitions are useful at his vertex:

Centroid Spanning Tree (CST): Given the Delaunay triangulation, the MST of edges which is a subgraph of the Delaunay has for each tetrahedron a centroid. For the equilateral tetrahedra, it is also a Voronoi vertex. The collection of tetrahedra with these centroids is referred to as a centroid spanning tree (CST). With irregular tetrahedra, the Voronoi vertex which is the center of a sphere through the tetrahedral vertices may lie outside the tetrahedral convex hull, yet with a regular tetrahedra, this problem will not be encountered.

Ribbon (\mathcal{R}): Within the CST, there are chains (links) of tetrahedra which for the sake of the argument will be called ribbons \mathcal{R} . They are ribbons of varying length ℓ_i and corresponding ρ^i which for the regular tetrahedra can be estimated without actually calculating the Steiner-tree solution since the ρ_3 is a decreasing function as was shown earlier.

Ribbon Junctions: Within the CST, the ribbons intersect at junctions $\delta_0, \delta_2, \delta_3$, and δ_4 of degree 0, 2, 3, and 4 of tetrahedra, respectively.

δ_0 -Junction: An “empty” junction of two ribbons with no common tetrahedron between them. However, the two ribbons share a common degenerate Steiner point.

δ_2 -Junction: A junction of 2 Delaunay tetrahedra with $\rho^{2j} = 0.8155$

δ_3 -Junction: A junction of 3 Delaunay tetrahedra with $\rho^{3j} = 0.8288$

δ_4 -Junction: A junction of 4 Delaunay tetrahedra with $\rho^{4j} = 0.9005$

The example illustrates an optimal SMT configuration where two of the given vertices act as degenerate Steiner points. While $N = 16$, there are only $M = 12$ Steiner points. Empty junctions are interesting because they allow the \mathcal{R} -sausage topologies to be independent, i.e., there is no loss in the Steiner ratio across the ribbons. These empty junctions deserve a study in their own right, since they may exist in natural phenomenon and may also be the foundation for certain organic and inorganic applications, e.g., single- and double-chain silicate compounds and other ceramic structures [54].

The other δ_i junctions are problematic since the sausage topologies get corrupted and the ρ_3 becomes less than the expected optimal value. Actually, however, for the heuristic algorithm, it is exactly how the junctions are dealt with that is the key problem in the ribbon decomposition process.

¹Following the buckyball rule, a *one-way* junction is simply the \mathcal{R} -sausage.

- Figure 22 represents a junction of 3 Delaunay tetrahedra with $\rho_{3j} = 0.8288$.

- The solid circles are vertices from the original set. Empty circles are additional Steiner points.

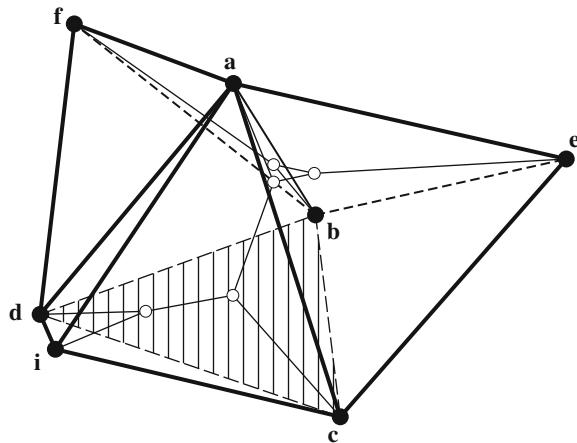


Fig. 22 3-way junction $\rho_3(V) = 0.8288$

4.3 Heuristic Algorithm

The algorithm is essentially a greedy one based on using the MST as a decomposition guide and the Delaunay triangulation to construct the ribbons of tetrahedra. It seeks to decompose the ribbon data structures in such a way as to minimize the Steiner ratio for each piece. The Steiner ratio depends on two factors: *the length of each ribbon and the degree of each junction*. Therefore, the algorithm tries to *maximize the minimum length ribbon*, but at the same time it seeks to reduce the degree of each junction in the ribbon data structure. Figure 24 illustrates diagrammatically how the heuristic works. The sawtooth line of Fig. 24 represents the MST and the tetrahedral ribbon data structure defined by the MST. The entire ribbon structure is to be decomposed while seeking to maximize the minimum length of ribbon, unioning the largest junctions of ribbons wherever possible and concatenating the local optimal solutions within the decomposition process.

In the following section, a formal description of the algorithm is presented. Data structures used in the algorithm are described in [74].

4.4 General Algorithm Description

The algorithm carefully examines the junctions of the ribbons and attempts to unite the local optimal solutions as efficiently as possible.

Step 1.0 Establish the CG Data Structure: Construct the convex hull and Delaunay triangulation data structure in E^3 , $DT(V)$.

Step 2.0 Establish an Upper Bound on the SMT: Solve the MST with the $DT(V)$ edge set.

- Figure 23 illustrates one cluster of three separate chains where each chain has three tetrahedra formed via the Bucky-ball rule and there is no degradation in ρ_3 for the entire vertex set i.e. the ρ_3 for each separate, ribbon is the optimal ρ_3 expected for 3 tetrahedra.

- The shaded triangles in the upper 3-d figure are the large ones in the lower figure, which is the 2-d representation of the same vertex set.

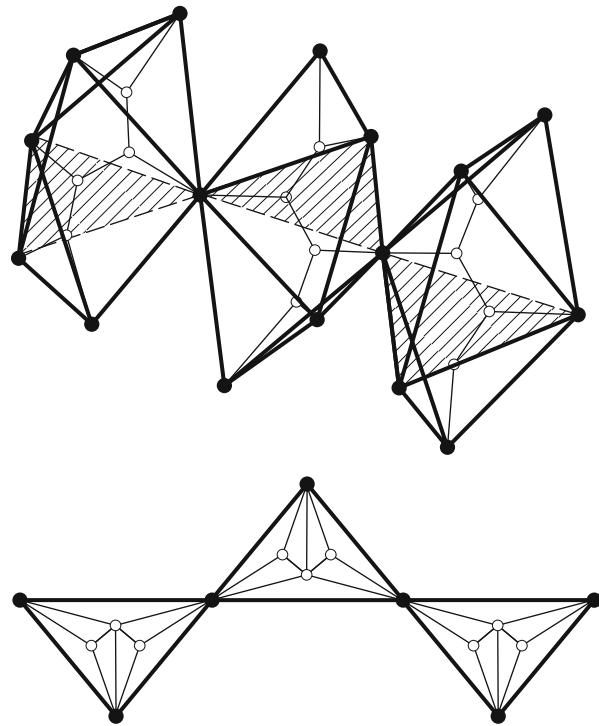


Fig. 23 3-disjoint ribbons with empty junctions $\rho_3(V) = 0.808065$

Step 3.0 *Construct the \mathcal{R} Data Structure:* Identify the tetrahedra t_i, t_j, \dots, t_q sharing edges in the MST.

Step 3.1: Utilize the Voronoi locus information to determine adjacent tetrahedra in the MST.

Step 3.2: Utilize the CST to construct the longest chains \mathcal{R}_i of tetrahedra in the MST.

Step 3.3: Identify ρ^i within each \mathcal{R}_i .

Step 3.4: Identify the junctions of the \mathcal{R}_i .

Step 4.0 *Local Optimal Solutions:* Create a priority queue \mathcal{Q} of \mathcal{R}_i sorted on their Steiner ratio.

Step 4.1: Select an \mathcal{R}_i and determine adjacent \mathcal{R}_j incident to \mathcal{R}_i at both ends if possible of \mathcal{R}_i

Step 4.2: Choose the largest junction and determine the adjacent \mathcal{R}_j which maximize the minimum length \mathcal{R}_j . If the sets of ribbons can be unioned together, go to Step 4.3, otherwise:

4.2.1: Using the longest \mathcal{R}_i at the junction, determine the face \mathcal{F}_i of the tetrahedron which ribbon \mathcal{R}_i is adjacent.

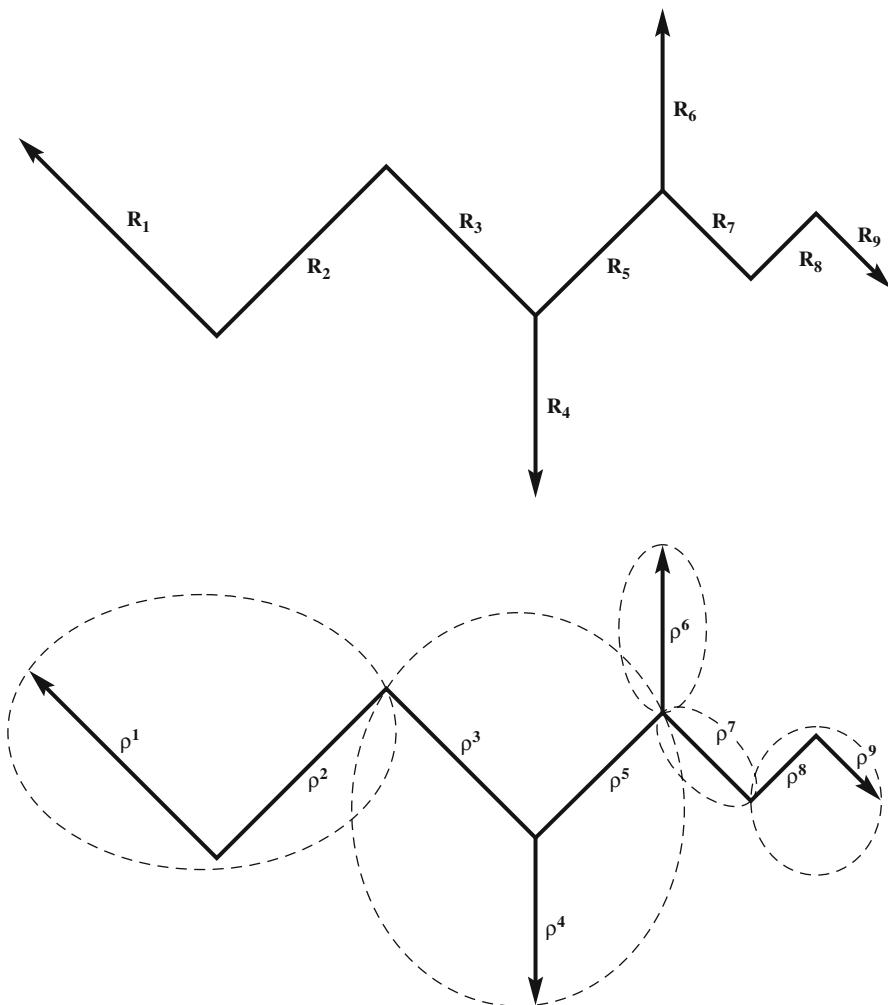


Fig. 24 \mathcal{R} -sausage decomposition process

4.2.2: Using this face, \mathcal{F}_i , determine the vertex of the tetrahedral junction nearest to the first three vertices of \mathcal{R}_i . Call this vertex v_k as the critical root vertex of the \mathcal{R}_i .

4.2.3: Using v_k to find a local optimal solution for the \mathcal{R}_i .

4.2.4: Find the local optimal solutions for the remaining \mathcal{R}_j , $j = 2, 3, 4$ using the appropriate root vertex of the junction tetrahedra.

Step 4.3: Unite the \mathcal{R}_i and \mathcal{R}_j , $j = 2, 3, 4$ and find the SMT of the union.

Step 4.4: Store the SMT solution and $k \leftarrow k + 1$ and return to 4.1.

Step 4.5: The process is complete once the priority queue \mathcal{Q} is exhausted.

4.5 Heuristic Example

The working of the heuristic is best described by way of an example. Consider the case of three ribbons connected by two 2-way junctions, as shown in Fig. 25. Figure 25 is a two-dimensional projection representation of the MST passing through the Delaunay data structure. The length of each ribbon is 3 tetrahedra, and they are numbered 0, 1, and 2.

1. *Locate the longest ribbon.* The program searches the list of ribbons, looking for the longest one. Since in this instance all ribbons are of identical length, the “chosen one” is \mathcal{R}_0 , the first one in the list of ribbons.
2. *Identify the ends of the chosen ribbon.* Check whether:
 - The chosen ribbon has junctions at both ends. If so, choose the junction with the larger number of ribbons and call it the “join.”
 - If all the ribbons at the join tetrahedron are as yet untagged, by which is meant that they have not yet been included in any collection of data vertices, then concatenate all the ribbons at this join, as well as the join itself into one data file. Mark all the concatenated ribbons as tagged.
 - If any of the ribbons that start/end at join are tagged, go to the other junction of the chosen ribbon, and repeat the above process. If this junction too has at least one previously tagged ribbon, return to the junction originally selected. In such cases, the vertices of the chosen ribbon will be the only vertices in the data set (Fig. 26).

In the example, the chosen ribbon, \mathcal{R}_0 , has junctions at both ends, both of which are two-way junctions. Neither of the other two ribbons has as yet been tagged, and so \mathcal{R}_0 are chosen and \mathcal{R}_1 as the ribbons to concatenate into a single data set, along with the junction tetrahedron, i.e., tetrahedron number 0.

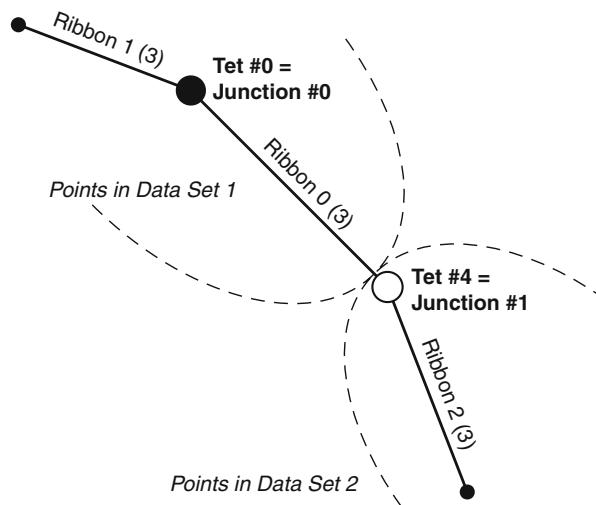


Fig. 25 Example two-way junctions

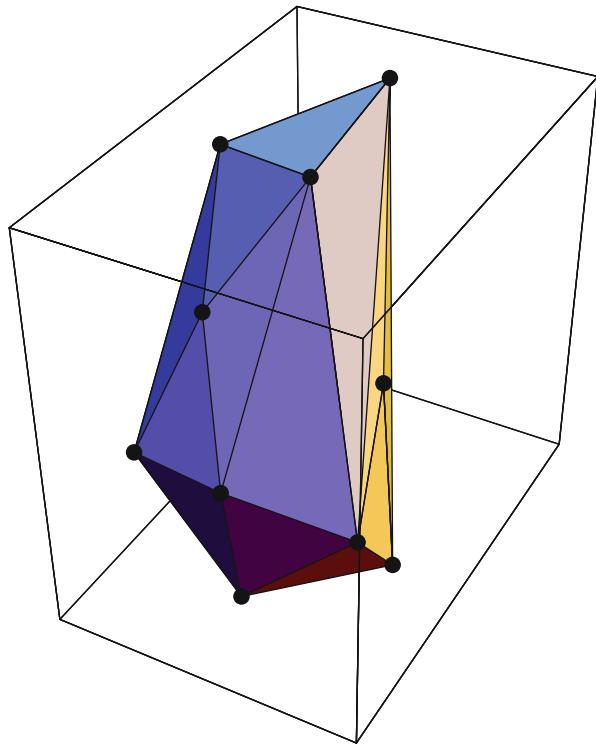


Fig. 26 Convex hull of example vertex set $N = 14$

3. *For each of the ribbons* that are to be concatenated together, one must check whether it has a junction at the other end. If so, one will need to “split out” that junction.
4. *Print out* the relevant vertices from each of the concatenated ribbons. These vertices are run through the SMT program, and the resultant tree is shown in Fig. 27 (left). The lighter lines represent the edges of the entire set of tetrahedra.
5. *The remaining vertices.* The algorithm is now run again on the remaining vertices in the data set, i.e., it searches from among the untagged ribbons the longest one and examines its junctions. In the example, the only remaining ribbon is \mathcal{R}_2 . Therefore, the second file of data vertices will consist of the vertices from \mathcal{R}_2 , along with the vertices of Junction 1, which is the connector to the data vertices in the first file. These vertices are now passed to the SMT program, and the resultant tree is in Fig. 27 (right).
6. *Calculating the ρ_3 ratio.* For each of the segregated data sets that the heuristic creates, a ρ_3 value is calculated. The SMT/MST ratio for the entire data set is then taken as the average of the individual ρ_3 ’s.
7. *Splitting out a junction.* This is a critical part of the heuristic. When a junction such as Junction 1 in the example above is to be split out, it is important to do it

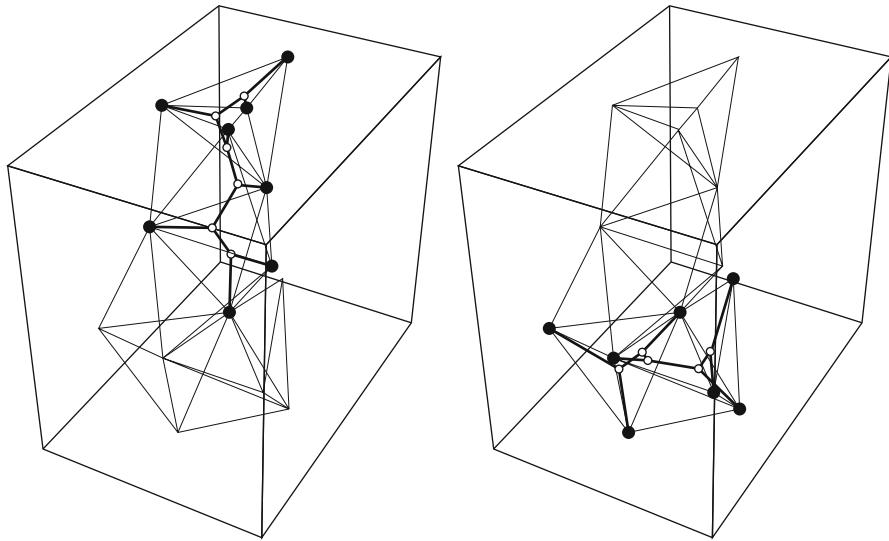


Fig. 27 SMT top part (*left*) and bottom part (*right*) of the data set

in a manner that allows one to maintain continuity between the different vertex sets, and also not introduce any additional distance into the overall tree. In the heuristic, this is achieved by examining the junction tetrahedron to be split out and identifying the vertex that is closest to the ribbon from which it is being separated. This vertex is then concatenated with the separated ribbon, so that a good connection vertex is available to link the Steiner minimal trees obtained from the different data sets.

The overall SMT is obtained by the superposition of the individual trees, and Fig. 28 on the left shows the overall SMT for the sample data set. The average ρ_3 value for the entire data set is 0.804496 and was obtained in 31 s.

By contrast, the tree that would have been obtained had the entire data set been passed directly to the SMT program is shown on the right in Fig. 28. The ρ_3 value thus obtained is 0.809361 and was obtained in 10 min (600 s) of computing time. This solution is obviously not guaranteed to be optimal.

4.6 Complexity Analysis

The worst case time bound for the three-dimensional $DT(V)$ must be $\Omega(n^2)$ [8]. An implementation of the Delaunay triangulation based on the work of Beichel and Sullivan [7] was employed. Steps 2.0 requires $O(N^2)$ time in the worst case, although some implementations of MST in E^3 will run faster. Step 3.0 is also $O(N^2)$ because there could be $O(N^2)$ tetrahedra. Step 4.0 and its substeps can be carried out in constant time if the search process is bounded above by a fixed

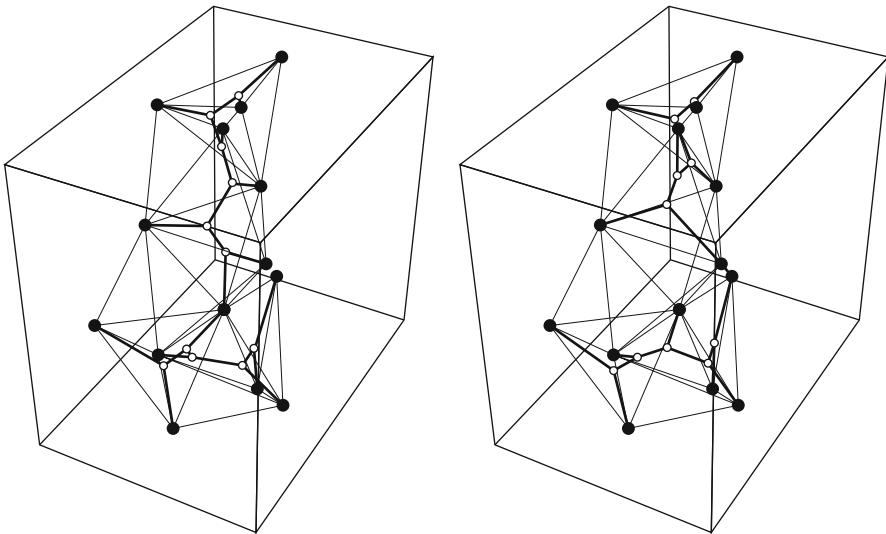


Fig. 28 Heuristic and optimal algorithm solutions

constant amount of time. Otherwise, it will be exponential; in fact, it will require $O(2^N N!)$ time since the numerical solution procedure is a branch-and-bound algorithm [68].

For the most general case where V are uniformly distributed within the unit cube and where the local optimal solution procedure is bounded by a constant search time C_R , then one can show:

Lemma 1 *The greedy heuristic will require $O(C_1 N^2)$ time.*

The result follows from the previous complexity analysis. The data structure is the key to the underlying complexity of the algorithm.

Since optimality is not being sought, and one knows the approximate length of the CST chains, then the local optimal search can be controlled, and one can achieve reasonably good solutions in polynomial time. The key factor here will be the expected ribbon length.

For the special case where equilateral tetrahedra are studied, one knows the ρ_3 function as a function of the length of R_i , then one can also achieve:

Lemma 2 *For the equilateral tetrahedra case, the greedy heuristic will require $O(C_2 N^2)$ time.*

It is this latter case that is tested in this heuristic procedure for this chapter, where the V have special structure. While one could be criticized for testing the heuristic

on this special equilateral case, this is a case where many degeneracies occur, so it is actually quite general.

Unfortunately, one cannot provide a performance guarantee like that provided in the E^2 because of the lack of knowing the optimal ρ_3 in E^3 .

4.7 Experiments

As one measure of the worst case running time performance of the heuristic process, the capability of the optimal seeking subroutine algorithm to solve long chains of tetrahedra needs to be assessed. To estimate the feasibility of this, the following experiments were performed on \mathcal{R} -sausage chains of lengths $N = 25, 50, 100$. Those values which are blank in [Table 2](#) represent the fact that the algorithm stopped without any solution obtained. Thus, as one sees, relatively short chains will be examined in order to keep the computing times reasonable, and this underscores the need for the heuristic. Also one sees that for $N = 100$, one is coming quite close to the conjectured optimal $\rho_3(V)$ for E^3 , although the above results are not guaranteed to be optimal. In order to test the heuristic, the following experimental design was established (see [Fig. 29](#)):

- Two-way \mathcal{R} – sausage networks • Three-way \mathcal{R} – sausage networks
- Four-way \mathcal{R} – sausage networks • Many-way \mathcal{R} – sausage networks

[Figure 29](#) represents $2-d$ projections of the MST and tetrahedral ribbons through which it passes. No δ_0 ribbon junction examples were included, since that would bring one into the domain of examining how the optimal sausage/ribbon topology should be combined to yield the best possible vertex sets *a la* Issue #2. This will be examined in a future monograph.

Also, the length of the \mathcal{R} -sausage chains was kept relatively small so that the local optimal search procedure is reasonably bounded in running time. Larger vertex sets are feasible with a consequent increase in running time although very reasonable. [Figure 30](#) on the left illustrates the performance of the heuristic in relation to the optimal seeking algorithm for short symmetric vertex sets of two-way, three-way, four-way, and multi-way junctions. [Figure 30](#) also illustrates on the right a series of experiments for short asymmetric vertex sets. [Figure 31](#) illustrates the

Table 2 Optimal seeking experiments

Time (min)	$N = 25$	$N = 50$	$N = 100$
1	0.793215	-?-	-?-
2	0.793215	-?-	-?-
3	0.793215	0.787845	-?-
10	0.793215	0.787845	-?-
15	0.793215	0.787845	0.786049
20	0.793215	0.787845	0.786049

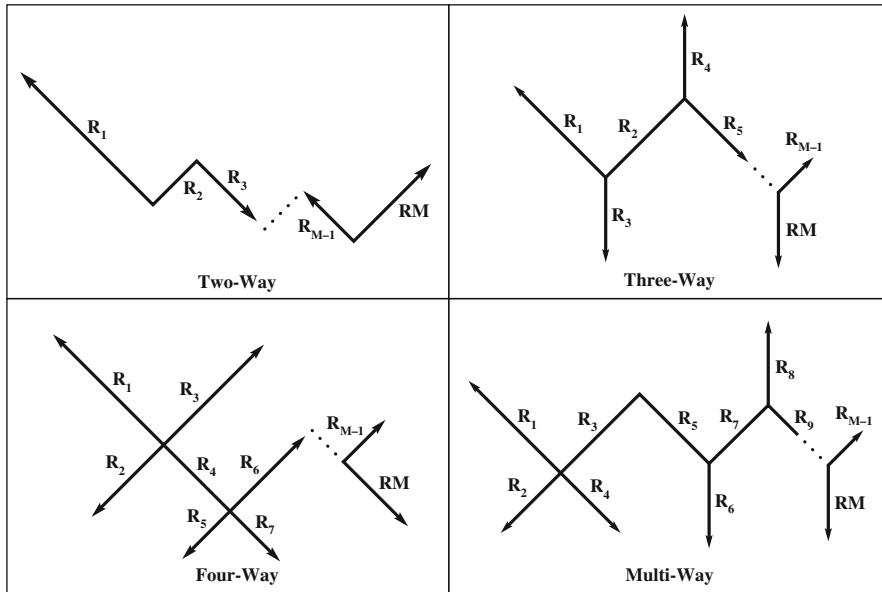


Fig. 29 Experimental design

longer experiments for symmetric and asymmetric vertex sets. Running times for the heuristic on all these vertex sets are arrayed in Table 3. Certainly for some of the smaller vertex sets, the optimal seeking program can do better, yet the percentage improvement over the heuristic is not significantly different. Also, for some of the longer more complex configurations, the run times and the ρ_3 values for the optimal seeking algorithm are much worse.

As for a more general heuristic for arbitrary point configurations of much larger cardinality, the reader is referred to the paper by Toppur and Smith [79] where they develop an $O(N^2)$ heuristic for points generally distributed in space. They utilize the \mathcal{R} -sausage to decompose the point set. The percentage reduction in the Steiner ratio is not as significant as that which occurs in the previously described heuristic, but this is understandable in light of the special structure nature of the previous heuristic.

5 Applications

Given what has been shown so far about Steiner trees in E^3 , it is quite natural to wonder whether the long extended structure of the conjectured optimal configuration might imply or have relevance to certain physical, biochemical, or engineering applications. The most obvious application that occurred was that the Steiner problem in E^3 might help explain the reason for the long molecular chains of atoms in proteins and DNA and other molecular structures found in

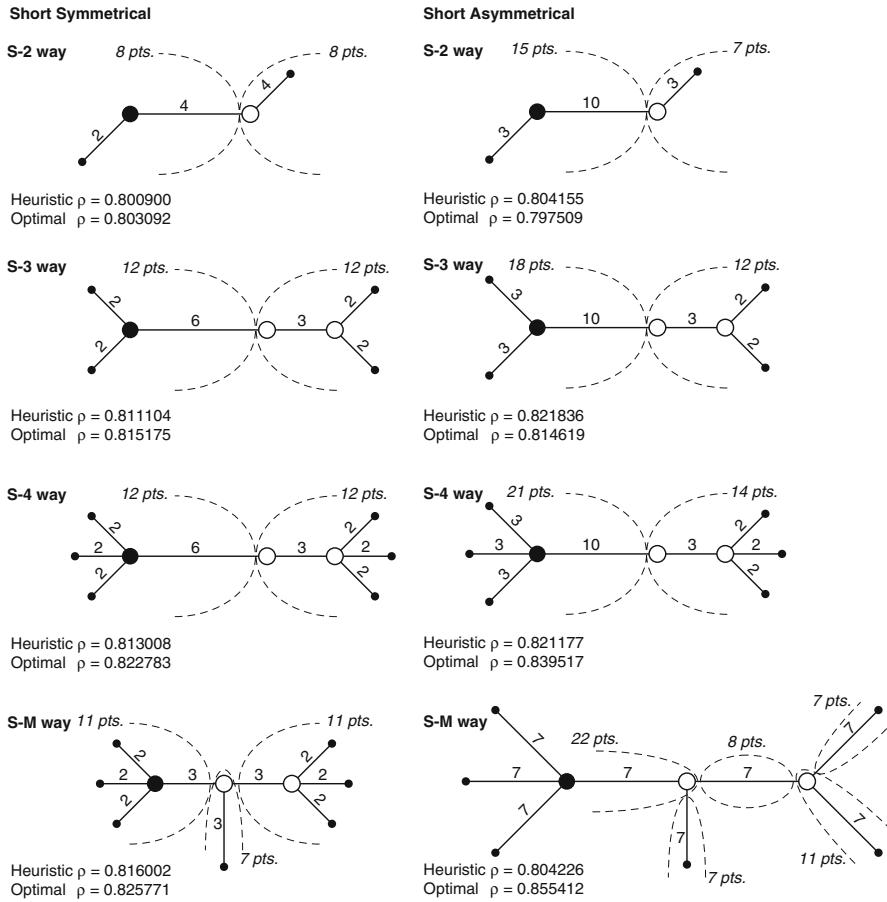
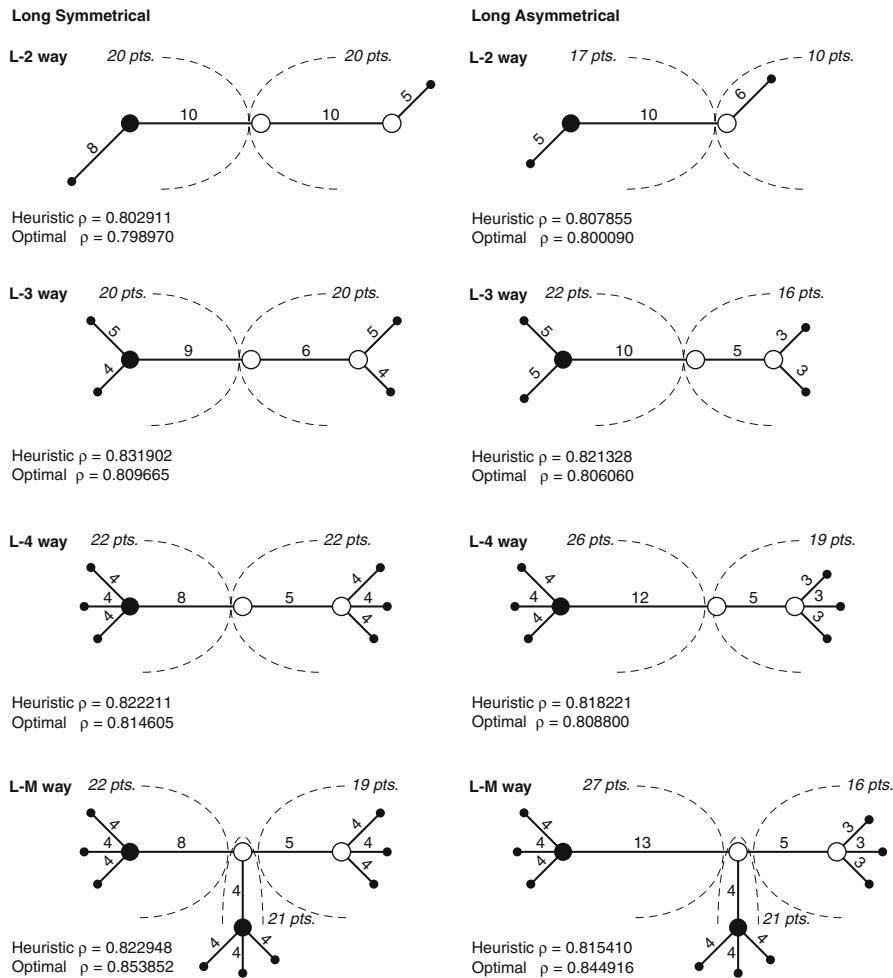


Fig. 30 Short-chain experiments

biochemistry [11, 45, 59, 71]. In order to examine this potential application area and others related to it, possible linkages between the objective function of the Steiner problem and objective functions of these applications in the physical, biochemical, or engineering sciences need to be examined.

5.1 Minimum Energy Configurations (MECs)

To clarify and delineate the connection between the scientific and engineering applications and the Steiner problem, one needs another property of the Steiner problem which was first shown in the classic paper on Steiner trees by Gilbert and Pollak [41]. It is recounted as Maxwell's theorem after the famous physicist.

**Fig. 31** Long chain experiments**Table 3** Computation times (secs.) heuristic vs. optimal algorithm

Topology	Short symmetric	Short asymmetric	Long symmetric	Long asymmetric
Two-way	65/120	57/120	120/240	129/240
Three-way	120/240	129/240	120/240	165/360
Four-way	120/240	131/240	121/240	273/360
Multi-way	128/600	174/600	300/600	678/720

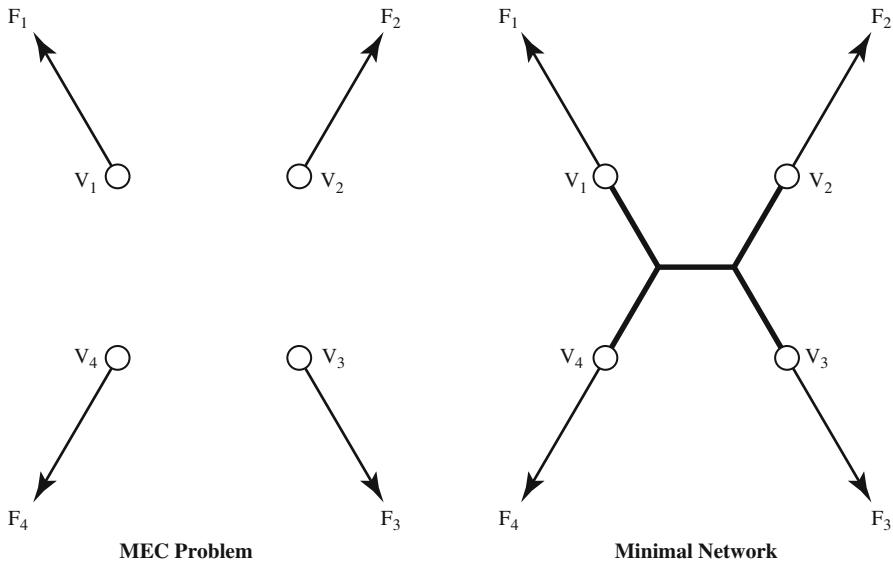


Fig. 32 Maxwell's theorem illustration

Let F_1, F_2, F_3 , and F_4 be unit forces acting at fixed vertices v_1, v_2, v_3 , and v_4 , respectively. Also, if one tried to design a network with moveable Steiner vertices to link up the fixed ends with elastic bands where each band will have a tensile force, then one seeks to find the network where these tensile forces will be in equilibrium. Figure 32 nicely illustrates the MEC.

Theorem 4 *If one draws unit vectors from a Steiner tree in the direction of each of the lines incident to v_1, v_2, \dots, v_n and lets F_i denote the sum of the unit vectors at v_i , then in mechanical terms, F_i is the external force needed at v_i to hold the tree in equilibrium. The length of the tree T has the simple formula*

$$T = \sum_{i=1}^n v_i \cdot F_i$$

The proof is in their paper [41].

What Maxwell's theorem implies is that the minimal-length Steiner tree is equivalent to the equilibrium configuration of vertices which minimizes the potential energy between them. Maxwell's application was to determine the minimum weight truss made from pin-jointed rigid rods and holding a given set of forces $\{F_1, \dots, F_n\}$. Maxwell's theorem is more general in that it applies to circuits as well as trees, and the forces need not be all uniform, although for the Steiner problem, the uniform forces are required [63].

5.2 Lower Bounds

The importance of the Steiner tree for MECs is that if there are unit forces acting at the nodes, then the Steiner tree which is the minimal-length network yields the optimal MEC. On the other hand, if the forces are not all uniform, e.g., some forces are in compression as well as tension or vary in order of magnitude, then the Steiner minimal tree yields only a lower bound. What will be shown in the experiments on proteins which follow is that this lower bound provided by the SMTs is surprisingly *tight*.

6 Fundamentals of Protein Structures

There are four major types of macromolecules of interest in the biological sciences: carbohydrates, nucleic acids, proteins, and lipids. Protein structures will be the focus because the structural geometry of proteins relates most closely to what has been found with the ideal Steiner R —sausage geometry, but as will be argued, a similar type of structural analysis could be mounted for the other types of macromolecules, especially DNA and RNA.

A key issue in biochemistry today, as pointed out in the introduction, is to predict the 3-D structure of proteins from the primary sequence of amino acids [11, 45]. Basically, two approaches to this problem have been developed over the years; see [22]: (i) threading or homological methods and (ii) hierarchical condensation methods. Threading methods rely on the comparison of unknown protein structures with known protein structures. Basically, researchers decompose a protein into its constituent parts, analyze the parts independently, then match the proteins to those in the known database; see [27, 66].

The other approach to protein prediction is to rely on the fundamental building blocks of protein structure, i.e., the amino acids and how they are assembled together in E^3 according to a set of rules. This approach to protein modeling harbors the most power and flexibility in dealing with unknown protein structures. For the hierarchical condensation methods, researchers view the problem with identifying and predicting protein structures in the following manner: First they define the fundamental building blocks of the amino acid sequence as the *primary* level. With the known amino acid sequence, the types of geometric structures these linked sequences can develop are viewed as the *secondary* level. Most typically, the geometry of the secondary structures can be either an alpha-helix, beta sheet, or combinations of these structures. The way the geometric structures pack or “fold up” in space is called the *tertiary* level, and finally, the *quaternary* level represents the final composite structure of all the secondary and tertiary structures within the protein. This hierarchical approach is the one considered here.

6.1 Steiner-Algorithm Approach

A computer algorithm similar to that used in [70] to evaluate the proteins will be utilized. It is a branch-and-bound optimization algorithm as in [68] for solving Steiner trees in E^d , and the version used to solve the problem in E^3 seeks an optimal solution. Since it seeks to find an optimal solution by adding a maximum of $N - 2$ Steiner points so that the structure is an FST, its running time is exponential. Performance for running time will be traded off since to know the optimal configuration of the amino acids and proteins, if possible, is most desirable. However, unless the terminal vertex sets are $N \leq 13$ or have special topological structure, optimality cannot be achieved within a reasonable amount of computing time. Running times for the amino acids and the proteins included in this chapter were on the order of 0–600 min of computing time.

In the experimental part of this chapter, the fundamental geometry of the individual amino acids will be examined for their Steiner properties, then proceeding to modeling protein secondary structures and perturbations for their structure. When HIV-1 protease is modeled, the tertiary structure will also be examined. Ultimately, how the Steiner properties transcend all the structural levels will be the goal, but in this chapter, the primary and secondary levels will be the main focus [77].

6.2 Primary Structure of Amino Acids

The individual amino acids will be first examined to see what Steiner properties result. After this, the robustness of different proteins with respect to how the Steiner ratio can be used to model the proteins will be evaluated.

The primary structure of a protein is made up of different amino acids coupled head to tail like beads on a string. Each amino acid has a carbon atom with an R group bonded to it, an amino group on the left, and a carboxyl group to its right. Each amino acid differs in its side chain, i.e., its R group. The carbon atom connecting the side chain is called the α carbon, and subsequent carbon atoms in the side chain are alphabetically ordered $\beta, \gamma, \delta, \dots$. A typical schematic diagram of an amino acid occurs in Fig. 33 on the left.

There are 20 amino acids that naturally occur in proteins, and these 20 are combined in a variety of ways to yield the various proteins and protein structures. For those not familiar with amino acids, a functional classification of the 20 amino acids appears in Fig. 33 from [22] on the right.

What has been done in the experimental part of this section is simply to take each amino acid and calculate its Steiner tree. Remember, in solving for the Steiner points, the algorithm does not know a priori where the Steiner points should be located. The algorithm systematically adds $N - 2$ Steiner points to the given terminal vertex set V in order to reduce the overall length of the spanning tree connecting V . It seeds the points through the midpoints of the edges of the set V , then tries to relocate them according to topology and derivative information from the objective

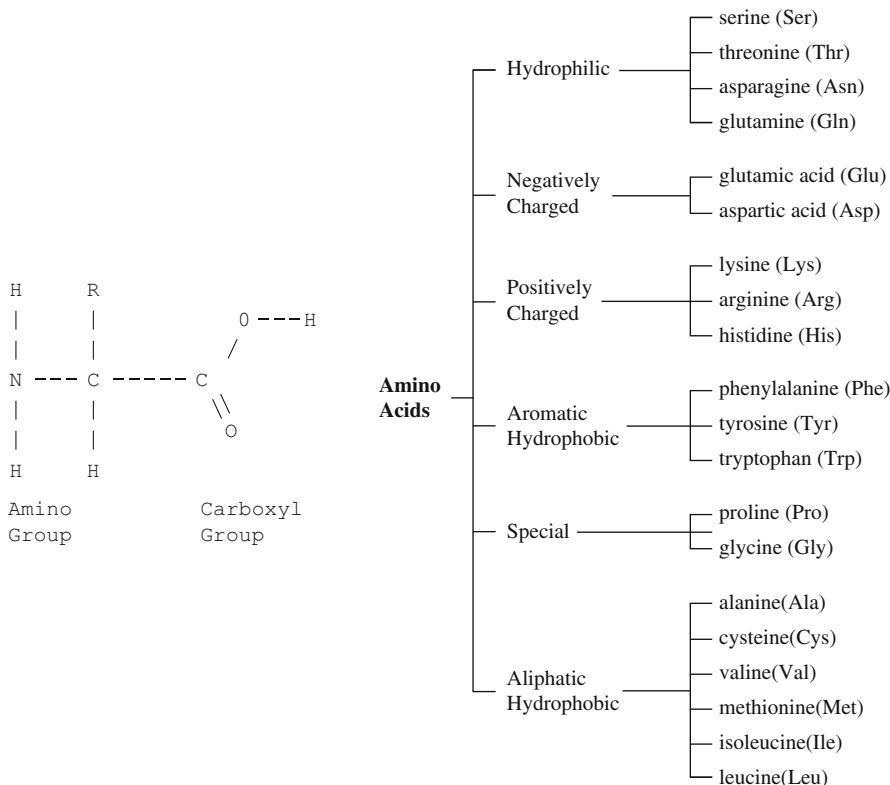


Fig. 33 Amino acid structure and classification

function of the Steiner-tree problem so as to satisfy the 120-degree-angle property of the edges incident to the Steiner points. The algorithm does not start off with any prior information on where the Steiner points should be located. It terminates after it examines all possible topologies and cannot reduce the SMT length further. As will be shown, the carbon and nitrogen atoms function as Steiner points in the amino acid structures. Table 4 illustrates the results found. All 20 different amino acids available from the education site of the [64] will be examined. It needs to be shown what information can be derived from computing the Steiner tree for each amino acid. Different data set realizations of the amino acids would reveal different SMT and MST solutions, yet similar topological results should occur as in Table 4. The detailed Steiner-tree outputs for each acid are available from the author.

In reading the rightmost two columns of Table 4, the $S' \subseteq S$ column represents the subset of Steiner points that correspond to the given carbon and nitrogen atoms in the amino acid, and the last column represents the combined number of carbon and nitrogen atoms within each acid. That the Steiner program basically identifies all the carbon and nitrogen atoms as Steiner points from first principles is quite

Table 4 Amino acid results

Name	SMT	MST	ρ	V	$S' \subseteq S$	$C + N$
Alanine (Ala)	14.394	14.478	0.9942	13	4	4
Arginine (Arg)	31.162	31.357	0.9938	27	10	10
Asparagine (Asn)	19.382	19.481	0.9949	17	6	6
Aspartic acid (Asp)	17.292	17.391	0.9943	15	5	5
Cysteine (Cys)	16.395	16.541	0.9912	14	4	4
Glutamine (Gln)	23.071	23.186	0.9950	20	7	7
Glutamic acid (Glu)	20.991	21.096	0.9950	18	6	6
Glycine (Gly)	10.720	10.775	0.9949	10	3	3
Histidine (His)	23.291	23.518	0.9904	20	8	9
Isoleucine (Ile)	25.417	25.586	0.9934	22	7	7
Leucine (Leu)	25.447	25.588	0.9945	22	7	7
Lysine (Lys)	28.912	29.102	0.9935	25	8	8
Methionine (Met)	23.801	23.990	0.9918	20	6	6
Phenylalanine (Phe)	27.222	27.300	0.9972	23	9	10
Proline (Pro)	19.491	19.756	0.9866	17	6	6
Serine (Ser)	15.696	15.792	0.9940	14	4	4
Threonine (Thr)	19.359	19.494	0.9931	17	5	5
Tryptophan (Trp)	32.253	32.478	0.9931	27	12	13
Tyrosine (Tyr)	28.473	28.562	0.9969	24	10	10
Valine (Val)	21.752	21.885	0.9939	19	6	6

remarkable. It is remarkable that one can predict the placement of certain atoms within a molecule without using quantum mechanics.

In [Table 4](#), the alanine and glycine amino acid SMTs are optimal solutions. It took 440 min of computing time to find optimality for the ($N = 13$) alanine while only 8 min for the glycine ($N = 10$). In the other amino acids, the results are upper bounds, since the algorithm did not converge within a reasonable amount of running time (≤ 600) min. For instance, since cysteine has $N = 14$ atoms, as a test, the program was run for 1,000 min of computing time, but it did not converge to optimality. The design of more efficient algorithms for SMTs ($N > 13$) is crucial to future progress in this area.

Alternatively, as was developed in [\[75\]](#), a heuristic with polynomial running time of $O(N^2)$ could be used, but it is the optimal structure for the proteins sought, and that is why the branch-and-bound algorithm was chosen. There are some theoretical heuristics by Arora [\[3\]](#) that are supposedly better than the $O(N^2)$ one, yet they are not practical, and no implementation is known that is either available or better than the algorithms used in this and previous papers. Also, the fact that optimal solutions are sought, especially for the small-cardinality points sets of the amino acids, implies that the branch-and-bound algorithm is the one desired. The program comes with an output file generation so that the Steiner-tree output of the solution can be viewed in *Mathematica*; see [\[87\]](#). The algorithm is available from the author upon written request.

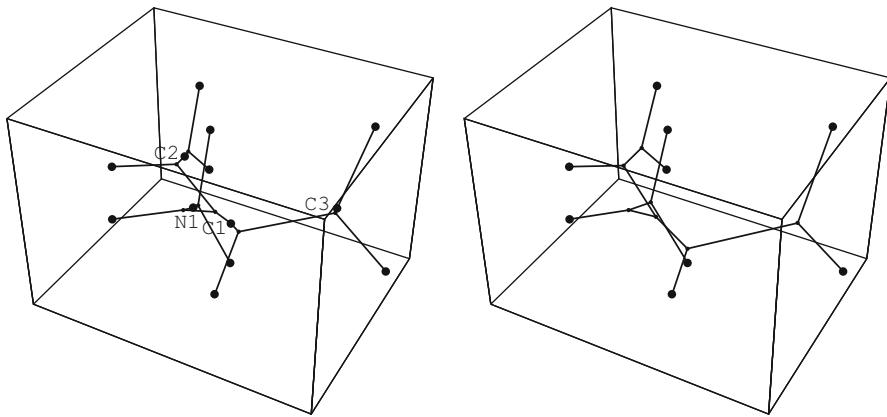


Fig. 34 Alanine native (*left*) and perturbed structure (*right*)

With the exception of histidine, phenylalanine, and tryptophan, the algorithm identified all the carbon and nitrogen atoms as Steiner points. Only for the histidine it did not locate the single nitrogen atom, and in the phenylalanine and tryptophan it did fail to identify one of the carbon atoms. The reasons for this are probably due to the inability of the computer program for the Steiner trees to check exhaustively all the full Steiner topologies and optimally locate all the correct Steiner points within a reasonable amount of computing time, since the number of atoms (terminals) in the histidine, phenylalanine, and tryptophan acids are relatively large. The number of possible full topologies for a given terminal vertex set is exponential in N ; see [41]. Remember, optimal results are sought here. In all other cases, it exactly identified the carbon and nitrogen atoms acting as Steiner points.

While some of the carbon and nitrogen atoms are Steiner points of degree 3, many of them are degenerate since their degree is 2 and their angles do not equal $2\pi/3$. Figures 34 on the left and 35 on the left illustrate this situation where some of the carbon and nitrogen atoms are degenerate Steiner points. All in all, it appears that the resulting Steiner network acts as a rigid structural framework for the amino acid, much in the spirit of Maxwell's theorem. The importance of this framework in the next section will be illustrated when the structure is perturbed.

None of the oxygen or sulfur atoms are ever identified as Steiner points. The hydrogens are terminal vertices of degree 1, so they cannot be Steiner points. An open research question here is the following: *Is there any significance to the Steiner points that do not correspond to the given atoms in V ?*

Also, of some note is the surprisingly high $\rho \approx 1.0$ value in the amino acids, which corresponds to some previous experimental results. When one considers the possible ρ associated with the R -sausage, one might think that the ρ of the amino acids should be closer to 0.7841903733 rather than 1. Why is there such a difference in the ρ values of the amino acids and the R -sausage? That the ρ value is also remarkably consistent in value across all the amino acids is indicative of

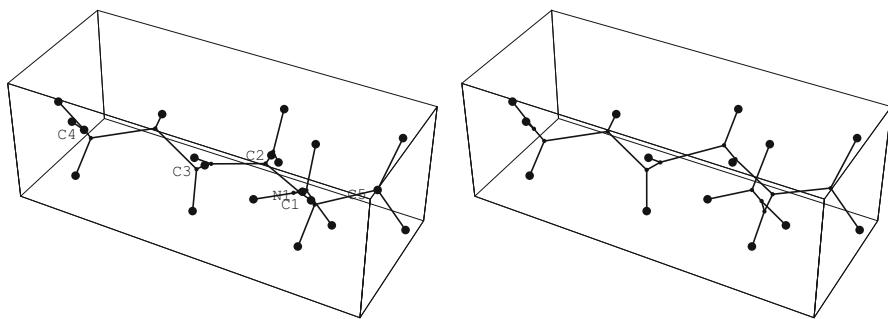


Fig. 35 Methionine amino acid native (*left*) and perturbed Steiner structure (*right*)

Table 5 Isoleucine SMT = 25.417093; MST = 25.586591; $\rho = 0.993376$

Atoms	Terminal coordinates				Steiner points			
(C) C1:	1.2250	0.3120	0.2470	→	s27:	1.2250	0.3120	0.2470
(C) C2:	-0.0650	-0.4640	-0.0120	→	s23:	-0.0650	-0.4640	-0.0120
(C) C3:	-0.2480	-1.5220	1.0720	→	s38:	-0.2480	-1.5220	1.0720
(C) C4:	-1.2510	0.4990	0.0100	→	s25:	-1.2510	0.4990	0.0100
(C) C5:	-2.5400	-0.2760	-0.2480	→	s41:	-2.5400	-0.2760	-0.2480
(N) N1:	1.4050	1.3490	-0.8150	→	s32:	1.4050	1.3490	-0.8150
(C) C6:	2.3980	-0.6390	0.2250	→	s28:	2.3980	-0.6390	0.2250
(O) v8:	3.1030	-0.7610	1.2130					
(O) v9:	2.6490	-1.3130	-0.8220		s24:	-0.0476	-0.1107	-0.0266
(H) v10:	1.1690	0.7950	1.2200		s26:	0.9814	0.4214	0.0203
(H) v11:	-0.0090	-0.9480	-0.9850		s29:	2.3980	-0.6390	0.2250
(H) v12:	2.2790	1.8740	-0.6400		s30:	1.3084	0.2917	0.3506
(H) v13:	0.6020	2.0010	-0.8000		s31:	-0.0862	-0.6323	-0.0489
(H) v14:	1.4590	0.8890	-1.7410		s33:	1.2104	1.3401	-0.7028
(H) v15:	-1.1210	1.2530	-0.7630		s34:	1.4905	1.3550	-0.8841
(H) v16:	-1.3080	0.9820	0.9830		s35:	-1.0692	0.5461	-0.1261
(H) v17:	-1.1670	-2.0750	0.8880		s36:	-1.3863	0.5040	0.1192
(H) v18:	0.5960	-2.2080	1.0560		s37:	-0.3553	-1.5083	0.9367
(H) v19:	-0.3050	-1.0390	2.0450		s39:	-0.1760	-1.5406	1.1596
(H) v20:	-3.3850	0.4090	-0.2320		s40:	-2.5394	-0.1073	-0.2096
(H) v21:	-2.6710	-1.0310	0.5250					
(H) v22:	-2.4840	-0.7600	-1.2210					

its importance in conformation of a protein, as will be argued in the next set of experiments.

Table 5 illustrates one detailed output from the Steiner program for the isoleucine acid describing the relationship between the carbon and nitrogen atoms and the Steiner points. When viewing the outputs of the amino acids, two columns were created, and those atoms identified as Steiner points (→) were aligned with those of

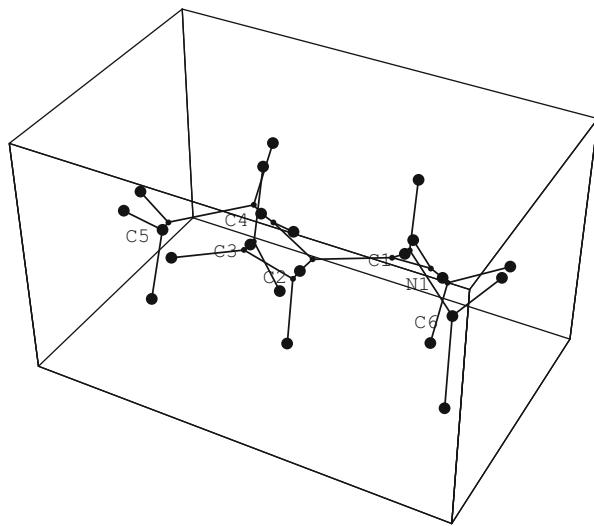


Fig. 36 Isoleucine acid Steiner tree

the input data set V , so the correspondence is clear. The indexing of the atoms in Fig. 36 corresponds to the order of the atoms given in the PDB file [65]. Since isoleucine has $N = 22$ atoms, the computer program added $M = 20$ Steiner vertices to create an FST, so that there should be a total of 42 vertices in all. Actually, the program found duplicates of Steiner point s_{38} so only 21 unique Steiner points are shown in Table 2. In fact, 7 of the 20 additional Steiner vertices that the program added happen to coincide exactly with the carbon and nitrogen atoms. Remember, when the program tries to find the additional vertices, it does not know beforehand the location of the Steiner points. Notice in Fig. 36 that atoms C_5 and C_6 are degree-3 Steiner points while the others are degenerate ones. Thus, the presence of the carbon and nitrogen atoms within the amino acids acting as Steiner points serves to move $\rho \rightarrow 1$, while in the R -sausage the Steiner points in the interior of the sausage are necessary, so $\rho \rightarrow 0.7841903733$.

6.3 Perturbed Amino Acids

If one generates an optimal SMT with a given terminal set V and Steiner points S , then one can create a new terminal vertex set $V' = V \cup S$ and resolve the SMT, and then $\rho = 1$, since there is no better way to improve the SMT.

In order to test the significance of the carbon and nitrogen atoms acting as Steiner points for the amino acids, they will be removed from the amino acids and examined to see what occurs experimentally as they are resolved with the Steiner program. Will the program replace the carbon and nitrogen atoms with Steiner points in the exact same locations?

Table 6 Perturbed amino acid results

Name	SMT	MST	ρ	SMT'	MST'	ρ'
Alanine (Ala)	14.394	14.478	0.9942	14.355	16.459	0.8721
Arginine (Arg)	31.162	31.357	0.9938	29.468	32.132	0.9171
Asparagine (Asn)	19.382	19.481	0.9949	19.180	20.857	0.9196
Aspartic acid (Asp)	17.292	17.391	0.9943	17.144	18.800	0.9119
Cysteine (Cys)	16.395	16.541	0.9915	16.347	18.425	0.8872
Glutamine (Gln)	23.071	23.186	0.9950	23.014	25.142	0.9154
Glutamic acid (Glu)	20.991	21.096	0.9950	20.907	23.908	0.9051
Glycine (Gly)	10.720	10.775	0.9949	10.708	12.307	0.8701
Histidine (His)	23.291	23.518	0.9904	21.011	22.575	0.9307
Isoleucine (Ile)	25.417	25.586	0.9934	25.135	28.009	0.8974
Leucine (Leu)	25.447	25.588	0.9945	25.276	27.812	0.9088
Lysine (Lys)	28.912	29.102	0.9935	28.674	32.636	0.8786
Methionine (Met)	23.801	23.990	0.9918	23.767	26.714	0.8897
Phenylalanine (Phe)	27.222	27.300	0.9972	26.132	26.969	0.9690
Proline (Pro)	19.491	19.756	0.9866	19.062	20.793	0.9168
Serine (Ser)	15.696	15.792	0.9940	15.660	17.719	0.8838
Threonine (Thr)	19.359	19.494	0.9931	19.255	21.330	0.9027
Tryptophan (Trp)	32.253	32.478	0.9931	29.774	29.976	0.9993
Tyrosine (Tyr)	28.473	28.562	0.9969	26.112	27.698	0.9427
Valine (Val)	21.752	21.885	0.9939	21.687	24.365	0.8901

As can be seen in [Table 6](#), removing the carbon and nitrogen atoms has a significant impact on the Steiner ratio. In most cases, the ρ for each acid dramatically drops to around $0.90 \pm 2\%$. This underscores the significant Steiner role of the carbon and nitrogen atoms.

In [Figures 34 and 35](#) on the right, the perturbed SMT solutions are comparable to the Steiner tree topology of the alanine and methionine native structures on the left in these respective figures. The algorithm roughly located new Steiner points in the locations abandoned by the carbon and nitrogen atoms. This further emphasizes the role of the Steiner tree and the Steiner ratio as a lower bound on the overall MEC topology since in all cases, when the amino acids were re-optimized, a better lower bound on the Steiner tree was realized.

Another interesting thing is that the actual length of the Steiner-tree in the perturbed structures SMT' is very close to the Steiner-tree length when the carbon and nitrogen atoms are included in the acid. In fact, in all cases, the $SMT' \leq SMT$ of the unperturbed structures in [Table 3](#). This lower bound is due to Maxwell's theorem.

This presence of the \mathcal{C} and \mathcal{N} atoms will increase the overall Steiner network length since the forces in the amino acids are not all uniform. Only for the arginine acid was the lower bound not very tight, i.e., the perturbed Steiner-tree length is much lower than the unperturbed structure. In [Fig. 37](#), it is seen that there is

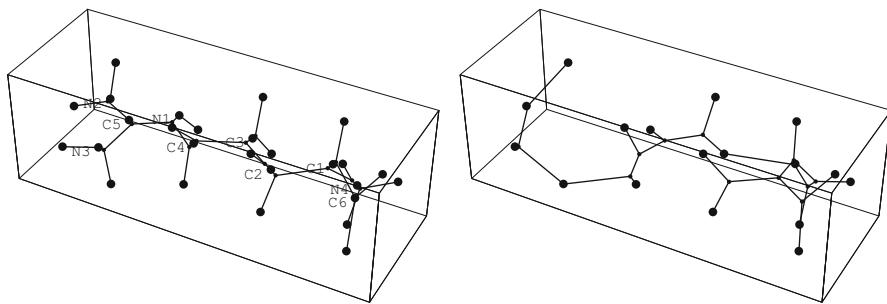


Fig. 37 Native arginine SMT (*left*) and perturbed SMT (*right*)

a dramatic change in the topology of the SMT from the native to the perturbed structure.

The Steiner points as generated in the acids without the carbon and nitrogen atoms are not “exactly” colocated with the coordinates of the displaced carbon and nitrogen atoms. There appear to be three reasons for this occurrence:

1. When the carbon and nitrogen atoms are removed from the amino acid, this reduces the cardinality of the amino acid N , and when the Steiner-tree program adds $N - 2$ Steiner points to make it a FST, the $N - 2$ Steiner points are not necessarily going to equal the number of carbon and nitrogen atoms that were removed. For instance, for alanine, $N = 13$ with four carbon and nitrogen atoms. If one removes the carbon and nitrogen atoms, $N = 9$ and so the Steiner program will try to add seven Steiner vertices rather than the 11 when alanine was unperturbed. Thus, the total number of Steiner atoms will not match the number of carbon and nitrogen atoms, so the Steiner points that are added will not likely coincide, and this might change the network structure dramatically, as it did in some cases.
2. The carbon and nitrogen atoms are functionally very important to each amino acid; they are really part and parcel of the terminal set V (witness the arginine acid in Fig. 37), and their coordinate presence is important to the entire acid structure and function. In Table 6, for the aromatic acids histidine, phenylalanine, tryptophan, and tyrosine, the MST' of the perturbed acid structure is below the SMT value of the unperturbed, which makes no sense, so removing the carbon and nitrogen atoms fundamentally changes the minimal-length network structure. This can probably best be seen in Fig. 38, with the amino acid tryptophan. The presence of the indole ring structure greatly affects the Steiner tree, because the carbon atoms are in the center part of the ring, and without them, the Steiner tree gravitates to the hydrogen atoms.
3. The forces in the amino acids are not all uniform, and this feature will cause variations in Maxwell’s theorem, so that the Steiner coordinates will likely be slightly different. The actual closeness of the Steiner lengths before and after the removal of the carbon and nitrogen atoms attests to these differences in Table 6.

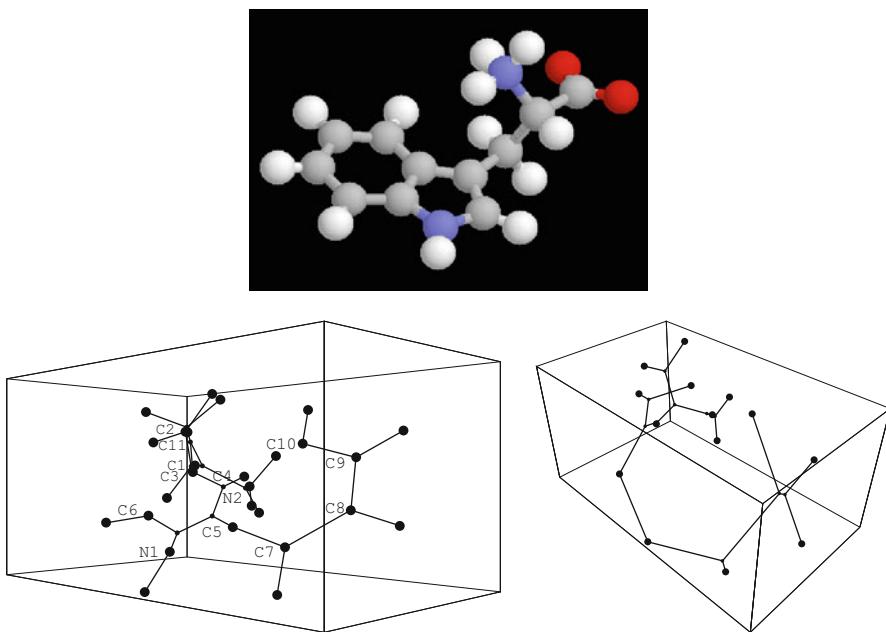


Fig. 38 Native tryptophan (*top*), Steiner (*left*), and perturbed structure (*right*)

Thus, as shown, the carbon and nitrogen atoms within the amino acids act as Steiner points in the chemical structure. What is needed here is to examine the Steiner role of the carbon and nitrogen atoms in the secondary structure of the proteins. In the experiments that follow, the perturbation experiments will be measured by programs that calculate the energies in the proteins. Thus, the importance of ρ and the energies in the proteins that give rise to the structures in 3-D are being conjoined.

7 Secondary Structure

The secondary structure of a protein is a linked sequence of rigid peptide groups; see Fig. 39 as described in [82]. Thus, as the amino acids are linked together across the peptide bond, and a water molecule is eliminated. As the amino acids are strung together, they are connected together by the peptide bond planes, which in effect are planar Steiner trees themselves. The geometry of the peptide bond planes is very important from the perspective of Steiner-tree modeling of the proteins. A chemical schematic diagram appears in Fig. 40. Figure 41 illustrates the three-dimensional orientations possible with two rigid amide planes and the two degrees of freedom based on the torsion angles Φ and Ψ they maintain, while Fig. 39 illustrates the typical conjoining of the amino acids in a protein with the amide planes and side

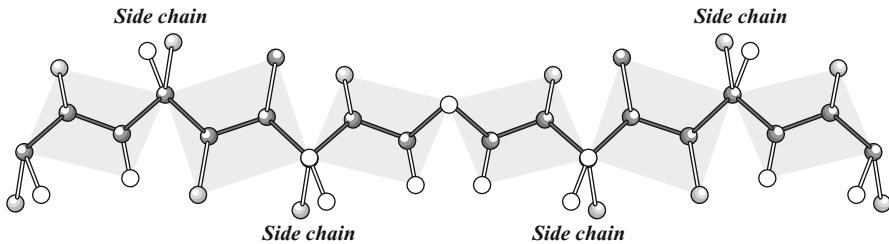


Fig. 39 Network of amide planes in a protein

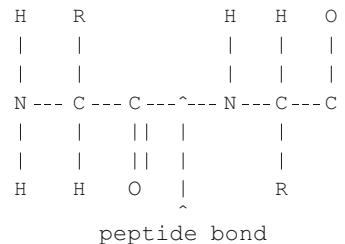


Fig. 40 Chemical diagram of amide planes in a protein

chains. That the rigid planes are strikingly similar to the SMT solution for $N = 4$ in E^2 is quite remarkable. For an instance of $N = 4$, see the right-hand illustration in Fig. 32.

To explore this further, what types of forces are at work in these proteins needs to be demonstrated. In protein modeling, the potential energy objective function often used to measure the MEC is the following, where K_{b_i} , K_{θ_i} , A_{ij} , B_{ij} , and ϵ are adjustable weights; see [22]. This potential energy objective function is based on the theoretical molecular mechanical force field model used to model most molecular structures; see [51]. It is interesting to see that the objective function is a sum of nonlinear terms with little interaction between the terms. The fact that the function is separable is of some debate, as discussed in [28]:

$$\begin{aligned}
E_{tot} = & \sum_i K_{b_i} (b_i - b_0)^2 && \text{bond lengths } [E_{bs}] \\
& + \sum_i K_{\theta_i} (\theta_i - \theta_0)^2 && \text{bond angles } [E_{ab}] \\
& + \sum_i K_{\tau_i} (\cos(3\tau_i - \gamma_0)) && \text{torsion angles } [E_{tor}] \\
& + \sum_i \sum_j A_{ij} d_{ij}^{-6} + B_{ij} d_{ij}^{-12} && \text{van der Waals } [E_{14vdW}] \\
& + \sum_i \sum_j V_i V_j / \epsilon d_{ij} && \text{electrostatic interactions } [E_{14e}]
\end{aligned}$$

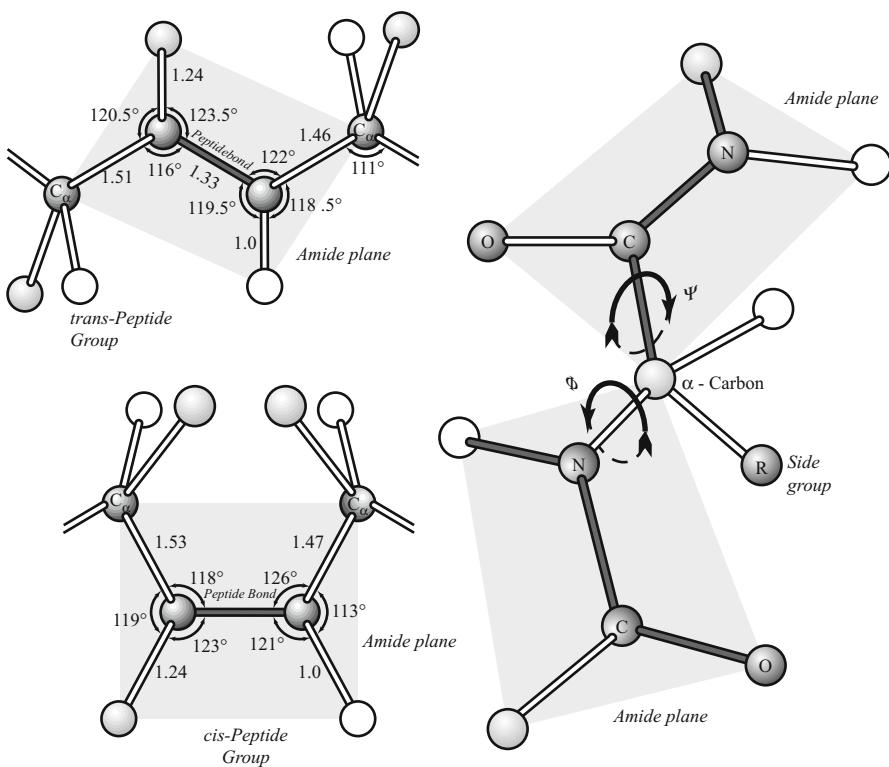


Fig. 41 Peptide or amide plane

where:

E_{bs} : The first term is the sum of energies arising from bond stretching or compression beyond the optimum bond length, and b_i and b_0 are the actual equilibrium bond lengths.

E_{ab} : The second term is the sum of energies for angles that are distorted from their optimum values, and θ_i and θ_0 are the equilibrium bond angles.

E_{tor} : The third term is the sum of the torsional energies that arise from rotations about each respective dihedral angle, and τ_i and γ_0 are the torsion angles.

E_{14vdW} : The sum of energies due to van der Waals interactions.

E_{14e} : The final term is for the electrostatic interactions.

The resulting geometry of the secondary structures can either be an alpha-helix, a beta sheet, random coils, or combinations of these structures. In this next part of the chapter, the results for modeling of collagen, silk, and myosin data sets will be reexamined to see how robust they were to changes in the given atom configurations, much as was done for the individual amino acids. If one changes the atomic structure from the PDB and perturbs the bond lengths, bond angles, and torsion angles, what

effects will occur on the Steiner ratio and on the individual terms in the potential energy equation, i.e., stretching energy, bending, torsion, vdw, and electrostatic?

The protocol for the secondary structure analysis was as follows: All the PDB files were downloaded from the Internet; the file names reflect their PDB code. To isolate one or more specific amino acids, such as Asp25 used in the HIV-1 protease experiments, the PDB file was read into Microsoft Word, and the text was edited. The PDB file could then be imported into [14] or read directly by Sculpt, which were the two programs used to evaluate the individual energies in the proteins. The atomic coordinates of the atoms were systematically perturbed within the proteins by separately changing (a) bond length, (b) bond angles, and (c) dihedral angles. Sculpt was used only to identify the amino acids in the active site fragment that was of interest in using the “tug” function, which allows one to change the location of single atoms; see [78]. Chembuilder calculated the energy values without the hydrogens. Steiner files were created by reading the PDB files into Word and creating a new document with the desired coordinates.

7.1 Collagen

Collagen consists of three polypeptide chains woven together in a triple helix structure, much like the *R*–sausage; see Fig. 42 [57]. Collagen contains 33 % Gly, 33 % proline and hydroxyproline, and 33 % Lys. The repeating unit is $(Gly - X - Y)_n$, where Y is Pro or HyPro. Lys is in position X . Glycine must be every third residue; see [26]. Each polypeptide chain in the collagen structure contains around

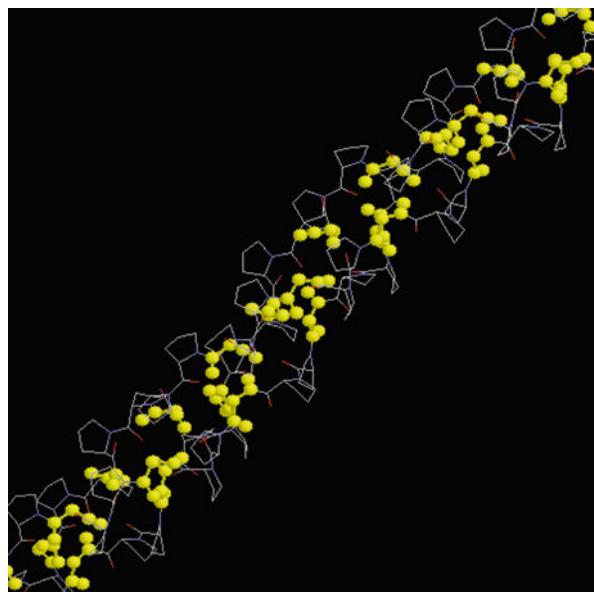


Fig. 42 Collagen structure

Table 7 Typical collagen results

Unperturbed collagen structure							
SMT = 10.1646	Atom	Atomic coordinates			Coordinates Steiner points		
MST = 10.1980	42	N1	1.6670	0.0000	0.0000	1.3770	1.1610
$\rho = 0.9967$	43	C1	1.3770	1.1610	0.8240	2.3010	1.2150
$E_{tot} = 741.78$	44	C2	2.3010	1.2150	2.0420	1.5519	0.0027
stretch = 724.75	45	O1	2.8690	0.1990	2.4400	1.4720	1.2878
bend = 6.39	46	H1	0.9690	-0.7160	0.0170	1.2490	1.0314
torsion = 5.31	47	H2	1.4940	2.0690	0.2330	2.2646	1.2913
vdw = 10.44	48	H3	0.3380	1.1270	1.1530	2.4700	2.3960
electro = 0.00	49	N2	2.4700	2.3960	2.6520		
str-bnd = -4.44	50	C3	3.3400	2.5220	3.8240		
bnd-bnd = -0.42							
tor-str = -0.26							
Perturbed collagen structure							
SMT = 8.9208	Atom	Atomic coordinates			Coordinates Steiner points		
MST = 9.5136	1	N1	0.9267	-0.9357	0.3155	0.2328	-0.5458
$\rho = 0.9377$	2	C1	-0.3067	-0.8414	0.1743	0.0264	0.3432
$E_{tot} = 1,004.61$	3	C2	0.0264	0.3432	0.2037	0.0611	0.8608
stretch = 773.29	4	O1	1.0962	1.1211	0.6883	0.5382	1.2832
bend = 89.39	5	N2	-0.4607	1.0262	-0.6433	-0.0530	-0.8024
torsion = 4.43	6	C3	0.5929	1.7666	-0.2791	-0.3067	-0.8414
vdw = 130.06	7	H1	-0.0141	-1.2813	-0.7905	0.8385	-0.7428
electro = 0.00	8	H2	-1.3018	-1.0551	0.5917		
str-bnd = 6.29	9	H3	1.4041	-0.4793	-0.4879		
bnd-bnd = 2.16							
tor-str = -1.01							

1,000 amino acids. The collagen helix is fully extended, unlike the alpha-helix, so it is much more rigid and is not as flexible as the alpha-helix; see [82].

Collagen is the most abundant protein in vertebrates, and it also occurs in invertebrate species in bone, skin, tendon, cornea, and basement membrane and is a rigid, strong *connective* ligament for transmitting the structural forces in these tissues; see [56] and [26]. That collagen is the *connective* network for transmitting structural forces in human and animal tissues is remarkable when you compare this with the mathematical properties and objective of the geometric Steiner problem and the recent discussion of Maxwell's theorem. Collagen is a natural implementation of the Steiner network problem.

Many, many perturbation experiments were carried out. A sample of the experiments appears in Table 7. In these experiments, since $N \leq 13$, the algorithm was able to converge to the optimal solution in a reasonable amount of running time. What can be seen in the comparison of results in Table 7 for the data set from [15] is that the Steiner ρ for the perturbed collagen structure decreased in value and the energy of the perturbed structure increased in value: from a $\rho = 0.996725$

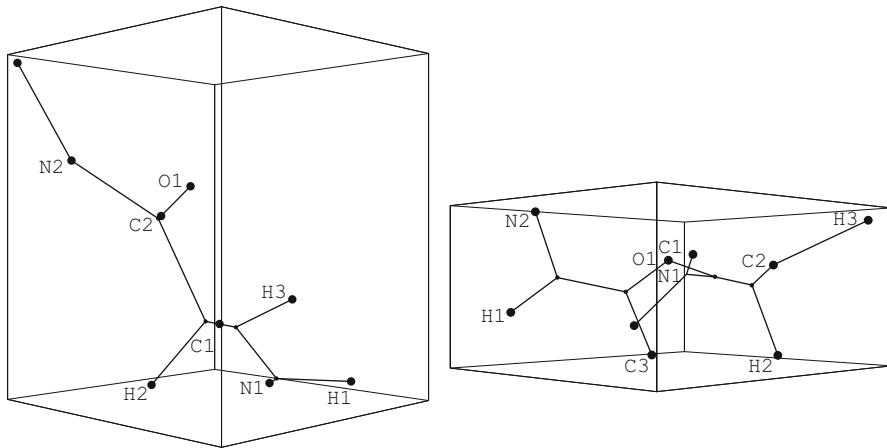


Fig. 43 Native collagen (*left*) perturbation (*right*) SMT

with a total MEC energy = 741.7761 kcal to a $\rho = 0.937688$ and a total MEC energy = 1,004.6121 kcal. Stretching, bending, and van der Waals energy levels increased in the perturbed residue. The experiment is based on systematically perturbing the atomic coordinates of the fragments in 3-D, as described in the protocol above. Since the terminal vertex set is small, the results are optimal from the Steiner algorithm. Figure 43 and Table 7 illustrate the before and after of the perturbation of the collagen fragment. Figure 43 illustrates the two data sets of Table 7. One can see that in the perturbed structure, different Steiner points are necessary, and the geometry of the Steiner tree is vastly different than the native structure.

This is what should be expected, i.e., that the minimum energy levels increase in the perturbed structure under stretching and rotation transformations. In Table 7, the Steiner points corresponding to the terminal vertices are marked with boldface coordinates. One may wonder why more Steiner points did not show up as nitrogen and carbon atoms, but this is probably due to the fact that the atoms were randomly selected within the collagen polypeptide chain, and the carbon and nitrogen atoms sometimes are acting as leaf vertices V in the chain rather than acting as true Steiner points.

7.2 Silk

This second structural protein is one of the most thoroughly studied proteins, called Fibroin, and is the material of silk; see [26] and [82]. The silk protein is a structural protein used by insects and spiders to create cocoons, webs, nests, etc. The peptide chains are arranged in antiparallel β -pleated structures in which the chains extend parallel to the fiber axis; see [82].

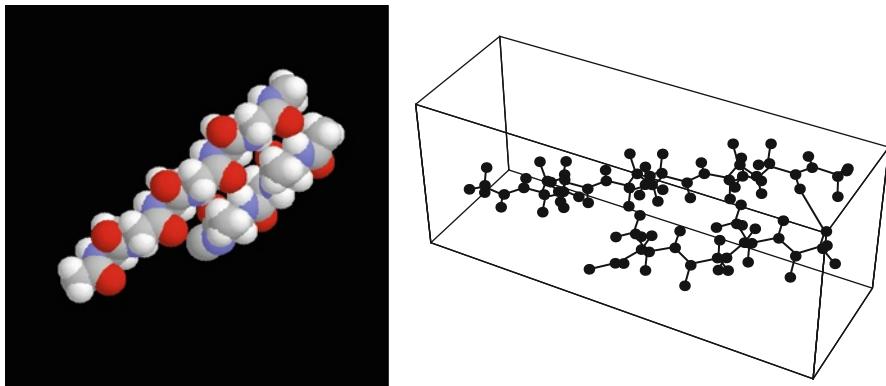


Fig. 44 Silk protein space-fill (*left*) and Steiner tree $n = 99$ (*right*)

Fibroin contains large amounts of glycine, alanine, tyrosine, and serine, as well as small amounts of other amino acids. The repeating elements of fibroin are $(Gly - Ser - Gly - Ala)_n$. Gly extends on one side of the sheet, Ala and Ser on the other, much as depicted in Fig. 44.

Again, as in the collagen results based on the data from [37], the perturbation of the bond lengths, bond angles, and torsion angles of the silk atoms lowers the ρ so improvements in the length of the Steiner tree can be made, yet these improvements in length cause an increase in the potential energy equation. The results contained in Table 8 are optimal.

Thus, once again, if a protein in the PDB is a MEC, the Steiner ratio should be close to ≈ 1.00 , and the carbon and nitrogen atoms should all be acting as Steiner-points. In this sense, Steiner-tree properties act as a very good evaluation tool for protein conformation.

To be more specific:

- ρ appears to vary inversely with bending energy (bond angles).
- ρ appears to be independent or insignificantly affected by stretching energy (bond length).

7.3 Myosin

Myosin occurs in muscle and, together with the protein actin, provides the catalytic and structural properties and mechanism to accommodate and transfer the structural forces within muscle; see Fig. 45. To produce force, cross-bridges from the myosin filaments associate with the actin [49] filament then rotate slightly to pull the filaments across each other (much like the oars of a rowboat pull across the water); see [1]. During the myosin–actin interaction, the overall effect is the contraction of the muscle.

Table 8 Typical silk results

silk_g9	silk_g9-mod (bond length only)
Length of SMT = 11.256445	Length of SMT = 13.168045
Length of MST = 11.280861	Length of MST = 13.192828
Rho = 0.997836	Rho = 0.998122
total energy = 149.2164	total energy = 333.7486
stretch = 142.43	stretch = 317.11
bend = 7.89	bend = 7.87
torsion = 3.51	torsion = 3.51
vdw = 1.82	vdw = -0.15
electro = 0.00	electro = 0.00
str-bnd = -5.90	str-bnd = 5.98
bnd-bnd = -0.39	bnd-bnd = -0.40
tor-str = -0.16	tor-str = -0.16
silk_g9-mod2 (bond angles only)	silk_g9-mod3 (torsional angles only)
Length of SMT = 10.933290	Length of SMT = 11.243819
Length of MST = 11.280684	Length of MST = 11.279127
Rho = 0.969205	Rho = 0.996870
total energy = 219.9434	total energy = 191.3742
stretch = 147.19	stretch = 142.49
bend = 60.06	bend = 7.89
torsion = 3.61	torsion = 3.48
vdw = 8.55	vdw = 43.97
electro = 0.00	electro = 0.00
str-bnd = -0.99	str-bnd = -5.90
bnd-bnd = 1.69	bnd-bnd = -0.39
tor-str = -0.16	tor-str = -0.18

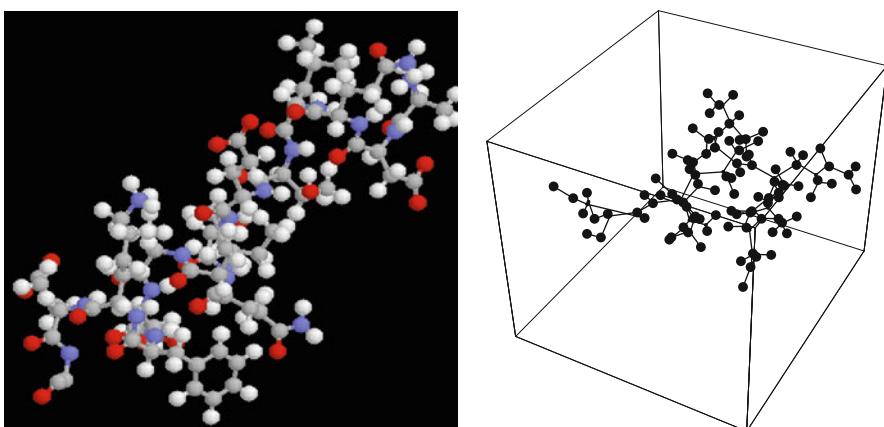
**Fig. 45** Myosin protein ball and stick (*left*) and Steiner tree $n = 99$ (*right*)

Table 9 Typical myosin results

myosin - regulatory domain			
1scm82-3		1scm82-3-mod (bond length only)	
Length of SMT = 14.035851		Length of SMT = 19.089484	
Length of MST = 14.156624		Length of MST = 19.222764	
Rho = 0.991469		Rho = 0.993067	
total energy = 164.9216		total energy = 5311.9234	
stretch= 144.47	bend= 14.89	stretch= 5275.35	bend= 14.81
torsion= 4.92	vdw= 7.45	torsion= 4.97	vdw= 3.02
electro= 0.00	str-bnd=-6.34	electro= 0.00	str-bnd= 14.24
bnd-bnd= -0.20	tor-str=-0.27	bnd-bnd= -0.20	tor-str= -0.26
1scm82-3-mod2 (bond angles only)		1scm82-3-mod3 (dihedral angles only)	
Length of SMT = 13.397818		Length of SMT = 14.020379	
Length of MST = 14.202967		Length of MST = 14.156530	
Rho = 0.943311		Rho = 0.990382	
total energy = 324.3677		total energy = 196.1716	
stretch= 137.35	bend= 144.10	stretch= 144.40	bend= 14.89
torsion= 4.94	vdw= 40.05	torsion= 5.51	vdw= 38.12
electro= 0.00	str-bnd= -4.97	electro= 0.00	str-bnd= -6.34
bnd-bnd= 3.13	tor-str= -0.23	bnd-bnd= -0.20	tor-str= -0.22

Since myosin is basically a catalytic protein with a globular and alpha-helix structure, it is an interesting protein to examine from a Steiner-tree point of view. The two previous proteins, collagen and silk, are considered structural proteins, whereas because myosin is catalytic, one would not necessarily think that the Steiner-tree properties found earlier with the structural proteins in collagen and silk would carry over, but they do.

As one can see again from the experimental results in Table 9, perturbations in the myosin atoms decrease the Steiner ρ value and increase the potential energy, except when the stretching is significant, as is the case here. Notice that the stretching increased the energy significantly and also the ρ value. In the limit, if all the myosin atoms were in a straight-line, then $\rho \rightarrow 1$. Also notice that the overall length increased for the SMT beyond the SMT of the native structure when the bonds were stretched. It appears that changing the bond stretching, bond angles, and torsion angles all individually and simultaneously affect the Steiner ratio and the potential energy equation.

So at this point in our journey, the following issues become important where the quantitative relationship between the Steiner ratio and the various energies of the amino acids in more detail. To be more specific:

- ρ appears to vary inversely with bending energy (bond angles).
- ρ appears to be independent or insignificantly affected by stretching energy (bond length).
- ρ appears to vary inversely with van der Waals energy.
- ρ may vary inversely with torsional energy.

7.4 HIV-1 Protease

Finally, the last secondary structure experimental evaluation involving HIV-1 protease is presented. HIV-1 protease is a catalytic protein (enzyme) that functions to cleave other proteins that are necessary to the life cycle of the HIV-1 virus. This cleavage serves to render these proteins functional. Therefore, HIV protease is absolutely essential to the survival and proliferation of the virus. For this reason, HIV protease has recently come under close scrutiny. In fact, one of the most popular drug therapies in the United States is a combination of protease inhibitors and another important enzyme, reverse transcriptase, inhibitors. Inhibitors are simply any chemical species or molecule that binds to the protein, either directly in the catalytic site or another site, and disrupts the proper function of the protein. Upon binding of any inhibitor (or the molecules that are involved in the reaction catalyzed by the enzyme), the conformation of the enzyme changes due to the forces involved in binding. A major obstacle to inhibitor therapy is mutation. Once an individual had been treated with a protease inhibitor, the viruses that have “normal” (wild-type) proteases are killed since the protease is no longer able to cleave the other essential viral proteins. However, there exists in the virus population mutants that have proteases that deviate by one to several amino acids. Despite the difference, the mutant enzyme may still be functional. In this case, the individual treated with inhibitors may be killing off viruses with the wild-type enzyme, but not viruses with functional mutant enzymes. Therefore, the virus is able to proliferate and continue causing disease. This phenomenon has led to increased attempts to understand and model the different enzyme/inhibitor complexes.

The HIV-1 protease enzyme is a dimeric protein – made up of two identical polypeptide chains referred to as subunits. Each subunit is 99 amino acid residues long. The active site, or the residues that are involved in protein cleavage, is found at the center of protein, symmetric between the two chains. At the bottom of the active site cleft lies two catalytically active Asp residues, Asp25 and Asp125. In these studies, how the binding of different inhibitors affects the conformation of Asp25 and 125 was to be examined, as well as a larger segment of the active site. It was expected that the conformational change induced upon binding of the inhibitors would cause a difference in the ρ values ([Table 10](#)).

Thus, as can be seen in the tables of results:

- *For the Asp25, when bound to different inhibitors, ρ did not significantly vary, $\approx 0.3\%$.*
- *For the active site fragments, again as in the Asp25 experiments, ρ did not significantly vary $\approx 0.3\%$.*

Also the chain binding symmetry is examined. Recall that the active site is symmetric between the two subunits. However, the inhibitors looked at are asymmetric and therefore bind to the two subunits in an asymmetric fashion. It was expected that the asymmetrical binding would cause a difference in ρ between the two subunits. What was found was:

- *For the chain and binding symmetry, ρ varied only between $\approx 0.001\%$ and $\approx 0.2\%$*

Table 10 Typical HIV-1 protease results

3PHV ASP25		Atom	Coordinates	Steiner points	
With no bound inhibitor		ATOM 1	N -1.4453	-1.1451	-0.4288 0.7857 -0.3196 ***
Length of SMT = 9.694515		ATOM 2	C -0.4288	0.7857	-0.3196 1.3927 -0.2877
Length of MST = 9.723531		ATOM 3	C 0.9124	1.4193	-0.2572 -0.2392 0.6847 -0.3329
Rho = 0.997016		ATOM 4	O 1.7517	1.2422	-1.1559 -0.1990 -0.6131 -0.4574
total energy = 165.7153		ATOM 5	C -0.3118	-0.7183	-0.6369 0.4841 -1.3541 0.4354
stretch= 159.79	bend= 7.04	ATOM 6	C 0.4742	-1.4618	0.4351 0.4742 -1.4618 0.4351 ***
torsion= 1.80	vdw= 2.97	ATOM 7	O 1.1872	-0.8308	1.2178
electro= 0.00	str-bnd=-5.42	ATOM 8	O 0.3979	-2.7064	0.5425
bnd-bnd= -0.45	tor-str= -0.01				
With bound inhibitor SDZ2283-910					
Length of SMT = 9.624662		ATOM 1	N -0.2092	-2.5294	-1.8228 -0.8588 -1.2708 -1.7076 ***
Length of MST = 9.647486		ATOM 2	C -0.8588	-1.2708	-1.7076 -2.1105 -1.1909 -0.8550 ***
Rho = 0.997634		ATOM 3	C -2.1105	-1.1909	-0.8550 -0.8101 -1.3734 -1.8843
total energy = 202.8486		ATOM 4	O -3.2259	-1.4534	-1.3067 -1.1326 -0.8122 -3.1369
stretch= 197.25	bend= 7.81	ATOM 5	C -1.1326	-0.8122	-3.1369 -1.4454 0.6714 -3.3016 ***
torsion= 2.01	vdw= 1.82	ATOM 6	C -1.4454	0.6714	-3.3016 -1.4401 0.6593 -3.3187
electro= 0.00	str-bnd=-5.65	ATOM 7	O -1.7525	1.3721	-2.3535
bnd-bnd= -0.38	tor-str= -0.00	ATOM 8	O -1.3909	1.1512	-4.4099

IHVR Asp25

With bound nonpeptide cyclic urea inhibitor XK263	
Length of SMT = 9.654778	ATOM 1 N -2.3499 0.4610 -0.8474 -1.2631 0.3847 0.1335 ***
Length of MST = 9.680506	ATOM 2 C -1.2631 0.3847 0.1335 -0.1747 1.4590 -0.0521 ***
Rho = 0.997342	ATOM 3 C -0.1747 1.4590 -0.0521 -1.3303 0.2262 -0.0062 ***
total energy = 204.1035	ATOM 4 O 0.7466 1.3481 -0.8589 -0.6887 -1.0240 -0.0115 ***
stretch= 199.05	ATOM 5 C -0.6887 -1.0240 -0.0115 0.2039 -1.4282 1.1166 ***
torsion= 1.99	ATOM 6 C 0.2039 -1.4282 1.1166 0.1835 -1.4567 1.0937 ***
electro= 0.00	ATOM 7 O 0.7986 -0.6007 1.7930
bnd-bnd=-0.38	ATOM 8 O 0.3064 -2.6030 1.3324

IAID Asp25

With bound nonpeptidic inhibitor THK

Length of SMT = 9.592264	ATOM 1 N 2.2188 0.0988 -1.2941 1.3826 0.5137 -0.2115 ***
Length of MST = 9.620918	ATOM 2 C 1.3826 0.5137 -0.2115 1.5949 -0.2172 1.0806
Rho = 0.997022	ATOM 3 C 1.5970 -0.2160 1.0829 1.3187 0.4406 -0.4190
total energy = 193.5667	ATOM 4 O 1.0040 -1.2648 1.2941 -0.0180 0.3633 -0.7401 ***
stretch= 187.97	ATOM 5 C -0.0180 0.3633 -0.7401 -1.1380 0.9339 0.0994 ***
torsion= 1.95	ATOM 6 C -1.1380 0.9339 0.0994 -1.1457 0.9190 0.0471 ***
electro= 0.00	ATOM 7 O -0.9750 1.2648 1.2745
bnd-bnd=-0.37	tor-str=-0.02 ATOM 8 O -2.2188 1.0412 -0.4545

*** The Steiner points correspond to an existing atom in the protein structure

The final analysis was a look at mutant enzymes. A mutant enzyme implies that the amino acid sequence of the protein has changed in some way – therefore changing the conformation somewhat. The aim of all of these studies was to determine whether or not these changes in conformation would bring about a change in the value of ρ . It would be of interest to determine the binding energies of the various inhibitors and see if there is a relationship with the ρ value:

- *For the mutants, ρ varied between $\approx 0.01\%$ and $\approx 0.1\%$ of the wild type.*

This would indicate a very slight difference between the conformational changes in binding of the various inhibitors. Thus, in all the HIV protease studies, the Steiner ratio varied little if not at all. This seems to indicate that all the HIV protease/inhibitor complexes were very stable MEC structures.

In the next section, the relationship between the Steiner ratio and the torsion energy in proteins is examined in some detail.

7.5 Steiner Ratio Versus Torsion Energy

What needs to be done in this section is to take a subset of the amino acids and perturb the atomic coordinates and examine how the energy in the acid changes with variations in the torsion angles of the acids. At the same time, the Steiner ratio will be evaluated for the changing torsion angle conformations. If it is true that variations in the torsion angle of the amino acids correlate with an increase/decrease in the Steiner ratio, then the properties of Steiner trees should be a useful approximation to the potential energy function.

In order for this to occur, one can argue that the native state of an amino acid correlates strongly with the minimum energy conformation of the acid. Changes in the torsion energy can then be also measured by the Steiner ratio.

7.5.1 Alanine Torsion Angle Variation

[Figure 46](#) illustrates the variation in torsion energy of alanine with respect to its side-chain torsion angle χ_1 . The energy measures are captured by the software program Chembuilder [14]. Variations in the potential energy curve are plotted with variations in the torsion angle χ_1 . [Figure 46](#) also illustrates the corresponding changes in the Steiner ratio as a function of variations in the torsion angle. From these two plots, it is observed that the Steiner ratio is inversely proportional to the torsion angle.

Why should this be the case? As in Maxwell's theorem, if a structure is a MEC, then perturbations in the Steiner structure will increase the Steiner ratio away from the MEC. The native alanine structure with a staggered conformation (set $\chi_1 = 0^\circ$) achieves the lowest energy value or consequently the highest Steiner ratio, whereas the highest energy value occurs when the conformation is not staggered ($\chi_1 = 60^\circ$). [Figure 47](#) shows the optimal Steiner tree results from the native alanine structure and the highest energy conformation, respectively. As before in the previous section, the planes of the atoms in the twist angles of alanine were utilized.

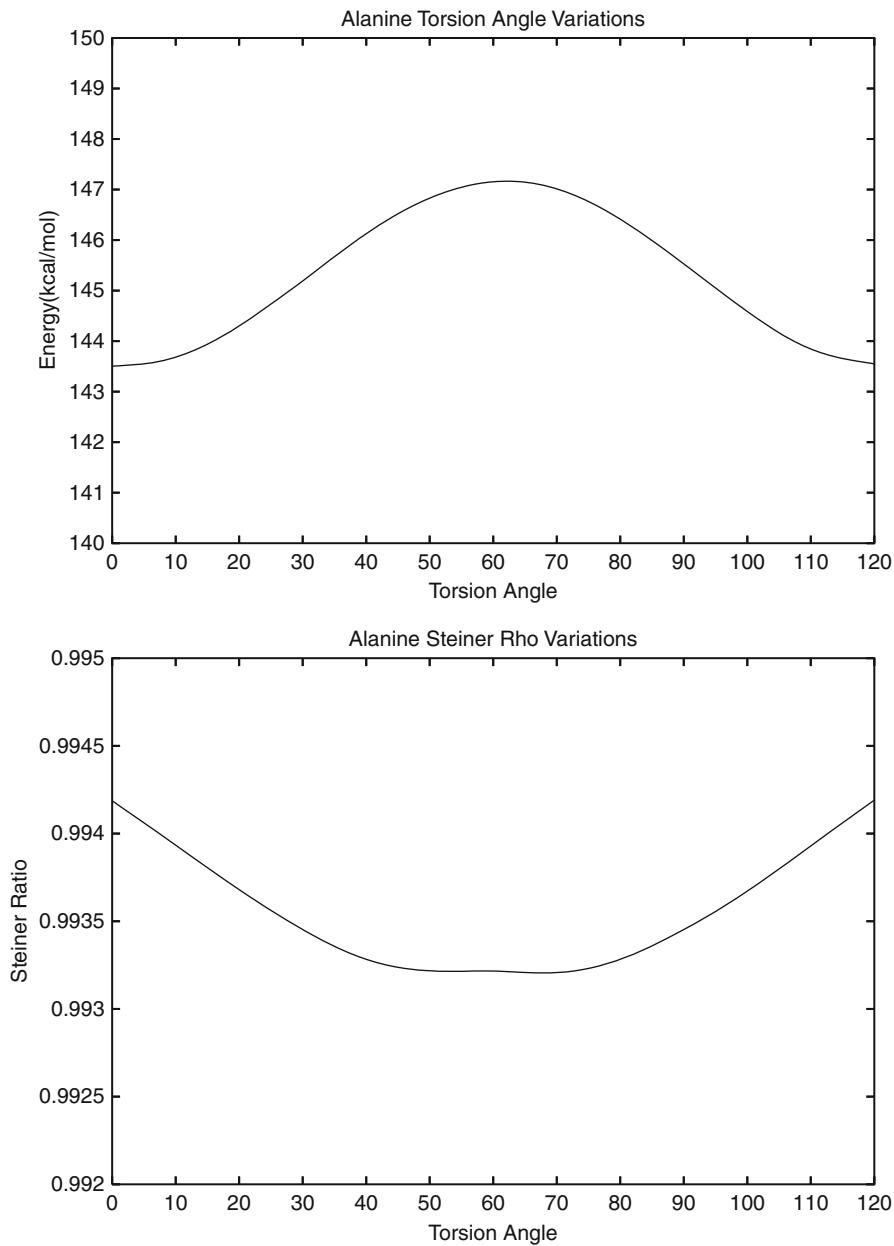


Fig. 46 Alanine torsion energy and Steiner ρ variations

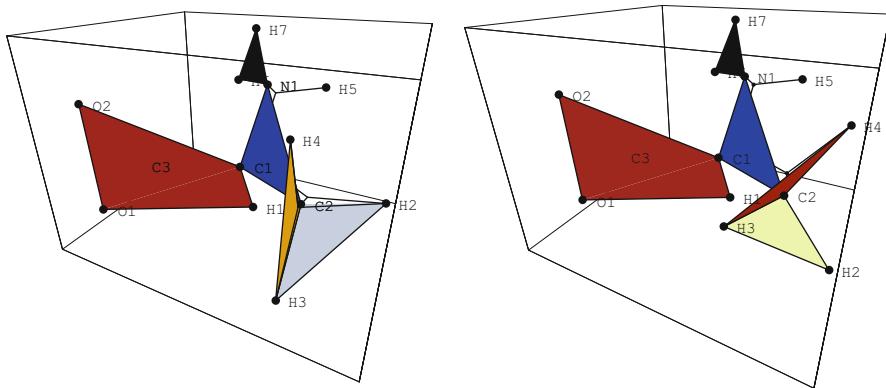


Fig. 47 Ala native conformation ($\chi_1 = 0^\circ$) (left) and MEC (right)

The most obvious difference of the non-perturbed conformation is the twisting of the planes involving the ${}_1H_\beta$, ${}_2H_\beta$, and ${}_3H_\beta$ atoms which results in actually a shorter Steiner tree ($SMT^{0^\circ} = 14.3942$ vs. $SMT^{60^\circ} = 14.3803$) and smaller Steiner ratio ($\rho^{0^\circ} = 0.9942$ vs. $\rho^{60^\circ} = 0.9932$). [Equation 19](#) displays the twist angle matrix for the perturbed Ala when $\chi_1 = 60^\circ$:

$$\begin{aligned}\mathcal{P}_1 &:= \{C_\alpha, N_1, C_2\} & \mathcal{P}_2 &:= \{C_\alpha, H_1, O_2\} \\ \mathcal{P}_3 &:= \{C_2, H_3, H_4\} & \mathcal{P}_4 &:= \{N_1, H_6, H_7\} \\ \mathcal{P}_5 &:= \{C_2, H_2, H_3\}\end{aligned}$$

$$\begin{array}{c|ccccc} \mathcal{P} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0.0 & 90.0 & 60.0 & 60.0 & 60.0 \\ 2 & & 0.0 & \mathbf{33.6} & 60.0 & \mathbf{33.6} \\ 3 & & & 0.0 & 90.0 & \mathbf{2.4} \\ 4 & & & & 0.0 & 90.0 \\ 5 & & & & & 0.0 \end{array} \quad (19)$$

One can see that there are some new plane twist angles of about 33.6° . They were 60° before in the native alanine structure. The value of 2.4° computed by the software program in the right-hand matrix is expected to be approximately zero.

7.5.2 Cysteine Torsion Angle Variation

As with the alanine variations, variations are performed of the torsion angle with cysteine. [Figure 48](#) illustrates the energy variations with respect to χ_1 , while the Steiner ratio still varies inversely proportional to the torsion energy. The plane twist angles for the minimum and maximum energy conformations are also examined in [Eq. 20](#), respectively. The irregular twist angles of $\{36.7^\circ, 54.8^\circ\}$ appear in the maximum energy or minimum Steiner ratio conformation. The tetrahedra and planes defined for Cys after solving for the Steiner minimal tree are as follows:

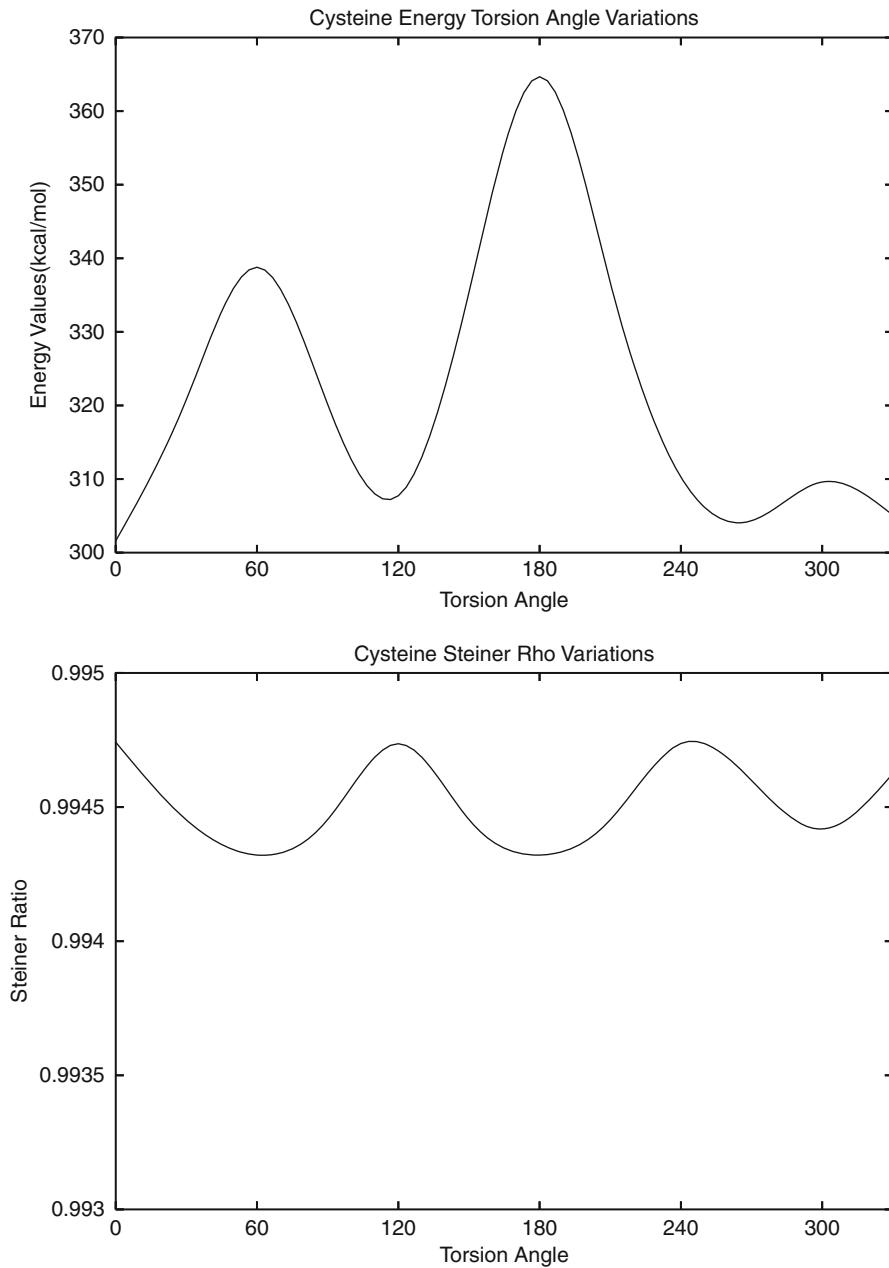


Fig. 48 Cysteine torsion energy and Steiner ρ variation

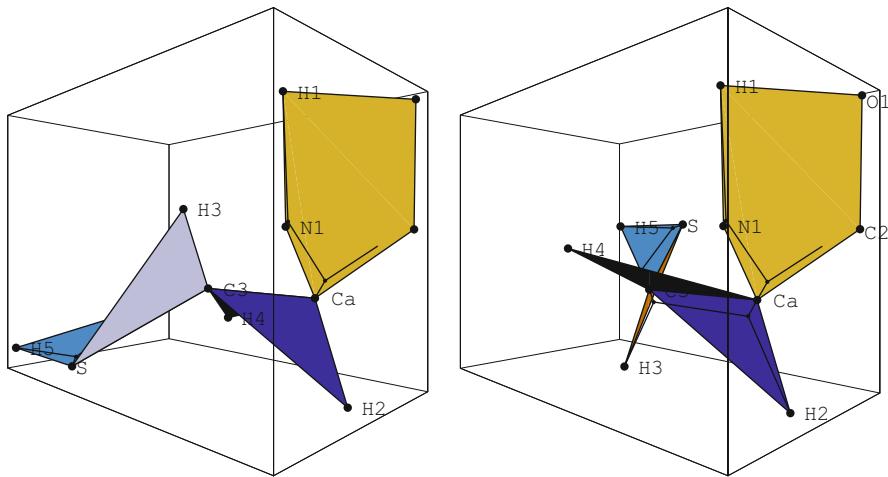


Fig. 49 Cysteine MEC *left* ($\chi_1 = 0^\circ$) and *right* ($\chi_1 = 180^\circ$)

$$\mathcal{T}_1 := \{O_1, O_2, C_\alpha, H_\alpha\}; \mathcal{P}_1 := \{H_\alpha, O_2, C_\alpha\};$$

$$\mathcal{P}_2 := \{C_\alpha, C_\beta, N\}; \mathcal{P}_3 := \{S_\gamma, C_{\beta,1}, H_\beta\};$$

$$\mathcal{P}_4 := \{N, H_1, H_2\}; \mathcal{P}_5 := \{N, H_2, H_3\};$$

\mathcal{P}	1	2	3	4	5	\mathcal{P}	1	2	3	4	5
1	0.0	90.0	60.0	60.0	60.0	1	0.0	90.0	36.7	60.0	60.0
2		0.0	60.0	60.0	60.0	2		0.0	54.8	60.0	60.0
3			0.0	60.0	90.0	3			0.0	54.8	90.0
4				0.0	60.0	4				0.0	60.0
5					0.0	5					0.0

(20)

From the experimental comparison of the two selected amino acids, it is postulated that the torsion energy varies inversely with the Steiner ratio. It is expected that if all the 20 amino acids were examined, a similar result would follow. Thus, if a native amino acid achieves a minimum energy conformation, variations in the torsion angle should be captured by the Steiner ratio denoted through its ρ value (Fig. 49).

7.6 Theoretical Versus Empirical ρ Values

Table 11 shows the ideal ρ_t values for all the 20 amino acids. This represents the best ρ values found so far with the computing capability presently available. While

Table 11 Ideal and empirical amino acid results

Name	SMT	MST	ρ_t	ρ_e^c	σ^d	V	$S' \subseteq S$	$C + N$
(Ala)	14.394	14.478	0.9942^a	0.99748	0.00099	13	4	4
(Arg)	31.162	31.357	0.9938	0.99751	0.00061	27	10	10
(Asn)	19.382	19.481	0.9949	0.99735	0.00070	17	6	6
(Asp)	17.292	17.391	0.9943^a	0.99734	0.00064	15	5	5
(Cys)	16.395	16.541	0.9912^a	0.99671	0.00108	14	4	4
(Gln)	23.071	23.186	0.9950	0.99754	0.00058	20	7	7
(Glu)	20.991	21.096	0.9950	0.99759	0.00056	18	6	6
(Gly)	10.720	10.775	0.9949^a	0.99823	0.00157	10	3	3
(His)	23.291	23.518	0.9904^b	0.98711	0.00046	20	8	9
(Ile)	25.417	25.586	0.9934	0.99603	0.00069	22	7	7
(Leu)	25.447	25.588	0.9945	0.99665	0.00072	22	7	7
(Lys)	28.912	29.102	0.9935	0.99665	0.00072	25	8	8
(Met)	23.801	23.990	0.9918	0.99245	0.00144	20	6	6
(Phe)	27.222	27.300	0.9972^b	0.99841	0.00037	23	9	10
(Pro)	19.491	19.756	0.9866	0.97649	0.00209	17	6	6
(Ser)	15.696	15.792	0.9940^a	0.99573	0.00100	14	4	4
(Thr)	19.359	19.494	0.9931	0.99556	0.00085	17	5	5
(Trp)	32.253	32.478	0.9931^b	0.99190	0.00042	27	12	13
(Tyr)	28.473	28.562	0.9969	0.99853	0.00042	24	10	10
(Val)	21.752	21.885	0.9939	0.99592	0.00076	19	6	6

^aThose acids marked with *a* are optimal structures^bThose acids where some of the C, N atoms are leaf vertices^cEmpirical ρ_e value is computed optimal for all acids except Trp^dEstimated standard deviation

not all the ρ_t values are optimal, they represent a best upper bound on the Steiner ratio, so they should still be useful benchmarks.

Table 11 also includes the empirical ρ_e values statistically obtained from thousands of randomly selected amino acids from the PDB. The standard deviation σ of the empirical data sets is also included in the table, and one recognizes that it is indeed very small. The mean and standard deviation will be useful in comparing whether the Steiner ρ_e value for certain PDB files is unusual or not. All 20 amino acids have been analyzed, and a sample result with basic statistics and a histogram of the sample acid is presented in [Fig. 50](#). These sample results for all the amino acids are also available from the authors.

In reading the last two columns of **Table 11**, the $S' \subseteq S$ column represents the subset of Steiner points that correspond to the given carbon and nitrogen atoms in the amino acid, and the last column represents the combined number of carbon and nitrogen atoms within the amino acid.

With certain exceptions, the empirical ρ_e values are larger than the theoretical values since the theoretical ρ_t values include the hydrogen atoms. The exceptions are interesting because they largely have some of the carbon and nitrogen atoms as

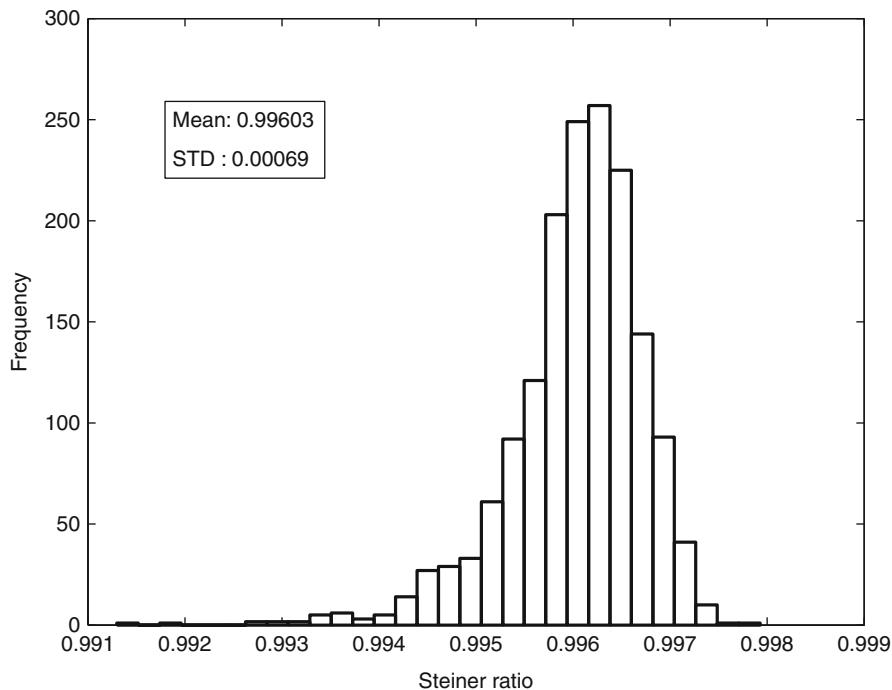


Fig. 50 Histogram of Ile empirical ρ values. Coordinates of 1,628 Ile residues are randomly sampled from the PDB, and those Steiner rho values are calculated individually. The average and standard deviation of Ile empirical rho values are also provided in the *inset box*

leaf vertices in the Steiner tree. The hydrogen atoms add more structure and can greatly affect the Steiner ratio. The empirical ρ_e value along with the theoretical measure ρ_t will be used as scoring markers in the next section to compare native and decoy protein structures.

7.7 Protein Fold Recognition Using Steiner Ratio

In order to test the Steiner ratio as a surrogate for the potential energy function, folded and misfolded protein data sets were examined. They were obtained from the Web site <http://dd.stanford.edu>. There are a total of 26 different data sets with folded and misfolded conformations. The given native folded structures were radically misfolded via a computer program where the main chains were swapped between folds and the side chains were annealed using a Monte Carlo process [46].

7.7.1 Native Versus Decoy Experiments

What is important here is to realize that in the misfolded proteins, all the detailed energy values in the potential energy equation such as the bond lengths, bond angles,

Table 12 Comparison of native and misfolded ρ values

1bp2				1bp2on2paz			
Acid	Single ρ	Pair ρ	Triple ρ	Acid	Single ρ	Pair ρ	Triple ρ
Ala	0.996846			Ala	0.997354		
Leu	0.994013	0.995389		Leu	0.996975	0.997179	
Trp	0.992520		0.994542	Trp	0.992484		0.996013
$\sum E$			323.93 KJ/mol	$\sum E$			234.22 KJ/mol
1cbh				1cbhon1ppt			
Acid	Single ρ	Pair ρ	Triple ρ	Acid	Single ρ	Pair ρ	Triple ρ
Thr	0.992456			Thr	0.996503		
Gln	0.995699	0.995652		Gln	0.997929	0.997174	
Ser	0.989046		0.995508	Ser	0.995815		0.996918
$\sum E$			199.09 KJ/mol	$\sum E$			104.37 KJ/mol
2paz				2pazon1bp2			
Acid	Single ρ	Pair ρ	Triple ρ	Acid	Single ρ	Pair ρ	Triple ρ
Glu	0.993448			Glu	0.998131		
Asn	0.998048	0.995849		Asn	0.997409	0.997937	
Ile	0.995611		0.996106	Ile	0.997272		0.997784
$\sum E$			234.66 KJ/mol	$\sum E$			102.13 KJ/mol

and torsion angles are being perturbed so that the Steiner ratio varies with all these quantities. While only a systematic analysis of the inverse relationship of the Steiner ratio with the side-chain torsion angle was carried out, it appears that, in general, the Steiner ratio varies inversely with the potential energy of the entire system.

Can the Steiner ratio ρ be used as a predictive measure? As a first detailed step, let us select three of the decoy data sets and compare the first three residues of the native with the decoy data sets. As one can see in [Table 12](#), when the residues 1–3 are examined for these three native proteins and their misfolded decoys, the ρ values for the decoys are mainly much higher than they should be for the native proteins. The energy computations are carried out for each structure using Swiss-Pdb Viewer, in which computations have been done in vacuo with the GROMOS96 43B1 parameter set without reaction field [44]. In all cases, each misfolded protein segment (dipeptide or tripeptide in this context) has a higher Steiner ratio but lower energy value than the corresponding folded one. This inverse relationship is as expected with the Steiner ratio and the energy within the protein as was demonstrated earlier in this chapter. A detailed printout of the experimental results for the first comparison of “1bp2” and “1bp2on2paz” is provided in [Table 12](#).

The energy comparison in [Table 12](#) probably misleads one to consider the misfolded decoy as the best candidate for the native one in the protein-folding prediction based on the energy minimization. This is counterintuitive to the belief that the native-folded protein abides by the minimum energy conformation, implying that the potential energy functions, most of which have been defined in empirical manner, are not perfect enough to be used as a predictive tool for the

Table 13 Comparison of detailed energies in folded and misfolded Protein

Native protein 1bp2							
Residue	Bonds	Angles	Torsion	Improper	Nonbonded	Electro	Total
HHT	0.000	6.183	7.540	0.000	0.00	6.52	E = 20.239
ALA	0.107	0.944	1.936	0.359	-2.67	132.56	E = 133.227
LEU	1.940	6.445	2.843	3.837	0.90	22.42	E = 38.390
TRP	37.273	26.807	3.906	9.613	-12.21	61.31	E = 126.706
OXT	0.000	0.000	0.000	0.000	-4.55	9.91	E = 5.367
KJ/mol	39.319	40.379	16.225	13.809	-18.53	232.72	E = 323.929
Misfolded protein 1bp2on2paz							
HHT	0.000	6.183	7.540	0.000	0.00	7.50	E = 21.223
ALA	0.204	2.545	0.218	0.158	-7.97	120.82	E = 115.980
LEU	0.467	1.914	3.621	0.655	-13.29	15.94	E = 9.303
TRP	1.894	26.546	3.260	6.073	-19.10	69.41	E = 88.087
OXT	0.000	0.000	0.000	0.000	-1.97	1.59	E = -0.373
KJ/mol	2.565	37.188	14.639	6.886	-42.32	215.26	E = 234.221

protein-folding problem. Alternatively, the Steiner ratio in this experiment shows the consistent result such that the ρ value for a native protein segment is much closer to the ideal value than that of the corresponding misfolded one. Here the ideal Steiner ratio is computed by averaging the ρ values of amino acids involved.

What is important here is to realize that the misfolded proteins vary the bond lengths, bond angles, torsion angles, van der Waals energies, and all the factors in the potential energy equation, so that the Steiner ratio varies with all these quantities. While only a systematic analysis of the inverse relationship of the Steiner ratio with the torsion angles has been done, it appears that, in general, the Steiner ratio varies inversely with the potential energy of the entire system. A detailed printout of the experimental results for the first comparison of *Ibp2* and *Ibp2on2paz* is provided in [Table 13](#).

At this point, all the amino acids of all the 26 decoy data sets and their native counterparts are compared. The Steiner ratios of each residue in both native and decoy structures are calculated. These observed rho values are compared with empirical and theoretical ρ values, respectively. [Table 14](#) presents the resulting root-mean-square-difference (RMSD) values of all 26 data sets. Decoy structures have significantly lower RMSD values with respect to empirical ρ values, while the RMSD values of native structures are slightly lower than those of decoy structures with respect to theoretical ρ values. This indicates that the native Steiner ratio is much closer to the theoretical (ideal) value than the decoy Steiner ratio is. [Figure 51](#) illustrates one sample experiment, namely, “1bp2” and its decoy structure “1bp2on2paz” isolated from the entire set of the native and decoy Steiner values for all 123 residues of this sample data set.

Table 14 RMSD of decoy sets with respect to empirical and theoretical Steiner ratios

Protein ^a	ρ_e vs. ρ_{native}	ρ_e vs. ρ_{decoy}	ρ_t vs. ρ_{native}	ρ_t vs. ρ_{decoy}
A	0.00226758084446	0.00080169662064	0.00307542421557	0.00380339175653
B	0.00252209632823	0.00086561543617	0.00294431229473	0.00389293920308
C	0.00899720953243	0.00084395144235	0.00806618892754	0.00446086803959
D	0.00202274768861	0.00071587065049	0.00309418402814	0.00387002790232
E	0.00234134600972	0.00069675754638	0.00242096050457	0.00327903313005
F	0.00273481490351	0.00076084043090	0.00379442693727	0.00389154181237
G	0.00266088230459	0.00139383439954	0.00298973701463	0.00429915675194
H	0.00307889201769	0.00095057552825	0.00296593827147	0.00369207755405
I	0.00218672115808	0.00085166858710	0.00292171495819	0.00364471880262
J	0.00206946838467	0.00082012901227	0.00306574382518	0.00346608680298
K	0.00247249335069	0.00077465697620	0.00330349898185	0.00401356341384
L	0.00247249335069	0.00078291542917	0.00330349898185	0.00424805526003
M	0.00095706956536	0.00076549658623	0.00290738480223	0.00329402594190
N	0.00210766152916	0.00075164613741	0.00348854407143	0.00381369332481
O	0.00101438455764	0.00087653677794	0.00339295324505	0.00360909024761
P	0.00101438455764	0.00083972931536	0.00339295324505	0.00371656865319
Q	0.00171440737911	0.00090986173929	0.00302500858994	0.00325749319617
R	0.00171440737911	0.00078263486918	0.00302500858994	0.00313377218503
S	0.00179952889265	0.00092412574496	0.00308037221334	0.00358489761612
T	0.00202460645252	0.00077607463340	0.00297905567680	0.00345544069529
U	0.00399540647016	0.00088136649262	0.00406085975718	0.00393308599333
V	0.00201920898040	0.00093025411562	0.00292958098021	0.00394733592996
W	0.00198190404311	0.00083021505716	0.00286781566249	0.00325789134694
X	0.00204184145641	0.00085082896671	0.00278786498424	0.00333892925091
Y	0.00209015991196	0.00091132128574	0.00308512686889	0.00363619416358
Z	0.00229482924807	0.00099494048770	0.00342678028616	0.00426266543070

^a 26 decoy sets are renamed in alphabetic order for convenience only

One may think of Fig. 51 as a statistical signature of the empirical and theoretical ρ values one should expect. The empirical ρ value acts as an upper bound of the Steiner ratio value of the native protein. Notice that the native Steiner ratio underestimates the decoy Steiner ratio and the decoy structure indicates a lower energy value (higher ρ) than that of the native structure. The experiment underscores the inverse relationship between the Steiner ratio and the energy level. Thus, the Steiner ratio for each individual acid in the PDB structure is a useful indicator of the energy levels of the entire protein.

An additional hypothesis is that it appears that the native structure more closely matches the theoretical ρ_t value, since actually the native structure has the hydrogens even though they are not recorded in the PDB structure. Therefore, while not a perfect measure, the theoretical Steiner ρ value by itself becomes a useful benchmark to distinguish correctly folded structures.

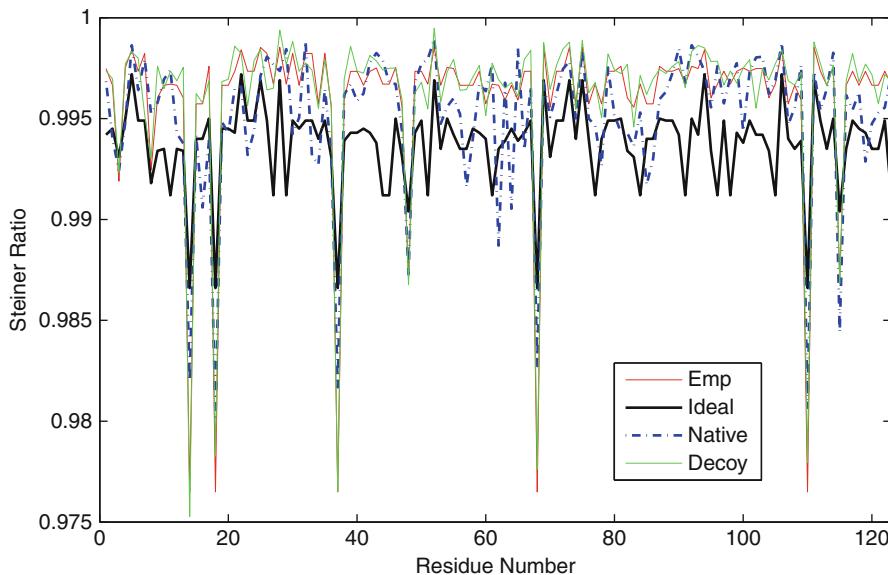


Fig. 51 Graphs of native and decoy Steiner ratios of data set A (1bp2 and 1bp2on2paz). The empirical ρ value acts as an *upper bound*, while the theoretical (ideal) ρ value is a *lower bound* of the Steiner ratio of the native protein. Decoy Steiner values are much closer to the empirical plot. Native Steiner values largely follow the theoretical plot with some marginal error

7.7.2 Experimental Error Detection

Besides the comparison of the native and decoy data structures, it appears that computing and graphing the Steiner ratio of all the amino acids in a native protein deposited on the PDB could indicate whether the PDB structure is a true accurate indicator of the native structure. The graphs of data sets C ("1fdx" and its decoy "1fdxon5rxn") and U ("2paz" and its decoy "2pazon1bp2"), although the latter are not shown, are examined in the comparison of the native and decoy data sets (see Fig. 52). Both proteins are interesting because they have statistical outliers that indicate the Steiner ratio for the amino acids in the PDB data are probabilistically unrealistic.

Using a standard normality test, Table 15 compares the ρ values of the observed amino acids which are outliers and computes the probability that the observed amino acid ρ values would occur. In all instances, the probabilities are essentially zero. Even if one assumed that the theoretical values were distributed in a similar manner as the empirical ρ values, the probabilities of the observed conformation would still be essentially zero.

Thus, the computed ρ values should be a very useful measure to indicate suspected conformational outliers or abnormalities in a protein structure obtained experimentally.

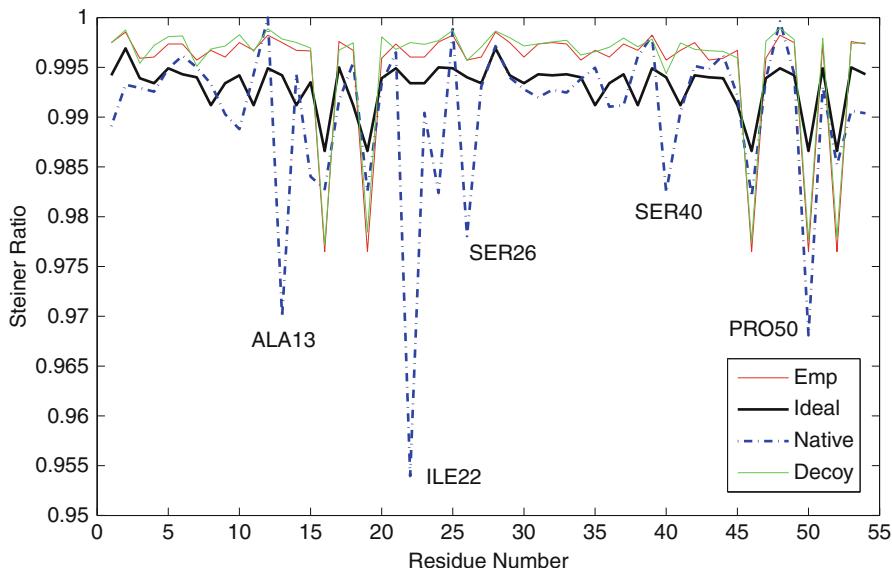


Fig. 52 Graphs of native and decoy Steiner ratios of data set C (1fdx and 1fdxon5rxn). Residue type and number of the five largest outliers are indicated. By comparison of the Steiner ratio, one can detect abnormalities of the native structure corrupted by experimental noise

Table 15 Statistical comparison of outliers in proteins

Protein C	Observed $\rho_{e'}$	ρ_t	ρ_e	σ	Z-score ^a
ALA13	0.969951	0.9942	0.99748	0.00099	-27.807
ILE22	0.953953	0.9934	0.99603	0.00069	-60.981
SER26	0.978052	0.9940	0.99573	0.00100	-17.678
SER40	0.982450	0.9940	0.99573	0.00100	-13.280
PRO50	0.968086	0.9866	0.97649	0.00209	-4.021
Protein U	$\rho_{e'}$	ρ_t	ρ_e	σ	Z-score
MET84	0.982605	0.9918	0.99245	0.00144	-6.837
SER104	0.974016	0.9940	0.99573	0.00100	-21.714

^aFrom the standardized normal distribution approximation of the probability distribution of the various acids, one can compute the Z-score $Z = \frac{x-\mu}{\sigma}$ which represents the probability of the particular acid having a ρ value below the empirical ρ_e value and in particular $P(X \leq a) = \Phi(\frac{a-\mu}{\sigma})$ of the standardized normal distribution. By convention, a Z-value of $Z \leq -3.4$ or less corresponds to essentially a zero probability value

8 Overall Protein Results

In order to summarize the results for protein modeling, let us posit the following:

Conjecture For an arbitrary sequence of n -amino acid residues denoted by $\{\rho_1, \rho_2, \dots, \rho_n\}$ within a protein chain, the smallest Steiner ratio is bounded below by the smallest Steiner ratio of the amino acid residues in the chain, i.e., $\rho_{Protein} \geq \min\{\rho_1, \rho_2, \dots, \rho_n\}$.

Obviously, if there are impurities and other unknowns acting within the protein, then perhaps this conjecture may not hold. Ideally, the property should hold because of the nature of assembling together the amino acids in the rigid peptide bond planes and the bottleneck property of most serial systems. The reasons why the conjecture should hold are:

- The Steiner-tree properties, i.e., ρ , give a dimensionless quantitative measure of the individual amino acids as well as the secondary structure of proteins. This is an important signature for each amino acid as well as for each protein.
- Steiner trees seek to find the shortest network possible. In Maxwell's theorem, this shortest network is equivalent to the minimum energy configuration that minimizes the potential energy in a system with uniform forces at each terminal vertex. Unfortunately, the forces are not all uniform in a protein, so the SMT can yield only a lower bound on the MEC topology. Especially in the amino acid experiments, however, it has been shown that this lower bound length for each amino acid is very tight.
- From the amino acid experiments of this chapter, the carbon and nitrogen atoms all act as Steiner points. This occurred in the unperturbed as well as in the perturbed acid structures. As displayed in Fig. 39 along with the previous experimental studies of secondary structure in [76], the rigid peptide bond planes are interconnected Steiner trees, and they maintain a constant Steiner ratio, so the only difference in ρ for a protein is due to the side chains.

This is felt to be significant, since in the secondary, tertiary, and quaternary evaluation of proteins, the carbon and nitrogen atoms should all be acting as Steiner points. It would also seem to indicate that, in an arbitrary sequence of amino acids, the Steiner ratio of the entire protein cannot be smaller than the Steiner ratio of the amino acid residues in the chain.

Future research will focus on studying larger systems of atoms that actually incorporate a good amount of secondary structure instead of just focusing on single amino acids. Still, however, improvements in the Steiner-tree algorithm need to be made to allow it to find the optimal solutions for larger N . One facet that may be valuable here for proteins is the regularity of the backbone topology and the knowledge of the topologies of the Steiner structure within each amino acid.

Also, more sophisticated tools and software techniques should be used in order to carry out the perturbations of the locations of the atoms and the measurement of the resultant energies. Finally, exploration of how SMTs can be used more directly to predict and evaluate low-energy conformation structures of proteins in the protein-folding problem should be conducted.

8.1 Other Applications

While the application of E^3 ESMT trees to other fields of science and engineering are speculative at this juncture, some mention of what is already known is worth exploring.

The structure of glass silicates exhibits a similar geometry to the triple helix pattern found with the conjectured optimal configuration for the Steiner problem [54].

As might be expected from what is known of collagen's structural properties, this may be applied to the design of columnar structures and space frame trusses. Buckminster Fuller [38, 60] laid down groundwork in this area, although his viewpoint was not from a Steiner perspective, but simply from the geometry of putting together tetrahedra. Fuller did not know the relationship of the triple helix to the Steiner problem and was simply fascinated by the helical shape when putting the tetrahedra together. He felt that such a regular repeating structure might have some stability characteristics other skeletal structures might not have, but he was not able to realize it in any manner. It would appear from the previous discussion on MECs and Maxwell's theorem that the Steiner coordinates and its network topology are crucial to the stability of the \mathcal{R} -sausage in any structural engineering application.

Of related interest to the above structural system properties includes the design of building systems, i.e., the heating, ventilating, air-conditioning, plumbing, and electrical systems that provide the utilities in normal building construction. Since minimizing overall length [69] tends to be the predominant objective of these building systems, the understanding of the optimal topology of ESMTs should be of major import for this application.

Other areas of science and engineering such as the design of polymers, antenna design (often helical antennas are employed), VLSI, and massively parallel computer network design would seem to be ideal candidates for examining how the ESMT and MEC problems relate, yet no more speculation at this point will be attempted.

9 Conclusion

Many of the properties, algorithmic methodologies, and applications of Steiner trees in E^3 have been collected together in this chapter. While the results so far are very new and just developing, they should give the reader an idea of the potential for Steiner trees for future theoretical and algorithmic research as well as the potential for their applications in science and engineering.

That all theory, methodological approaches (algebraic, geometric, and optimization), and application areas related to MECs and Steiner trees are closely intertwined is both fascinating and compelling. The tools of minimal-length network algorithms should help in the future verification and illumination of computational biological structures and perhaps other problems and applications in science and engineering.

Cross-References

- Gradient-Constrained Minimum Interconnection Networks
- Protein Docking Problem as Combinatorial Optimization Using Beta-Complex
- Steiner Minimal Trees: An Introduction, Parallel Computation, and Future Work

Recommended Reading

1. B. Alberts, D. Bray, J. Lewis, K. Roberts, J. Watson, *Molecular Biology of the Cell*, 2nd edn. (Garland Publishing, New York, 1983)
2. A.A. Aly, D.C. Kay, D.W. Litwhiler, Location dominance on spherical surfaces. *Oper. Res.* **27**, 972–981 (1979)
3. S. Arora, Polynomial time approximation scheme for Euclidean traveling salesman and other geometric problems. *J. ACM* **45**, 753–782 (1998)
4. F. Aurenhammer, Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Comput. Surv.* **23**, 345–405 (1991). Technical Report B-90-09, Department of Mathematics, Free University of Berlin (1990)
5. J.E. Beasley, A Greedy heuristic for the Euclidean and rectilinear Steiner problem. *EJOR* **58**, 284–292 (1992)
6. J.E. Beasley, F. Goffinet, A Delaunay triangulation based heuristic for the Euclidean Steiner problem. *Networks* **24**, 215–224 (1994)
7. I. Beichel, F. Sullivan, Fast triangulation via empty spheres. Working Paper, Computing and Applied Mathematics Laboratory National Institute of Standards and Technology, Gaithersburg, MD 20899 (1992)
8. M. Bern, D. Eppstein, Mesh generation and optimal triangulation, in *Computing and Euclidean Geometry*, ed. by D.Z. Du, F.K. Hwang (World Scientific, Singapore, 1992), pp. 23–90
9. M. Bern, R. Graham, The shortest-network problem. *Sci. Am.* **260**(1), 84–89 (1989)
10. B.N. Boots, *Voronoi (Thiessen) Polygons* (Geo Books/W.H. Hutchins & Sons, Norwich, 1986)
11. C. Brandon, J. Tooze, *Introduction to Protein Structure* (Gabriel Publishing, New York, 1991)
12. M. Brazil, J.H. Rubinstein, N. Wormald, D. Thomas, J. Weng, T. Cole, Minimal Steiner trees for $2^k \times 2^k$ square lattices. *J. Comb. Theory Ser. A* **73**, 91–110 (1996)
13. S.K. Chang, The generation of minimal trees with a Steiner topology. *J. ACM* **19**(4), 699–711 (1972)
14. Chembuilder, Version 1.1. *Interactive Simulations* (San Diego, 1997)
15. J.M. Chen, C.E. Kung, S.E. Feairheller, E.M. Brown, An energetic evaluation ... collagen microfibril model. *J. Protein Chem.* **10**, 535 (1991)
16. D. Cheriton, R.E. Tarjan, Finding minimal spanning trees. *SIAM J. Comput.* **5**(4), 724–742 (1976)
17. F.R.K. Chung, R.L. Graham, Steiner trees for ladders. *Ann. Discret. Math.* **2**, 173–200 (1978)
18. F.R.K. Chung, F.K. Hwang, A lower bound for the Steiner tree problem. *SIAM J. Appl. Math.* **34**(1), 27–36 (1976)
19. E.J. Cockayne, On Fermat's problem on the surface of a sphere. *Math. Mag.* **45**, 216–219 (1972)
20. E.J. Cockayne, Z.A. Melzak, Steiner's problem for set terminals. *J. Appl. Math.* **26**(2), 213–218 (1968)
21. E.J. Cockayne, Z.A. Melzak, Euclidean constructability in graph minimization problems. *Math. Mag.* **42**, 206–208 (1969)
22. F.E. Cohen, Folding the sheets: using computational methods to predict the structure of proteins, in *Calculating the Secrets of Life*, ed. by E. Lander (National Academy of Sciences, Washington, DC, 1995), pp. 236–271
23. D.R. Courant, H. Robbins, *What Is Mathematics?* (Oxford University Press, New York, 1941)

24. H.S.M. Coxeter, *Introduction to Geometry* (Wiley, New York, 1961)
25. P. Diaconis, R. Graham, *Magical Mathematics: The Mathematical Ideas That Animate Great Magic Tricks*. (Princeton University Press, Princeton, New Jersey, 2012)
26. R.E. Dickerson, I. Geis, *The Structure and Action of Proteins* (Harper and Row Publishers, New York, 1969)
27. B.W. Dijkstra, K.H. Kalk, W.G.J. Hol, J. Drenth, Structure of bovine pancreatic phospholipase-a2 at 1.7 angstroms resolution. *J. Mol. Biol.* **147**, 97–123 (1981)
28. K.A. Dill, Additivity principles in biochemistry. *J. Biol. Chem.* **272**, 701–704 (1997)
29. J. Dolan, R. Weiss, J.M. Smith, Minimal length tree networks on the unit sphere. *Ann. Oper. Res.* **33**, 503–535 (1991)
30. J.D.H. Donnay, Spherical trigonometry, in *Encyclopedia of Mathematics* (Interscience, New York, 1945)
31. Z. Drezner, G.O. Wesolowsky, Facility location on a sphere. *J. Oper. Res. Soc.* **29**, 997–1004 (1979)
32. D.Z. Du, Disproving Gilbert-Pollak conjecture in higher dimensional spaces. Manuscript, Computer Science Department, University of Minnesota, Sept 1992
33. D.Z. Du, F.K. Hwang, A proof of the Gilbert-Pollak conjecture on the Steiner ratio. *Algorithmica* **7**, 121–135 (1992)
34. D.Z. Du, W.D. Smith, Disproofs of generalized Gilbert-Pollak conjecture on the Steiner ratio in three or more dimensions. *J. Comb. Theory Ser. A* **74**(1), 115–130 (1996)
35. D.Z. Du, F.K. Hwang, J.F. Weng, Steiner minimal trees on zig-zag lines. *Trans. Am. Math. Soc.* **278**(1), 149–156 (1982)
36. S.J. Fortune, Voronoi diagrams and Delaunay triangulations, in *Computing and Euclidean Geometry*, ed. by D.Z. Du, F.K. Hwang (World Scientific, Singapore, 1992), pp. 193–233
37. S.A. Fossey, G. Nemethy, K.D. Gibson, H.A. Scheraga, Conformational energy studies of beta-sheets of model silk fibroin peptides. I. Sheets of poly(Ala-Gly) chains. *Biopolymers* **31**, 1529 (1991)
38. R.B. Fuller, *No More Secondhand God* (Southern Illinois University Press, Carbondale, 1963)
39. M.R. Garey, D.S. Johnson, *Computers and Intractability; A Guide to the Theory of Np-Completeness* (W.H. Freeman and Company, San Francisco, 1979)
40. M.R. Garey, R.L. Graham, D.S. Johnson, The complexity of computing Steiner minimal trees. *SIAM J. Appl. Math.* **32**(4), 835–859 (1977)
41. E.N. Gilbert, H.O. Pollak, Steiner minimal trees. *SIAM J. Appl. Math.* **16**, 1–29 (1968)
42. R.L. Graham, F.K. Hwang, Remarks on Steiner minimal trees. *Bull. Inst. Math. Acad. Sin.* **4**, 177–182 (1976)
43. M.G. Greening, Solution to problem E2233. *Am. Math. Mon.* **4**, 303–304 (1971)
44. N. Guex, M.C. Peitsch, SWISS-MODEL and the Swiss-PdbViewer: an environment for comparative protein modeling. *Electrophoresis* **18**, 2714–2723 (1997)
45. B.A. Hendrickson, The molecule problem: determining conformation from pairwise distances. Ph.D. thesis #90-1159, Department of Computer Science, Cornell University, Ithaca, NY 14853–7501, 1990
46. L. Holm, C. Sander, Evaluation of protein models by atomic solvation preference. *J. Mol. Biol.* **225**, 93–105 (1992)
47. F.W. Hwang, D. Richards, Steiner tree problems. *Networks* **22**(1), 55–89 (1989)
48. A.O. Ivanov, A.A. Tuzhilin, The Steiner Ratio Gilbert-Pollack Conjecture is Still Open. *Algorithmica* **62**(1), 630–632 (2012)
49. W. Kabsch et. al., Atomic structure of the actin/DNASE I complex. *Nature* **347**, 37 (1990)
50. H.W. Kuhn, A note on Fermat's problem. *Math. Program.* **4**, 98–107 (1973)
51. A.R. Leach, *Molecular Modeling – Principles and Applications*, 2nd. edn. (Prentice-Hall, New York, 1996)
52. D.W. Litwhiler, Steiner's problem and Fagnano's result on the sphere. *Math. Program.* **18**, 286–290 (1980)
53. R.F. Love, J.G. Morris, G.O. Wesolowsky, *Facilities Location* (North-Holland, New York, 1988)

54. I.J. McColm, *Ceramic Science for Materials Technologists* (Chapman and Hall, New York, 1983)
55. Z.A. Melzak, On the problem of Steiner. *Can. Math. Bull.* **4**, 143–148 (1961)
56. M.H. Miller, H.A. Scheraga, Calculation of the structures of collagen models. . . . *J. Polym. Sci. Polym. Symp.* **54**, 171–200 (1976)
57. G. Nemethy et.al., Energy parameters in polypeptides. . . . *J. Phys. Chem.* **96**, 6472–6484 (1992)
58. A. Okabe, B. Boots, K. Sugihara, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams* (Wiley, New York, 1992)
59. L. Pauling, The Architecture of Molecules. *Proc. Nat. Acad. Sci. USA.* **51**(5), 977–984 (1964)
60. P. Pearce, *Structure in Nature is a Strategy for Design* (MIT, Cambridge, 1978)
61. F. Preparata, M.I. Shamos, *Computational Geometry* (Springer, New York, 1985)
62. R.C. Prim, Shortest connecting networks and some generalizations. *BSTJ* **36**, 1389–1401 (1957)
63. Private communication
64. Protein Data Bank, Education Section (2004), <http://www.rothamsted.bbsrc.ac.uk/notebook/courses/guide/aa.htm>
65. Protein Explorer, (2004), <http://molvis.sdsc.edu/protexpl/frntdoor.htm>
66. K. Ray, T. Gaasterland, R. Overbeek, Automation and determination of 3-D protein structure. Preprint MCS-P417-0294, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL (1994)
67. P. Schorn, Private Communication, 1994
68. W.D. Smith, How to find Steiner minimal trees in Euclidean d-space. *Algorithmica* **7**, 137–177 (1992)
69. J.M. Smith, J.S. Liebman, Steiner trees, Steiner circuits, and the interference problem in building design. *Eng. Optim.* **4**(1), 15–36 (1979)
70. W.D. Smith, J.M. Smith, On the Steiner ratio in 3-space. *J. Comb. Theory Ser. A* **69**(2), 301–332 (1992)
71. J.M. Smith, B. Toppur, Euclidean Steiner minimal trees, minimum energy configurations, and the embedding problem of weighted graphs in E^3 . *Discret. Appl. Mat.* **71**, 187–215 (1996)
72. J.M. Smith, P. Winter, Computational geometry and topological network design, in *Computing in Euclidean Geometry*, ed. by D.-Z. Du and F. Hwang. Lecture Note Series in Computing, vol. 4, 2nd edn. (World Scientific, Singapore/River Edge, 1995), pp. 351–451
73. J.M. Smith, D.T. Lee, J.S. Liebman, An $O(N\log N)$ heuristic for Steiner minimal tree problems on the Euclidean metric. *Networks* **11**(1), 23–39 (1981)
74. J.M. Smith, R. Weiss, M. Patel, An $O(N^2)$ heuristic for Steiner minimal trees in E^3 . *Networks* **25**, 273–289 (1995)
75. J.M. Smith, R. Weiss, M. Patel, An $O(N^2)$ heuristic for the Steiner minimal tree problem in E^3 . *Networks* **25**, 273–289 (1995)
76. J.M. Smith, R. Weiss, B. Toppur, N. Maculan, Characterization of protein structure and 3-d Steiner networks, *Proceedings of the II ALIO/EURO Workshop on Practical Combinatorial Optimization*, Faculty of Engineering, Catholic University of Valparaiso, School of Industrial Engineering, Valparaiso, Chile, Nov 1996, pp. 37–44
77. C. Stanton, J.M. Smith, Steiner trees and 3-D macromolecular conformation. *Informs J. Comput.* **16**, 470–485 (2004)
78. M.C. Surles, J.S. Richardson, D.C. Richardson, F.P. Brooks, Sculpting proteins interactively: continual energy minimization embedded in a graphical modeling system. *Protein Sci* **3**, 198–210 (1994)
79. B. Toppur, J.M. Smith, A sausage heuristic for Steiner minimal trees in three-dimensional Euclidean space. *J. Math. Model. Algorithms* **4**, 199–217 (2005)
80. F. Tóth, Research problem 13. *Period. Math. Hung.* **6**, 197–199 (1975)

81. P.M. Vaidya, Minimum spanning trees in k -dimensional space. SIAM J. Comput. **17**(3), 572–582 (1988)
82. D. Voet, J. Voet, *Biochemistry* (Wiley, New York, 1990)
83. D.M. Warme, P. Winter, and M. Zachariasen, Exact algorithms for plane Steiner tree problems: a computational study, in *Advances in Steiner Trees*, ed. by D.Z. Du, J.M. Smith, J.H. Rubinstein (Kluwer, Dordrecht/Boston, 2000), pp. 81–116
84. G.O. Wesolowsky, Location problems on a sphere. Reg. Sci. Urban Econ. **12**, 495–508 (1982)
85. J.M. Wills, On the density of finite packing. Acta Math. Hung. **46**, 205–210 (1985)
86. P. Winter, Steiner problem in networks: a survey. Networks **17**, 129–167 (1987)
87. S. Wolfram, *Mathematica*, 3rd edn. (Cambridge University Press, Cambridge/New York, 1996)

Tabu Search*

Fred Glover and Manuel Laguna

Contents

1	Introduction	3263
2	Tabu Search Features and Relevance	3263
2.1	General Tenets	3265
2.2	Use of Memory	3266
2.3	Intensification and Diversification	3267
3	Tabu Search Foundations and Short-Term Memory	3268
3.1	Memory and Tabu Classifications	3269
3.2	Recency-Based Memory	3270
3.3	A First-Level Tabu Search Approach	3275
3.4	Recency-Based Memory for Add/Drop Moves	3279
3.5	Tabu Tenure	3283
3.6	Aspiration Criteria and Regional Dependencies	3287
3.7	Concluding Observations for the Min k-Tree Example	3290
4	Additional Aspects of Short-Term Memory	3291
4.1	Tabu Search and Candidate List Strategies	3291
4.2	Some General Classes of Candidate List Strategies	3292
4.3	Connections Between Candidate Lists, Tabu Status, and Aspiration Criteria	3297
4.4	Logical Restructuring	3297
5	Longer-Term Memory	3302
5.1	Frequency-Based Approach	3303
5.2	Intensification Strategies	3306
5.3	Diversification Strategies	3307

*The material of this chapter is in part adapted from the book *Tabu Search*, by Fred Glover and Manuel Laguna, Kluwer Academic Publishers, 1997.

F. Glover (✉)
OptTek Systems, Inc, Boulder, CO, USA
e-mail: glover@opttek.com

M. Laguna
Leeds School of Business, University of Colorado, Boulder, CO, USA
e-mail: laguna@colorado.edu

5.4 Strategic Oscillation.....	3310
5.5 Path Relinking.....	3313
5.6 The Intensification/Diversification Distinction.....	3317
5.7 Some Basic Memory Structures for Longer-Term Strategies.....	3319
6 Connections, Hybrid Approaches, and Learning.....	3323
6.1 Simulated Annealing.....	3323
6.2 Genetic Algorithms.....	3324
6.3 Scatter Search.....	3327
6.4 Greedy Randomized Adaptive Search Procedures (GRASP).....	3333
6.5 Neural Networks.....	3335
6.6 Target Analysis.....	3336
7 Neglected Tabu Search Strategies.....	3346
7.1 Candidate List Strategies.....	3346
7.2 Intensification Approaches.....	3347
7.3 Diversification Approaches.....	3349
7.4 Strategic Oscillation.....	3352
7.5 Clustering and Conditional Analysis.....	3353
7.6 Referent-Domain Optimization.....	3355
8 Conclusion.....	3357
Cross-References.....	3359
Recommended Reading.....	3359

Abstract

Tabu search, also called adaptive memory programming, is a method for solving challenging problems in the field of optimization. The goal is to identify the best decisions or actions in order to maximize some measure of merit (such as maximizing profit, effectiveness, quality, and social or scientific benefit) or to minimize some measure of demerit (cost, inefficiency, waste, and social or scientific loss).

Practical applications in optimization addressed by tabu search are exceedingly challenging and pervade the fields of business, engineering, economics, and science. Everyday examples include problems in resource management, financial and investment planning, healthcare systems, energy and environmental policy, pattern classification, biotechnology, and a host of other areas. The complexity and importance of such problems has motivated a wealth of academic and practical research throughout the past several decades, in an effort to discover methods that are able to find solutions of higher quality than many found in the past and capable of producing such solutions within feasible time limits or at reduced computational cost.

Tabu search has emerged as one of the leading technologies for handling optimization problems that have proved difficult or impossible to solve with classical procedures that dominated the attention of textbooks and were considered the mainstays of available alternatives until recent times. A key feature of tabu search, underscored by its *adaptive memory programming* alias, is the use of special strategies designed to exploit adaptive memory. The idea is that an effective search for optimal solutions should involve a process of flexibly responding to the solution landscape in a manner that permits it to learn

appropriate directions to take along with appropriate departures to explore new terrain. The adaptive memory feature of tabu search allows the implementation of procedures that are capable of searching this terrain economically and effectively.

1 Introduction

Faced with the challenge of solving hard optimization problems that abound in the real world, classical methods often encounter great difficulty. Vitally important applications in business, engineering, economics, and science cannot be tackled with any reasonable hope of success, within practical time horizons, by solution methods that have been the predominant focus of academic research throughout the past three decades (and which are still the focus of many textbooks).

The meta-heuristic approach called tabu search (TS) is dramatically changing our ability to solve problems of practical significance. Current applications of TS span the realms of resource planning, telecommunications, VLSI design, financial analysis, scheduling, space planning, energy distribution, molecular engineering, logistics, pattern classification, flexible manufacturing, waste management, mineral exploration, biomedical analysis, environmental conservation, and scores of others. In recent years, journals in a wide variety of fields have published tutorial articles and computational studies documenting successes by tabu search in extending the frontier of problems that can be handled effectively – yielding solutions whose quality often significantly surpasses that obtained by methods previously applied. **Table 1** gives a partial catalog of example applications. A more comprehensive list, including summary descriptions of gains achieved from practical implementations, can be found in Glover and Laguna [31]. Recent TS developments and applications can also be found in the Tabu Search Vignettes section of the web page <http://spot.colorado.edu/~glover>.

2 Tabu Search Features and Relevance

A distinguishing feature of tabu search is embodied in its exploitation of adaptive forms of memory, which equips it to penetrate complexities that often confound alternative approaches. Yet we are only beginning to tap the rich potential of adaptive memory strategies, and the discoveries that lie ahead promise to be as important and exciting as those made to date. The knowledge and principles that have emerged from the TS framework give a foundation to create practical systems whose capabilities markedly exceed those available earlier. At the same time, there are many untried variations that may lead to further advances. A conspicuous feature of tabu search is that it is dynamically growing and evolving, drawing on important contributions by many researchers.

Table 1 Illustrative tabu search applications

<i>Scheduling</i>	<i>Telecommunications</i>
Flow-time cell manufacturing	Call routing
Heterogeneous processor scheduling	Bandwidth packing
Workforce planning	Hub facility location
Classroom scheduling	Path assignment
Machine scheduling	Network design for services
Flow shop scheduling	Customer discount planning
Job shop scheduling	Failure immune architecture
Sequencing and batching	Synchronous optical networks
<i>Design</i>	<i>Production, inventory and investment</i>
Computer-aided design	Flexible manufacturing
Fault tolerant networks	Just-in-time production
Transport network design	Capacitated MRP
Architectural space planning	Part selection
Diagram coherency	Multi-item inventory planning
Fixed charge network design	Volume discount acquisition
Irregular cutting problems	Fixed mix investment
<i>Location and allocation</i>	<i>Routing</i>
Supply chain analysis	Vehicle routing
Multicommodity location/allocation	Capacitated routing
Quadratic assignment	Time window routing
Quadratic semi-assignment	Multi-mode routing
Multilevel generalized assignment	Mixed fleet routing
Lay-out planning	Traveling salesman
Off-shore oil exploration	Traveling purchaser
<i>Logic and artificial intelligence</i>	<i>Graph optimization</i>
Maximum satisfiability	Graph partitioning
Probabilistic logic	Graph coloring
Clustering	Clique partitioning
Pattern recognition/classification	Maximum clique problems
Data integrity	Maximum planner graphs
Neural network Training and design	P-median problems
<i>Technology</i>	<i>General combinational optimization</i>
Seismic inversion	Zero-one programming
Electrical power distribution	Fixed charge optimization
Engineering structural design	Nonconvex nonlinear programming
Coordination of energy resources	All-or-none networks
Space station construction	Bilevel programming
DNA sequencing	Multi-objective discrete optimization
Circuit cell placement	Hyperplane splitting
Computer aided molecular design	General mixed integer optimization

2.1 General Tenets

The word *tabu* (or *taboo*) comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga island to indicate things that cannot be touched because they are sacred. According to Webster's Dictionary, the word now also means "a prohibition imposed by social custom as a protective measure" or of something "banned as constituting a risk." These current more pragmatic senses of the word accord well with the theme of tabu search. The risk to be avoided in this case is that of following a counterproductive course, including one which may lead to entrapment without hope of escape. On the other hand, as in the broader social context where "protective prohibitions" are capable of being superseded when the occasion demands, the "tabus" of tabu search are to be overruled when evidence of a preferred alternative becomes compelling.

The most important association with traditional usage, however, stems from the fact that tabus as normally conceived are transmitted by means of a social memory which is subject to modification over time. This creates the fundamental link to the meaning of "tabu" in tabu search. The forbidden elements of tabu search receive their status by reliance on an evolving memory, which allows this status to shift according to time and circumstance.

More particularly, tabu search is based on the premise that problem solving, in order to qualify as intelligent, must incorporate *adaptive memory* and *responsive exploration*. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Since local choices are guided by information collected during the search, TS contrasts with memoryless designs that heavily rely on semirandom processes that implement a form of sampling. Examples of memoryless methods include semi-greedy heuristics and the prominent "genetic" and "annealing" approaches inspired by metaphors of physics and biology. Adaptive memory also contrasts with rigid memory designs typical of branch and bound strategies (It can be argued that some types of evolutionary procedures that operate by combining solutions, such as genetic algorithms, embody a form of implicit memory. Special links with evolutionary methods, and implications for establishing more effective variants of them, are discussed in [Sect. 6](#)).

The emphasis on responsive exploration in tabu search, whether in a deterministic or probabilistic implementation, derives from the supposition that a bad strategic choice can yield more information than a good random choice. In a system that uses memory, a bad choice based on strategy can provide useful clues about how the strategy may profitably be changed (Even in a space with significant randomness, a purposeful design can be more adept at uncovering the imprint of structure).

Responsive exploration integrates the basic principles of intelligent search, that is, exploiting good solution features while exploring new promising regions. Tabu search is concerned with finding new and more effective ways of taking advantage of

the mechanisms associated with both adaptive memory and responsive exploration. The development of new designs and strategic mixes makes TS a fertile area for research and empirical study.

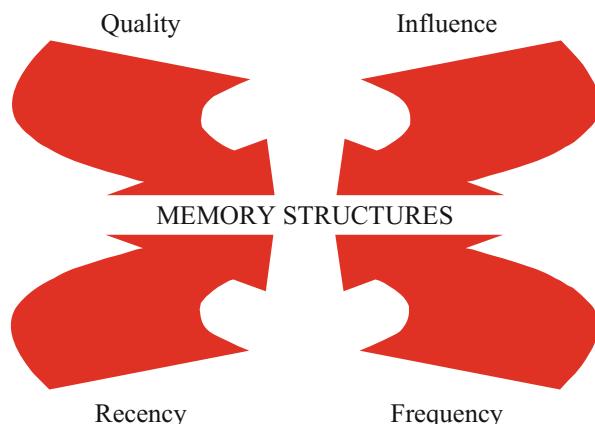
2.2 Use of Memory

The memory structures in tabu search operate by reference to four principal dimensions, consisting of recency, frequency, quality, and influence (Fig. 1). *Recency-based* and *frequency-based* memory complement each other and have important characteristics we amplify in later sections. The *quality* dimension refers to the ability to differentiate the merit of solutions visited during the search. In this context, memory can be used to identify elements that are common to good solutions or to paths that lead to such solutions. Operationally, quality becomes a foundation for incentive-based learning, where inducements are provided to reinforce actions that lead to good solutions and penalties are provided to discourage actions that lead to poor solutions. The flexibility of these memory structures allows the search to be guided in a multi-objective environment, where the goodness of a particular search direction may be determined by more than one function. The tabu search concept of quality is broader than the one implicitly used by standard optimization methods.

The fourth dimension, *influence*, considers the impact of the choices made during the search, not only on quality but also on structure (In a sense, quality may be regarded as a special form of influence). Recording information about the influence of choices on particular solution elements incorporates an additional level of learning. By contrast, in branch and bound, for example, the separation rules are prespecified and the branching directions remain fixed, once selected, at a given node of a decision tree. It is clear however that certain decisions have more influence than others as a function of the neighborhood of moves employed and the way that this neighborhood is negotiated (e.g., choices near the root of a branch and bound tree are quite influential when using a depth-first strategy). The assessment and exploitation of influence by a memory more flexible than embodied in such tree searches is an important feature of the TS framework.

The memory used in tabu search is both *explicit* and *attributive*. Explicit memory records complete solutions, typically consisting of elite solutions visited during the search. An extension of this memory records highly attractive but unexplored neighbors of elite solutions. The memorized elite solutions (or their attractive neighbors) are used to expand the local search, as indicated in Sect. 4. In some cases, explicit memory has been used to guide the search and avoid visiting solutions more than once. This application is limited, because clever data structures must be designed to avoid excessive memory requirements.

Alternatively, TS uses attributive memory for guiding purposes. This type of memory records information about solution attributes that change in moving from one solution to another. For example, in a graph or network setting, attributes can consist of nodes or arcs that are added, dropped, or repositioned by the moving mechanism. In production scheduling, the index of jobs may be used as attributes to inhibit or encourage the method to follow certain search directions.

Fig. 1 Four TS dimensions

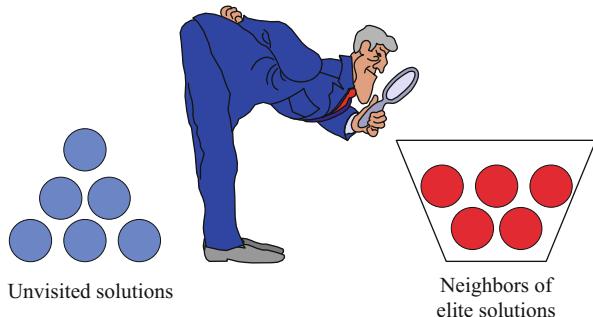
2.3 Intensification and Diversification

Two highly important components of tabu search are intensification and diversification strategies. Intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good. They may also initiate a return to attractive regions to search them more thoroughly. Since elite solutions must be recorded in order to examine their immediate neighborhoods, explicit memory is closely related to the implementation of intensification strategies. As Fig. 2 illustrates, the main difference between intensification and diversification is that during an intensification stage, the search focuses on examining neighbors of elite solutions.

Here the term “neighbors” has a broader meaning than in the usual context of “neighborhood search.” That is, in addition to considering solutions that are adjacent or close to elite solutions by means of standard move mechanisms, intensification strategies generate “neighbors” by either grafting together components of good solution or by using modified evaluation strategies that favor the introduction of such components into a current (evolving) solution. The diversification stage on the other hand encourages the search process to examine unvisited regions and to generate solutions that differ in various significant ways from those seen before. Again, such an approach can be based on generating subassemblies of solution components that are then “fleshed out” to produce full solutions or can rely on modified evaluations as embodied, for example, in the use of penalty/incentive functions.

Intensification strategies require a means for identifying a set of elite solutions as basis for incorporating good attributes into newly created solutions. Membership in the elite set is often determined by setting a threshold which is connected to the objective function value of the best solution found during the search. However, considerations of clustering and “anti-clustering” are also relevant for generating such a set and more particularly for generating subsets of solutions that may be used for specific phases of intensification and diversification. In the following sections, we show how the treatment of such concerns can be enhanced by making use of

Fig. 2 Intensification and diversification



special memory structures. The TS notions of intensification and diversification are beginning to find their way into other meta-heuristics, and it is important to keep in mind (as we subsequently demonstrate) that these ideas are somewhat different than the old control theory concepts of “exploitation” and “exploration,” especially in their implications for developing effective problem-solving strategies.

3 Tabu Search Foundations and Short-Term Memory

Tabu search can be applied directly to verbal or symbolic statements of many kinds of decision problems, without the need to transform them into mathematical formulations. Nevertheless, it is useful to introduce mathematical notation to express a broad class of these problems, as a basis for describing certain features of tabu search. We characterize this class of problems as that of optimizing (minimizing or maximizing) a function $f(x)$ subject to $x \in X$, where $f(x)$ may be linear or nonlinear, and the set X summarizes constraints on the vector of decision variables x . The constraints may include linear or nonlinear inequalities and may compel all or some components of x to receive discrete values. While this representation is useful for discussing a number of problem-solving considerations, we emphasize again that in many applications of combinatorial optimization, the problem of interest may not be easily formulated as an objective function subject to a set of constraints. The requirement $x \in X$, for example, may specify logical conditions or interconnections that would be cumbersome to formulate mathematically but may be better left as verbal stipulations that can be then coded as rules.

Tabu search begins in the same way as ordinary local or neighborhood search, proceeding iteratively from one point (solution) to another until a chosen termination criterion is satisfied. Each $x \in X$ has an associated neighborhood $N(x) \subset X$, and each solution $x' \in N(x)$ is reached from x by an operation called a *move*.

As an initial point of departure, we may contrast TS with a simple descent method where the goal is to minimize $f(x)$ (or a corresponding ascent method where the goal is to maximize $f(x)$). Such a method only permits moves to neighbor solutions that improve the current objective function value and ends when no improving solutions can be found. A pseudo-code of a generic descent method is presented in Fig. 3. The final x obtained by a descent method is called a local optimum, since

Fig. 3 Descent method

- 1) Choose $x \in \mathbf{X}$ to start the process.
- 2) Find $x' \in N(x)$ such that $f(x') < f(x)$.
- 3) If no such x' can be found, x is the local optimum and the method stops.
- 4) Otherwise, designate x' to be the new x and go to 2).

it is at least as good or better than all solutions in its neighborhood. The evident shortcoming of a descent method is that such a local optimum in most cases will not be a global optimum, that is, it usually will not minimize $f(x)$ over all $x \in X$.

The version of a descent method called *steepest descent* scans the entire neighborhood of x in search of a neighbor solution x' that gives a smallest $f(x')$ value over $x' \in N(x)$. Steepest descent implementations of some types of solution approaches (such as certain path augmentation algorithms in networks and matroids) are guaranteed to yield globally optimal solutions for the problems they are designed to handle, while other forms of descent may terminate with local optima that are not global optima. In spite of this attractive feature, in certain settings, steepest descent is sometimes impractical because it is computationally too expensive, as where $N(x)$ contains many elements or each element is costly to retrieve or evaluate. Still, it is often valuable to choose an x' at each iteration that yields a “good” if not smallest $f(x')$ value.

The relevance of choosing good solutions from current neighborhoods is magnified when the guidance mechanisms of tabu search are introduced to go beyond the locally optimal termination point of a descent method. Thus, an important first-level consideration for tabu search is to determine an appropriate *candidate list strategy* for narrowing the examination of elements of $N(x)$, in order to achieve an effective trade-off between the quality of x' and the effort expended to find it. Here quality may involve considerations beyond those narrowly reflected by the value of $f(x')$. If a neighborhood space is totally random, then of course nothing will work better than a totally random choice (In such a case, there is no merit in trying to devise an effective solution procedure). Assuming that neighborhoods can be identified that are reasonably meaningful for a given class of problems, the challenge is to define solution quality appropriately so that evaluations likewise will have meaning. By the TS orientation, the ability to use history in creating such evaluations then becomes important for devising effective methods

To give a foundation for understanding the basic issues involved, we turn our attention to the following illustrative example, which will also be used as a basis for illustrating various aspects of tabu search in later sections.

3.1 Memory and Tabu Classifications

An important distinction in TS arises by differentiating between short-term memory and longer-term memory. Each type of memory is accompanied by its own special

strategies. However, the effect of both types of memory may be viewed as modifying the neighborhood $\mathbf{N}(x)$ of the current solution x . The modified neighborhood, which we denote by $\mathbf{N}^*(x)$, is the result of maintaining a selective history of the states encountered during the search.

In the TS strategies based on short-term considerations, $\mathbf{N}^*(x)$ characteristically is a subset of $\mathbf{N}(x)$, and the tabu classification serves to identify elements of $\mathbf{N}(x)$ excluded from $\mathbf{N}^*(x)$. In TS strategies that include longer-term considerations, $\mathbf{N}^*(x)$ may also be expanded to include solutions not ordinarily found in $\mathbf{N}(x)$. Characterized in this way, TS may be viewed as a dynamic neighborhood method. This means that the neighborhood of x is not a static set, but rather a set that can change according to the history of the search. This feature of a dynamically changing neighborhood also applies to the consideration of selecting different component neighborhoods from a *compound* neighborhood that encompasses multiple types or levels of moves and provides an important basis for parallel processing. Characteristically, a TS process based strictly on short-term strategies may allow a solution x to be visited more than once, but it is likely that the corresponding reduced neighborhood $\mathbf{N}^*(x)$ will be different each time. With the inclusion of longer-term considerations, the likelihood of duplicating a previous neighborhood upon revisiting a solution, and more generally of making choices that repeatedly visit only a limited subset of \mathbf{X} , is all but nonexistent. From a practical standpoint, the method will characteristically identify an optimal or near-optimal solution long before a substantial portion of \mathbf{X} is examined.

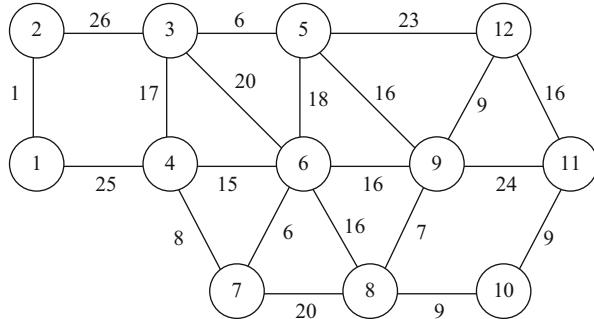
A crucial aspect of TS involves the choice of an appropriate definition of $\mathbf{N}^*(x)$. Due to the exploitation of memory, $\mathbf{N}^*(x)$ depends upon the trajectory followed in moving from one solution to the next (or upon a collection of such trajectories in a parallel processing environment).

The approach of storing complete solutions (*explicit* memory) generally consumes an enormous amount of space and time when applied to each solution generated. A scheme that emulates this approach with limited memory requirements is given by the use of hash functions (Also, as will be seen, explicit memory has a valuable role when selectively applied in strategies that record and analyze certain “special” solutions). Regardless of the implementation details, short-term memory functions provide one of the important cornerstones of the TS methodology. These functions give the search the opportunity to continue beyond local optima, by allowing the execution of nonimproving moves coupled with the modification of the neighborhood structure of subsequent solutions. However, instead of recording full solutions, these memory structures are generally based on recording attributes (*attributive* memory). In addition, short-term memory is often based on the most recent history of the search trajectory.

3.2 Recency-Based Memory

The most commonly used short-term memory keeps track of solutions attributes that have changed during the recent past and is called *recency-based* memory. This

Fig. 4 Weighted undirected graph



is the kind of memory that is included in most short descriptions of tabu search in the literature (although a number of its aspects are often left out by popular summaries).

To exploit this memory, selected attributes that occur in solutions recently visited are labeled *tabu-active*, and solutions that contain tabu-active elements, or particular combinations of these attributes, are those that become tabu. This prevents certain solutions from the recent past from belonging to $\mathbf{N}^*(x)$ and hence from being revisited. Other solutions that share such tabu-active attributes are also similarly prevented from being visited. Note that while the tabu classification strictly refers to solutions that are forbidden to be visited, by virtue of containing tabu-active attributes (or more generally by violating certain restriction based on these attributes), we also often refer to moves that lead to such solutions as being tabu. We illustrate these points with the following example.

Minimum k -Tree Problem Example

The *Minimum k -Tree* problem seeks a tree consisting of k edges in a graph so that the sum of the weights of these edges is minimum [49]. An instance of this problem is given in Fig. 4, where nodes are shown as numbered circles and edges are shown as lines that join pairs of nodes (the two “endpoint” nodes that determine the edge). Edge weights are shown as the numbers attached to these lines. A tree is a set of edges that contains no cycles, that is, that contains no paths that start and end at the same node (without retracing any edges).

Assume that the move mechanism is defined by edge swapping, as subsequently described, and that a greedy procedure is used to find an initial solution. The greedy construction starts by choosing the edge (i, j) with the smallest weight in the graph, where i and j are the indexes of the nodes that are the endpoints of the edge. The remaining $k-1$ edges are chosen successively to minimize the increase in total weight at each step, where the edges considered meet exactly one node from those that are endpoints of edges previously chosen. For $k = 4$, the greedy construction performs the steps in Table 2.

The construction starts by choosing edge $(1, 2)$ with a weight of 1 (the smallest weight of any edge in the graph). After this selection, the candidate edges are those that connect the nodes in the current partial tree with those nodes not in the tree (i.e., edges $(1, 4)$ and $(2, 3)$). Since edge $(1, 4)$ minimizes the weight increase, it is

Table 2 Greedy construction.

Step	Candidates	Selection	Total weight
1	(1,2)	(1,2)	1
2	(1,4), (2,3)	(1,4)	26
3	(2,3), (3,4), (4,6), (4,7)	(4,7)	34
4	(2,3), (3,4), (4,6), (6,7), (7,8)	(6,7)	40

chosen to be part of the partial solution. The rest of the selections follow the same logic, and the construction ends when the tree consists of 4 edges (i.e., the value of k). The initial solution in this particular case has a total weight of 40.

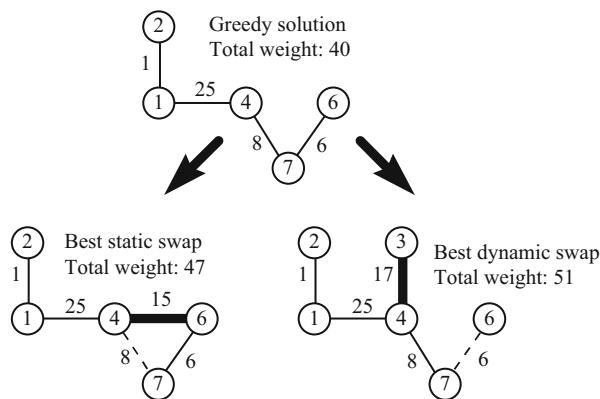
The swap move mechanism, which is used from this point onward, replaces a selected edge in the tree by another selected edge outside the tree, subject to requiring that the resulting subgraph is also a tree. There are actually two types of such edge swaps, one that maintains the current nodes of the tree unchanged (static) and one that results in replacing a node of the tree by a new node (dynamic). [Figure 5](#) illustrates the best swap of each type that can be made starting from the greedy solution. The added edge in each case is shown by a heavy line, and the dropped edge is shown by a dotted line.

The best move of both types is the static swap of [Fig. 5](#), where for our present illustration, we are defining *best* solely in terms of the change on the objective function value. Since this best move results in an increase of the total weight of the current solution, the execution of such move abandons the rules of a descent approach and sets the stage for a tabu search process. (The feasibility restriction that requires a tree to be produced at each step is particular to this illustration, since in general the TS methodology may include search trajectories that violate various types of feasibility conditions.)

Given a move mechanism, such as the swap mechanism we have selected for our example, the next step is to choose the key attributes that will be used for the tabu classification. Tabu search is very flexible at this stage of the design. Problem-specific knowledge can be used as guidance to settle on a particular design. In problems where the moves are defined by adding and deleting elements, the labels of these elements can be used as the attributes for enforcing tabu status. Here, in the present example, we can simply refer to the edges as attributes of the move, since the condition of being *in* or *out of the tree* (which is a distinguishing property of the current solution) may be assumed to always be automatically known by a reasonable solution representation.

Choosing Tabu Classifications

Tabu classifications do not have to be symmetric, that is, the tabu structure can be designed to treat added and dropped elements differently. Suppose, for example, that after choosing the static swap of [Fig. 5](#), which adds edge (4,6) and drops edge (4,7), a tabu status is assigned to both of these edges. Then one possibility is to classify both of these edges tabu-active for the same number of iterations. The tabu-active status has different meanings depending on whether the edge is added or dropped. For an added edge, tabu-active means that this edge is not allowed to be dropped

Fig. 5 Swap move types

from the current tree for the number of iterations that defines its tabu tenure. For a dropped edge, on the other hand, tabu-active means the edge is not allowed to be included in the current solution during its tabu tenure. Since there are many more edges outside the tree than in the tree, it seems reasonable to implement a tabu structure that keeps a recently dropped edge tabu-active for a longer period of time than a recently added edge. Notice also that for this problem, the tabu-active period for added edges is bounded by k , since if no added edge is allowed to be dropped for k iterations, then within k steps, all available moves will be classified tabu.

The concept of creating asymmetric tabu classifications can be readily applied to settings where add/drop moves are not used.

Illustrative Tabu Classifications for the Min k-Tree Problem

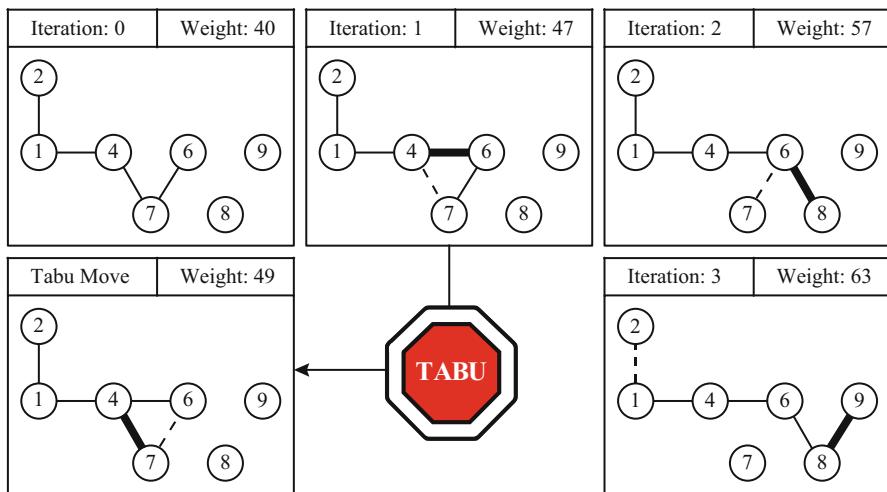
As previously remarked, the tabu-active classification may in fact prevent the search from visiting solutions that have not been examined yet. We illustrate this phenomenon as follows. Suppose that in the Min k -Tree problem instance of Fig. 4, dropped edges are kept tabu-active for two iterations, while added edges are kept tabu-active for only one iteration (The number of iterations an edge is kept tabu-active is called the *tabu tenure* of the edge). Also assume that we define a swap move to be tabu if either its added or dropped edge is tabu-active. If we examine the full neighborhood of available edge swaps at each iteration, and always choose the best that is not tabu, then the first three moves are as shown in Table 3 below (starting from the initial solution found by the greedy construction heuristic). The move of iteration 1 is the static swap move previously identified in Fig. 5. Diagrams showing the successive trees generated by these moves, starting with the initial greedy solution, are given in Fig. 6.

The net tenure values of 1 and 2 in Table 3 for the currently tabu-active edges indicate the number of iterations that these edges will remain tabu-active (including the current iteration).

At iteration 2, the reversal of the move of iteration 1 (i.e., the move that now adds (4,7) and drops (4,6)) is clearly tabu, since both of its edges are tabu-active at iteration 2. In addition, the move that adds (4,7) and drops (6,7) is also classified tabu, because it contains the tabu-active edge (4,7) (with a net tenure of 2).

Table 3 TS iterations

Iteration	Tabu-active net tenure		Add	Drop	Weight
	1	2			
1			(4,6)	(4,7)	47
2	(4,6)	(4,7)	(6,8)	(6,7)	57
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	63

**Fig. 6** Effects of attributive short-term memory

This move leads to a solution with a total weight of 49, a solution that clearly has not been visited before (see Fig. 6). The tabu-active classification of (4,7) has modified the original neighborhood of the solution at iteration 2 and has forced the search to choose a move with an inferior objective function value (i.e., the one with a total weight of 57). In this case, excluding the solution with a total weight of 49 has little effect on the quality of the best solution found (since we have already obtained one with a weight of 40).

In other situations, however, additional precautions must be taken to avoid missing good solutions. These strategies are known as aspiration criteria and are the subject of Sect. 3.6. For the moment, we observe simply that if the tabu solution encountered at the current step instead had a weight of 39, which is better than the best weight of 40 so far seen, then we would allow the tabu classification of this solution to be overridden and consider the solution admissible to be visited. The aspiration criterion that applies in this case is called the *improved-best* aspiration criterion (It is important to keep in mind that aspiration criteria do not compel particular moves to be selected, but simply make them available, or alternately rescind evaluation penalties attached to certain tabu classifications).

One other comment about tabu classification deserves to be made at this point. In our preceding discussion of the Min k -Tree problem, we consider a swap move

tabu if either its added edge or its dropped edge is tabu-active. However, we could instead stipulate that a swap move is tabu only if both its added and dropped edges are tabu-active. In general, the tabu status of a move is a function of the tabu-active attributes of the move (i.e., of the new solution produced by the move).

3.3 A First-Level Tabu Search Approach

We now have on hand enough ingredients for a first-level tabu search procedure. Such a procedure is sometimes implemented in an initial phase of a TS development to obtain a preliminary idea of performance and calibration features or simply to provide a convenient staged approach for the purpose of debugging solution software. While this naive form of a TS method omits a number of important short-term memory considerations and does not yet incorporate longer-term concerns, it nevertheless gives a useful starting point for demonstrating several basic aspects of tabu search.

We start from the solution with a weight of 63 as shown previously in Fig. 6 which was obtained at iteration 3. At each step, we select the least-weight non-tabu move from those available and use the improved-best aspiration criterion to allow a move to be considered admissible in spite of leading to a tabu solution. The reader may verify that the outcome leads to the series of solutions shown in Table 4, which continues from iteration 3, just executed. For simplicity, we select an arbitrary stopping rule that ends the search at iteration 10.

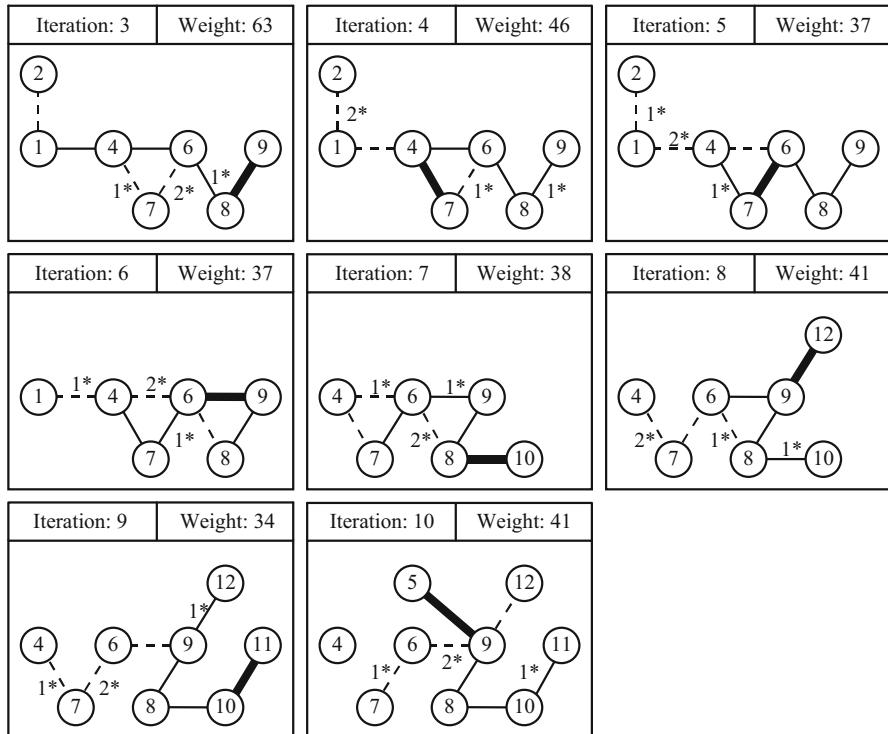
The successive solutions identified in Table 4 are shown graphically in Fig. 7 below. In addition to identifying the dropped edge at each step as a dotted line, we also identify the dropped edge from the immediately preceding step as a dotted line which is labeled 2^* to indicate its current net tabu tenure of 2. Similarly, we identify the dropped edge from one further step back by a dotted line which is labeled 1^* to indicate its current net tabu tenure of 1. Finally, the edge that was added on the immediately preceding step is also labeled 1^* to indicate that it likewise has a current net tabu tenure of 1. Thus, the edges that are labeled with tabu tenures are those which are currently tabu-active and which are excluded from being chosen by a move of the current iteration (unless permitted to be chosen by the aspiration criterion).

As illustrated in Table 4 and Fig. 7, the method continues to generate different solutions, and over time, the best-known solution (denoted by an asterisk) progressively improves. In fact, it can be verified for this simple example that the solution obtained at iteration 9 is optimal (In general, of course, there is no known way to verify optimality in polynomial time for difficult discrete optimization problems, i.e., those that fall in the class called NP-hard. The Min k -Tree problem is one of these).

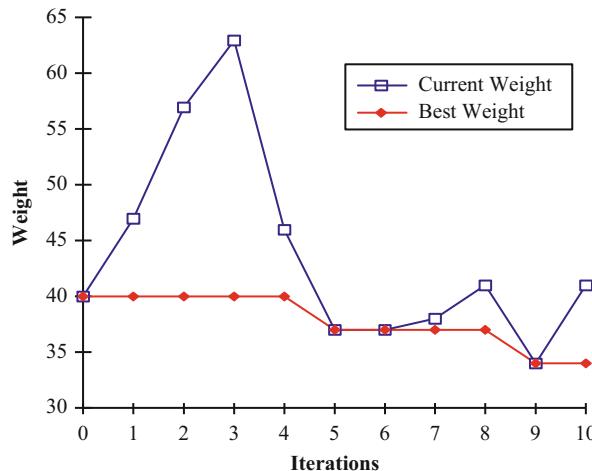
It may be noted that at iteration 6, the method selected a move with a move value of zero. Nevertheless, the configuration of the current solution changes after the execution of this move, as illustrated in Fig. 7.

Table 4 Iterations of a first-level TS procedure

Iteration	Tabu-active net tenure		Add	Drop	Move value	Weight
	1	2				
3	(6,8), (4,7)	(6,7)	(8,9)	(1,2)	6	63
4	(6,7), (8,9)	(1,2)	(4,7)	(1,4)	-17	46
5	(1,2), (4,7)	(1,4)	(6,7)	(4,6)	-9	37*
6	(1,4), (6,7)	(4,6)	(6,9)	(6,8)	0	37
7	(4,6), (6,9)	(6,8)	(8,10)	(4,7)	1	38
8	(6,8), (8,10)	(4,7)	(9,12)	(6,7)	3	41
9	(4,7), (9,12)	(6,7)	(10,11)	(6,9)	-7	34*
10	(6,7), (10,11)	(6,9)	(5,9)	(9,12)	7	41

**Fig. 7** Graphical representation of TS iterations

The selection of moves with certain move values, such as zero move values, may be strategically controlled to limit their selection as added insurance against cycling in special settings. We will soon see how considerations beyond this first-level implementation can lead to an improved search trajectory, but the non-monotonic,

Fig. 8 TS search trajectory

gradually improving, behavior is characteristic of TS in general. [Figure 8](#) provides a graphic illustration of this behavior for the current example.

We have purposely chosen the stopping iteration to be small to illustrate an additional relevant feature and to give a foundation for considering certain types of longer-term considerations. One natural way to apply TS is to periodically discontinue its progress, particularly if its rate of finding new best solutions falls below a preferred level, and to restart the method by a process designated to generate a new sequence of solutions.

Classical restarting procedures based on randomization evidently can be used for this purpose, but TS often derives an advantage by employing more strategic forms of restarting. We illustrate a simple instance of such a restarting procedure, which also serves to introduce a useful memory concept.

3.3.1 Critical Event Memory

Critical event memory in tabu search, as its name implies, monitors the occurrence of certain *critical events* during the search and establishes a memory that constitutes an aggregate summary of these events. For our current example, where we seek to generate a new starting solution, a critical event that is clearly relevant is the generation of the previous starting solution. Correspondingly, if we apply a restarting procedure multiple times, the steps of generating all preceding starting solutions naturally qualify as critical events. That is, we would prefer to depart from these solutions in some significant manner as we generate other starting solutions.

Different degrees of departure, representing different levels of *diversification*, can be achieved by defining solutions that correspond to critical events in different ways (and by activating critical event memory by different rules). In the present setting, we consider it important that new starting solutions not only differ from preceding starting solutions but that they also differ from other solutions generated during previous passes. One possibility is to use a blanket approach that considers each

complete solution previously generated to represent a critical event. The aggregation of such events by means of critical event memory makes this entirely practicable, but often, it is quite sufficient (and, sometimes preferable) to isolate a smaller set of solutions.

For the current example, therefore, we will specify that the critical events of interest consist of generating not only the starting solution of the previous pass(es) but also each subsequent solution that represents a “local TS optimum,” that is whose objective function value is better (or no worse) than that of the solution immediately before and after it. Using this simple definition, we see that four solutions qualify as critical (i.e., are generated by the indicated critical events) in the first solution pass of our example: the initial solution and the solutions found at iterations 5, 6, and 9 (with weights of 40, 37, 37, and 34, respectively).

Since the solution at iteration 9 happens to be optimal, we are interested in the effect of restarting before this solution is found. Assume we had chosen to restart after iteration 7, without yet reaching an optimal solution. Then the solutions that correspond to critical events are the initial solution and the solutions of iterations 5 and 6. We treat these three solutions in aggregate by combining their edges to create a subgraph that consists of the edges (1,2), (1,4), (4,7), (6,7), (6,8), (8,9), and (6,9) (Frequency-based memory, as discussed in Sect. 5, refines this representation by accounting for the number of times each edge appears in the critical solutions and allows the inclusion of additional weighting factors).

To execute a restarting procedure, we penalize the inclusion of the edges of this subgraph at various steps of constructing the new solution. It is usually preferable to apply this penalty process at early steps, implicitly allowing the penalty function to decay rapidly as the number of steps increases. It is also sometimes useful to allow one or more intervening steps after applying such penalties before applying them again.

For our illustration, we will use the memory embodied in the subgraph of penalized edges by introducing a large penalty that effectively excludes all these edges from consideration on the first two steps of constructing the new solution. Then, because the construction involves four steps in total, we will not activate the critical event memory on subsequent construction steps but will allow the method to proceed in its initial form.

Applying this approach, we restart the method by first choosing edge (3,5), which is the minimum weight edge not in the penalized subgraph. This choice and the remaining choices that generate the new starting solution are shown in Table 5.

Beginning from the solution constructed in Table 5, and applying the first-level TS procedure exactly as it was applied on the first pass, generates the sequence of solutions shown in Table 6 and depicted in Fig. 10 (Again, we have arbitrarily limited the total number of iterations, in this case to 5).

It is interesting to note that the restarting procedure generates a better solution (with a total weight of 38) than the initial solution generated during the first construction (with a total weight of 40). Also, the restarting solution contains 2 “optimal edges” (i.e., edges that appear in the optimal tree). This starting solution allows the search trajectory to find the optimal solution in only two iterations,

Table 5 Restarting procedure

Step	Candidates	Selection	Total weight
1	(3,5)	(3, 5)	6
2	(2,3), (3,4), (3,6), (5,6), (5,9), (5,12)	(5, 9)	22
3	(2,3), (3,4), (3,6), (5,6), (5,12), (6,9), (8,9), (9,12)	(8, 9)	29
4	(2,3), (3,4), (3,6), (5,6), (5,12), (6,8), (6,9), (7,8), (8,10), (9,12)	(8, 10)	38

Table 6 TS iterations following restarting

Iteration	Tabu-active net tenure		Add	Drop	Move value	Weight
	1	2				
1			(9,12)	(3,5)	3	41
2	(9,12)	(3,5)	(10,11)	(5,9)	-7	34*
3	(3,5), (10,11)	(5,9)	(6,8)	(9,12)	7	41
4	(5,9), (6,8)	(9,12)	(6,7)	(10,11)	-3	38
5	(9,12), (6,7)	(10,11)	(4,7)	(8,10)	-1	37

illustrating the benefits of applying a critical event memory within a restarting strategy. As will be seen in Sect. 5, related memory structures can also be valuable for strategies that drive the search into new regions by “partial restarting” or by directly continuing a current trajectory (with modified decision rules).

Now we return from our example to examine elements of TS that take us beyond these first-level concerns and open up possibilities for creating more powerful solution approaches. We continue to focus primarily on short-term aspects and begin by discussing how to generalize the use of recency-based memory when neighborhood exploration is based on add/drop moves. From these foundations, we then discuss issues of logical restructuring, tabu-activation rules, and ways of determining tabu tenure. We then examine the important area of aspiration criteria, together with the role of influence.

3.4 Recency-Based Memory for Add/Drop Moves

To understand procedurally how various forms of recency-based memory work, and to see their interconnections, it is useful to examine a convenient design for implementing the ideas illustrated so far. Such a design for the Min k -Tree problem creates a natural basis for handling a variety of other problems for which add/drop moves are relevant. In addition, the ideas can be adapted to settings that are quite different from those where add/drop moves are used.

As a step toward fuller generality, we will refer to items added and dropped as *elements*, though we will continue to make explicit reference to edges (as particular types of elements) within the context of the Min k -Tree problem example (Elements

are related to, but not quite the same as, solution attributes. The difference will be made apparent shortly). There are many settings where operations of adding and dropping paired elements are the cornerstone of useful neighborhood definitions. For example, many types of exchange or swap moves can be characterized by such operations. Add/drop moves also apply to the omnipresent class of *multiple choice* problems, which require that exactly one element must be chosen from each member set from a specified disjoint collection. Add/drop moves are quite natural in this setting, since whenever a new element is chosen from a given set (and hence is “added” to the current solution), the element previously chosen from that set must be replaced (and hence “dropped”). Such problems are represented by discrete *generalized upper bound* (GUB) formulations in mathematical optimization, where various disjoint sets of 0-1 variables must sum to 1 (hence exactly one variable from each set must equal 1, and the others must equal 0). An add/drop move in this formulation consists of choosing a new variable to equal 1 (the “add move”) and setting the associated (previously selected) variable equal to 0 (the “drop move”).

Add/drop moves further apply to many types of problems that are not strictly discrete, that is, which contain variables whose values can vary continuously across specified ranges. Such applications arise by taking advantage of *basis exchange* (pivoting) procedures, such as the simplex method of linear programming. In this case, an add/drop move consists of selecting a new variable to enter (add to) the basis and identifying an associated variable to leave (drop from) the basis. A variety of procedures for nonlinear and mixed-integer optimization rely on such moves and have provided a useful foundation for a number of tabu search applications. Additional related examples will be encountered throughout the course of this book.

3.4.1 Some Useful Notation

The approach used in the Min k -Tree problem can be conveniently described by means of the following notation. For a pair of elements that is selected to perform an add/drop move, let *Added* denote the element that is added and *Dropped* the element that is dropped. Also denote the current iteration at which this pair is selected by *Iter*. We maintain a record of *Iter* to identify when *Added* and *Dropped* start to be tabu-active. Specifically, at this step we set

$$\text{TabuDropStart}(\text{Added}) = \text{Iter}$$

$$\text{TabuAddStart}(\text{Dropped}) = \text{Iter}.$$

Thus, *TabuDropStart* records the iteration where *Added* becomes tabu-active (to prevent this element from later being dropped), and *TabuAddStart* records the iteration where *Dropped* becomes tabu-active (to prevent this element from later being added).

For example, in the Min k -Tree problem illustration of [Table 4](#), where the edge (4,6) was added and the edge (4,7) was dropped on the first iteration, we would establish the record (for $\text{Iter} = 1$)

$$\text{TabuDropStart}(4, 6) = 1$$

$$\text{TabuAddStart}(4, 7) = 1.$$

To identify whether or not an element is currently tabu-active, let *TabuDropTenure* denote the tabu tenure (number of iterations) to forbid an element to be dropped (once added), and let *TabuAddTenure* denote the tabu tenure to forbid an element from being added (once dropped) (In our Min k -Tree problem example of Sect. 3.2, we selected *TabuAddTenure*=2 and *TabuDropTenure*=1).

As a point of clarification, when we speak of an element as being tabu-active, our terminology implicitly treats elements and attributes as if they are the same. However, to be precise, each element is associated with two different attributes, one where the element belongs to the current solution and one where the element does not. Elements may be viewed as corresponding to variables and attributes as corresponding to specific value assignments for such variables. There is no danger of confusion in the add/drop setting, because we always know when an element belongs or does not belong to the current solution, and hence, we know which of the two associated attributes is currently being considered.

We can now identify precisely the set of iterations during which an element (i.e., its associated attribute) will be tabu-active. Let *TestAdd* and *TestDrop* denote a candidate pair of elements, whose members are respectively under consideration to be added and dropped from the current solution. If *TestAdd* previously corresponded to an element *Dropped* that was dropped from the solution and *TestDrop* previously corresponded to an element *Added* that was added to the solution (not necessarily on the same step), then it is possible that one or both may be tabu-active and we can check their status as follows. By means of the records established on earlier iterations, where *TestAdd* began to be tabu-active at iteration *TabuAddStart* (*TestAdd*) and *TestDrop* began to be tabu-active at iteration *TabuDropStart* (*TestDrop*), we conclude that as *Iter* grows, the status of these elements will be given by

TestAdd is tabu-active when:

$$\text{Iter} \leq \text{TabuAddStart}(\text{TestAdd}) + \text{TabuAddTenure}$$

TestDrop is tabu-active when:

$$\text{Iter} \leq \text{TabuDropStart}(\text{TestDrop}) + \text{TabuDropTenure}.$$

Consider again the Min k -Tree problem illustration of Table 4. As previously noted, the move of Iteration 1 that added edge (4,6) and dropped edge (4,7) was accompanied by setting the *TabuDropStart*(4,6)=1 and *TabuAddStart*(4,7)=1 to record the iteration where these two edges start to be tabu-active (to prevent (4,6) from being dropped and (4,7) from being added). The edge (4,6) will then remain tabu-active on subsequent iterations, in the role of *TestDrop* (as a candidate to be dropped), as long as

$$\text{Iter} \leq \text{TabuDropStart} (4, 6) + \text{TabuDropTenure}.$$

Hence, since we selected $\text{TabuDropTenure} = 1$ (to prevent an added edge from being dropped for 1 iteration), it follows that (4,6) remains tabu-active as long as

$$\text{Iter} \leq 2.$$

Similarly, having selected $\text{TabuAddTenure} = 2$, we see that the edge (4,7) remains tabu-active, to forbid it from being added back, as long as

$$\text{Iter} \leq 3.$$

An initialization step is needed to be sure that elements that have never been previously added or dropped from the solutions successively generated will not be considered tabu-active. This can be done by initially setting TabuAddStart and TabuDropStart equal to a large negative number for all elements. Then, as Iter begins at 1 and successively increases, the inequalities that determine the tabu-active status will not be satisfied, and hence will correctly disclose that an element is not tabu-active, until it becomes one of the elements *Added* or *Dropped* (Alternately, TabuAddStart and TabuDropStart can be initialized at 0, and the test of whether an element is tabu-active can be skipped when it has a 0 value in the associated array).

3.4.2 Streamlining

The preceding ideas can be streamlined to allow a more convenient implementation. First, we observe that the two arrays, TabuAddStart and TabuDropStart , which we have maintained separately from each other to emphasize their different functions, can be combined into a single array TabuStart . The reason is simply that we can interpret $\text{TabuStart}(E)$ to be the same as $\text{TabuDropStart}(E)$ when the element E is in the current solution and to be the same as $\text{TabuAddStart}(E)$ when E is not in the current solution (There is no possible overlap between these two states of E and hence no danger of using the TabuStart array incorrectly). Consequently, from now on, we will let the single array TabuStart take the role of both TabuAddStart and TabuDropStart . For example, when the move is executed that (respectively) adds and drops the elements *Added* and *Dropped*, the appropriate record consists of setting

$$\text{TabuStart}(\text{Added}) = \text{Iter}$$

$$\text{TabuStart}(\text{Dropped}) = \text{Iter}.$$

The TabuStart array has an additional function beyond that of monitoring the status of tabu-active elements (As shown in Sect. 5, this array is also useful for determining a type of frequency measure called a *residence frequency*). However, sometimes it is convenient to use a different array, TabuEnd , to keep track of tabu-active status for recency-based memory, as we are treating here. Instead of recording

when the tabu-active status starts, *TabuEnd* records when it ends. Thus, in place of the two assignments to *TabuStart* shown above, the record would consist of setting

$$\text{TabuEnd}(\text{Added}) = \text{Iter} + \text{TabuDropTenure}$$

$$\text{TabuEnd}(\text{Dropped}) = \text{Iter} + \text{TabuAddTenure}.$$

(The element *Added* is now available to be dropped, and the element *Dropped* is now available to be added). In conjunction with this, the step that checks for whether a candidate pair of elements *TestAdd* and *TestDrop* are currently tabu-active becomes:

TestAdd is tabu-active when

$$\text{Iter} \leq \text{TabuEnd}(\text{TestAdd}).$$

TestDrop is tabu-active when

$$\text{Iter} \leq \text{TabuEnd}(\text{TestDrop}).$$

This is a simpler representation than the one using *TabuStart*, and so, it is appealing when *TabuStart* is not also used for additional purposes (Also, *TabuEnd* can simply be initialized at 0 rather than at a large negative number).

As will be discussed more fully in the next section, the values of *TabuAddTenure* and *TabuDropTenure* (which are explicitly referenced in testing tabu-active status with *TabuStart* and implicitly referenced in testing this status with *TabuEnd*) are often preferably made variable rather than fixed. The fact that we use different tenures for added and dropped elements discloses that it can be useful to differentiate the tenures applied to elements of different classes. This type of differentiation can also be based on historical performance, as tracked by frequency-based measures. Consequently, tenures may be individually adjusted for different elements (as well as modified over time). Such adjustment can be quite effective in some settings. These basic considerations can be refined to create effective implementations and also can be extended to handle additional move structures, as shown in Glover and Laguna [31].

3.5 Tabu Tenure

In general, recency-based memory is managed by creating one or several tabu lists, which record the tabu-active attributes and implicitly or explicitly identify their current status. Tabu tenure can vary for different types or combinations of attributes and can also vary over different intervals of time or stages of the search. This varying tenure makes it possible to create different kinds of trade-offs between short-term and longer-term strategies. It also provides a dynamic and robust form of search.

The choice of appropriate types of tabu lists depends on the context. Although no single type of list is uniformly best for all applications, some guidelines can be formulated. If memory space is sufficient (as it often is) to store one piece of information (e.g., a single integer) for each solution attribute used to define the tabu-activation rule, it is usually advantageous to record the iteration number that identifies when the tabu-active status of an attribute starts or ends as illustrated by the add/drop data structure described in Sects. 3.3 and 3.4. This typically makes it possible to test the tabu status of a move in constant time. The necessary memory space depends on the attributes and neighborhood size, but it does not depend on the tabu tenure.

Depending on the size of the problem, it may not be feasible to implement the preceding memory structure in combination with certain types of attributes. In general, storing one piece of information for each attribute becomes unattractive when the problem size increases or attribute definition is complex. Sequential and circular tabu lists are used in this case, which store the identities of each tabu-active attribute and explicitly (or implicitly, by list position) record associated tabu tenures.

Effective tabu tenures have been empirically shown to depend on the size of the problem instance. However, no single rule has been designed to yield an effective tenure for all classes of problems. This is partly because an appropriate tabu tenure depends on the strength of the tabu-activation rule employed (where more restrictive rules are generally coupled with shorter tenures). Effective tabu tenures and tabu-activation rules can usually be determined quite easily for a given class of problems by a little experimentation. Tabu tenures that are too small can be recognized by periodically repeated objective function values or other function indicators, including those generated by hashing, that suggest the occurrence of cycling. Tenures that are too large can be recognized by a resulting deterioration in the quality of the solutions found (within reasonable time periods). Somewhere in between typically exists a robust range of tenures that provide good performance.

Once a good range of tenure values is located, first-level improvements generally result by selecting different values from this range on different iterations (A smaller subrange, or even more than one subrange, may be chosen for this purpose). Problem structures are sometimes encountered where performance for some individual fixed tenure values within a range can be unpredictably worse than for other values in the range, and the identity of the isolated poorer values can change from problem to problem. However, if the range is selected to be good overall, then a strategy that selects different tenure values from the range on different iterations typically performs at a level comparable to selecting one of the best values in the range, regardless of the problem instance.

Short-term memory refinements subsequently discussed, and longer-term considerations introduced in later sections, transform the method based on these constructions into one with considerable power. Still, it occasionally happens that even the initial short-term approach by itself leads to exceptionally high-quality solutions. Consequently, some of the TS literature has restricted itself only to this initial part of the method.

In general, short tabu tenures allow the exploration of solutions “close” to a local optimum, while long tenures can help to break free from the vicinity of a local optimum. These functions illustrate a special instance of the notions of *intensification* and *diversification* that will be explored in more detail later. Varying the tabu tenure during the search provides one way to induce a balance between closely examining one region and moving to different parts of the solution space.

In situations where a neighborhood may (periodically) become fairly small, or where a tabu tenure is chosen to be fairly large, it is entirely possible that iterations can occur when all available moves are classified tabu. In this case, an *aspiration-by-default* is used to allow a move with a “least tabu” status to be considered admissible. Such situations rarely occur for most problems, and even random selection is often an acceptable form of aspiration-by-default. When tabu status is translated into a modified evaluation criterion, by penalties and inducements, then of course aspiration-by-default is handled automatically, with no need to monitor the possibility that all moves are tabu.

There are several ways in which a *dynamic* tabu tenure can be implemented. These implementations may be classified into *random* and *systematic* dynamic tabu tenures.

3.5.1 Random Dynamic Tenure

Random dynamic tabu tenures are often given one of two forms. Both of these forms use a tenure range defined by parameters t_{\min} and t_{\max} . The tabu tenure t is randomly selected within this range, usually following a uniform distribution. In the first case, the chosen tenure is maintained constant for αt_{\max} iterations, and then a new tenure is selected by the same process. The second form draws a new t for every attribute that becomes tabu at a given iteration. The first form requires more bookkeeping than the second one, because one must remember the last time that the tabu tenure was modified.

Either of the two arrays *TabuStart* or *TabuEnd* discussed in Sect. 3.4 can be used to implement these forms of dynamic tabu tenure. For example, a 2-dimensional array *TabuEnd* can be created to control a dynamic recency-based memory for the sequencing problem introduced at the beginning of this section. As in the case of the Min k -Tree problem, such an array can be used to record the time (iteration number) at which a particular attribute will be released from its tabu status. Suppose, for example, that $t_{\min} = 5$ and $t_{\max} = 10$ and that swaps of jobs are used to move from one solution to another in the sequencing problem. Also, assume that *TabuEnd* (j, p) refers to the iteration that job j will be released from a tabu restriction that prevents it from being assigned to position p . Then, if at iteration 30, job 8 in position 2 is swapped with job 12 in position 25, we will want to make the attribute (8,2) and (12,25) tabu-active for some number of iterations to prevent a move that will return one or both of jobs 8 and 12 from reoccupying their preceding positions. If t is assigned a value of 7 from the range $t_{\min} = 5$ and $t_{\max} = 10$, then upon making the swap at iteration 30, we may set *TabuEnd* (8,2)=37 and *TabuEnd* (12,25)=37.

This is not the only kind of *TabuEnd* array that can be used for the sequencing problem, and we examine other alternatives and their implications in Sect. 4.

Nevertheless, we warn against a potential danger. An array $\text{TabuEnd}(i, j)$ that seeks to prevent jobs i and j from exchanging positions, without specifying what these positions are, does not truly refer to attributes of a sequencing solution and hence entails a risk if used to determine tabu status (The pair (i, j) here constitutes an attribute of a move, in a lose sense, but does not serve to distinguish one solution from another). Thus, if at iteration 30, we were to set $\text{TabuEnd}(8,12)=37$, in order to prevent jobs 8 and 12 from exchanging positions until after iteration 37, this still might not prevent job 8 from returning to position 2 and job 12 from returning to position 25. In fact, a sequence of swaps could be executed that could return to precisely the same solution visited before swapping jobs 8 and 12.

Evidently, the TabuEnd array can be used by selecting a different t from the interval (t_{\min}, t_{\max}) at every iteration. As remarked in the case of the Min k -Tree problem, it is also possible to select t differently for different solution attributes.

3.5.2 Systematic Dynamic Tenure

Dynamic tabu tenures based on a random scheme are attractive for their ease of implementation. However, relying on randomization may not be the best strategy when specific information about the context is available. In addition, certain diversity-inducing patterns can be achieved more effectively by not restricting consideration to random designs. A simple form of *systematic* dynamic tabu tenure consists of creating a sequence of tabu search tenure values in the range defined by t_{\min} and t_{\max} . This sequence is then used, instead of the uniform distribution, to assign the current tabu tenure value. Suppose it is desired to vary t so that its value alternately increases and decreases (Such a pattern induces a form of diversity that will rarely be achieved randomly). Then the following sequence can be used for the range defined above:

$$\{5, 8, 6, 9, 7, 10\}.$$

The sequence may be repeated as many times as necessary until the end of the search, where additional variation is introduced by progressively shifting and/or reversing the sequence before repeating it (In a combined random/systematic approach, the decision of the shift value and the forward or backward direction can itself be made random). Another variation is to retain a selected tenure value from the sequence for a variable number of iterations before selecting the next value. Different sequences can be created and identified as effective for particular classes of problems.

The foregoing range of values (from 5 to 10) may seem relatively small. However, some applications use even smaller ranges but adaptively increase and decrease the midpoint of the range for diversification and intensification purposes. Well-designed adaptive systems can significantly reduce or even eliminate the need to discover a best range of tenures by preliminary calibration. This is an important area of study.

These basic alternatives typically provide good starting tabu search implementations. In fact, most initial implementations apply only the simplest versions of these ideas.

3.6 Aspiration Criteria and Regional Dependencies

Aspiration criteria are introduced in tabu search to determine when tabu-activation rules can be overridden, thus removing a tabu classification otherwise applied to a move (The improved-best and aspiration-by-default criteria, as previously mentioned, are obvious simple instances). The appropriate use of such criteria can be very important for enabling a TS method to achieve its best performance levels. Early applications employed only a simple type of aspiration criterion, consisting of removing a tabu classification from a trial move when the move yields a solution better than the best obtained so far. This criterion remains widely used. However, other aspiration criteria can prove effective for improving the search.

A basis for one of these criteria arises by introducing the concept of *influence*, which measures the degree of change induced in solution structure or feasibility. This notion can be illustrated for the Min k -Tree problem as follows. Suppose that the current solution includes edges (1,2), (1,4), (4,7), and (6,7), as illustrated in Fig. 9, following. A high-influence move, which significantly changes the structure of the current solution, is exemplified by dropping edge (1,2) and replacing it by edge (6,9). A low-influence move, on the other hand, is exemplified by dropping edge (6,7) and adding edge (4,6). The weight difference of the edges in the high-influence move is 15, while the difference is 9 for the low-influence move. However, it is important to point out that differences on weight or cost are not the only – or even the primary – basis for distinguishing between moves of high and low influence. In the present example, the move we identify as a low-influence move creates a solution that consists of the same set of nodes included in the current solution, while the move we identified as a high-influence move includes a new

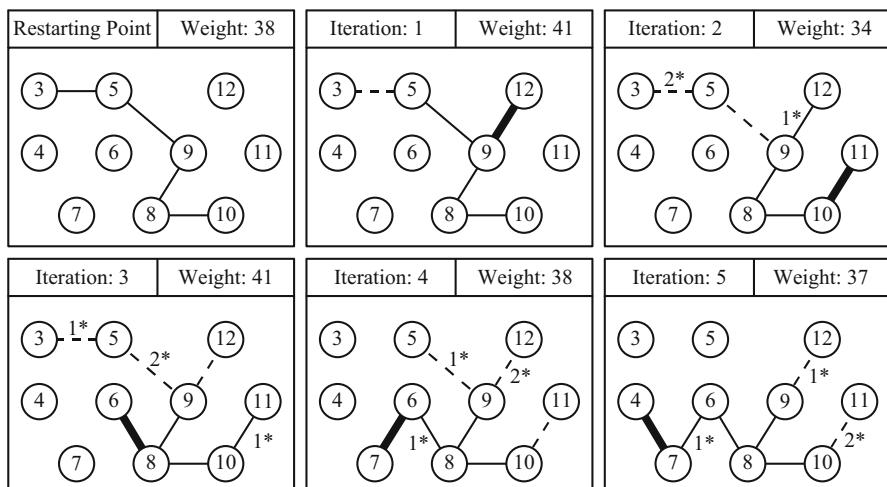


Fig. 9 Graphical representation of TS iterations after restarting

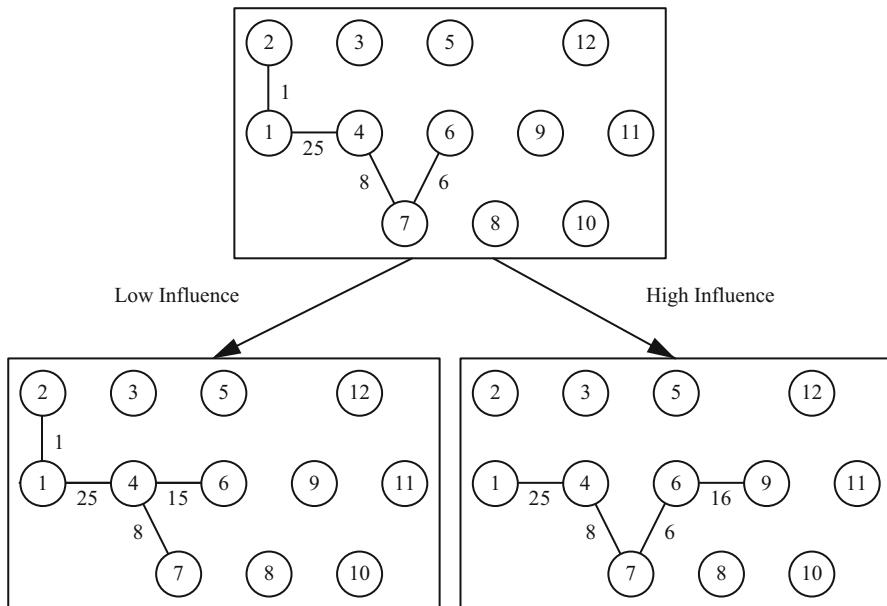


Fig. 10 Influence level of two moves

node (number 9) from which new edges can be examined (These moves correspond to those labeled static and dynamic in Fig. 5).

As illustrated here, high-influence moves may or may not improve the current solution, though they are less likely to yield an improvement when the current solution is relatively good. But high-influence moves are important, especially during intervals of breaking away from local optimality, because a series of moves that is confined to making only small structural change is unlikely to uncover a chance for significant improvement. Executing the high-influence move in Fig. 10, for example, allows the search to reach the optimal edges (8,9) and (9,12) in subsequent iterations. Of course, moves of much greater influence than those shown can be constructed by considering compound moves. Such considerations are treated in later sections.

Influence often is associated with the idea of move distance. Although important, move influence is only one of several elements that commonly underlie the determination of aspiration criteria. We illustrate a few of these elements in Table 7.

Aspirations such as those shown in Table 7 can be applied according to two implementation categories: aspiration by move and aspirations by attribute. A move aspiration, when satisfied, revokes the move's tabu classification. An attribute aspiration, when satisfied, revokes the attribute's tabu-active status. In the latter case, the move may or may not change its tabu classification, depending on whether the tabu-activation rule is triggered by more than one attribute. For example, in our sequencing problem, if the swap of jobs 3 and 6 is forbidden because a

Table 7 Illustrative aspiration criteria

Aspiration by	Description	Example
<i>Default</i>	If all available moves are classified tabu and are not rendered admissible by some other aspiration criteria, then a “least tabu” move is selected	Revoke the tabu status of all moves with minimum <i>TabuEnd</i> value
<i>Objective</i>	<i>Global</i> : A move aspiration is satisfied if the move yields a solution better than the best obtained so far <i>Regional</i> : A move aspiration is satisfied if the move yields a solution better than the best found in the region where the solution lies	<i>Global</i> : The best total tardiness found so far is 29. The current sequence is (4, 1, 5, 3, 6, 2) with $T = 39$. The move value of the tabu swap (5,2) is -20. Then, the tabu status of the swap is revoked and the search moves to the new best sequence (4, 1, 2, 3, 6, 5) with $T = 19$ <i>Regional</i> : The best sequence found in the region defined by all sequences (1, 2, 3, *, *, *) is (1, 2, 3, 6, 4, 5) with $T = 31$. The current solution is (1, 4, 3, 2, 6, 5) with $T = 23$. The swap (4, 2) with move value of 6 is tabu. The tabu status is revoked because a new regional best (1, 2, 3, 4, 6, 5) with $T = 29$ can be found
<i>Search direction</i>	An attribute can be added and dropped from a solution (regardless of its tabu status) if the direction of the search (improving or nonimproving) has not changed	For the Min k -Tree problem, the edge (11,12) has been recently dropped in the current improving phase making its addition a tabu-active attribute. The improving phase can continue if edge (11,12) is now added; therefore its tabu status may be revoked
<i>Influence</i>	The tabu status of a low-influence move may be revoked if a high-influence move has been performed since establishing the tabu status for the low-influence move	If the low-influence swap (1,4) described in Table 2.7 is classified tabu, its tabu status can be revoked after the high-influence swap (4,5) is performed

tabu-activation rule prevents job 3 from moving at all, then an attribute aspiration that revokes job 3's tabu-active status also revokes the move's tabu classification. However, if the swap (3,6) is classified tabu because both job 3 and job 6 are not allowed to move, then revoking job 3's tabu-active status does not result in overriding the tabu status of the entire move.

Different variants of the aspiration criteria presented in Table 7 are possible. For example, the regional aspiration by objective can be defined in terms of bounds on

the objective function value. These bounds determine the region being explored, and they are modified to reflect the discovery of better (or worse) regions. Another possibility is to define regions with respect to time. For example, one may record the best solution found during the recent past (defined as a number of iterations) and use this value as the aspiration level.

3.7 Concluding Observations for the Min k-Tree Example

Influence of Tabu Tenures

The tabu tenures used to illustrate the first-level TS approach for the Min k -Tree problem of course are very small. The risk of using such tenures can be demonstrated in this example from the fact that changing the weight of edge (3,6) in Fig. 4 from 20 to 17 will cause the illustrated TS approach with $\text{TabuAddTenure} = 2$ and $\text{TabuDropTenure} = 1$ to go into a cycle that will prevent the optimal solution from being found. The intuition that TabuDropTenure has a stronger influence than the TabuAddTenure for this problem is supported by the fact that the use of tenures of $\text{TabuAddTenure} = 1$ and $\text{TabuDropTenure} = 2$ in this case will avoid the cycling problem and allow an optimal solution to be found.

Alternative Neighborhoods

The relevance of considering alternative neighborhoods can be illustrated by reference to the following observation. For any given set of $k + 1$ nodes, an optimal (min weight) k -tree over these nodes can always be found by using the greedy constructive procedure illustrated in Table 2 to generate a starting solution (restricted to these nodes) or by beginning with an arbitrary tree on these nodes and performing a succession of static improving moves (which do not change the node set). The absence of a static-improving move signals that no better solution can be found on this set.

This suggests that tabu search might advantageously be used to guide the search over a “node-swap” neighborhood instead of an “edge-swap” neighborhood, where each move consists of adding a non-tree node i and dropping a tree node j , followed by finding a min weight solution on the resulting node set (Since the tree node j may not be a leaf node, and the reconnections may also not make node i a leaf node in the new tree, the possibilities are somewhat different than making a dynamic move in the edge-swap neighborhood). The tabu tenures may reasonably be defined over nodes added and dropped, rather than over edges added and dropped.

Critical Event Memory

The type of critical event memory used in the illustration of restarting the TS approach in Sect. 3.3.1 may not be best. Generally, it is reasonable to expect that the type of critical event memory used for restarting should be different from that used to continue the search from the current solution (when both are applied to drive the search into new regions). Nevertheless, a form that is popularly used in both situations consists of remembering *all* elements contained in solutions previously examined. One reason is that it is actually easier to maintain such memory than to keep track of elements that only occur in selected solutions. Also, instead of

keeping track only of which elements occur in past solution, critical event memory is more usually designed to monitor the frequency that elements have appeared in past solutions. Such considerations are amplified in [Sect. 5](#).

4 Additional Aspects of Short-Term Memory

We began the discussion of short-term memory for tabu search by contrasting the TS designs with those of memoryless strategies such as simple or iterated descent and by pointing out how candidate list strategies are especially important for applying TS in the most effective ways. We now describe types of candidate list strategies that often prove valuable in tabu search implementations. Then we examine the issues of logical restructuring, which provide important bridges to longer-term considerations.

4.1 Tabu Search and Candidate List Strategies

The aggressive aspect of TS is manifest in choice rules that seek the best available move that can be determined with an appropriate amount of effort. As addressed in [Sect. 3](#), the meaning of best in TS applications is customarily not limited to an objective function evaluation. Even where the objective function evaluation may appear on the surface to be the only reasonable criterion to determine the best move, the non-tabu move that yields a maximum improvement or least deterioration is not always the one that should be chosen. Rather, as we have noted, the definition of best should consider factors such as move influence, determined by the search history and the problem context.

For situations where $N^*(x)$ is large or its elements are expensive to evaluate, candidate list strategies are essential to restrict the number of solutions examined on a given iteration. In many practical settings, TS is used to control a search process that may involve the solution of relatively complex subproblems by way of linear programming or simulation. Because of the importance TS attaches to selecting elements judiciously, efficient rules for generating and evaluating good candidates are critical to the search process. The purpose of these values is to isolate regions of the neighborhood containing moves with desirable features and to put these moves on a list of candidates for current examination.

Before describing the kinds of candidate list strategies that are particularly useful in tabu search implementations, we note that the efficiency of implementing such strategies often can be enhanced by using relatively straightforward memory structures to give efficient updates of move evaluations from one iteration to another. Appropriately coordinated, such updates can appreciably reduce the effort of finding best or near-best moves.

In sequencing, for example, the move values often can be calculated without a full evaluation of the objective function. Intelligent updating can be useful even where candidate list strategies are not used. However, the inclusion of explicit

candidate list strategies, for problems that are large, can significantly magnify the resulting benefits. Not only search speed but also solution quality can be influenced by the use of appropriate candidate list strategies. Perhaps surprisingly, the importance of such approaches is often overlooked.

4.2 Some General Classes of Candidate List Strategies

Candidate lists can be constructed from context-related rules and from general strategies. In this section, we focus on rules for constructing candidate lists that are context-independent. We emphasize that the effectiveness of a candidate list strategy should not be measured in terms of the reduction of the computational effort in a single iteration. Instead, a preferable measure of performance for a given candidate list is the quality of the best solution found given a specified amount of computer time. For example, a candidate list strategy intended to replace an exhaustive neighborhood examination may result in more iterations per unit of time but may require many more iterations to match the solution quality of the original method. If the quality of the best solution found within a desirable time limit (or across a graduated series of such limits) does not improve, we conclude that the candidate list strategy is not effective.

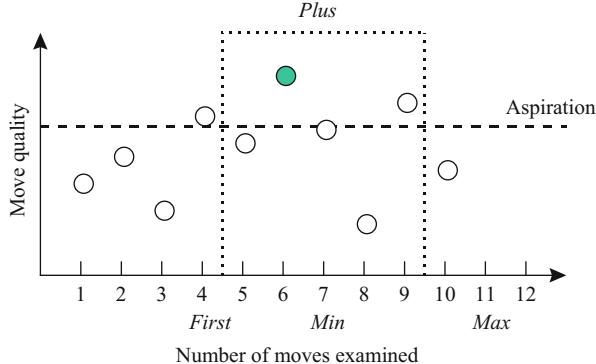
4.2.1 Aspiration Plus

The Aspiration Plus strategy establishes a threshold for the quality of a move, based on the history of the search pattern. The procedure operates by examining moves until finding one that satisfies this threshold. Upon reaching this point, additional moves are examined, equal in number to the selected value *Plus*, and the best move overall is selected.

To assure that neither too few nor too many moves are considered, this rule is qualified to require that at least *Min* moves and at most *Max* moves are examined, for chosen values of *Min* and *Max*. The interpretation of *Min* and *Max* is as follows. Let *First* denote the number of moves examined when the aspiration threshold is first satisfied. Then if *Min* and *Max* were not specified, the total number of moves examined would be *First* + *Plus*. However, if *First* + *Plus* < *Min*, then *Min* moves are examined while if *First* + *Plus* > *Max*, then *Max* moves are examined (This conditions may be viewed as imposing limits on the move that is “effectively” treated as the *First* move. For example, if as many as *Max* – *Plus* moves are examined without finding one that satisfies the aspiration threshold, then *First* effectively becomes the same as *Max* – *Plus*).

This strategy is graphically represented in Fig. 11. In this illustration, the fourth move examined satisfies the aspiration threshold and qualifies as *First*. The value of *Plus* has been selected to be 5, and so 9 moves are examined in total, selecting the best over this interval. The value of *Min*, set at 7, indicates that at least 7 moves will be examined even if *First* is so small that *First* + *Plus* < 7 (In this case, *Min* is not very restrictive, because it only applies if *First* < 2). Similarly, the value of *Max*, set at 11, indicates that at most 11 moves will be examined even if *First* is so large

Fig. 11 Aspiration Plus strategy



that $\text{First} + \text{Plus} > 11$ (Here, Max is strongly restrictive). The sixth move examined is the best found in this illustration.

The “Aspiration” line in this approach is an established threshold that can be dynamically adjusted during the search. For example, during a sequence of improving moves, the aspiration may specify that the next move chosen should likewise be improving, at a level based on other recent moves and the current objective function value. Similarly, the values of Min and Max can be modified as a function of the number of moves required to meet the threshold.

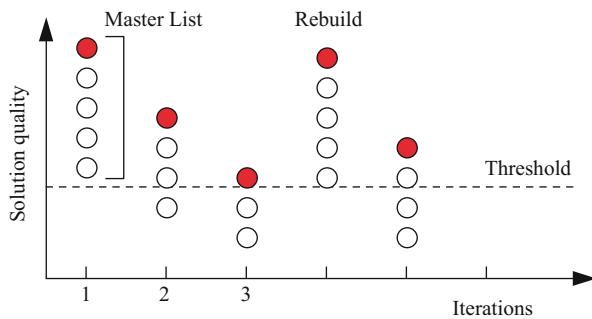
During a nonimproving sequence, the aspiration of the Aspiration Plus rule will typically be lower than during an improving phase but rise toward the improving level as the sequence lengthens. The quality of currently examined moves can shift the threshold, as by encountering moves that significantly surpass or that uniformly fall below the threshold. As an elementary option, the threshold can simply be a function of the quality of the initial Min moves examined on the current iteration.

The Aspiration Plus strategy includes several other strategies as special cases. For example, a *first-improving* strategy results by setting $\text{Plus} = 0$ and directing the aspiration threshold to accept moves that qualify as improving, while ignoring the values of Min and Max. Then *First* corresponds to the first move that improves the current value of the objective, if such a move can be found. A slightly more advanced strategy can allow *Plus* to be increased or decreased according to the variance in the quality of moves encountered from among some initial number examined. In general, in applying the Aspiration Plus strategy, it is important to assure on each iteration that new moves are examined which differ from those just reviewed. One way of achieving this is to create a circular list and start each new iteration where the previous examination left off.

4.2.2 Elite Candidate List

The elite candidate list approach first builds a Master List by examining all (or a relatively large number of) moves, selecting the k best moves encountered, where k is a parameter of the process. Then at each subsequent iteration, the current best move from the Master List is chosen to be executed, continuing until such a move

Fig. 12 Elite candidate list strategy



falls below a given quality threshold or until a given number of iterations have elapsed. Then a new Master List is constructed and the process repeats. This strategy is depicted in Fig. 12 below.

This technique is motivated by the assumption that a good move, if not performed at the present iteration, will still be a good move for some number of iterations. More precisely, after an iteration is performed, the nature of a recorded move implicitly may be transformed. The assumption is that a useful proportion of these transformed moves will inherit attractive properties from their antecedents.

The evaluation and precise identity of a given move on the list must be appropriately monitored, since one or both may change as result of executing other moves from the list. For example, in the Min k -Tree problem, the evaluations of many moves can remain unchanged from one iteration to the next. However, the identity and evaluation of specific moves will change as a result of deleting and adding particular edges, and these changes should be accounted for by appropriate updating (applied periodically if not at each iteration). An elite candidate list strategy can be advantageously extended by a variant of the Aspiration Plus strategy, allowing some additional number of moves outside the Master List to be examined at each iteration, where those of sufficiently high quality may replace elements of the Master List.

4.2.3 Successive Filter Strategy

Moves can often be broken into component operations, and the set of moves examined can be reduced by restricting consideration to those that yield high-quality outcomes for each operation separately. For example, the choice of an exchange move that includes an “add component” and a “drop component” may restrict attention only to exchanges created from a relatively small subset of “best add” and “best drop” components. The gain in efficiency can be considerable. If there are 100 add possibilities and 100 drop possibilities, the number of add/drop combinations is 10,000. However, by restricting attention to the 8 best add and drop moves, considered independently, the number of combinations to examine is only 64 (Values of 8 and even smaller have been found effective in some practical applications).

The evaluations of the separate components often will give only approximate information about their combined evaluation. Nevertheless, if this information is good enough to insure a significant number of the best complete moves will result by combining these apparently best components, then the approach can yield quite good outcomes. Improved information may be obtained by sequential evaluations, as where the evaluation of one component is conditional upon the prior (restricted) choices of another. Such strategies of subdividing compound moves into components, and then restricting consideration of complete compound moves only to those assembled from components that pass selected thresholds of quality, have proved quite effective in TS methods for partitioning problems and for telecommunication channel-balancing problems.

Conditional uses of component evaluations are also relevant for sequencing problems, where a measure can be defined to identify preferred attributes using information such as due dates, processing times, and delay penalties. If swap moves are being used, then some jobs are generally better candidates than others to move early or later in the sequence. The candidate list considers those swaps whose composition includes at least one of these preferred attributes.

In the context of the traveling salesman problem, good solutions are often primarily composed of edges that are among the 20–40 shortest edges meeting one of their endpoints (depending on various factors). Some studies have attempted to limit consideration entirely to tours constructed from such a collection of edges. The successive filter strategy, by contrast, offers greater flexibility by organizing moves that do not have to be entirely composed of such special elements, provided one or more of these elements is incorporated as part of the move. This approach can be frequently controlled to require little more time than the more restricted standard approach, while affording a more desirable set of alternatives to consider.

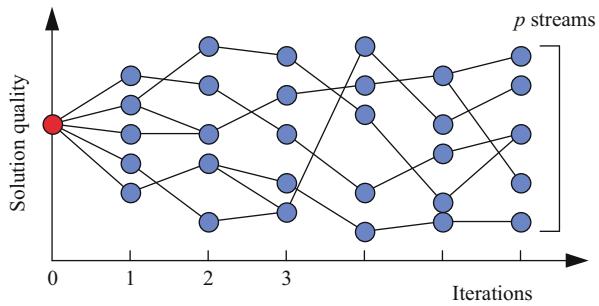
4.2.4 Sequential Fan Candidate List

A type of candidate list that is highly exploitable by parallel processing is the sequential fan candidate list. The basic idea is to generate some p best alternative moves at a given step and then to create a fan of solution streams, one for each alternative. The several best available moves for each stream are again examined, and only the p best moves overall (where many or no moves may be contributed by a given stream) provide the p new streams at the next step.

In the setting of tree search methods, such a sequential fanning process is sometimes called *beam search*. For use in the tabu search framework, TS memory and activation rules can be carried forward with each stream and hence inherited in the selected continuations. Since a chosen solution can be assigned to more than one stream, different streams can embody different missions in TS. Alternatively, when two streams merge into the same solution, other streams may be started by selecting a neighbor adjacent to one of the current streams.

The process is graphically represented in Fig. 13. Iteration 0 constructs an initial solution or alternatively may be viewed as the starting point for constructing a solution. That is, the sequential fan approach can be applied using one type of move to create a set of initial solutions and then can continue using another type

Fig. 13 Sequential fan candidate lis



of move to generate additional solutions (We thus allow a “solution” to be a partial solution as well as a complete solution). The best moves from this solution are used to generate p streams. Then at every subsequent iteration, the overall best moves are selected to lead the search to p different solutions. Note that since more than one move may lead the search to the same solution, more than p moves may be necessary to continue the exploration of p distinct streams.

A more intensive form of the sequential fan candidate list approach, which is potentially more powerful but requires more work, is to use the process illustrated in Fig. 13 as a “look-ahead” strategy. In this case, a limit is placed on the number of iterations that the streams are generated beyond iteration 0. Then the best outcome at this limiting iteration is used to identify a “best current move” (a single first branch) from iteration 0. Upon executing this move, the step shown as iteration 1 in Fig. 13 becomes the new iteration 0, that is, iteration 0 always corresponds to the current iteration. Then this solution becomes the source of p new streams, and the process repeats.

There are a number of possible variants of this sequential fan strategy. For example, instead of selecting a single best branch at the limiting iteration, the method can select a small number of best branches and thus give the method a handful of candidates from which to generate p streams at the new iteration 0.

The iteration limit that determines depth of the look ahead can be variable, and the value of p can change at various depths. Also the number of successors of a given solution that are examined to determine candidates for the p best continuations can be varied as by progressively reducing this number at greater depths.

The type of staging involved in successive solution runs of each stream may be viewed as a means of defining levels in the context of the Proximate Optimality Principle commonly associated with the strategic oscillation component of tabu search. Although we will study this principle in more detail later, we remark that the sequential fan candidate list has a form that is conveniently suited to exploit it.

4.2.5 Bounded Change Candidate List

A bounded change candidate list strategy is relevant in situations where an improved solution can be found by restricting the domain of choices so that no solution

component changes by more than a limited degree on any step. A bound on this degree, expressed by a distance metric appropriate to the context, is selected large enough to encompass possibilities considered strategically relevant. The metric may allow large changes along one dimension but limit the changes along another so that choices can be reduced and evaluated more quickly. Such an approach offers particular benefits as part of an intensification strategy based on decomposition, where the decomposition itself suggests the limits for bounding the changes considered.

4.3 Connections Between Candidate Lists, Tabu Status, and Aspiration Criteria

It is useful to summarize the short-term memory considerations embodied in the interaction between candidate lists, tabu status, and aspiration criteria. The operations of these TS short-term elements are shown in [Fig. 14](#). The representation of penalties in [Fig. 14](#) either as “large” or “very small” expresses a thresholding effect: Either the tabu status yields a greatly deteriorated evaluation or else it chiefly serves to break ties among solutions with highest evaluations. Such an effect of course can be modulated to shift evaluations across levels other than these extremes. If all moves currently available lead to solutions that are tabu (with evaluations that normally would exclude them from being selected), the penalties result in choosing a “least tabu” solution.

The sequence of the *tabu test* and the *aspiration test* in [Fig. 14](#) evidently can be reversed (i.e., by employing the tabu test only if the aspiration threshold is not satisfied). Also, the tabu evaluation can be modified by creating inducements based on the aspiration level, just as it is modified by creating penalties based on tabu status. In this sense, aspiration conditions and tabu conditions can be conceived roughly as “mirror images” of each other.

For convenience, [Fig. 14](#) expresses tabu restrictions solely in terms of penalized evaluations, although we have seen that tabu status is often permitted to serve as an all-or-none threshold, without explicit reference to penalties and inducements (by directly excluding tabu options from being selected, subject to the outcome of aspiration tests). Whether or not modified evaluations are explicitly used, the selected move may not be the one with the best objective function value, and consequently, the solution with the best objective function value encountered throughout the search history is recorded separately.

4.4 Logical Restructuring

Logical restructuring is an important element of adaptive memory solution approaches, which gives a connection between short- and long-term strategies. Logical restructuring is implicit in strategic oscillation and path relinking, which we examine in subsequent sections, but its role and significance in these strategies is

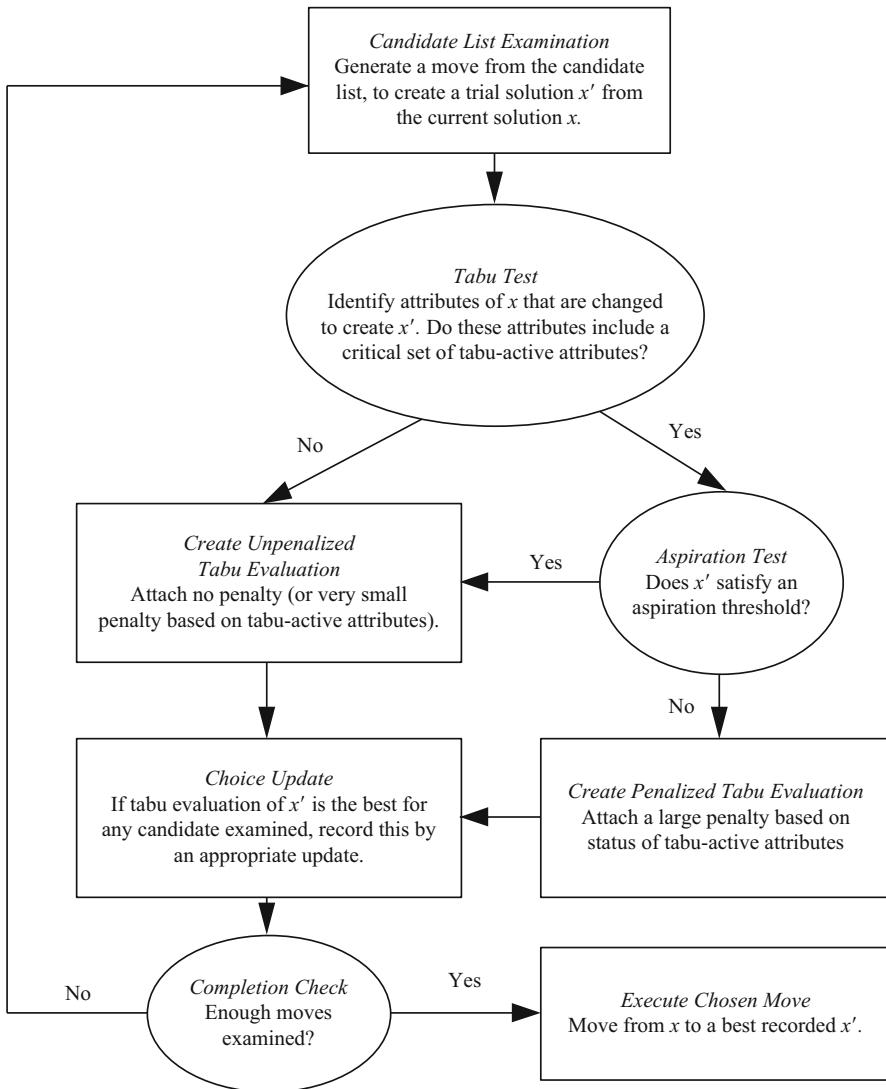
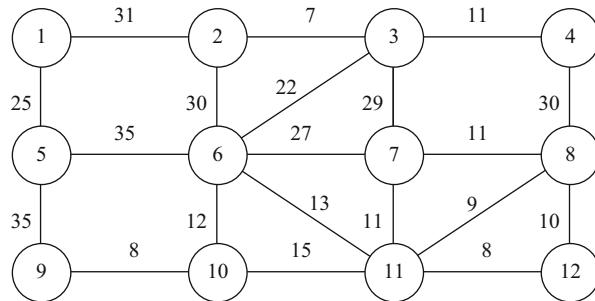


Fig. 14 Short-term memory operation

often overlooked. By extension, the general usefulness of logical restructuring is also often not clearly understood. We examine some of its principal features before delving into longer-term considerations and show how it can also be relevant for improving the designs of short-term strategies.

Logical restructuring emerges as a way to meet the combined concerns of quality and influence. Its goal is to exploit the ways in which influence (structural, local, and global) can uncover improved routes to high-quality solutions. For this purpose, a critical step is to redesign standard strategies to endow them with the

Fig. 15 Illustrative Min k -Tree problem



power to ferret out opportunities otherwise missed. This step particularly relies on integrating two elements: (1) the identification of changes that satisfy properties that are essential (and limiting) in order to achieve improvement, in contrast to changes that simply depart from what has previously been seen and (2) the use of anticipatory (“means-ends”) analysis to bring about such essential changes. Within the context of anticipatory analysis, logical restructuring seeks to answer the following questions: “What conditions assure the existence of a trajectory that will lead to an improved solution?” and “What intermediate moves can create such conditions?” The “intermediate moves” of the second question may be generated either by modifying the evaluations used to select transitions between solutions or by modifying the neighborhood structure that determines these transitions.

To illustrate the relevant considerations, we return again to the example of the Min k -Tree problem discussed in previous sections. We replace the previous graph by the one shown in Fig. 15 but continue to consider the case of $k = 4$.

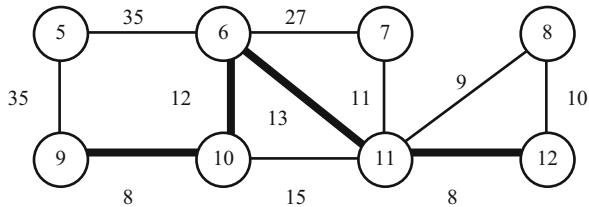
The same rules to execute a first-level tabu search approach as in our earlier illustrations (including the rules for generating a starting solution) produce a sequence of steps that quickly reaches the vicinity of the optimal solution but require some effort actually find this solution. In fact, it is readily verified that applying these rules will cause all edges of the optimal solution except one, edge (10,11), to be contained in the union of the two solutions obtained on iterations 4 and 5. Yet an optimal solution will not be found until iteration 11.

This delayed process of finding a route to an optimal solution (which can be greatly magnified for larger or more complex problems) can be substantially accelerated by means of logical restructuring. More generally, such restructuring can make it possible to uncover fertile options that can otherwise be missed entirely.

4.4.1 Restructuring by Changing Evaluations and Neighborhoods

The first type of logical restructuring we illustrate makes use both of modified evaluations and an amended neighborhood structure. As pointed out in Sect. 3.2 earlier, the swap moves we have employed for the Min k -Tree problem may be subdivided into two types: *static* swaps, which leave the nodes of the current tree unchanged, and *dynamic* swaps, which replace one of the nodes currently in the tree with another that is not in the tree. This terminology was chosen to reflect the effect that each swap type has on the nodes of the tree. Since dynamic swaps in a sense

Fig. 16 Solution and candidate edges to add to iteration 4 tree



are more influential, we give them special consideration. We observe that a dynamic swap can select an edge to be dropped only if it is a *terminal edge*, that is, one that meets a *leaf node* of the tree, which is a node that is met by only a single tree edge (the terminal edge).

Although it is usually advantageous to drop an edge with a relatively large weight, this may not be possible. Thus, we are prompted to consider an “anticipatory goal” of making moves that cause more heavily weighted edges to become terminal edges and hence eligible to be dropped. By this means, static swaps can be used to set up desirable conditions for dynamic swaps.

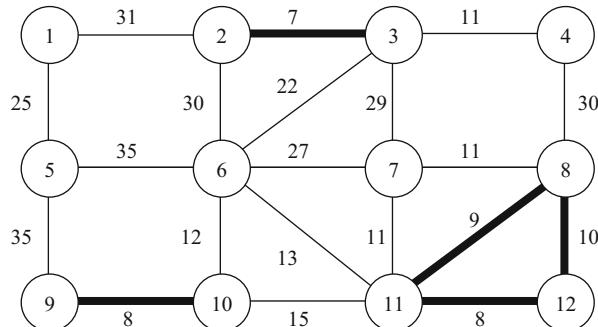
The solution obtained at iteration 4 of the process for solving the example problem of Fig. 15 gives a basis for showing what is involved. We clarify the situation by showing the current solution at this iteration in Fig. 16 (without bothering to identify the solutions obtained at other iterations), where edges contained in the current tree are shown as heavy edges and the candidate edges to add to the tree are shown as light edges.

The move that changes the tree at iteration 4 to that of iteration 5 – if the rules illustrated in Sect. 3 are used – is a dynamic swap that adds edge (8,11) with a weight of 9 and drops edge (9,10) with a weight of 8. We make use of information contained in this choice to construct a more powerful move using logical restructuring, as follows.

Having identified (8,11) as a candidate to be added, the associated anticipatory goal is to identify a static swap that will change a larger weight edge into a terminal edge. Specifically, the static swap that adds edge (10,11) and drops edge (6,10), with a move value of 3, produces a terminal edge from the relatively high weight edge (6,11) (which has a weight of 13). Since the candidate edge (8,11) to be added has a weight of 9, the result of joining the indicated static swap with the subsequent dynamic swap (that respectively adds and drops (8,11) and (6,11)) will be a net gain (The static move value of 3 is joined with the dynamic move value of -4, yielding a result of -1).

Effectively, such anticipatory analysis leads to a way to extract a fruitful outcome from a relatively complex set of options by focusing on a simple set of features. It would be possible to find the same outcome by a more ponderous approach that checks all sequences in which a dynamic move follows a static move. This requires a great deal of computational effort, in fact, considerably more than involved in the approach without logical restructuring that succeeded in finding an optimal solution at iteration 11 (considering the trade-off between number of iterations and work per iteration).

Fig. 17 Threshold generated components



By contrast, the use of logical restructuring allows the anticipatory analysis to achieve the benefits of a more massive exploration of alternatives but without incurring the burden of undue computational effort. In this example, the restructuring is accomplished directly as follows. First, it is only necessary to identify the two best edges to add for a dynamic swap (independent of matching them with an edge to drop), subject to requiring that these edges meet different nodes of the tree (In the tree of iteration 4, seen in Fig. 17, these two edges are (8,11) and (8,12)). Then at the next step, during the process of looking at candidate static swaps, a modified “anticipatory move value” is created for each swap that creates a terminal edge, by subtracting the weight of this edge from the standard move value.

This gives all that is needed to find (and evaluate) a best “combined move sequence” of the type we are looking for. In particular, every static move that generates a terminal edge can be combined with a dynamic move that drops this edge and then adds one of the two “best edges” identified in first of the two preceding steps. Hence, the restructuring is completed by adding the anticipatory move value to the weight of one of these two edges (appropriately identified) thereby determining a best combined move. The illustrated process therefore achieves restructuring in two ways: by modifying customary move values and by fusing certain sequences of moves into a single compound move.

Although this example appears on the surface to be highly problem specific, its basic features are shared by applications that arise in a variety of problem settings. Later, the reader will see how variants of logical restructuring embodied in this illustration are natural components of the strategies of path relinking and ejection chain constructions.

4.4.2 Threshold-Based Restructuring and Induced Decomposition

The second mode of logical restructuring that we illustrate by reference to the Min k -Tree problem example is more complex (in the sense of inducing a more radical restructuring) but relatively easy to sketch and also potentially more powerful.

Consider again the solution produced at iteration 4. This is a local optimum and also the best solution found up to the current stage of search. We seek to identify a property that will be satisfied by at least one solution that has a smaller weight than the weight of this solution (41) and which will impose useful limits

on the composition of such a solution. A property that in fact must be shared by all “better” solutions can be expressed as a threshold involving the average weight of the tree edges. This average weight must be less than the threshold value of 41/4 (i.e., 10 1/4). Since some of the edges in any improved solution must have weights less than this threshold, we are motivated to identify such “preferred” edges as a foundation for a restructured form of the solution approach. In this type of restructuring, we no longer confine attention to swap moves but look for ways to link the preferred edges to produce an improved solution (Such a restructuring can be based on threshold values derived from multiple criteria).

When the indicated strategy is applied to the present example, a large part of the graph is eliminated, leaving only 3 separate connected components: (a) the edge (2,3), (b) the edge (9,10), and (c) the three edges (8,11), (8,12), and (11,12). The graph that highlights these components is shown in Fig. 17. At this point, a natural approach is to link such components by shortest paths and then shave off terminal edges if the trees are too large, before returning to the swapping process. Such an approach will immediately find the optimal solution that previously was not found until iteration 11.

This second illustrated form of restructuring is a fundamental component of the strategic oscillation approach which we describe in more detail in the next section. A salient feature of this type of restructuring is its ability to create an *induced decomposition* of either the solution space or the problem space. This outcome, coupled with the goal of effectively joining the decomposed components to generate additional solution alternatives, is also a basic characteristic of path relinking, which is also examined in the next section. More particularly, the special instance of path relinking known as vocabulary building, which focuses on assembling fragments of solutions into larger units, offers a direct model for generalizing the “threshold decomposition” strategy illustrated here.

In some applications, specific theorems can be developed about the nature of optimal solutions and can be used to provide relevant designs for restructuring. The Min k -Tree problem is one for which such a theorem is available (Glover and Laguna, [31]). Interestingly, the second form of restructuring we have illustrated, which is quite basic, exploits several aspects of this theorem, although without “knowing” what the theorem is. In general, logical restructuring and the TS strategies such as path relinking and strategic oscillation which embody it appear to behave as if they similarly have a capacity to exploit underlying properties of optimal solutions in broader contexts, contexts whose features are not sufficiently uniform or easily characterized to permit the nature of optimal solutions to be expressed in the form of a theorem.

5 Longer-Term Memory

In some applications, the short-term TS memory components are sufficient to produce very high-quality solutions. However, in general, TS becomes significantly stronger by including longer-term memory and its associated strategies. In the longer-term TS strategies, the modified neighborhood produced by tabu search

may contain solutions not in the original one, generally consisting of selected elite solutions (high-quality local optima) encountered at various points in the solution process. Such elite solutions typically are identified as elements of a regional cluster in intensification strategies and as elements of different clusters in diversification strategies. In addition, elite solution components, in contrast to the solutions themselves, are included among the elements that can be retained and integrated to provide inputs to the search process.

Perhaps surprisingly, the use of longer-term memory does not require long solution runs before its benefits become visible. Often, its improvements begin to be manifest in a relatively modest length of time and can allow solution efforts to be terminated somewhat earlier than otherwise possible, due to finding very high-quality solutions within an economical time span. The fastest methods for some types of routing and scheduling problems, for example, are based on including longer-term TS memory. On the other hand, it is also true that the chance of finding still better solutions as time grows, in the case where an optimal solution is not already found, is enhanced by using longer-term TS memory in addition to short-term memory.

5.1 Frequency-Based Approach

Frequency-based memory provides a type of information that complements the information provided by recency-based memory, broadening the foundation for selecting preferred moves. Like recency, frequency often is weighted or decomposed into subclasses by taking account of the dimensions of solution quality and move influence. Also, frequency can be integrated with recency to provide a composite structure for creating penalties and inducements that modify move evaluations (Although recency-based memory is often used in the context of short-term memory, it can also be a foundation of longer-term forms of memory).

For our present purposes, we conceive frequencies to consist of ratios, whose numerators represent counts expressed in two different measures: a *transition measure*, the number of iterations where an attribute changes (enters or leaves) the solutions visited on a particular trajectory, and a *residence measure*, the number of iterations where an attribute belongs to solutions visited on a particular trajectory or the number of instances where an attribute belongs to solutions from a particular subset. The denominators generally represent one of three types of quantities: (1) the total number of occurrences of all events represented by the numerators (such as the total number of associated iterations), (2) the sum (or average) of the numerators, and (3) the maximum numerator value. In cases where the numerators represent weighted counts, some of which may be negative, denominator (3) is expressed as an absolute value and denominator (2) is expressed as a sum of absolute values (possibly shifted by a small constant to avoid a zero denominator). The ratios produce *transition frequencies* that keep track of how often attributes change and *residence frequencies* that keep track of how often attributes are members of solutions generated. In addition to referring to such frequencies, thresholds based on the numerators alone can be useful for indicating when phases of greater

diversification are appropriate (The thresholds for particular attributes can shift after a diversification phase is executed).

Residence frequencies and transition frequencies sometimes convey related information but in general carry different implications. They are sometimes confused (or treated identically) in the literature. A noteworthy distinction is that residence measures, by contrast to transition measures, are not concerned with the characteristics of a particular solution attribute or whether it is an attribute that changes in moving from one solution to another. For example, in the Min k -Tree problem, a residence measure may count the number of times edge (i, j) was part of the solution, while a transition measure may count the number of times edge (i, j) was added to the solution (More complex joint measures, such as the number of times edge (i, j) was accompanied in the solution by edge (k, l) or was deleted from the solution in favor of edge (k, l) , can also selectively be generated. Such frequencies relate to the issues of creating more complex attributes out of simpler ones and to the strategies of vocabulary building).

A high residence frequency may indicate that an attribute is highly attractive if the domain consists of high-quality solutions or may indicate the opposite if the domain consists of low quality solutions. On the other hand, a residence frequency that is high (or low) when the domain is chosen to include both high- and low-quality solutions may point to an entrenched (or excluded) attribute that causes the search space to be restricted and that needs to be jettisoned (or incorporated) to allow increased diversity. For example, an entrenched attribute may be a job that is scheduled in the same position during a sequence of iterations that include both low- and high quality-objective function evaluations.

As a further useful distinction, a high transition frequency, in contrast to a high residence frequency, may indicate an associated attribute and is a “crack filler” that shifts in and out of solutions to perform a fine-tuning function. In this context, a transition frequency may be interpreted as a measure of volatility. For example, the Min k -Tree problem instance in Fig. 4 of Sect. 3 contains a number of edges whose weight may give them the role of crack fillers. Specifically, edges (3,5) and (6,7) both have a weight of 6, which makes them attractive relative to other edges in the graph. Since these edges are not contained in an optimal solution, there is some likelihood that they may repeatedly enter and leave the current solution in a manner to lure the search away from the optimal region. In general, crack fillers are determined not simply by cost or quality but by structure, as in certain forms of connectivity (Hence, e.g., the edge (3,5) of Fig. 4 does not repeatedly enter and leave solutions in spite of its cost). Some subset of such elements is also likely to be a part of an optimal solution. This subset can typically be identified with much less difficulty once other elements are in place. On the other hand, a solution (full or partial) may contain the “right” crack fillers but offer little clue as to the identity of the other attributes that will transform the solution into one that is optimal.

We use a sequencing problem and the Min k -Tree problem as contexts to further illustrate both residence and transition frequencies. Only numerators are indicated, understanding that denominators are provided by the conditions (1)–(3) previously defined. The measures are given in Table 8.

Table 8 Example of frequency measures

<i>Problem</i>	<i>Residence measure</i>	<i>Transition measure</i>
Sequencing	Number of times job j has occupied position $\pi(j)$	Number of times job i has exchanged positions with job j
	Sum of tardiness of job j when this job occupies position $\pi(j)$	Number of times job j has been moved to an earlier position in the sequence
Min k -Tree problem	Number of times edge (i, j) has been part of the current solution	Number of times edge (i, j) has been deleted from the current solution when edge (k, l) has been added
	Sum of total solution weight when edge (i, j) is part of the solution	Number of times edge (i, j) has been added during improving moves

Attributes that have greater frequency measures, just as those that have greater recency measures (i.e., that occur in solutions or moves closer to the present), can trigger a tabu-activation rule if they are based on consecutive solutions that end with the current solution. However, frequency-based memory often finds its most productive use as part of a longer-term strategy, which employs incentives as well as restrictions to determine which moves are selected. In such a strategy, tabu-activation rules are translated into evaluation penalties, and incentives become evaluation enhancements, to alter the basis for qualifying moves as attractive or unattractive.

To illustrate, in a scheduling setting where a swap neighborhood is used, an attribute such as a job j with a high residence frequency in position $\pi(j)$ may be assigned a strong incentive (“profit”) to serve as a *swap attribute*, thus resulting in the choice of a move that yields a new sequence π' with $\pi'(j) \neq \pi(j)$. Such an incentive is particularly relevant in the case where the *TabuEnd* value of job j is small compared to the current iteration, since this value (minus the corresponding tabu tenure) identifies the latest iteration that job j was a *swap attribute*, and hence discloses that job j has occupied position $\pi(j)$ in every solution since.

Frequency-based memory therefore is usually applied by introducing graduated tabu states, as a foundation for defining penalty and incentive values to modify the evaluation of moves. A natural connection exists between this approach and the recency-based memory approach that creates tabu status as an all-or-none condition. If the tenure of an attribute in recency-based memory is conceived as a conditional threshold for applying a very large penalty, then the tabu classifications produced by such memory can be interpreted as the result of an evaluation that becomes strongly inferior when the penalties are activated. Conditional thresholds are also relevant to determining the values of penalties and incentives in longer-term strategies. Most applications at present, however, use a simple linear multiple of a frequency measure to create a penalty or incentive term. The multiplier is adjusted to create the right balance between the incentive or penalty and the cost (or profit) coefficients of the objective function.

5.2 Intensification Strategies

Intensification strategies are based on modifying choice rules to encourage move combinations and solution features historically found good. They may also initiate a return to attractive regions to search them more thoroughly. A simple instance of this second type of intensification strategy is shown in Fig. 18. The strategy for selecting elite solutions is italicized in Fig. 18 due to its importance. Two variants have proved quite successful. One introduces a diversification measure to assure the solutions recorded differ from each other by a desired degree and then erases all short-term memory before resuming from the best of the recorded solutions. A diversification measure may be related to the number of moves that are necessary to transform one solution into another. Or the measure may be defined independently from the move mechanism. For example, in sequencing, two solutions may be considered diverse if the number of swaps needed to move from one to the other is “large.” On the other hand, the diversification measure may be the number of jobs that occupy a different position in the two sequences being compared (This shows that intensification and diversification often work together, as elaborated in the next section).

The second variant that has also proved successful keeps a bounded length sequential list that adds a new solution at the end only if it is better than any previously seen. The current last member of the list is always the one chosen (and removed) as a basis for resuming search. However, TS short-term memory that accompanied this solution is also saved, and the first move also forbids the move previously taken from this solution, so that a new solution path will be launched.

A third variant of the approach of Fig. 18 is related to a strategy that resumes the search from unvisited neighbors of solutions previously generated. Such a strategy keeps track of the quality of these neighbors to select an elite set and restricts attention to specific types of solutions, such as neighbors of local optima or neighbors of solutions visited on steps immediately before reaching such local optima. This type of “unvisited neighbor” strategy has been little examined. It is noteworthy, however, that the two variants previously indicated have provided solutions of remarkably high quality.

Another type of intensification approach is *intensification by decomposition*, where restrictions may be imposed on parts of the problem or solution structure in order to generate a form of decomposition that allows a more concentrated focus on other parts of the structure. A classical example is provided by the traveling salesman problem, where edges that belong to the intersection of elite tours may be “locked into” the solution, in order to focus on manipulating other parts of the tour. The use of intersections is an extreme instance of a more general strategy for exploiting frequency information, by a process that seeks to identify and constrain the values of *strongly determined* and *consistent variables*. We discuss the identification and use of such variables in Sect. 4.4.1.

Intensification by decomposition also encompasses other types of strategic considerations, basing the decomposition not only on indicators of strength and consistency but also on opportunities for particular elements to interact productively.

Fig. 18 Simple TS intensification approach

```

Apply TS short term memory.
Apply an elite selection strategy.
do {
    Choose one of the elite solutions.
    Resume short term memory TS from chosen solution.
    Add new solutions to elite list when applicable.
} while (iterations < limit and list not empty)

```

Within the context of a permutation problem as in scheduling or routing, for example, where solutions may be depicted as selecting one or more sequences of edges in a graph, a decomposition may be based on identifying subchains of elite solution, where two or more subchains may be assigned to a common set if they contain nodes that are “strongly attracted” to be linked with nodes of other subchains in the set. An edge disjoint collection of subchains can be treated by an intensification process that operates in parallel on each set, subject to the restriction that the identity of the endpoints of the subchains will not be altered. As a result of the decomposition, the best new sets of subchains can be reassembled to create new solutions. Such a process can be applied to multiple alternative decompositions in broader forms of intensification by decomposition.

These ideas are lately finding favor in other procedures and may provide a bridge for interesting components of tabu search with components of other methodologies. We address the connections with these methodologies in Sect. 6.

5.3 Diversification Strategies

Search methods based on local optimization often rely on diversification strategies to increase their effectiveness in exploring the solution space defined by a combinatorial optimization problem. Some of these strategies are designed with the chief purpose of preventing searching processes from *cycling*, that is, from endlessly executing the same sequence of moves (or more generally, from endlessly and exclusively revisiting the same set of solutions). Others are introduced to impart additional robustness or vigor to the search. Genetic algorithms use randomization in component processes such as combining population elements and applying crossover (as well as occasional mutation), thus providing an approximate diversifying effect. Simulated annealing likewise incorporates randomization to make diversification a function of temperature, whose gradual reduction correspondingly diminishes the directional variation in the objective function trajectory of solutions generated. Diversification in GRASP (greedy randomized adaptive search procedures) is achieved in a certain sense within repeated construction phases by means of a random sampling over elements that pass a threshold of attractiveness by a greedy criterion.

In tabu search, diversification is created to some extent by short-term memory functions but is particularly reinforced by certain forms of longer-term memory. TS diversification strategies, as their name suggests, are designed to drive the search

into new regions. Often, they are based on modifying choice rules to bring attributes into the solution that are infrequently used. Alternatively, they may introduce such attributes by periodically applying methods that assemble subsets of these attributes into candidate solutions for continuing the search or by partially or fully restarting the solution process. Diversification strategies are particularly helpful when better solutions can be reached only by crossing barriers or “humps” in the solution space topology.

5.3.1 Modifying Choice Rules

Consider a TS method designed to solve a graph-partitioning problem which uses full and partial swap moves to explore the local neighborhood. The goal of this problem is to partition the nodes of the graph into two equal subsets so that the sum of the weights of the edges that join nodes in one subset to nodes in the other subset is minimized. Full swaps exchange two nodes that lie in two different sets of the partition. Partial swaps transfer a single node from one set to the other set. Since full swaps do not modify the number of nodes in the two sets of the partition, they maintain feasibility, while partial swaps do not. Therefore, under appropriate guidance, one approach to generate diversity is to periodically disallow the use of nonimproving full swaps for a chosen duration (after an initial period where the search “settles down”). The partial swaps must of course be coordinated to allow feasibility to be recovered after achieving various degrees of infeasibility (This relates to the approach of strategic oscillation, described in Sect. 5.4). Implemented appropriately, this strategy has the effect of intelligently perturbing the current solution, while escaping from a local optimum, to an extent that the search is directed to a region that is different than the one being currently explored. The implementation of this strategy as applied to experimental problems has resulted in significant improvements in problem-solving efficacy.

The incorporation of partial swaps in place of full swaps in the previous example can be moderated by using the following penalty function:

$$\text{MoveValue}' = \text{MoveValue} + d * \text{Penalty}.$$

This type of penalty approach is commonly used in TS, where the *Penalty* value is often a function of frequency measures such as those indicated in Table 8 and d is an adjustable diversification parameter. Larger d values correspond to a desire for more diversification (For example, nodes that change sets more frequently are penalized more heavily to encourage the choice of moves that incorporate other nodes. Negative penalties, or “inducements,” may also be used to encourage low-frequency elements). The penalty can be applied to classes of moves as well as to attributes of moves. Thus, during a phase where full swaps moves are excluded, all such moves receive a large penalty (with a value of d that is effectively infinite).

In some applications where d is used to inhibit the selection of “feasibility preserving” moves, the parameter can be viewed as the reciprocal of a Lagrangian multiplier in that “low” values result in nearly infinite costs for constraint violation, while “high” values allow searching through infeasible regions. The adjustment of

such a parameter can be done in a way to provide a strategic oscillation around the feasibility boundary, again as discussed in [Sect. 5.4](#). The parameter can also be used to control the amount of randomization in probabilistic versions of tabu search.

In TS methods that incorporate the simplex method of linear programming, as in “adjacent extreme point approaches” for solving certain nonlinear and mixed-integer programming problems, a diversification phase can be designed based on the number of times variables become basic. For example, a diversification step can give preference to bringing a nonbasic variable into the basis that has remained out of the basis for a relatively long period (cumulatively, or since its most recent inclusion, or a combination of the two). The number of successive iterations such steps are performed, and the frequency with which they are initiated, are design considerations of the type that can be addressed, for example, by the approach of target analysis (see [Sect. 6](#)).

5.3.2 Restarting

Frequency information can be used in different ways to design restarting mechanisms within tabu search. In a sequencing problem, for example, the overall frequency of jobs occupying certain positions can be used to bias a construction procedure and generate new restarting points.

In a TS method for a location/allocation problem, a diversification phase can be developed using frequency counts on the number of times a depot has changed its status (from open to closed or vice versa). The diversification phase can be started from the best solution found during the search. Based on the frequency information, d depots with the lowest counts are selected and their status is changed. The search starts from the new solution which differs from the best by exactly d components. To prevent a quick return to the best solution, the status of the d depots is also recorded in short-term memory (This is another case where residence frequency measures may provide useful alternatives or supplements to transition frequency measures).

Additional forms of memory functions are possible when a restarting mechanism is implemented. For example, in the location/allocation problem, it is possible to keep track of recent sets of depots that were selected for diversification and avoid the same selection in the next diversification phase. Similarly, in a sequencing problem, the positions occupied by jobs in recent starting points can be recorded to avoid future repetition. This may be viewed as a very simple form of the critical event memory discussed in [Sect. 3](#), and more elaborate forms will often yield greater benefits. The exploitation of such memory is very important in TS designs that are completely deterministic, since in these cases, a given starting point will always produce the same search path. Experience also shows, however, that uses of TS memory to guide probabilistic forms of restarting can likewise yield benefits [[14, 50, 64](#)].

Before concluding this section, it is appropriate to provide a word of background about the orientation underlying diversification strategies within the tabu search framework. Often there appears to be a hidden assumption that diversification is somehow tantamount to randomization. Certainly, the introduction of a random element to achieve a diversifying effect is a widespread theme among search

procedures and is fundamental to the operation of simulated annealing and genetic algorithms. From an abstract standpoint, there is clearly nothing wrong with equating randomization and diversification, but to the extent that diversity connotes differences among elements of a set and to the extent that establishing such differences is relevant to an effective search strategy, then the popular use of randomization is at best a convenient proxy (and at worst a haphazard substitute) for something quite different.

When randomization is used as part of a restarting mechanism, for example, frequency information can be employed to approximate probability distributions that bias the construction process. In this way, randomization is not a “blind” mechanism, but instead, it is guided by search history. We examine inappropriate roles of randomization in [Sect. 5.6](#), where we also explore the intensification/diversification distinction more thoroughly.

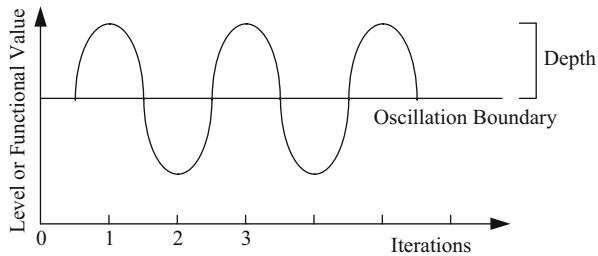
5.4 Strategic Oscillation

Strategic oscillation is closely linked to the origins of tabu search and provides a means to achieve an effective interplay between intensification and diversification over the intermediate to long term. The recurring usefulness of this approach documented in a variety of studies warrants a more detailed examination of its characteristics.

Strategic oscillation operates by orienting moves in relation to a critical level, as identified by a stage of construction or a chosen interval of functional values. Such a critical level or *oscillation boundary* often represents a point where the method would normally stop. Instead of stopping when this boundary is reached, however, the rules for selecting moves are modified to permit the region defined by the critical level to be crossed. The approach then proceeds for a specified depth beyond the oscillation boundary and turns around. The oscillation boundary again is approached and crossed, this time from the opposite direction, and the method proceeds to a new turning point (see [Fig. 19](#)).

The process of repeatedly approaching and crossing the critical level from different directions creates an oscillatory behavior, which gives the method its name. Control over this behavior is established by generating modified evaluations and rules of movement, depending on the region navigated and the direction of search. The possibility of retracing a prior trajectory is avoided by standard tabu search mechanisms, like those established by recency-based and frequency-based memory functions.

A simple example of this approach occurs for the multidimensional knapsack problem, where values of zero-one variables are changed from 0 to 1 until reaching the boundary of feasibility. The method then continues into the infeasible region using the same type of changes but with a modified evaluator. After a selected number of steps, the direction is reversed by choosing moves that change variables from 1 to 0. Evaluation criteria to drive toward improvement vary according to whether the movement occurs inside or outside the feasible region (and whether it is

Fig. 19 Strategic oscillation

directed toward or away from the boundary), accompanied by associated restrictions on admissible changes to values of variables. The turnaround toward feasibility can also be triggered by a maximum infeasibility value, which defines the depth of the oscillation beyond the critical level (i.e., the feasibility boundary).

A somewhat different type of application occurs for graph theory problems where the critical level represents a desired form of graph structure, capable of being generated by progressive additions (or insertions) of basic elements such as nodes, edges, or subgraphs. One type of strategic oscillation approach for this problem results by a constructive process of introducing elements until the critical level is reached and then introducing further elements to cross the boundary defined by the critical level. The current solution may change its structure once this boundary is crossed (as where a forest becomes transformed into a graph that contains loops), and hence, a different neighborhood may be required, yielding modified rules for selecting moves. The rules again change in order to proceed in the opposite direction, removing elements until again recovering the structure that defines the critical level.

In the Min k -Tree problem, for example, edges can be added beyond the critical level defined by k . Then a rule to delete edges must be applied. The rule to delete edges will typically be different in character from the one used for adding (i.e., will not simply be its “inverse”). In this case, all feasible solutions lie on the oscillation boundary, since any deviation from this level results in solutions with more or less than k edges.

Such rule changes are typical features of strategic oscillation and provide an enhanced heuristic vitality. The application of different rules may be accompanied by crossing a boundary to different depths on different sides. An option is to approach and retreat from the boundary while remaining on a single side, without crossing (i.e., electing a crossing of “zero depth”).

These examples constitute a constructive/destructive type of strategic oscillation, where constructive steps “add” elements (or set variables to 1) and destructive steps “drop” elements (or set variables to 0) (Types of TS memory structures for add/drop moves discussed in Sect. 3 are relevant for such procedures). One-sided oscillations (that remain on a single side of a critical boundary) are appropriate in a variety of scheduling and graph-related applications, where constructive processes are traditionally applied. The alternation with destructive processes that

strategically dismantle and then rebuild successive trial solutions affords a potent enhancement of more traditional procedures. In both one-sided and two-sided oscillation approaches, it is frequently important to spend additional search time in regions close to the critical level and especially to spend time at the critical level itself. This may be done by inducing a sequence of tight oscillations about the critical level, as a prelude to each larger oscillation that proceeds to a greater depth. Alternately, if greater effort is permitted for evaluating and executing each move, the method may use “exchange moves” (broadly interpreted) to stay at the critical level for longer periods. In the case of the Min k -Tree problem, for example, once the oscillation boundary has been reached, the search can stay on it by performing swap moves (either of nodes or edges). An option is to use such exchange moves to proceed to a local optimum each time the critical level is reached.

When the level or functional values in Fig. 19 refer to degrees of feasibility and infeasibility, a vector-valued function associated with a set of problem constraints can be used to control the oscillation. In this case, controlling the search by bounding this function can be viewed as manipulating a parameterization of the selected constraint set. A preferred alternative is often to make the function a Lagrangian or surrogate constraint penalty function, avoiding vector-valued functions and allowing trade-offs between degrees of violation of different component constraints.

Intensification processes can readily be embedded in strategic oscillation by altering choice rules to encourage the incorporation of particular attributes, or at the extreme, by locking such attributes into the solution for a period. Such processes can be viewed as designs for exploiting *strongly determined* and *consistent* variables. A strongly determined variable is one that cannot change its value in a given high-quality solution without seriously degrading quality or feasibility, while a consistent variable is one that frequently takes on a specific value (or a highly restricted range of values) in good solutions. The development of useful measures of “strength” and “consistency” is critical to exploiting these notions, particularly by accounting for trade-offs determined by context. However, straightforward uses of frequency-based memory for keeping track of consistency, sometimes weighted by elements of quality and influence, have produced methods with very good performance outcomes.

An example of where these kinds of approaches are also beginning to find favor in other settings occurs in recently developed variants of genetic algorithms for sequencing problems. The more venturesome of these approaches are coming to use special forms of “crossover” to assure offspring will receive attributes shared by good parents, thus incorporating a type of intensification based on consistency. Extensions of such procedures using TS ideas of identifying elements that qualify as consistent and strongly determined according to broader criteria, and making direct use of memory functions to establish this identification, provide an interesting area for investigation (Additional links to GA methods, and ways to go beyond current explorations of such methods, are discussed in Sect. 6).

Longer-term processes, following the type of progression customarily found beneficial in tabu search, may explicitly introduce supplemental diversification strategies into the oscillation pattern. When oscillation is based on constructive

and destructive processes, the repeated application of constructive phases (rather than moving to intermediate levels using destructive moves) embodies an extreme type of oscillation that is analogous to a restart method. In this instance, the restart point is always the same (i.e., a null state) instead of consisting of different initial solutions, and hence, it is important to use choice rule variations to assure appropriate diversification.

A connection can also be observed between an extreme version of strategic oscillation, in this case a relaxed version, and the class of procedures known as *perturbation* approaches. An example is the subclass known as “large-step simulated annealing” or “large-step Markov chain” methods [39, 52–54]. Such methods try to drive an SA procedure (or an iterated descent procedure) out of local optimality by propelling the solution a greater distance than usual from its current location.

Perturbation methods may be viewed as loosely structured procedures for inducing oscillation, without reference to intensification and diversification and their associated implementation strategies. Similarly, perturbation methods are not designed to exploit trade-offs created by parametric variations in elements such as different types of infeasibility and measures of displacement from different sides of boundaries. Nevertheless, at a first level of approximation, perturbation methods seek goals similar to those pursued by strategic oscillation.

5.5 Path Relinking

A useful integration of intensification and diversification strategies occurs in the approach called *path relinking*. This approach generates new solutions by exploring trajectories that connect elite solutions by starting from one of these solutions, called an *initiating solution*, and generating a path in the neighborhood space that leads toward the other solutions, called *guiding solutions*. This is accomplished by selecting moves that introduce attributes contained in the guiding solutions.

The approach may be viewed as an extreme (highly focused) instance of a strategy that seeks to incorporate attributes of high-quality solutions, by creating inducements to favor these attributes in the moves selected. However, instead of using an inducement that merely encourages the inclusion of such attributes, the path-relinking approach subordinates all other considerations to the goal of choosing moves that introduce the attributes of the guiding solutions, in order to create a “good attribute composition” in the current solution. The composition at each step is determined by choosing the best move, using customary choice criteria, from the restricted set of moves that incorporate a maximum number (or a maximum weighted value) of the attributes of the guiding solutions. As in other applications of TS, aspiration criteria can override this restriction to allow other moves of particularly high quality to be considered.

Specifically, upon identifying a collection of one or more elite solutions to guide the path of a given solution, the attributes of these guiding solutions are assigned preemptive weights as inducements to be selected. Larger weights are assigned to

attributes that occur in greater numbers of the guiding solutions, allowing bias to give increased emphasis to solutions with higher quality or with special features (e.g., complementing those of the solution that initiated the new trajectory).

More generally, it is not necessary for an attribute to occur in a guiding solution in order to have a favored status. In some settings, attributes can share degrees of similarity, and in this case, it can be useful to view a solution vector as providing “votes” to favor or discourage particular attributes. Usually, the strongest forms of aspiration criteria are relied upon to overcome this type of choice rule.

In a given collection of elite solutions, the role of initiating solution and guiding solutions can be alternated. The distinction between initiating solutions and guiding solutions effectively vanishes in such cases. For example, a set of current solutions may be generated simultaneously, extending different paths and allowing an initiating solution to be replaced (as a guiding solution for others) whenever its associated current solution satisfies a sufficiently strong aspiration criterion.

Because their roles are interchangeable, the initiating and guiding solutions are collectively called *reference solutions*. These reference solutions can have different interpretations depending on the solution framework under consideration. Reference points can be created by any of a number of different heuristics that result in high-quality solutions.

An idealized form of such a process is shown in Fig. 20. The chosen collection of reference solutions consists of the three members, A, B, and C. Paths are generated by allowing each to serve as initiating solution and by allowing either one or both of the other two solutions to operate as guiding solutions. Intermediate solutions encountered along the paths are not shown. The representation of the paths as straight lines of course is oversimplified, since choosing among available moves in a current neighborhood will generally produce a considerably more complex trajectory. Intensification can be achieved by generating paths from similar solutions, while diversification is obtained creating paths from dissimilar solutions. Appropriate aspiration criteria allow deviation from the paths at attractive neighbors.

As Fig. 20 indicates, at least one path continuation is allowed beyond each initiating/guiding solution. Such a continuation can be accomplished by penalizing the inclusion of attributes dropped during a trajectory, including attributes of guiding solutions that may be compelled to be dropped in order to continue the path (An initiating solution may also be repelled from the guiding solutions by penalizing the inclusion of their attributes from the outset). Probabilistic TS variants operate in the path-relinking setting, as they do in others, by translating evaluations for deterministic rules into probabilities of selection, strongly biased to favor higher evaluations.

Promising regions are searched more thoroughly in path relinking by modifying the weights attached to attributes of the guiding solutions and by altering the bias associated with solution quality and selected solution features. Figure 21 depicts the type of variation that can result, where the point X represents an initiating solution, the points A, B, and C represent guiding solutions, and the dashed, dotted, and solid lines are different searching paths. For appropriate choices of the reference points (and neighborhoods for generating paths from them), the notion called the *Principle*

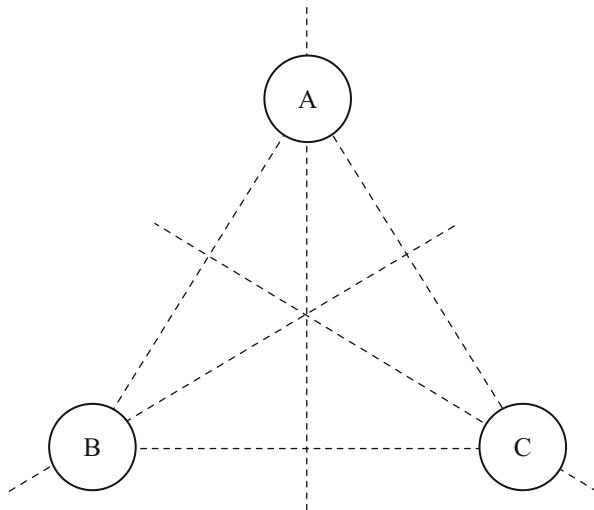


Fig. 20 Paths relinking in neighborhood space

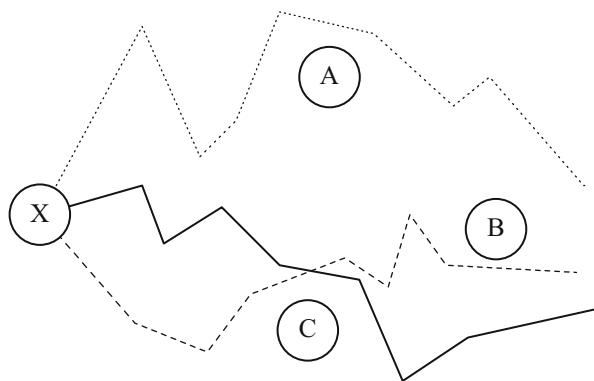


Fig. 21 Path relinking by attribute bias

of Proximate Optimality [31] suggests that additional elite points are likely to be found in the regions traversed by the paths, upon launching new searches from high quality points on these paths.

5.5.1 Roles in Intensification and Diversification

Path relinking, in common with strategic oscillation, gives a natural foundation for developing intensification and diversification strategies. Intensification strategies in this setting typically choose reference solutions to be elite solutions that lie in a common region or that share common features. Similarly, diversification strategies

based on path relinking characteristically select reference solutions that come from different regions or that exhibit contrasting features. Diversification strategies may also place more emphasis on paths that go beyond the reference points. Collections of reference points that embody such conditions can be usefully determined by clustering and conditional analysis methods.

These alternative forms of path relinking also offer a convenient basis for parallel processing, contributing to the approaches for incorporating intensification and diversification trade-offs into the design of parallel solution processes generally.

5.5.2 Incorporating Alternative Neighborhoods

Path-relinking strategies in tabu search can occasionally profit by employing different neighborhoods and attribute definitions than those used by the heuristics for generating the reference solutions. For example, it is sometimes convenient to use a constructive neighborhood for path relinking, that is, one that permits a solution to be built in a sequence of constructive steps (as in generating a sequence of jobs to be processed on specified machines using dispatching rules). In this case, the initiating solution can be used to give a beginning partial construction, by specifying particular attributes (such as jobs in particular relative or absolute sequence positions) as a basis for remaining constructive steps. Similarly, path relinking can make use of destructive neighborhoods, where an initial solution is “overloaded” with attributes donated by the guiding solutions, and such attributes are progressively stripped away or modified until reaching a set with an appropriate composition.

When path relinking is based on constructive neighborhoods, the guiding solution(s) provides the attribute relationships that give options for subsequent stages of construction. At an extreme, a full construction can be produced, by making the initiating solution a *null solution*. The destructive extreme starts from a “complete set” of solution elements. Constructive and destructive approaches differ from transition approaches by typically producing only a single new solution, rather than a sequence of solutions, on each path that leads from the initiating solution toward the others. In this case, the path will never reach the additional solutions unless a transition neighborhood is used to extend the constructive neighborhood.

Constructive neighborhoods can often be viewed as a special case of feasibility restoring neighborhoods, since a null or partially constructed solution does not satisfy all conditions to qualify as feasible. Similarly, destructive neighborhoods can also represent an instance of a feasibility restoring function, as where an excess of elements may violate explicit problem constraints. A variety of methods have been devised to restore infeasible solutions to feasibility, as exemplified by flow augmentation methods in network problems, subtour elimination methods in traveling salesman and vehicle routing problems, alternating chain processes in degree-constrained subgraph problems, and value incrementing and decrementing methods in covering and multidimensional knapsack problems. Using neighborhoods that permit restricted forms of infeasibilities to be generated, and then using associated neighborhoods to remove these infeasibilities, provides a form of path relinking with useful diversification features. Upon further introducing transition neighborhoods,

with the ability to generate successive solutions with changed attribute mixes, the mechanism of path relinking also gives a way to tunnel through infeasible regions. The following is a summary of the components of path relinking:

- Step 1. Identify the neighborhood structure and associated solution attributes for path relinking (possibly different from those of other TS strategies applied to the problem).
- Step 2. Select a collection of two or more reference solutions, and identify which members will serve as the initiating solution and the guiding solution(s) (Reference solutions can be infeasible, such as “incomplete” or “overloaded” solution components treated by constructive or destructive neighborhoods).
- Step 3. Move from the initiating solution toward (or beyond) the guiding solution(s), generating one or more intermediate solutions as candidates to initiate subsequent problem-solving efforts (If the first phase of this step creates an infeasible solution, apply an associated second phase with a feasibility restoring neighborhood).

In Sect. 6, we will see how the path-relinking strategy relates to a strategy called scatter search, which provides additional insights into the nature of both approaches.

5.6 The Intensification/Diversification Distinction

The relevance of the intensification/diversification distinction is supported by the usefulness of TS strategies that embody these notions. Although both operate in the short term as well as the long term, we have seen that longer-term strategies are generally the ones where these notions find their greatest application.

In some instances, we may conceive of intensification as having the function of an intermediate term strategy, while diversification applies to considerations that emerge in the longer run. This view comes from the observation that in human problem solving, once a short-term strategy has exhausted its efficacy, the first (intermediate term) response is often to focus on the events where the short-term approach produced the best outcomes and to try to capitalize on elements that may be common to those events. When this intensified focus on such events likewise begins to lose its power to uncover further improvement, more dramatic departures from a short-term strategy are undertaken (Psychologists do not usually differentiate between intermediate and longer-term memory, but the fact that memory for intensification and diversification can benefit from such differentiation suggests that there may be analogous physical or functional differences in human memory structures). Over the truly long term, however, intensification and diversification repeatedly come into play in ways where each depends on the other, not merely sequentially, but also simultaneously.

There has been some confusion between the terms intensification and diversification, as applied in tabu search, and the terms *exploitation* and *exploration*, as popularized in the literature of genetic algorithms. The differences between these

two sets of notions deserve to be clarified, because they have substantially different consequences for problem solving.

The exploitation/exploration distinction comes from control theory, where exploitation refers to following a particular recipe (traditionally memoryless) until it fails to be effective, and exploration then refers to instituting a series of random changes – typically via multiarmed bandit schemes – before reverting to the tactical recipe (The issue of exploitation versus exploration concerns how often and under what circumstances the randomized departures are launched).

By contrast, intensification and diversification in tabu search are both processes that take place when simpler exploitation designs play out and lose their effectiveness – although as we have noted, the incorporation of memory into search causes intensification and diversification also to be manifest in varying degrees even in the short range (Similarly, as we have noted, intensification and diversification are not opposed notions, for the best form of each contains aspects of the other, along a spectrum of alternatives).

Intensification and diversification are likewise different from the control theory notion of exploration. Diversification, which is sometimes confused with exploration, is not a recourse to a game of chance for shaking up the options invoked but is a collection of strategies – again taking advantage of memory – designed to move purposefully rather than randomly into uncharted territory.

The source of these differences is not hard to understand. Researchers and practitioners in the area of search methods have had an enduring love affair with randomization, perhaps influenced by the much publicized Heisenberg Uncertainty Principle in Quantum Mechanics. Einstein's belief that God does not roll dice is out of favor, and many find a special enchantment in miraculous events where blind purposelessness creates useful order (We are less often disposed to notice that this way of producing order requires an extravagant use of time and that order, once created, is considerably more effective than randomization in creating still higher order).

Our “scientific” reports of experiments with nature reflect our fascination with the role of chance. When apparently chaotic fluctuations are brought under control by random perturbations, we seize upon the random element as the key, while downplaying the importance of attendant restrictions on the setting in which randomization operates. The diligently concealed message is that under appropriate controls, *perturbation* is effective for creating desired patterned outcomes – and in fact, if the system and attendant controls are sufficiently constrained, perturbation works even when random (Instead of accentuating differences between workable and unworkable kinds of perturbation, in our quest to mold the universe to match our mystique, we portray the central consideration to be randomization versus nonrandomization).

The tabu search orientation evidently contrasts with this perspective. As manifest in the probabilistic TS variant, elements subjected to random influence are preferably to be strongly confined, and uses of randomization are preferably to be modulated through well differentiated probabilities. In short, the situations where randomization finds a place are very highly structured. From this point of view, God may play with dice, but beyond any question, the dice are *loaded*.

5.7 Some Basic Memory Structures for Longer-Term Strategies

To give a foundation for describing fundamental types of memory structures for longer-term strategies, we first briefly review the form of the recency-based memory structure introduced in Sect. 3 for handling add/drop moves. However, we slightly change the notation to provide a convenient way to refer to a variety of other types of moves.

5.7.1 Conventions

Let $S = \{1, 2, \dots, s\}$ denote an index set for a collection of solution attributes. For example, the indexes $i \in S$ may correspond to indexes of zero-one variables x_i , or they may be indexes of edges that may be added to or deleted from a graph, or the job indexes in a production scheduling problem. More precisely, by the attribute;element distinction discussed in Sect. 3, the attributes referenced by S in these cases consist of the specific values assigned to the variables, the specific add/drop states adopted by the edges, or positions occupied by the jobs. In general, to give a correspondence with developments of Sect. 4, an index $i \in S$ can summarize more detailed information, for example, by referring to an ordered pair (j, k) that summarizes a value assignment $x_j = k$ or the assignment of job j to position k . Hence, broadly speaking, the index i may be viewed as a notational convenience for representing a pair or a vector.

To keep our description at the simplest level, suppose that each $i \in S$ corresponds to a 0–1 variable x_i . As before, we let Iter denote the counter that identifies the current iteration, which starts at 0 and increases by 1 each time a move is made.

For recency-based memory, following the approach indicated in Sect. 3, when a move is executed that causes a variable x_i to change its value, we record $\text{TabuStart}(i) = \text{Iter}$ immediately after updating the iteration counter (This means that if the move has resulted in $x_i = 1$, then the attribute $x_i = 0$ becomes tabu-active at the iteration $\text{TabuStart}(i)$). Further, we let $\text{TabuTenure}(i)$ denote the number of iterations this attribute will remain tabu-active. Thus, by our previous design, the recency-based tabu criterion says that the previous value of x_i is tabu-active throughout all iterations such that

$$\text{TabuStart}(i) + \text{TabuTenure}(i) \leq \text{Iter}.$$

Similarly, in correspondence with earlier remarks, the value $\text{TabuStart}(i)$ can be set to 0 before initiating the method, as a convention to indicate no prior history exists. Then, we automatically avoid assigning a tabu-active status to any variable with $\text{TabuStart}(i) = 0$ (since the starting value for variable x_i has not yet been changed).

5.7.2 Frequency-Based Memory

By our foregoing conventions, allowing the set $S = \{1, \dots, s\}$ for illustration purposes to refer to indexes of 0–1 variables x_i , we may indicate structures to handle frequency-based memory as follows.

Transition frequency-based memory is by far the simplest to handle. A transition memory, $Transition(i)$, to record the number of times x_i changes its value, can be maintained simply in the form of a counter for x_i that is incremented at each move where such a change occurs. Since x_i is a zero-one variable, $Transition(i)$ also discloses the number of times x_i changes to and from each of its possible assigned values. In more complex situations, by the conventions already noted, a matrix memory $Transition(j, k)$ can be used to determine numbers of transitions involving assignments such as $x_j = k$. Similarly, a matrix memory may be used in the case of the sequencing problem where both the index of job j and position k may be of interest. In the context of the Min k -Tree problem, an array dimensioned by the number of edges can maintain a transition memory to keep track of the number of times that specific edges have been brought in and out of the solution. A matrix based on the edges can also identify conditional frequencies. For example, the matrix $Transition(j, k)$ can be used to count the number of times edge j replaced edge k . It should be kept in mind in using transition frequency memory that penalties and inducements are often based on *relative* numbers (rather than absolute numbers) of transitions, hence requiring that recorded transition values are divided by the total number of iterations (or the total number of transitions). As noted earlier, other options include dividing by the current maximum transition value. Raising transition values to a power, as by squaring, is often useful to accentuate the differences in relative frequencies.

Residence memory requires only slightly more effort to maintain than transition memory, by taking advantage of the recency-based memory stored in $TabuStart(i)$. The following approach can be used to track the number of solutions in which $x_i = 1$, thereby allowing the number of solutions in which $x_i = 0$ to be inferred from this. Start with $Residence(i) = 0$ for all i . Then, whenever x_i changes from 1 to 0, after updating $Iter$ but before updating $TabuStart(i)$, set

$$Residence(i) = Residence(i) + Iter - TabuStart(i).$$

Then, during iterations when $x_i = 0$, $Residence(i)$ correctly stores the number of earlier solutions in which $x_i = 1$. During iterations when $x_i = 1$, the true value of $Residence(i)$ is the right-hand side of the preceding assignment; however, the update only has to be made at the indicated points when x_i changes from 1 to 0. [Table 9](#) illustrates how this memory structure works when used to track the assignments of a variable x during 100 iterations. The variable is originally assigned to a value of zero by a construction procedure that generates an initial solution. In iteration 10, a move is made that changes the assignment of x from zero to one; however, the $Residence$ value remains at zero. $Residence$ is updated at iterations 22 and 73, when moves are made that change the assignment of x from 1 to 0. At iteration 65, for example, x has received a value of 1 for 27 iterations (i.e., $Residence + Iter - TabuStart = 12 + 65 - 50 = 27$), while at iteration 90, the count is 35 (i.e., the value of $Residence$).

As with transition memory, residence memory should be translated into a relative as a basis for creating penalties and inducements.

Table 9 Illustrative residence memory

<i>Iter</i>	<i>Assignment</i>	<i>Residence</i>
0	$x = 0$	0
10	$x = 1$	0
22	$x = 0$	$22 - 10 = 12$
50	$x = 1$	12
73	$x = 0$	$12 + 73 - 50 = 35$

The indicated memory structures can readily be applied to multivalued variables (or multistate attributes) by the extended designs illustrated in Sect. 4. In addition, the 0–1 format can be adapted to reference the number of times (and last time) a more general variable changed its value, which leads to more restrictive tabu conditions and more limiting (“stronger”) uses of frequency-based memory than by referring separately to each value the variable receives. As in the case of recency-based memory, the ability to affect larger numbers of alternative moves by these more aggregated forms of memory can be useful for larger problems, not only for conserving memory space but also for providing additional control over solutions generated.

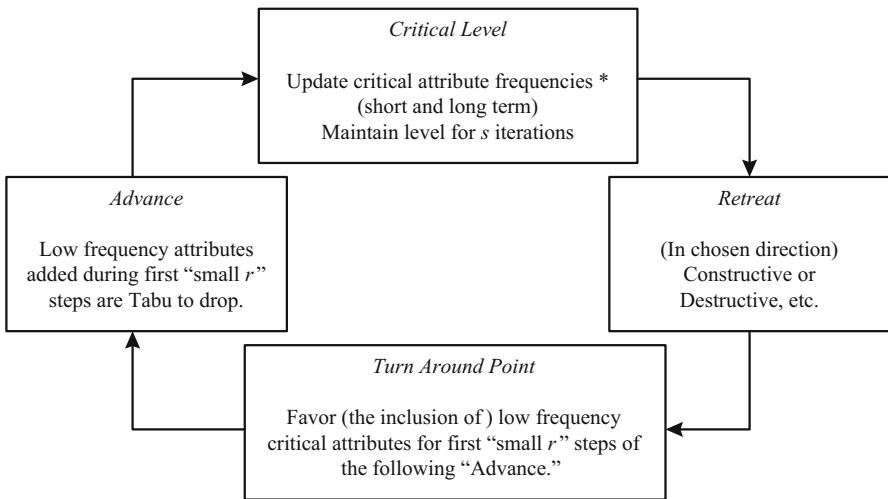
5.7.3 Critical Event Memory

Strategic oscillation offers an opportunity to make particular use of both short-term and long-term frequency-based memory. To illustrate, let $A(\text{Iter})$ denote a zero-one vector whose j th component has the value 1 if attribute j is present in the current solution and has the value 0 otherwise. The vector A can be treated “as if” it is the same as the solution vector for zero-one problems, though implicitly it is twice as large, since $x_j = 0$ is a different attribute from $x_j = 1$. This means that rules for operating on the full A must be reinterpreted for operating on the condensed form of A . The sum of the A vectors over the most recent t critical events provides a simple memory that combines recency and frequency considerations. To maintain the sum requires remembering $A(k)$, for k ranging over the last t iterations. Then the sum vector A^* can be updated quite easily by the incremental calculation:

$$A^* = A^* + A(\text{Iter}) - A(\text{Iter} - t + 1).$$

Associated frequency measures, as noted earlier, may be normalized, in this case, for example, by dividing A^* by the value of t . A long-term form of A^* does not require storing the $A(k)$ vectors but simply keeps a running sum. A^* can also be maintained by exponential smoothing.

Such frequency-based memory is useful in strategic oscillation where critical events are chosen to be those of generating a complete (feasible) construction or in general of reaching the targeted boundary (or a best point within a boundary region). Instead of using a customary recency-based TS memory at each step of an oscillating pattern, greater flexibility results by disregarding tabu restrictions until reaching the turning point, where the oscillation process alters its course to follow a path toward the boundary. At this point, assume a choice rule is applied to introduce an attribute that was not contained in any recent solution at the critical level. If this



* For selected part of critical level iterations: e.g., for first and best solutions of current block

Fig. 22 Strategic oscillation illustrative memory

attribute is maintained in the solution by making it tabu to be dropped, then upon eventually reaching the critical level, the solution will be different from any seen over the horizon of the last t critical events. Thus, instead of updating A^* at each step, the updating is done only for critical level solutions, while simultaneously enhancing the flexibility of making choices.

In general, the possibility occurs that no attribute exists that allows this process to be implemented in the form stated. That is, every attribute may already have a positive associated entry in A^* . Thus, at the turn-around point, the rule instead is to choose a move that introduces attributes which are least frequently used (Note that “infrequently used” can mean either “infrequently present” or “infrequently absent,” depending upon the current direction of oscillation). This again can be managed conveniently by using penalties and inducements. Such an approach has been found very effective for multidimensional knapsack problems and 0–1 quadratic optimization problems in [30, 33].

For greater diversification, this rule can be applied for r steps after reaching the turnaround point. Normally, r should be a small number, for example, with a baseline value of 1 or 2, which is periodically increased in a standard diversification pattern. Shifting from a short-term A^* to a long-term A^* creates a global diversification effect. A template for this approach is given in Fig. 22.

The approach of Fig. 22 is not symmetric. An alternative form of control is to seek immediately to introduce a low-frequency attribute upon leaving the critical level, to increase the likelihood that the solution at the next turn around will not duplicate a solution previously visited at that point. Such a control enhances diversity, though duplication at the turn around will already be inhibited by starting from different solutions at the critical level.

6 Connections, Hybrid Approaches, and Learning

Relationships between tabu search and other procedures like simulated annealing and genetic algorithms provide a basis for understanding similarities and contrasts in their philosophies and for creating potentially useful hybrid combinations of these approaches. We offer some speculation on preferable directions in this regard and also suggest how elements of tabu search can add a useful dimension to neural network approaches.

From the standpoint of evolutionary strategies, we trace connections between population-based models for combining solutions, as in genetic algorithms, and ideas that emerged from surrogate constraint approaches for exploiting optimization problems by combining constraints. We show how this provides the foundation for methods that give additional alternatives to genetic-based frameworks, specifically as embodied in the scatter search approach, which is the “primal complement” to the dual strategy of surrogate constraint approaches. Recent successes by integrating scatter search (and its path-relinking extensions) with tabu search disclose potential advantages for evolutionary strategies that incorporate adaptive memory.

Finally, we describe the learning approach called target analysis, which provides a way to determine decision parameters for deterministic and probabilistic strategies – and thus affords an opportunity to create enhanced solution methods.

6.1 Simulated Annealing

The contrasts between simulated annealing and tabu search are fairly conspicuous, though undoubtedly, the most prominent is the focus on exploiting memory in tabu search that is absent from simulated annealing. The introduction of this focus entails associated differences in search mechanisms and in the elements on which they operate. Accompanying the differences directly attributable to the focus on memory, and also magnifying them, several additional elements are fundamental for understanding the relationship between the methods. We consider three such elements in order of increasing importance.

First, tabu search emphasizes scouting successive neighborhoods to identify moves of high quality, as by candidate list approaches of the form described in Sect. 4. This contrasts with the simulated annealing approach of randomly sampling among these moves to apply an acceptance criterion that disregards the quality of other moves available (Such an acceptance criterion provides the sole basis for sorting the moves selected in the SA method). The relevance of this difference in orientation is accentuated for tabu search, since its neighborhoods include linkages based on history and therefore yield access to information for selecting moves that is not available in neighborhoods of the type used in simulated annealing.

Next, tabu search evaluates the relative attractiveness of moves not only in relation to objective function change but in relation to additional factors that represent quality, which are balanced over time with factors that represent influence. Both types of measures are affected by the differentiation among move attributes, as

embodied in tabu-activation rules and aspiration criteria, and in turn by relationships manifested in recency, frequency, and sequential interdependence (hence, again, involving recourse to memory). Other aspects of the state of search also affect these measures, as reflected in the altered evaluations of strategic oscillation, which depend on the direction of the current trajectory and the region visited.

Finally TS emphasizes guiding the search by reference to multiple thresholds, reflected in the tenures for tabu-active attributes and in the conditional stipulations of aspiration criteria. This may be contrasted to the simulated annealing reliance on guiding the search by reference to the single threshold implicit in the temperature parameter. The treatment of thresholds by the two methods compounds this difference between them. Tabu search varies its thresholds nonmonotonically, reflecting the conception that multidirectional parameter changes are essential to adapt to different conditions and to provide a basis for locating alternatives that might otherwise be missed. This contrasts with the simulated annealing philosophy of adhering to a temperature parameter that only changes monotonically.

Hybrids are now emerging that are taking preliminary steps to bridge some of these differences, particularly in the realm of transcending the simulated annealing reliance on a monotonic temperature parameter. A hybrid method that allows temperature to be strategically manipulated, rather than progressively diminished, has been shown to yield improved performance over standard SA approaches. A hybrid method that expands the SA basis for move evaluations also has been found to perform better than standard simulated annealing. Consideration of these findings invites the question of whether removing the memory scaffolding of tabu search and retaining its other features may yield a viable method in its own right. For example, experience cited in some of the studies reported in Glover and Laguna [31] suggests that, while a memoryless version of tabu search called tabu thresholding can outperform a variety of alternative heuristics, it generally does not match the performance of TS methods that appropriately exploit memory.

6.2 Genetic Algorithms

Genetic algorithms offer a somewhat different set of comparisons and contrasts with tabu search. GAs are based on selecting subsets (traditionally pairs) of solutions from a population, called parents, and combining them to produce new solutions called children. Rules of combination to yield children are based on the genetic notion of crossover, which in the classical form consists of interchanging solution values of particular variables, together with occasional operations such as random value changes. Children that pass a survivability test, probabilistically biased to favor those of superior quality, are then available to be chosen as parents of the next generation. The choice of parents to be matched in each generation is based on random or biased random sampling from the population (in some parallel versions executed over separate subpopulations whose best members are periodically exchanged or shared). Genetic terminology customarily refers to solutions as chromosomes, variables as genes, and values of variables as alleles.

By means of coding conventions, the genes of genetic algorithms may be compared to attributes in tabu search. Introducing memory in GAs to track the history of genes and their alleles over subpopulations would provide an immediate and natural way to create a hybrid with TS.

Some important differences between genes and attributes are worth noting, however. The implicit differentiation of attributes into *from* and *to* components, each having different memory functions, does not have a counterpart in genetic algorithms. A *from* attribute is one that is part of the current solution but is not included in the next solution once a move is made. A *to* attribute is one that is not part of the current solution but becomes part of the next solution once a move is made. The lack of this type of differentiation in GAs results because these approaches are organized to operate without reference to moves (although, strictly speaking, combination by crossover can be viewed as a special type of move).

A contrast to be noted between genetic algorithms and tabu search arises in the treatment of context, that is, in the consideration given to structure inherent in different problem classes. For tabu search, context is fundamental, embodied in the interplay of attribute definitions and the determination of move neighborhoods and in the choice of conditions to define tabu restrictions. Context is also implicit in the identification of amended evaluations created in association with longer-term memory and in the regionally dependent neighborhoods and evaluations of strategic oscillation.

At the opposite end of the spectrum, GA literature has traditionally stressed the freedom of its rules from the influence of context. Crossover, in particular, is supposedly a *context neutral* operation, which assumes no reliance on conditions that solutions must obey in a particular problem setting, just as genes make no reference to the environment as they follow their instructions for recombination (except, perhaps, in the case of mutation). Practical application, however, generally renders this an inconvenient assumption, making solutions of interest difficult to find. Consequently, a good deal of effort in GA implementation is devoted to developing “special crossover” operations that compensate for the difficulties created by context, effectively reintroducing it on a case-by-case basis.

The chief method by which modern genetic algorithms handle structure is by relegating its treatment to some other method. For example, genetic algorithms combine solutions by their parent-children processes at one level, and then a descent method may be introduced to operate on the resulting solutions to produce new solutions. These new solutions in turn are submitted to be recombined by the GA processes. In these versions, genetic algorithms already take the form of hybrid methods. Hence, there is a natural basis for marrying GA and TS procedures in such approaches. But genetic algorithms and tabu search also can be joined in a more fundamental way.

Specifically, tabu search strategies for intensification and diversification are based on the following question: How can information be extracted from a set of good solutions to help uncover additional (and better) solutions? From one point of view, GAs provide an approach for answering this question, consisting of putting solutions together and interchanging components (in some loosely defined sense,

if traditional crossover is not strictly enforced). Tabu search, by contrast, seeks an answer by utilizing processes that specifically incorporate neighborhood structures into their design.

Augmented by historical information, neighborhood structures are used as a basis for applying penalties and incentives to induce attributes of good solutions to become incorporated into current solutions. Consequently, although it may be meaningless to interchange or otherwise incorporate a set of attributes from one solution into another in a wholesale fashion, as attempted in traditional GA recombination operations, a stepwise approach to this goal through the use of neighborhood structures is entirely practicable. This observation provides a motive for creating *structured combinations* of solutions that embody desired characteristics such as feasibility – as is automatically achieved by the TS approach of path relinking discussed in Sect. 5. Instead of being compelled to create new types of crossover to remove deficiencies of standard operators upon being confronted by changing contexts, this approach addresses context directly and makes it an essential part of the design for generating combinations.

The current trend of genetic algorithms seems to be increasingly compatible with this perspective and could provide a basis for a useful hybrid combination of genetic algorithm and tabu search ideas. However, a fundamental question emerges, as posed in the development of the next sections, about whether there is any advantage to introducing genetic crossover-based ideas over introducing the apparently more flexible and exploitable path-relinking ideas.

6.2.1 Models of Nature: Beyond “Genetic Metaphors”

An aspect of tabu search that is often misunderstood concerns the relation between a subset of its strategies and certain approaches embodied in genetic algorithms. TS researchers have tended sometimes to overlook the part of the adaptive memory focus that is associated with strategies for combining sets of elite solutions. Complementing this, GA researchers have been largely unaware that such a collection of strategies outside their domain exists. This has quite possibly been due to the influence of the genetic metaphor, which on the one hand has helped to launch a number of useful problem-solving ideas and on the other hand has also sometimes obscured fertile connections to ideas that come from different foundations.

To understand the relevant ties, it is useful to go back in time to examine the origins of the GA framework and of an associated set of notions that became embodied in TS strategies. We will first sketch the original genetic algorithm design (see Fig. 23), as characterized in Holland [38]. Our description is purposely somewhat loose to be able to include approaches more general than the specific proposals that accompanied the introduction of GAs. Many variations and changes have come about over the years, as we subsequently observe.

A somewhat different model for combining elements of a population comes from a class of relaxation strategies in mathematical optimization known as surrogate constraint methods [19]. The goal of these approaches is to generate new constraints that capture information not contained in the original problem constraints taken independently, but which is implied by their union. We will see

Fig. 23 Genetic algorithm template

- 1) Begin with a population of binary vectors.
- 2) Operate repeatedly on the current generation of vectors, for a selected number of steps, choosing two “parent vectors” at random. Then mate the parents by exchanging certain of their components to produce offspring. (The exchange, called “crossover,” was originally designed to reflect the process by which chromosomes exchange components in genetic mating and, in common with the step of selecting parents themselves, was organized to rely heavily on randomization. In addition, a “mutation” operation is occasionally allowed to flip bits at random.)
- 3) Apply a measure of fitness to decide which offspring survive to become parents for the next generation. When the selected number of matings has been performed for the current generation, return to the start of Step 2 to initiate the mating of the resulting new set of parents.
- 4) Carry out the mating-and-survival operation of Steps 2 and 3 until the population becomes stable or until a chosen number of iterations has elapsed.

that some unexpected connections emerge between this development and that of genetic algorithms.

The information-capturing focus of the surrogate constraint framework has the aim of developing improved methods for solving difficult optimization problems by means of (a) providing better criteria for choice rules to guide a search for improved solutions and (b) inferring new bounds (constraints with special structures) to limit the space of solutions examined (The basic framework and strategies for exploiting it are given in Glover [19–21], Greenberg and Pierskalla [36,37], Karwan and Rardin [40,41], and Freville and Plateau [15,16]). Based on these objectives, the generation of new constraints proceeds as indicated in Fig. 24.

A natural first impression is that the surrogate constraint design is quite unrelated to the GA design, stemming from the fact that the concept of combining constraints seems inherently different from the concept of combining vectors. However, in many types of problem formulations, including those where surrogate constraints were first introduced, constraints are summarized by vectors. More particularly, over time, as the surrogate constraint approach became embedded in both exact and heuristic methods, variations led to the creation of a “primal counterpart” called *scatter search*. The scatter search approach combines solution vectors by rules patterned after those that govern the generation of new constraints and specifically inherits the strategy of exploiting linear combinations and inference [22].

6.3 Scatter Search

The scatter search process, building on the principles that underlie the surrogate constraint design, is organized to (1) capture information not contained separately

Fig. 24 Surrogate constraint template

- 1) Begin with an initial set of problem constraints (chosen to characterize all or a special part of the feasible region for the problem considered).
- 2) Create a measure of the relative influence of the constraints as basis for combining subsets to generate new constraints. The new (surrogate) constraints, are created from nonnegative linear combinations of other constraints, together with cutting planes inferred from such combinations. (The goal is to determine surrogate constraints that are most effective for guiding the solution process.)
- 3) Change the way the constraints are combined, based on the problem constraints that are not satisfied by trial solutions generated relative to the surrogate constraints, accounting for the degree to which different source constraints are violated. Then process the resulting new surrogate constraints to introduce additional inferred constraints obtained from bounds and cutting planes. (Weaker surrogate constraints and source constraints that are determined to be redundant are discarded.)
- 4) Change the way the constraints are combined, based on the problem constraints that are not satisfied by trial solutions generated relative to the surrogate constraints, accounting for the degree to which different source constraints are violated. Then process the resulting new surrogate constraints to introduce additional inferred constraints obtained from bounds and cutting planes. (Weaker surrogate constraints and source constraints that are determined to be redundant are discarded.)

in the original vectors and (2) take advantage of auxiliary heuristic solution methods to evaluate the combinations produced and to generate new vectors.

The original form of scatter search may be sketched as in Fig. 25.

Three particular features of scatter search deserve mention. First, the linear combinations are structured according to the goal of generating weighted centers of selected subregions, allowing for nonconvex combinations that project these centers into regions external to the original reference solutions. The dispersion pattern created by such centers and their external projections is particularly useful for mixed-integer optimization. Second, the strategies for selecting particular subsets of solutions to combine in Step 2 are designed to make use of clustering, which allows different types of strategic variation by generating new solutions “within clusters” and “across clusters.” Third, the method is organized to use supporting heuristics that are able to start from infeasible solutions and hence which remove the restriction that solutions selected as starting points for reapplying the heuristic processes must be feasible. In sum, scatter search is founded on the following premises.

- (P1) Useful information about the form (or location) of optimal solutions is typically contained in a suitably diverse collection of elite solutions.

Fig. 25 Scatter search procedure

- 1) Generate a starting set of solution vectors by heuristic processes designed for the problem considered, and designate a subset of the best vectors to be *reference solutions*. (Subsequent iterations of this step, transferring from Step 3 below, incorporate advanced starting solutions and best solutions from previous history as candidates for the reference solutions.)
- 2) Create new points consisting of linear combinations of subsets of the current reference solutions. The linear combinations are:
 - (a) chosen to produce points both inside and outside the convex regions spanned by the reference solutions.
 - (b) modified by generalized rounding processes to yield integer values for integer-constrained vector components.
- 3) Extract a collection of the best solutions generated in Step 2 to be used as starting points for a new application of the heuristic processes of Step 1. Repeat these steps until reaching a specified iteration limit.

- (P2) When solutions are combined as a strategy for exploiting such information, it is important to provide for combinations that can extrapolate beyond the regions spanned by the solutions considered and further to incorporate heuristic processes to map combined solutions into new points (This serves to provide both diversity and quality).
- (P3) Taking account of multiple solutions simultaneously, as a foundation for creating combinations, enhances the opportunity to exploit information contained in the union of elite solutions.

The fact that the heuristic processes of scatter search are not restricted to a single uniform design, but represent a varied collection of procedures, affords additional strategic possibilities. This theme also shares a link with the original surrogate constraint proposal, where heuristics for surrogate relaxations are introduced to improve the application of exact solution methods. In combination, the heuristics are used to generate strengthened surrogate constraints and, iteratively applied, to generate trial solutions for integer programming problems.

The catalog in Fig. 26 traces the links between the conceptions underlying scatter search and conceptions that have been introduced over time as amendments to the GA framework.

These innovations in the GA domain, which have subsequently been incorporated in a wide range of studies, are variously considered to be advances or heresies according to whether they are viewed from liberal or traditional perspectives. Significantly, their origins are somewhat diffuse, rather than integrated within a single framework.

It is clear that a number of the elements of the scatter search approach remain outside of the changes brought about by these proposals. A simple example is the approach of introducing adaptive rounding processes for mapping fractional

Fig. 26 Scatter search features (1977) incorporated into non-traditional GA approaches

- Introduction of “flexible crossover operations.” (Scatter search combinations include all possibilities generated by the early GA crossover operations, and also include all possibilities embedded in the more advanced “uniform” and “Bernoulli” crossovers (Ackley (1987), Spears and DeJong (1991)). Path relinking descendants of scatter search provide further possibilities, noted subsequently.)
- Use of heuristic methods to improve solutions generated from processes for combining vectors (Muhlenbein et al. (1988), Ulster et al. (1991)), (Whitley, Gordon and Mathias (1994)).
- Exploitation of vector representations that are not restricted to binary representations (Davis (1989), Eschelman and Schaffer (1992)).
- Introduction of special cases of linear combinations for operating on continuous vectors (Davis (1989), Wright (1990), Bäck et al. (1991), Michalewicz and Janikow (1991)).
- Use of combinations of more than two parents simultaneously to produce offspring (Eiben et al. (1994), Mühlenbein and Voigt (1996)).
- Introduction of strategies that subdivide the population into different groupings (Mühlenbein and Schlierkamp-Voosen (1994)).

components into integers. There also has conspicuously been no GA counterpart to the use of clustering to create strategic groupings of points, nor (as a result) to the notion of combining points according to distinctions between membership in different clusters (The closest approximation to this has been the use of “island populations” that evolve separately but without concern for analyzing or subdividing populations based on inference and clustering).

The most important distinction, however, is the link between scatter search and the theme of exploiting history. The prescriptions for combining solutions within scatter search are part of a larger design for taking advantage of information about characteristics of previously generated solutions to guide current search. In retrospect, it is perhaps not surprising that such a design should share an intimate association with the surrogate constraint framework, with its emphasis on extracting and coordinating information across different solution phases. This orientation, which takes account of elements such as the recency, frequency, and quality of particular value assignments, clearly shares a common foundation with notions incorporated within tabu search (The same reference on surrogate constraint strategies that is the starting point for scatter search is also often cited as a source of early TS conceptions). By this means, the link between tabu search and so-called “evolutionary” approaches also becomes apparent. The term *evolutionary* has undergone an interesting evolution of its own. By a novel turn, the term

“mutation” in the GA terminology has become reinterpreted to refer to any form of change, including the purposeful change produced by a heuristic process. As a result, all methods that apply heuristics to multiple solutions, whether or not they incorporate strategies for combining solutions, are now considered kindred to genetic algorithms, and the enlarged collection is labeled “evolutionary methods” (This terminology accordingly has acquired the distinction of embracing nearly every kind of method conceivable).

6.3.1 Modern Forms and Applications of Scatter Search

Recent implementations of scatter search (cited below) have taken advantage of the implicit learning capabilities provided by the tabu search framework, leading to refined methods for determining reference points and for generating new points. Current scatter search versions have also introduced more sophisticated mechanisms to map fractional values into integer values. This work is reinforced by new theorems about searches over spaces of zero-one integer variables. Special models have also been developed to allow both heuristic and exact methods to transform infeasible trial points into feasible points. Finally, scatter search is the source of the broader class of path-relinking methods, as described in [Sect. 5](#), which offer a wide range of mechanisms for creating productive combinations of reference solutions. A brief summary of some of these developments appears in [Fig. 27](#).

Implementation of various components of these extensions has provided advances for solving general nonlinear mixed discrete optimization problems with both linear and nonlinear constraints, as noted in the references cited under Recommended Reading.

- Tabu search memory is used to select current reference points from a historical pool (Glover, 1989, 1994a).
- Tabu search intensification and diversification strategies guide the generation of new points (Fleurent et al. 1996; Glover, Laguna and Marti, 2000).
- Solutions generated as “vector combinations” are further improved by explicit tabu search guidance (Trafalidis and Al-Harkan, 1995; Glover, Kelly and Laguna, 1996; Fleurent et al., 1996; Cung, et al. 1997).
- Directional rounding processes focus the search for feasible zero-one solutions allowing them to be mapped into convex subregions of hyperplanes produced by valid cutting plane inequalities (Glover, 1995a).
- Neural network learning is applied to filter out promising and unpromising points for further examination, and pattern analysis is used to predict the location of promising new solutions (Glover, Kelly and Laguna, 1996).
- Mixed integer programming models generate sets of diversified points, and yield refined procedures for mapping infeasible points into feasible points (Glover, Kelly and Laguna, 1996).
- Structured combinations of points take the role of linear combinations, to expand the range of alternatives generated (Glover, 1994a).

Fig. 27 Scatter search extensions

6.3.2 Scatter Search and Path-Relinking Interconnections

The relation between scatter search and path relinking sheds additional light on the character of these approaches. As already remarked, path relinking is a direct extension of scatter search. The way this extension comes about is as follows.

From a spatial orientation, the process of generating linear combinations of a set of reference points may be characterized as generating paths between and beyond these points (where points on such paths also serve as sources for generating additional points). This leads to a broader conception of the meaning of *combinations* of points. That is, by natural extension, we may conceive such combinations to arise by generating paths between and beyond selected points in neighborhood space, rather than in Euclidean space.

The form of these paths in neighborhood space is easily specified by reference to attribute-based memory, as used in tabu search. The path-relinking strategy thus emerges as a direct consequence. Just as scatter search encompasses the possibility to generate new solutions by weighting and combining more than two reference solutions at a time, path relinking includes the possibility to generate new solutions by multi-parent path constructions that incorporate attributes from a set of guiding solutions, where these attributes are weighted to determine which moves are given higher priority, as we have seen in Sect. 5. The name *path relinking* comes from the fact that the generation of such paths in neighborhood space characteristically “relinks” previous points in ways not achieved in the previous search history.

The relevance of these concepts as a foundation for evolutionary procedures is illustrated by recent applications of scatter search and path relinking which have disclosed the promise of these approaches for solving a variety of optimization problems. A sampling of such applications includes:

- Vehicle routing – Rochat and Taillard [64]; Taillard [66]
- Quadratic assignment – Cung et al. [7]
- Financial product design – Consiglio and Zenios [6]
- Neural network training – Kelly et al. [42]
- Job shop scheduling – Yamada and Nakano [71]
- Flow shop scheduling – Yamada and Reeves [72]
- Graph drawing – Laguna and Martí [46]
- Linear ordering – Laguna et al. [48]
- Unconstrained continuous optimization – Fleurent et al. [14]
- Bit representation – Rana and Whitley [62]
- Optimizing simulation – Glover et al. [32]
- Complex system optimization – Laguna [43]

It is additionally useful to note that reexpressing scatter search relative to neighborhood space – as done in path relinking – also leads to more general forms of scatter search in Euclidean space. The form of path relinking manifested in vocabulary building (which results by using constructive and destructive neighborhoods to create and reassemble components of solutions) also suggests the relevance of combining solutions in Euclidean space by allowing different linear combinations to be created for different solution components. The design considerations that underlie vocabulary building generally carry over to this particular instance (see [31]).

The broader conception of solution combinations provided by path relinking has useful implications for evolutionary procedures. The exploitation of neighborhood space and attribute-based memory gives specific, versatile mechanisms for achieving such combinations and provides a further interesting connection between tabu search proposals and genetic algorithm proposals. In particular, many recently developed “crossover operators,” which have no apparent relation between each other in the GA setting, can be shown to arise as special instances of path relinking, by restricting attention to two reference points (taken as parents in GAs) and by replacing the strategic neighborhood guidance of path relinking with a reliance on randomization. In short, the options afforded by path relinking for combining solutions are more unified, more systematic, and more encompassing than those provided by the “crossover” concept, which changes from instance to instance and offers no guidance for how to take advantage of any given context.

6.4 Greedy Randomized Adaptive Search Procedures (GRASP)

The GRASP methodology was developed in the late 1980s, and the acronym was coined by Tom Feo [13]. It was first used to solve computationally difficult set-covering problems [12]. Each GRASP iteration consists of constructing a trial solution and then applying an exchange procedure to find a local optimum (i.e., the final solution for that iteration). The construction phase is iterative, greedy, and adaptive. It is iterative because the initial solution is built considering one element at a time. It is greedy because the addition of each element is guided by a greedy function. It is adaptive because the element chosen at any iteration in a construction is a function of those previously chosen (That is, the method is adaptive in the sense of updating relevant information from iteration to iteration, as in most constructive procedures). The improvement phase typically consists of a local search procedure.

For illustration purposes, consider the design of a GRASP for the 2-partition problem (see, e.g., [44]). This problem consists of clustering the nodes of a weighted graph into two equal-sized sets such that the weight of the edges between the two sets is minimized. In this context, the iterative, greedy, and adaptive elements of the GRASP construction phase may be interpreted as follows. The initial solution is built considering one node at a time. The addition of each node is guided by a greedy function that minimizes the augmented weight of the partition. The node chosen at any iteration in the construction is a function of the adjacencies of previously chosen nodes. There is also a probabilistic component in GRASP that is applied to the selection of elements during the construction phase. After choosing the first node for one set, all nonadjacent nodes are of equal quality with respect to the given greedy function. If one of those nodes is chosen by some deterministic rule, then every GRASP iteration will repeat this selection. In such stages within a construction where there are multiple greedy choices, choosing any one of them will not compromise the greedy approach, yet each will often lead to a very different solution.

To generalize this strategy, consider forming a *candidate list* (at each stage of the construction) consisting of high-quality elements according to an adaptive greedy function. Then, the next element to be included in the initial solution is randomly selected from this list. A similar strategy has been categorized as a cardinality-based semi-greedy heuristic.

The solution generated by a greedy randomized adaptive construction can generally be improved by the application of an improvement phase following selected construction phases, as by using a descent method based on an exchange mechanism, since usually the result of the construction phase is not a local minimum with respect to simple exchange neighborhoods. There is an obvious computational trade-off between the construction and improving phases. An intelligent construction requires fewer improving exchanges to reach a local optimum, and therefore, it results in a reduction of the total CPU time required per GRASP iteration. The exchange mechanism can also be used as a basis for a hybrid method, as by incorporating elements of other methodologies such as simulated annealing or tabu search. In particular, given that the GRASP constructions inject a degree of diversification to the search process, the improvement phase may consist of a short-term memory tabu search that is fine-tuned for intensification purposes. Other connections may be established with methods such as scatter search or the path-relinking strategy of tabu search, by using the GRASP constructions (or their associated local optima) as reference points.

Performing multiple GRASP iterations may be interpreted as a means of strategically sampling the solution space. Based on empirical observations, it has been found that the sampling distribution generally has a mean value that is inferior to the one obtained by a deterministic construction, but the best over all trials dominates the deterministic solution with a high probability. The intuitive justification of this phenomenon is based on the ordering statistics of sampling. GRASP implementations are generally robust in the sense that it is difficult to find or devise pathological instances for which the method will perform arbitrarily bad. The robustness of this method has been well documented in applications to production, flight scheduling, equipment and tool selection, location, and maximum independent sets.

An interesting connection exists between GRASP and probabilistic tabu search (PTS). If PTS is implemented in a memoryless form and restricted to operate only in the constructive phase of a multistart procedure (stripping away memory, and even probabilistic choice, from the improving phase), then a procedure resembling GRASP results. The chief difference is that the probabilities used in PTS are rarely chosen to be uniform over members of the candidate list, but generally seek to capture variations in the evaluations, whenever these variations reflect anticipated differences in the effective quality of the moves considered.

This connection raises the question of whether a multistart variant of probabilistic tabu search may offer a useful alternative to memoryless multistart approaches like GRASP. A study of this issue for the quadratic assignment problem, where GRASP has been reported to perform well, was conducted by Fleurent and Glover [14]. To provide a basis for comparison, the improving phases of the PTS multistart

method excluded the use of TS memory and guidance strategies and were restricted to employ a standard descent procedure. Probabilistic tabu search mechanisms were used in the constructive phases, incorporating frequency-based intensification to improve the effectiveness of successive constructions. The resulting multistart method proved significantly superior to other multistart approaches previously reported for the quadratic assignment problem. However, it also turned out to be not as effective as the leading tabu search methods that use memory in the improving phases as well as (or instead of) in the constructive phases. Nevertheless, it seems reasonable to conjecture that classes of problems exist where increased reliance on restarting will prove advantageous and where the best results may be obtained from appropriately designed multistart strategies such as based on greedy randomized search and multistart variants of PTS.

6.5 Neural Networks

Neural networks have a somewhat different set of goals than tabu search, although some overlaps exist. We indicate how tabu search can be used to extend certain neural net conceptions, yielding a hybrid that may have both hardware and software implications. The basic transferable insight from tabu search is that memory components with dimensions such as recency and frequency can increase the efficacy of a system designed to evolve toward a desired state. We suggest the merit of fusing neural network memory with tabu search memory as follows (A rudimentary acquaintance with neural network ideas is assumed).

Recency-based considerations can be introduced from tabu search into neural networks by a *time-delay feedback loop* from a given neuron back to itself (or from a given synapse back to itself, by the device of interposing additional neurons). This permits firing rules and synapse weights to be changed only after a certain time threshold, determined by the length of the feedback loop. Aspiration thresholds of the form conceived in tabu search can be embodied in inputs transmitted on a secondary level, giving the ability to override the time delay for altering firing thresholds and synaptic weights. Frequency-based effects employed in tabu search similarly may be incorporated by introducing a form of cumulative averaged feedback.

Time-delay feedback mechanisms for creating recency and frequency effects also can have other functions. In a problem-solving context, for example, it may be convenient to disregard one set of options to concentrate on another, while retaining the ability to recover the suppressed options after an interval. This familiar type of human activity is not a customary part of neural network design, but can be introduced by the time-dependent functions previously indicated. In addition, a threshold can be created to allow a suppressed option to “go unnoticed” if current activity levels fall in a certain range, effectively altering the interval before the option reemerges for consideration. Neural network designs to incorporate those features may directly make use of the TS ideas that have made these elements effective in the problem-solving domain.

Tabu search strategies that introduce longer-term intensification and diversification concerns are also relevant to neural network processes. As a foundation for blending these approaches, it is useful to adopt an orientation where a collection of neurons linked by synapses with various activation weights is treated as a set of attribute variables which can be assigned alternative values. Then the condition that synapse j (from a specified origin neuron to a specified destination neuron) is assigned an activation weight in interval p can be coded by the assignment $y_j = p$, where y_j is a component of an attribute vector y as identified in the discussion of attribute creation processes in connection with vocabulary building. A similar coding identifies the condition under which a neuron fires (or does not fire) to activate its associated synapses. As a neural network process evolves, a sequence of these attribute vectors is produced over time. The association between successive vectors may be imagined to operate by reference to a neighborhood structure implicit in the neural architecture and associated connection weights. There also may be an implicit association with some (unknown) optimization problem or a more explicit association with a known problem and set of constraints. In the latter case, attribute assignments (neuron firings and synapse activation) can be evaluated for efficacy by transformation into a vector x , to be checked for feasibility by $x \in \mathbf{X}$ (We maintain a distinction between y and x since there may not be a one-one association between them).

Time records identifying the quality of outcomes produced by recent firings, and identifying the frequency particular attribute assignments produce the highest quality firing outcomes, yield a basis for delaying changes in certain weight assignments and for encouraging changes in others. The concept of influence, in the form introduced in tabu search, should be considered in parallel with quality of outcomes.

Early designs to incorporate tabu search into neural networks are provided in the work of de Werra and Hertz [9] and Beyer and Ogier [5]. These applications, which respectively treat visual pattern identification and nonconvex optimization, are reported to significantly reduce training times and increase the reliability of outcomes generated. More recent uses of tabu search to enhance the function of neural networks are provided by the studies reported in Glover and Laguna [31].

6.6 Target Analysis

Target analysis [29] links artificial intelligence and operation research perspectives to give heuristic or exact solution procedures the ability to learn what rules are best to solve a particular class of problems. Many existing solution methods have evolved by adopting, *a priori*, a somewhat limited characterization of appropriate rules for evaluating decisions. An illustration is provided by restricting the definition of a “best” move to be one that produces the most attractive objective function change. However, this strategy does not guarantee that the selected move will lead the search in the direction of the optimal solution. In fact, in some settings, it has been shown

that the merit of such a decision rule diminishes as the number of iterations increases during a solution attempt.

As seen earlier, the tabu search philosophy is to select a best admissible move (from a strategically controlled candidate list) at each iteration, interpreting best in a broad sense that goes beyond the use of objective function measures, and relies upon historical parameters to aid in composing an appropriate evaluation. Target analysis provides a means to exploit this broader view. For example, target analysis can be used to create a dynamic evaluation function that incorporates a systematic process for diversifying the search over the longer term.

A few examples of the types of questions that target analysis can be used to answer are:

1. Which decision rule from a collection of proposed alternatives should be selected to guide the search? (In an expanded setting, how should the rules from the collection be combined? By interpreting “decision rule” broadly, this encompasses the issue of selecting a neighborhood, or a combination of neighborhoods, as the source of a move at a given stage). Similarly, which parameter values should be chosen to provide effective instances of the decision rules?
2. What attributes are most relevant for determining tabu status, and what associated tabu restrictions, tabu tenures, and aspiration criteria should be used?
3. What weights should be assigned to create penalties or inducements (e.g., as a function of frequency-based memory), and what thresholds should govern their application?
4. Which measures of quality and influence are most appropriate, and which combinations of these lead to the best results in different search phases?
5. What features of the search trajectory disclose when to focus more strongly on intensification and when to focus more strongly on diversification? (In general, what is the best relative emphasis between intensification and diversification, and under what conditions should this emphasis change?)

Motivation for using target analysis to answer such questions is provided by contrasting target analysis with the way answers are normally determined. Typically, an experimenter begins with a set of alternative rules and decision criteria which are intended to capture the principal elements of a given method, often accompanied by ranges of associated parameters for implementing the rules. Then various combinations of options are tried to see how each one works for a preliminary set of test problems. However, even a modest number of rules and parameters may create a large number of possibilities in combination, and there is usually little hope of testing these with any degree of thoroughness. As a result, such testing for preferred alternatives generally amounts to a process of blind groping. Where methods boast the lack of optional parameters and rules, typically it is because the experimenter has already done the advance work to settle upon a particular combination that has been hardwired for the user, at best with some degree of adaptiveness built in, but the process that led to this hardwiring still raises the prospect that another set of options may be preferable.

More importantly, in an adaptive memory approach, where information from the history of the search is included among the inputs that determine current

choices, a trial-and-error testing of parameters may overlook key elements of timing and yield no insights about relationships to be exploited. Such a process affords no way to uncover or characterize the circumstances encountered during the search that may cause a given rule to perform well or badly and consequently gives no way to anticipate the nature of rules that may perform better than those originally envisioned. Target analysis replaces this by a systematic approach to create *hindsight before the fact* and then undertakes to “reverse engineer” the types of rules that will lead to good solutions.

6.6.1 Target Analysis Features

The main features of target analysis may briefly be sketched by viewing the approach as a five-phase procedure (see Fig. 28). *Phase 1* of target analysis is devoted to applying existing methods to determine optimal or exceptionally high-quality solutions to representative problems of a given class. In order to allow subsequent analysis to be carried out more conveniently, the problems are often selected to be relatively small, provided this can be done in a way to assure these problems will exhibit features expected to be encountered in hard problems from the class examined.

Although this phase is straightforward, the effort allotted to obtaining solutions of the specified quality will generally be somewhat greater than would be allotted during the normal operation of the existing solution procedures, in order to assure that the solutions have the quality sought (Such an effort may be circumvented in cases where optimal solutions to a particular testbed of problems are known in advance).

Phase 2 uses the solutions produced by Phase 1 as *targets*, which become the focus of a new set of solution passes. During these passes, each problem is solved again, this time scoring all available moves (or a high-ranking subset) on the basis of their ability to progress effectively toward the target solution. The scoring can be a simple classification, such as “good” or “bad,” or it may capture more refined gradations. In the case where multiple best or near-best solutions may reasonably qualify as targets, the scores may be based on the target that is “closest to” the current solution.

In some implementations, choices during Phase 2 are biased to select moves that have high scores, thereby leading to a target solution more quickly than the customary choice rules. In other implementations, the method is simply allowed to make its regular moves. In either case, the goal is to generate information during this solution effort which may be useful in inferring the solution scores. That is, the scores provide a basis for creating modified evaluations – and more generally, for creating new rules to generate such evaluations in order to more closely match them with the measures that represent “true goodness” (for reaching the targets).

In the case of tabu search intensification strategies such as elite solution recovery approaches, scores can be assigned to parameterized rules for determining the types of solutions to be saved. For example, such rules may take account of characteristics of clustering and dispersion among elite solutions. In environments where data bases can be maintained of solutions to related problems previously encountered, the

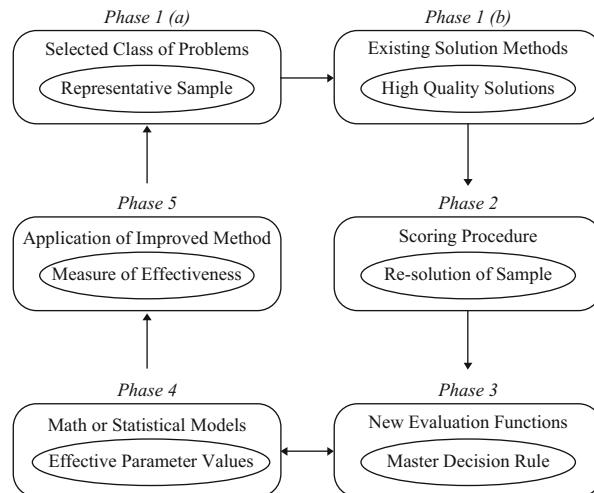


Fig. 28 Overview of the target analysis methodology

scores may be assigned to rules for recovering and exploiting particular instances of these past solutions and for determining which new solutions will be added to the data bases as additional problems are solved (The latter step, which is part of the target analysis and not part of the solution effort, can be performed “off line.”) An integration of target analysis with a generalized form of sensitivity analysis for these types of applications has been developed and implemented in financial planning and industrial engineering by Glover et al. [34]. Such designs are also relevant, for example, in applications of linear and nonlinear optimization based on simplex method subroutines, to identify sets of variables to provide crash-basis starting solution.

In path-relinking strategies, scores can be applied to rules for matching initiating solutions with guiding solutions. As with other types of decision rules produced by target analysis, these will preferably include reference to parameters that distinguish different problem instances. The parameter-based rules similarly can be used to select initiating and guiding solutions from preexisting solutions pools. Tunneling applications of path relinking, which allow traversal of infeasible regions, and strategic oscillation designs that purposely drive the search into and out of such regions, are natural accompaniments for handling recovered solutions that may be infeasible.

Phase 3 constructs parameterized functions of the information generated in *Phase 2*, with the goal of finding values of the parameters to create a *master decision rule*. This rule is designed to choose moves that score highly, in order to achieve the goal that underlies *Phase 2*. It should be noted that the parameters available for constructing a master decision rule depend on the search method employed. Thus, for example, tabu search may include parameters that embody various elements of recency-based and frequency-based memory, together with measures of influence

linked to different classes of attributes or to different regions from which elite solutions have been derived.

Phase 4 transforms the general design of the master decision rule into a specific design by applying a model to determine effective values for its parameters. This model can be a simple set of relationships based on intuition or can be a more rigorous formulation based on mathematics or statistics (such as a goal programming or discriminant analysis model, or even a “connectionist” model based on neural networks).

The components of phases 2, 3, and 4 are not entirely distinct and may be iterative. On the basis of the outcomes of these phases, the master decision rule becomes the rule that drives the solution method. In the case of tabu search, this rule may use feedback of outcomes obtained during the solution process to modify its parameters for the problem being solved.

Phase 5 concludes the process by applying the master decision rule to the original representative problems and to other problems from the chosen solution class to confirm its merit. The process can be repeated and nested to achieve further refinement.

Target analysis has an additional important function. On the basis of the information generated during its application, and particularly during its final confirmation phase, the method produces empirical frequency measures for the probabilities that choices with high evaluations will lead to an optimal (or near-optimal) solution within a certain number of steps. These decisions are not only at tactical levels but also at strategic levels, such as when to initiate alternative solution phases and which sources of information to use for guiding these phases (e.g., whether from processes for tracking solution trajectories or for recovering and analyzing solutions). By this means, target analysis can provide inferences concerning expected solution behavior, as a supplement to classical “worst case” complexity analysis. These inferences can aid the practitioner by indicating how long to run a solution method to achieve a solution desired quality, according to specified empirical probability.

One of the useful features of target analysis is its capacity for taking advantage of human interaction. The determination of key parameters, and the rules for connecting them, can draw directly on the insight of the observer as well as on supplementary analytical techniques. The ability to derive inferences from preestablished knowledge of optimal or near-optimal solutions, instead of manipulating parameters blindly (without information about the relation of decisions to targeted outcomes), can save significant investment in time and energy. The key, of course, is to coordinate the phases of solution and guided re-solution to obtain knowledge that has the greatest utility. Many potential applications of target analysis exist, and recent applications suggest the approach holds considerable promise for developing improved decision rules for difficult optimization problems.

6.6.2 Illustrative Application and Implications

An application of target analysis to a production scheduling problem [45] provides a basis for illustrating some of the relevant considerations of the approach. In this study, the moves consisted of a combination of swap and insert moves, and scores

were generated to identify the degree to which a move brought a solution closer to the target solution (which consisted of the best-known solution before improving the method by means of target analysis). In the case of a swap move, for example, a move might improve or worsen (or, by the measure used, leave unchanged) the “positional value” of each component of the swap, and by the simplification of assigning scores of 1, 0, or -1 to each component, a move could accordingly receive a score ranging from 2 to -2 . The application of target analysis then proceeded by tracking the scores of the 10 highest evaluation moves at each iteration, to determine the circumstances under which the highest evaluations tended to correspond to the highest scores. Both tabu and non-tabu moves were included in the analysis, to see whether tabu status was also appropriately defined.

At an early stage of the analysis, a surprising relationship emerged. Although the scores of the highest evaluation non-tabu moves ranged across both positive and negative values, the positive values were largely associated with moves that improved the schedule while the negative values were largely associated with moves that worsened the schedule. In short, the highest evaluations were significantly more “accurate” (corresponded more closely to high scores) during phases where the objective function value of the schedule improved than during phases when it deteriorated.

A simple diversification strategy was devised to exploit this discovery. Instead of relying on the original evaluations during “disimproving phases,” the strategy supplemented the evaluations over these intervals by assigning penalties to moves whose component jobs had been moved frequently in the past. The approach was initiated at a local optimum after the progress of the search began to slow (as measured by how often a new best solution was found) and was de-activated as soon as a move was executed that also was an improving move (to become reactivated the next time that all available moves were disimproving moves). The outcome was highly effective, producing new solutions that were often superior to the best previously found, especially for larger problems, and also finding the highest quality solutions more quickly.

The success of this application, in view of its clearly limited scope, provides an incentive for more thorough applications. For example, a more complete analysis would reasonably proceed by first seeking to isolate the high-scoring moves during the disimproving phases and to determine how frequency-based memory and other factors could be used to identify these moves more effectively. Comparisons between evaluations proposed in this manner and their associated move scores would then offer a foundation for identifying more intelligent choices. Classifications to segregate the moves based on criteria other than “improving” and “disimproving” could also be investigated. Additional relevant factors that may profitably be taken into account are examined in the illustration of the next subsection.

A Hypothetical Illustration. The following hypothetical example embodies a pattern related to the one uncovered in the scheduling application cited above. However, the pattern in this case is slightly more ambiguous and less clearly points to options that it may be exploited.

Table 10 Moves throughout the search history

Move rank	1	2	3	4	5
Percent of moves with “good” scores	22	14	10	20	16

Table 11 Moves during improving phases

Move rank	1	2	3	4	5
Percent of moves with “good” scores	34	21	9	14	7

Table 12 Moves during disimproving phases

Move rank	1	2	3	4	5
Percent of moves with “good” scores	8	7	11	26	25

For simplicity in this illustration, suppose that moves are scored to be either “good” or “bad” (If each move changes the value of a single 0-1 variable, for instance, a move may be judged good or bad depending on whether the assigned value is the same as in the target solution. More generally, a threshold can be used to differentiate the two classifications).

Table 10 indicates the percent of time each of the five highest evaluation moves, restricting attention in this case to those that are non-tabu, receives a good score during the search history (At a first stage of conducting the target analysis, this history could be for a single hard problem or for a small collection of such problems). The *move rank* in the table ranges from 1 to 5, corresponding to the highest evaluation move, the 2nd highest evaluation move, and so on to the fifth highest evaluation move.

The indicated percent values do not total 100 because good scores may also be assigned to moves with lower evaluations, whose ranks are not included among those shown. Also, it may be expected that some non-tabu moves will also receive good scores (A fuller analysis would similarly show ranks and scores for these moves).

At first glance, the table appears to suggest that the fourth- and fifth-ranked moves are almost as good as the first-ranked move, although the percent of moves that receives good scores is not particularly impressive for any of the ranks. Without further information, a strategy might be contemplated that allocates choices probabilistically among the first-, fourth-, and fifth-ranked moves (though such an approach would not be assured to do better than choosing the first ranked move at each step). **Tables 11** and **12** below provide more useful information about choices that are potentially favorable, by dividing the iterations into improving and disimproving phases as in the scheduling study previously discussed.

These tables are based on a hypothetical situation where improving and disimproving moves are roughly equal in number, so that the percent values shown in Table 9.1 are the average of the corresponding values in Tables 9.2 and 9.3 (For definiteness, moves that do not change the problem objective function may be assumed to be included in the improving phase, though a better analysis might treat them separately).

The foregoing outcomes to an extent resemble those found in the scheduling study, though with a lower success rate for the highest evaluation improving moves.

Clearly, [Tables 11](#) and [12](#) give information that is more exploitable than the information in [Table 10](#). According to these latter tables, it would be preferable to focus more strongly on choosing one of the two highest evaluation moves during an improving phase and one of the fourth or fifth highest evaluation moves during a disimproving phase. This conclusion is still weak in several respects, however, and we examine considerations that may lead to doing better.

Refining the Analysis. The approach of assigning scores to moves, as illustrated in [Tables 10–12](#), disregards the fact that some solution attributes (such as assignments of values to particular 0–1 variables) may be fairly easy to choose “correctly,” while others may be somewhat harder. Separate tables of the type illustrated should therefore be created for easy and hard attributes (as determined by how readily their evaluations lead to choices that would generate the target solution), since the preferred rules for evaluating moves may well differ depending on the types of attributes the moves contain. Likewise, an effective strategy may require that easy and hard attributes become the focus of different search phases. The question therefore arises as to how to identify such attributes.

As a first approximation, we may consider an easy attribute to be one that often generates an evaluation that keeps it out of the solution if it belongs out or that brings it into the solution if it belongs in. A hard attribute behaves oppositely. Thus, a comparison between frequency-based memory and move scores gives a straightforward way to differentiate these types of attributes. Both residence and transition frequencies are relevant, though residence measures are probably more usually appropriate. For example, an attribute that belongs to the current solution a high percentage of the time, and that also belongs to the target solution, would evidently qualify as easy. On the other hand, the number of times the attribute is accepted or rejected from the current solution may sometimes be less meaningful than how long it stays in or out. The fact that residence and transition frequencies are characteristically used in tabu search makes them conveniently available to assist in differentiations that can improve the effectiveness of target analysis.

6.6.3 Conditional Dependencies Among Attributes

[Tables 10–12](#) suggest that the search process that produced them is relatively unlikely to find the target solution. Even during improving phases, the highest evaluation move is almost twice as likely to be bad as good. However, this analysis is limited and discloses a limitation of the tables themselves. In spite of first appearances, it is entirely possible that these tables could be produced by a search process that successfully obtains the target solution (by a rule that chooses a highest evaluation move at each step). The reason is that the relation between scores and evaluations may change over time. While there may be fairly long intervals where choices are made poorly, there may be other shorter intervals where the choices are made more effectively – until eventually one of these shorter intervals succeeds in bringing all of the proper attributes into the solution.

Such behavior is likely to occur in situations where correctly choosing some attributes may pave the way for correctly choosing others. The interdependence

of easy and hard attributes previously discussed is carried a step farther by these conditional relationships, because an attribute that at one point deserves to be classified hard may later deserve to be classified easy, once the appropriate foundations are laid.

Instead of simply generating tables that summarize results over long periods of the search history, therefore, it can be important to look for blocks of iterations where the success rate of choosing good moves may differ appreciably from the success rate overall. These blocks provide clues about intermediate solution compositions that may transform hard attributes into easy ones and thus about preferred sequences for introducing attributes that may exploit conditional dependencies. The natural step then is to see which additional types of evaluation information may independently lead to identifying such sequences.

A simple instance of this type of effect occurs where the likelihood that a given attribute will correctly be selected (to enter or leave the solution) depends roughly on the number of attributes that already correctly belong to the solution. In such situations, the appropriate way to determine a “best choice” is therefore also likely to depend on this number of attributes correctly in solution. Even though such information will not generally be known during the search, it may be possible to estimate it and adjust the move evaluations accordingly. Such relationships, as well as the more general ones previously indicated, are therefore worth ferreting out by target analysis.

6.6.4 Differentiating Among Targets

In describing the steps of target analysis, it has already been noted that scores should not always be rigidly determined by only one specific target, but may account for alternative targets, and in general may be determined by the target that is closest to the current solution (by a metric that depends on the context). Acknowledging that there may be more than one good solution that is worth finding, such a differentiation among targets can prove useful. Yet even in the case where a particular solution is uniquely the one to be sought (as where its quality may be significantly better than that of all others known), alternative targets may still be valuable to consider in the role of intermediate solutions and may provide a springboard to finding additional solutions that are better. Making reference to intermediate targets is another way of accounting for the fact that conditional dependencies may exist among the attributes, as previously discussed. However, such dependencies in some cases may be more easily exploited by explicitly seeking constructions at particular stages that may progressively lead to a final destination.

Some elite solutions may provide better targets than others because they are easier to obtain – completely apart from developing strategies to reach ultimate targets by means of intermediate ones. However, some care is needed in making the decision to focus on such easier targets as a basis for developing choice rules. As in the study of Lokketangen and Glover [51], it may be that focusing instead on the harder targets will yield rules that likewise cause the easier targets to be found more readily, and these rules may apply to a wider spectrum of problems than those derived by focusing on easier targets.

6.6.5 Generating Rules by Optimization Models

Target analysis can use optimization models to generate decision rules by finding weights for various decision criteria to create a composite (master) rule. To illustrate, let G and B , respectively, denote index sets for good moves and bad moves, as determined from move scores, as in the classification embodied in [Tables 10–12](#). Incorporate the values of the different decision criteria in a vector A_i for $i \in G$ and $i \in B$; that is, the j th component a_{ij} of A_i is the value assigned to move i by the decision criterion j . These components need not be the result of rules, but can simply correspond to data considered relevant to constructing rules. In the tabu search setting, such data can include elements of recency-based and frequency-based memory. Then we may consider a master rule which is created by applying a weight vector w to each vector A_i to produce a composite decision value $A_i w = \sum_j a_{ij} w_j$. An ambitious objective is to find a vector w that yields

$$\begin{aligned} A_i w &> 0 \text{ for } i \in G \\ A_i w &\leq 0 \text{ for } i \in B. \end{aligned}$$

If such a weight vector w could be found, then all good moves would have higher evaluations by the composite criterion than all bad moves, which of course is normally too much to ask. A step toward formulating a more reasonable goal is as follows. Let $G(\text{iter})$ and $B(\text{iter})$ identify the sets G and B for a given iteration iter . Then an alternative objective is to find a w so that, at each such iteration, at least one $i \in G(\text{iter})$ would yield

$$A_i w > A_k w \text{ for all } k \in B(\text{iter})$$

or equivalently

$$\text{Max}\{A_i w : i \in G(\text{iter})\} > \text{Max}\{A_k w : k \in B(\text{iter})\}.$$

This outcome would insure that a highest evaluation move by the composite criterion will always be a good move. Naturally, this latter goal is still too optimistic. Nevertheless, it is possible to devise goal programming models (related to LP discriminant analysis models) that can be used to approximate this goal. A model of this type has proved to be effective for devising branching rules to solve a problem of refueling nuclear reactors [[17](#)].

A variety of opportunities exist for going farther in such strategies. For example, issues of creating nonlinear and discontinuous functions to achieve better master rules can be addressed by using trial functions to transform components of A_i vectors into new components, guided by LP sensitivity and postoptimality analysis. Target analysis ideas previously indicated can also be useful in this quest.

The range of possibilities for taking advantage of target analysis is considerable, and for the most part, only the most rudimentary applications of this learning approach have been initiated. The successes of these applications make further exploration of this approach attractive.

7 Neglected Tabu Search Strategies

We briefly review several key strategies in tabu search that are often neglected (especially in beginning studies) but which are important for producing the best results.

Our purpose is to call attention to the relevance of particular elements that are mutually reinforcing but which are not always discussed “side by side” in the literature and which deserve special emphasis. In addition, observations about useful directions for future research are included.

A comment regarding implementation is as follows: First steps do not have to include the most sophisticated variants of the ideas discussed in the following sections, but the difference between “some inclusion” and “no inclusion” can be significant. Implementations that incorporate simple instances of these ideas will often disclose the manner in which refined implementations can lead to improved performance.

The material that follows brings together ideas described in preceding sections to provide a perspective on how they interrelate. In the process, a number of additional observations are introduced.

7.1 Candidate List Strategies

Efficiency and quality can be greatly affected by using intelligent procedures for isolating effective candidate moves, rather than trying to evaluate every possible move in a current neighborhood of alternatives. This is particularly true when such a neighborhood is large or expensive to examine. The gains to be achieved by using candidate lists have been widely documented, yet many TS studies overlook their relevance.

Careful organization in applying candidate lists, as by saving evaluations from previous iterations and updating them efficiently, can also be valuable for reducing overall effort. Time saved in these ways allows a chance to devote more time to higher-level features of the search.

While the basic theme of candidate lists is straightforward, there are some subtleties in the ways candidate list strategies may be used. Considerable benefit can result by being aware of fundamental candidate list approaches, such as the *subdivision strategy*, the *Aspiration Plus strategy*, the *elite candidate list strategy*, the *bounded change strategy* and the *sequential fan strategy* (as discussed in Sect. 4).

An effective integration of a candidate list strategy with the rest of a tabu search method will typically benefit by using TS memory designs to facilitate functions to

be performed by the candidate lists. This applies especially to the use of frequency-based memory. A major mistake of some TS implementations, whether or not they make use of candidate lists, is to consider only the use of recency-based memory. Frequency-based memory – which itself takes different forms in intensification phases and diversification phases – cannot only have a dramatic impact on the performance of the search in general but also can often yield gains in the design of candidate list procedures. A useful way to meld different candidate list procedures is described in Glover [27].

7.2 Intensification Approaches

Intensification strategies, which are based on recording and exploiting elite solutions or, characteristically, specific features of these solutions, have proved very useful in a variety of applications. Some of the relevant forms of such strategies and considerations for implementing them are as follows.

7.2.1 Restarting with Elite Solutions

The simplest intensification approach is the strategy of recovering elite solutions in some order, each time the search progress slows, and then using these solutions as a basis for reinitiating the search. The list of solutions that are candidates to be recovered is generally limited in size, often in the range of 20–40 (although in parallel processing applications, the number is characteristically somewhat larger). The size chosen for the list in serial TS applications also corresponds roughly to the number of solution recoveries anticipated to be done during the search and so may be less or more depending on the setting. When an elite solution is recovered from the list, it is removed, and new elite solutions are allowed to replace less attractive previous solutions – usually dropping the worst of the current list members. However, if a new elite solution is highly similar to a solution presently recorded, instead of replacing the current worst solution, the new solution will compete directly with its similar counterpart to determine which solution is saved.

This approach has been applied very effectively in job shop and flow shop scheduling, in vehicle routing, and in telecommunication design problems. One of the best approaches for scheduling applications keeps the old TS memory associated with the solution but makes sure the first new move away from this solution goes to a different neighbor than the one visited after encountering this solution the first time. Another effective variant does not bother to save the old TS memory but uses a probabilistic TS choice design.

The most common strategy is to go through the list from best to worst, but in some cases, it has worked even better to go through the list in the other direction. In this approach, it appears effective to allow two passes of the list. On the first pass, when a new elite solution is found that falls below the quality of the solution currently recovered, but which is still better than the worst already examined on the list, the method still adds the new solution to the list and displaces the worst solution. Then a second pass, after reaching the top of the list, recovers any added solutions not previously recovered.

7.2.2 Frequency of Elite Solutions

Another primary intensification strategy is to examine elite solutions to determine the frequency in which particular solution attributes occur (where the frequency is typically weighted by the quality of the solutions in which the attributes are found).

This strategy was originally formulated in the context of identifying “consistent” and “strongly determined” variables – where, loosely speaking, consistent variables are those more frequently found in elite solutions, while strongly determined variables are those that would cause the greatest disruption by changing their values (as sometimes approximately measured by weighting the frequencies based on solution quality). The idea is to isolate the variables that qualify as more consistent and strongly determined (according to varying thresholds) and then to generate new solutions that give these variables their “preferred values.” This can be done either by rebuilding new solutions in a multistart approach or by modifying the choice rules of an ongoing solution effort to favor the inclusion of these value assignments.

Keeping track of the frequency that elite solutions include particular attributes (such as edges of tours, assignments of elements to positions, and narrow ranges of values taken on by variables), and then favoring the inclusion of the highest frequency elements effectively allows the search to concentrate on finding the best supporting uses and values of other elements. A simple variant is to “lock in” a small subset of the most attractive attributes (value assignments) – allowing this subset to change over time or on different passes.

A Relevant Concern: In the approach that starts from a current (good) solution and tries to bring in favored elements, it is important to introduce an element that yields a best outcome from among the current contenders (where, as always, best is defined to encompass considerations that are not solely restricted to objective function changes). If an attractive alternative move shows up during this process, which does not involve bringing in one of these elements, aspiration criteria may determine whether such a move should be taken instead. Under circumstances where the outcome of such a move appears sufficiently promising, the approach may be discontinued and allowed to enter an improving phase (reflecting a decision that enough intensification has been applied and it is time to return to searching by customary means).

Intensification of this form makes it possible to determine what percent of “good attributes” from prior solutions should be included in the solution currently generated. It also gives information about which subsets of these attributes should go together, since it is preferable not to choose attributes during this process that cause the solution to deteriorate compared to other choices. This type of intensification strategy has proved highly effective in the settings of vehicle routing and zero-one mixed-integer optimization.

7.2.3 Memory and Intensification

It is clearly somewhat more dangerous to hold elements “in” solution than to hold them “out” (considering that a solution normally is composed of a small fraction of available elements – as where a tree contains only a fraction of the edges of

a graph). However, there is an important exception, previously intimated. As part of a longer-term intensification strategy, elements may be selected very judiciously to be “locked in” on the basis of having occurred with high frequency in the best solutions found. In that case, choosing different mutually compatible (and mutually reinforcing) sets to lock in can be quite helpful. This creates a *combinatorial implosion* effect (opposite to a combinatorial explosion effect) that shrinks the solution space to a point where best solutions over the reduced space are likely to be found more readily.

The key to this type of intensification strategy naturally is to select an appropriate set of elements to lock in, but the chances appear empirically to be quite high that some subset of those with high frequencies in earlier best solutions will be correct. Varying the subsets selected gives a significant likelihood of picking a good one (More than one subset can be correct, because different subsets can still be part of the same complete set). Aspiration criteria make it possible to drop elements that are supposedly locked in, to give this approach more flexibility.

7.2.4 Relevance of Clustering for Intensification

A search process over a complex space is likely to produce clusters of elite solutions, where one group of solutions gives high frequencies for one set of attributes and another group gives high frequencies for a different set. It is important to recognize this situation when it arises. Otherwise, there is a danger that an intensification strategy may try to compel a solution to include attributes that work against each other. This is particularly true in a strategy that seeks to generate a solution by incorporating a collection of attributes “all at once,” rather than using a step-by-step evaluation process that is reapplied at each move through a neighborhood space (Stepping through a neighborhood has the disadvantage of being slower but may compensate by being more selective. Experimentation to determine the circumstances under which each of these alternative intensification approaches may be preferable would be quite valuable).

A strategy that incorporates a block of attributes together may yield benefits by varying both the size and composition of the subsets of high-frequency “attractive” attributes, even if these attributes are derived from solutions that lie in a common cluster, since the truly best solutions may not include them all. Threshold-based forms of logical restructuring, as discussed in Sect. 4, may additionally lead to identifying elements to integrate into solutions that may not necessarily belong to solutions previously encountered. The vocabulary building theme becomes important in this connection. The relevance of clustering analysis for logical restructuring and vocabulary building is reinforced by the use of a related conditional analysis, which is examined subsequently in Sect. 7.5.

7.3 Diversification Approaches

Diversification processes in tabu search are sometimes applied in ways that limit their effectiveness, due to overlooking the fact that diversification is not just

“random” or “impulsive” but depends on a purposeful blend of memory and strategy. As noted in Sect. 4, recency- and frequency-based memory are both relevant for diversification. Historically, these ideas stem in part from proposals for exploiting surrogate constraint methods. In this setting, the impetus is not simply to achieve diversification but to derive appropriate weights in order to assure that evaluations will lead to solutions that satisfy required conditions (see Sect. 6). Accordingly, it is important to account for elements such as how often, to what extent, and how recently particular constraints have been violated, in order to determine weights that produce more effective valuations.

The implicit *learning effects* that underlie such uses of recency, frequency, and influence are analogous to those that motivate the procedures used for diversification (and intensification) in tabu search. Early strategic oscillation approaches exploited this principle by driving the search to various depths outside (and inside) feasibility boundaries and then employing evaluations and directional search to move toward preferred regions.

In the same way that these early strategies bring diversification and intensification together as part of a continuously modulated process, it is important to stress that these two elements should be interwoven in general. A common mistake in many TS implementations is to apply diversification without regard for intensification. “Pure” diversification strategies are appropriate for truly long-term strategies, but over the intermediate term, diversification is generally more effective if it is applied by heeding information that is also incorporated in intensification strategies. In fact, intensification by itself can sometimes cause a form of diversification, because intensifying over part of the space allows a broader search of the rest of the space. A few relevant concerns are as follows.

7.3.1 Diversification and Intensification Links

A simple and natural diversification approach is to keep track of the frequency that attributes occur in nonelite solutions, as opposed to solutions encountered in general, and then to periodically discourage the incorporation of attributes that have modest to high frequencies (giving greater penalties to larger frequencies). The reference to nonelite solutions tends to avoid penalizing attributes that would be encouraged by an intensification strategy.

More generally, for a “first-level” balance, an intermediate-term memory matrix may be used, where the high-frequency items in elite solutions are not penalized by the long-term values, but may even be encouraged. The trade-offs involved in establishing the degree of encouragement, or the degree of reducing the penalties, represent an area where a small amount of preliminary testing can be valuable. This applies as well to picking thresholds to identify high-frequency items (Simple guesses about appropriate parameter values can often yield benefits, and tests of such initial guesses can build an understanding that leads to increasingly effective strategies).

By extension, if an element has never or rarely been in a solution generated, then it should be given a higher evaluation for being incorporated in a diversification approach if it was “almost chosen” in the past but did not make the grade.

This observation has not been widely heeded, but is not difficult to implement, and is relevant to intensification strategies as well. The relevant concerns are illustrated in the discussion of “Persistent Attractiveness” and “Persistent Voting” in Chap. 7 of Glover and Laguna [31].

7.3.2 Implicit Conflict and the Importance of Interactions

Current evaluations also should not be disregarded while diversification influences are activated. Otherwise, a diversification process may bring elements together that conflict with each other and make it harder rather than easier to find improved solutions.

For example, a design that gives high penalties to a wide range of elements, without considering interactions, may drive the solution to avoid good combinations of elements. Consequently, diversification – especially in intermediate term phases – should be carried out for a limited number of steps, accompanied by watching for and sidestepping situations where indiscriminately applying penalties would create incompatibilities or severe deterioration of quality. To repeat the theme, even in diversification, attention to quality is important. And as in “medical remedies,” sometimes small doses are better than large ones. Larger doses (i.e., more radical departures from previous solutions), which are normally applied less frequently, can still benefit by coordinating the elements of quality and change.

7.3.3 Reactive Tabu Search

An approach called reactive tabu search (RTS) developed by Battiti and Tecchiolli [3, 4] deserves additional consideration as a way to achieve a useful blend of intensification and diversification. RTS incorporates hashing in a highly effective manner to generate attributes that are very nearly able to differentiate among distinct solutions. That is, very few solutions contain the same hashed attribute, applying standard hash function techniques. Accompanying this, Battiti and Tecchiolli use an automated tabu tenure, which begins with the value of 1 (preventing a hashed attribute from being reinstated if this attribute gives the “signature” of the solution visited on the immediately preceding step). This tenure is then increased if examination shows the method is possibly cycling, as indicated by periodically generating solutions that produce the same hashed attribute.

The tabu tenure, which is the same for all attributes, is increased exponentially when repetitions are encountered and decreased gradually when repetitions disappear. Under circumstances where the search nevertheless encounters an excessive number of repetitions within a given span (i.e., where a moving frequency measure exceeds a certain threshold), a diversification step is activated, which consists of making a number of random moves proportional to a moving average of the cycle length.

The reported successes of this approach invite further investigations of its underlying ideas and related variants. As potential bases for generating such variants, attributes created by hashing may be viewed as *fine grain* attributes, which give them the ability to distinguish among different solutions. By contrast, “standard” solution attributes, which are the raw material for hashing, may be viewed as *coarse grain*

attributes, since each may be contained in (and hence provide a signature for) many different solutions. Experience has shown that tabu restrictions based on coarse grain attributes are often advantageous for giving increased vigor to the search (There can exist a variety of ways of defining and exploiting attributes, particularly at coarser levels, which complicates the issue somewhat). This raises the question of when particular degrees of granularity are more effective than others.

It seems reasonable to suspect that fine grain attributes may yield greater benefits if they are activated in the vicinity of elite solutions, thereby allowing the search to scour “high-quality terrain” more minutely. This effect may also be achieved by reducing tabu tenures for coarse grain attributes – or basing tabu restrictions on attribute conjunctions – and using more specialized aspiration criteria. Closer scouring of critical regions can also be brought about by using strongly focused candidate list strategies, such as a sequential fan candidate list strategy (Empirical comparisons of such alternatives to hashing clearly would be of interest). On the other hand, as documented by Nonobe and Ibaraki [59, 60], the use of “extra coarse grain” attributes (those that prohibit larger numbers of moves when embodied in tabu restrictions) can prove advantageous for solving large problems over a broadly defined problem domain.

Another type of alternative to hashing also exists, which is to create new attributes by processes that are not so uniform as hashing. A potential drawback of hashing is its inability to distinguish the relative importance (and appropriate influence) of the attributes that it seeks to map into others that are fine grained. A potential way to overcome this drawback is to make use of vocabulary building [31] and of conditional analysis ([Sect. 7.5](#)).

7.4 Strategic Oscillation

A considerable amount has been written on strategic oscillation and its advantages. However, one of the uses of this approach that is frequently overlooked involves the idea of oscillating among alternative choice rules and neighborhoods. As stressed in [Sect. 5](#), an important aspect of strategic oscillation is the fact that there naturally arise different types of moves and choice rules that are appropriate for negotiating different regions and different directions of search. Thus, for example, there are many constructive methods in graph and scheduling problems, but strategic oscillation further leads to the creation of complementary “destructive methods” which can operate together with their constructive counterparts. Different criteria emerge as relevant for selecting a move to take on a constructive step versus one to take on a destructive step. Similarly, different criteria apply according to whether moves are chosen within a feasible region or outside a feasible region (and whether the search is moving toward or away from a feasibility boundary).

The variation among moves and evaluations introduces an inherent vitality into the search that provides one of the sources underlying the success of strategic oscillation approaches. This reinforces the motivation to apply strategic oscillation to the choice of moves and evaluation criteria themselves, selecting moves from a

pool of possibilities according to rules for transitioning from one choice to another. In general, instead of picking a single rule, a process of invoking multiple rules provides a range of alternatives that run all the way from “strong diversification” to “strong intensification.”

This form of oscillation has much greater scope than may at first be apparent, because it invokes the possibility of simultaneously integrating decision rules and neighborhoods, rather than only visiting them in a strategically determined sequence.

Such concepts are beginning to find counterparts in investigations being launched by the computer science community. The “agent” terminology is being invoked in such applications to characterize different choice mechanisms and neighborhoods as representing different agents. Relying on this representation, different agents then are assigned to work on (or “attack”) the problem serially or in parallel. The CS community has begun to look upon this as a significant innovation – unaware of the literature where such ideas were introduced a decade or more ago – and the potential richness and variation of these ideas still seems not to be fully recognized. For example, there have not yet been any studies that consider the idea of “strategically sequencing” rules and neighborhoods, let alone those that envision the notion of parametric integration. The further incorporation of adaptive memory structures to enhance the application of such concepts also lies somewhat outside the purview of most current CS proposals. At the same time, however, TS research has also neglected to conduct empirical investigations of the broader possibilities. This is clearly an area that deserves fuller study.

7.5 Clustering and Conditional Analysis

To reinforce the theme of identifying opportunities for future research, we provide an illustration to clarify the relevance of clustering and conditional analysis, particularly as a basis for intensification and diversification strategies in tabu search.

An Example: Suppose 40 elite solutions have been saved during the search and each solution is characterized as a vector x of zero-one variables x_j , for $j \in N = \{1, \dots, n\}$. Assume the variables that receive positive values in at least one of the elite solutions are indexed x_1 to x_{30} (Commonly in such circumstances, n may be expected to be somewhat larger than the number of positive valued variables, e.g., in this case, reasonable values may be $n = 100$ or $1,000$).

For simplicity, we restrict attention to a simple weighted measure of consistency which is given by the frequency that the variables x_1 to x_{30} receive the value 1 in these elite solutions (We temporarily disregard weightings based on solution quality and other aspects of “strongly determined” assignments). Specifically, assume the frequency measures are as shown in [Table 13](#).

Since each of x_1 to x_{15} receives a value of 1 in 24 of the 40 solutions, these variables tie for giving “most frequent” assignments. An intensification strategy that favors the inclusion of some number of such assignments would give equal

Table 13 Frequency measures

Variables $x_j = 1$	Number of solutions
x_1 to x_{15}	24
x_{16} to x_{20}	21
x_{21} to x_{25}	17
x_{26} to x_{30}	12

Table 14 Frequency measures for two subsets

Subset 1 (20 solutions)		Subset 2 (20 solutions)	
Variables $x_j = 1$	No. of solutions	Variables $x_j = 1$	No. of solutions
x_{11} to x_{15}	20	x_{16} to x_{20}	20
x_{21} to x_{25}	16	x_6 to x_{10}	16
x_1 to x_5	12	x_1 to x_5	12
x_6 to x_{10}	8	x_{26} to x_{30}	8
x_{26} to x_{30}	4	x_{11} to x_{15}	4
x_{16} to x_{20}	1	x_{21} to x_{25}	1

bias to introducing each of x_1 to x_{15} at the value 1 (Such a bias would typically be administrated either by creating modified evaluations or by incorporating probabilities based on such evaluations).

To illustrate the relevance of clustering, suppose the collection of 40 elite solutions can be partitioned into two subsets of 20 solutions each, whose characteristics are summarized in Table 14.

A very different picture now emerges. The variables x_1 to x_{15} no longer appear to deserve equal status as “most favored” variables. Treating them with equal status may be a useful source of diversification, as opposed to intensification, but the clustered data provide more useful information for diversification concerns as well. In short, clustering gives a relevant contextual basis for determining the variables (and combinations of variables) that should be given special treatment.

7.5.1 Conditional Relationships

To go a step beyond the level of differentiation provided by cluster analysis, it is useful to sharpen the focus by referring explicitly to interactions among variables. Such interactions can often be identified in a very straightforward way and can form a basis for more effective clustering. In many types of problems, the number of value assignments (or the number of “critical attributes”) needed to specify a solution is relatively small compared to the total number of problem variables (For example, in routing, distribution, and telecommunication applications, the number of links contained in feasible constructions is typically a small fraction of those contained in the underlying graph). Using a 0–1 variable representation of possibilities, it is not unreasonable in such cases to create a *cross-reference* matrix, which identifies variables (or coded attributes) that simultaneously receive a value of 1 in a specific collection of elite solutions.

To illustrate, suppose the index set $P = \{1, \dots, p\}$ identifies the variables x_j that receive a value of 1 in at least r solutions from the collection of elite solutions under consideration (Apart from other strategic considerations, the parameter r can also be used to control the size of p , since larger values of r result in smaller values of p).

Then create a $p \times p$ symmetric matrix \mathbf{M} whose entries m_{ij} identify the number of solutions in which x_i and x_j are both 1 (Thus, row M_i of \mathbf{M} represents the sum of the solution vectors in which $x_i = 1$, restricted to components x_j for $j \in P$). The value m_{ii} identifies the total number of elite solutions in which $x_i = 1$, and the value m_{ij}/m_{ii} represents the “conditional probability” that $x_j = 1$ in this subset of solutions. Because p can be controlled to be of modest size, as by the choice of r and the number of solutions admitted to the elite set, the matrix \mathbf{M} is not generally highly expensive to create or maintain.

By means of the conditional probability interpretation, the entries of \mathbf{M} give a basis for a variety of analyses and choice rules for incorporating preferred attributes into new solutions. Once an assignment $x_j = 1$ is made in a solution currently under consideration (which may be either partly or completely constructed), an updated conditional matrix \mathbf{M} can be created by restricting attention to elite solution vectors for which $x_j = 1$ (Restricted updates of this form can also be used for look-ahead purposes). Weighted versions of \mathbf{M} , whose entries additionally reflect the quality of solutions in which specific assignments occur, likewise can be used.

Critical event memory provides a convenient mechanism to maintain appropriate variation when conditional influences are taken into account. The “critical solutions” associated with such memory in the present case are simply those constituting a selected subset of elite solutions. Frequency measures for value assignments can be obtained by summing these solution vectors for problems with 0–1 representations, and the critical event control mechanisms can then assure assignments are chosen to generate solutions that differ from those of previous elite solutions.

Conditional analysis, independent of such memory structures, can also be a useful foundation for generating solution fragments to be exploited by vocabulary-building processes.

7.6 Referent-Domain Optimization

Referent-domain optimization is based on introducing one or more optimization models to strategically restructure the problem or neighborhood, accompanied by auxiliary heuristic or algorithmic process to map the solutions back to the original problem space. The optimization models are characteristically devised to embody selected heuristic goals (e.g., of intensification, diversification, or both), within the context of particular classes of problems.

There are several ways to control the problem environment as a basis for applying referent-domain optimization. A natural control method is to limit the structure and range of parameters that define a neighborhood (or the rules used to navigate through a neighborhood) and to create an optimization model that operates under these restricted conditions.

The examples that follow assume the approach starts from a current trial solution, which may or may not be feasible. The steps described yield a new solution, and then the step is repeated, using tabu search as a master guiding strategy to avoid cycling and to incorporate intensification and diversification.

Example 1 A heuristic selects k variables to change values, holding other variables constant. An exact method determines the (conditionally) optimal new values of the k -selected variables.

Example 2 A heuristic identifies a set of restrictive bounds that bracket the values of the variables in the current trial solution (where the bounds may compel some variables to take on a single value). An exact method determines an optimal solution to the problem as modified to include these bounds.

Example 3 A heuristic selects a restructured and exploitable region around the current solution to search for an alternative solution. An exact method finds the best solution in this region.

Example 4 For add/drop neighborhoods, a heuristic chooses k elements to add (or to drop). For example, the heuristic may operate by both adding and dropping k -specific elements, as in k -opt moves for the TSP or k -swap moves for graph bipartitioning that add and drop k nodes. Then, attention is restricted to consider only the subset of elements added or the subset of elements dropped (and further restricted in the case of a bipartitioning problem to just one of the two sets). Then an exact method identifies the remaining k elements to drop (or to add) that will complete the move optimally.

Example 5 A heuristic chooses a modified problem formulation that also admits the current trial solution as a trial solution (For example, the heuristic may relax some part of the formulation and/or restrict another part). An exact method then finds an optimal solution to the modified formulation. An illustration occurs where a two-phase exact algorithm first finds an optimal solution to a relaxed portion of the problem and then finds an optimal solution to a restricted portion. Finally, a small part of the feasible region of the original problem close to or encompassing this latter solution is identified, and an exact solution method finds an optimal solution in this region.

Example 6 The use of specially constructed neighborhoods (and aggregations or partitions of integer variables) permits the application of mixed-integer programming (MIP) models to identify the best options from all moves of depth at most k (or from associated collections of at most k variables). When k is sufficiently small, such MIP models can be quite tractable and produce moves considerably more powerful than those provided by lower level heuristics.

Example 7 In problems with graph-related structures, the imposition of directionality or nonlooping conditions gives a basis for devising generalized shortest path (or

dynamic programming) models to generate moves that are optimal over a significant subclass of possibilities. This type of approach gives rise to a combinatorial leverage phenomenon, where a low-order effort (e.g., linear or quadratic) can yield solutions that dominate exponential numbers of alternatives (See, e.g., [24, 61, 63]).

Example 8 A broadly applicable control strategy, similar to that of a relaxation procedure but more flexible, is to create a proxy model that resembles the original problem of interest and which is easier to solve. Such an approach must be accompanied with a method to transform the solution to the proxy model into a trial solution for the original problem. A version of such an approach, which also induces special structure into the proxy model, can be patterned after layered surrogate/Lagrangian decomposition strategies for mixed-integer optimization.

Referent-domain optimization can also be applied in conjunction with target analysis to create more effective solution strategies. In this case, a first-stage learning model, based on controlled solution attempts, identifies a set of desired properties of good solutions, together with target solutions (or target regions) that embody these properties. Then a second-stage model is devised to generate neighborhoods and choice rules to take advantage of the outcomes of the learning model. Useful strategic possibilities are created by basing these two models on a proxy model for referent-domain optimization, to structure the outcomes so that they may be treated by one of the control methods indicated in the foregoing examples.

8 Conclusion

It is natural to be tempted to implement the most rudimentary forms of a method. More than a few papers on tabu search examine only a small portion of the elements of short-term memory and examine little or nothing at all of longer-term memory. Unfortunately, in some cases, these papers also present themselves as embodying the essence of tabu search.

A factor that has reinforced the tendency to examine a limited part of tabu search (aside from convenience, which can be sensible in early stages of an investigation) is that such a focus has sometimes produced very appealing results. When reasonably decent outcomes can be found without great effort, the motive to look further is diminished. The danger, of course, lies in failing to discover significant gains that are likely to be achieved by a more complete approach.

It is appropriate to acknowledge that attention may be given to a limited subset of ideas from an overall search framework for the following reasons:

1. Such a focus may help to uncover a better form for the strategies associated with this subset.
2. Weaknesses of this subset, when studied in isolation from other ideas, may stand out more clearly, thus yielding insights into the features of a more complete approach that are required to produce a better method.

3. For methods which are susceptible to highly “modular” implementations, as typically occurs for tabu search, simpler designs can readily be made a part of more complex designs.

Nevertheless, in many settings, tabu search implementations that incorporate a more comprehensive set of its basic strategies typically perform appreciably better than implementations that restrict consideration to a narrow set of such strategies.

A great deal remains to be learned about tabu search. Evidently, we also still know very little about how we ourselves use memory in our problem solving. It is not inconceivable that discoveries about effective uses of memory within our search methods will provide clues about strategies that humans are adept at employing – or may advantageously be taught to employ. The potential links between the areas of heuristic search and psychology have scarcely been examined. Unquestionably, in the realm of optimization, we have not yet investigated the strategic possibilities at a level that comes close to disclosing their full potential. The numerous successes of tabu search implementations provide encouragement that such issues are profitable to probe more fully. Some of the opportunities and challenges involved are discussed in Glover [28].

Recent fundamental advances in applications of tabu search have been assembled in a collection of “Tabu Search Vignettes” which can be accessed via the internet at <http://spot.colorado.edu/~glover>. These include summaries of key developments in a variety of areas, including:

Constraint Solving and Its Applications (Resource Assignment, Planning and Timetabling, Integer Programming Feasibility, Satisfiability, Mobile Network Frequency Assignment)

Chemical Industry Applications (Computer Aided Molecular Design (CAMD), Heat Exchanger Network (HEN) Synthesis, Phase Equilibrium Calculations, Gibbs Free Energy Minimization, Optimal Component Lumping Problems)

Classification

Feature Selection

Satellite Range Scheduling

Maritime Transportation for International Trade

Conservation Area Network Design

High Level Synthesis

Graph Coloring

Delivery

Routing with Loading and Inventory Constraints

Heterogeneous Routing and Scheduling

Capacitated Facility Location

Multi-period Forest Harvesting

Manpower Scheduling

DNA Sequencing

Airline Disruption Management

Internet Traffic Engineering

Matrix Bandwidth Minimization

Generalized Assignment

Constraint Satisfaction (Work Shift Scheduling, Set-Covering and Nurse Scheduling)
Resource-Constrained Project Scheduling
Dynamic Optimization (Trade Market Prediction, Meteorological Forecast, Robotics Motion Control)

Additional topics and references related to tabu search, including these vignettes, will also be featured in the website <http://www.tabusearch.info/> which is scheduled to debut in November 2012.

Cross-References

- ▶ [Algorithms and Metaheuristics for Combinatorial Matrices](#)
 - ▶ [Binary Unconstrained Quadratic Optimization Problem](#)
 - ▶ [Fuzzy Combinatorial Optimization Problems](#)
 - ▶ [Neural Network Models in Combinatorial Optimization](#)
-

Recommended Reading

1. D. Ackley, *A Connectionist Model for Genetic Hillclimbing* (Kluwer Academic Publishers, Dordrecht, 1987)
2. T. Bäck, F. Hoffmeister, H. Schwefel, A survey of evolution strategies, in *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, ed. by R. Belew, L. Booker (1991), pp. 2–9
3. R. Battiti, G. Tecchiolli, Parallel based search for combinatorial optimization: genetic algorithms and tabu search. *Microprocess. Microsyst.* **16**, 351–367 (1992)
4. R. Battiti, G. Tecchiolli, The reactive tabu search. *ORSA J. Comput.* **6**(2), 126–140 (1994)
5. D. Beyer, R. Ogier, Tabu learning: a neural network search method for solving nonconvex optimization problems, in *Proceedings of the International Conference in Neural Networks* (IEEE/INNS, Singapore, 1991)
6. A. Consiglio, S.A. Zenios, Designing portfolios of financial products via integrated simulation and optimization models. *Oper. Res.* **47**(2), 195–208 (1999)
7. V.-D. Cung, T. Mautor, P. Michelon, A. Tavares, Scatter search for the Quadratic assignment problem, Laboratoire PRISM-CNRS URA 1525, 1996
8. L. Davis, Adapting operator probabilities in genetic algorithms, in *Proceedings of the Third International Conference on Genetic Algorithms* (Morgan Kaufmann, San Mateo, 1989), pp. 61–69
9. D. De Werra, A. Hertz, Tabu search techniques: a tutorial and applications to neural networks. *OR Spectrum* **11**, 131–141 (1989)
10. A.E. Eiben, P.-E. Raue, Z. Ruttkay, Genetic algorithms with multi-parent recombination, in *Proceedings of the third international conference on parallel problem solving from nature* (PPSN), ed. by Y. Davidor, H.-P. Schwefel, R. Manner (Springer, New York, 1994), pp. 78–87
11. L.J. Eschelman, J.D. Schaffer, Real-coded genetic algorithms and interval-schemata, Technical report, Phillips Laboratories, 1992
12. T. Feo, M.G.C. Resende, A probabilistic Heuristic for a computationally difficult set covering problem *Oper. Res. Lett.* **8**, 67–71 (1989)
13. T. Feo, M.G.C. Resende, Greedy randomized adaptive search procedures. *J. Global Opt.* **2**, 1–27 (1995)

14. C. Fleurent, F. Glover, P. Michelon, Z. Valli, A scatter search approach for unconstrained continuous optimization, in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation* (1996), pp. 643–648
15. A. Freville, G. Plateau, Heuristics and reduction methods for multiple constraint 0-1 linear programming problems. *Eur. J. Oper. Res.* **24**, 206–215 (1986)
16. A. Freville, G. Plateau, An exact search for the solution of the surrogate dual of the 0-1 bidimensional Knapsack problem. *Eur. J. Oper. Res.* **68**, 413–421 (1993)
17. F. Glover, D. Klingman, N Phillips, Netform modeling and applications, Special issue on the practice of mathematical programming. *Interfaces* **20**(1), 7–27 (1990)
18. F. Glover, Parametric combinations of local job shop rules. Chapter IV, ONR Research Memorandum no. 117, GSIA, Carnegie Mellon University, Pittsburgh, PA, 1963
19. F. Glover, A multiphase-dual algorithm for the zero-one integer programming problem. *Oper. Res.* **13**(6), 879–919 (1965)
20. F. Glover, Surrogate constraints. *Oper. Res.* **16**, 741–749 (1968)
21. F. Glover, Surrogate constraint duality in mathematical programming. *Oper. Res.* **23**, 434–451 (1975)
22. F. Glover, Heuristics for integer programming using surrogate constraints. *Decis. Sci.* **8**(1), 156–166 (1977)
23. F. Glover, Tabu search—part I. *ORSA J. Comput.* **1**, 190–206 (1989)
24. F. Glover, Ejection chains, reference structures and alternating path methods for traveling salesman problems. University of Colorado. Shortened version published in *Discret. Appl. Math.* **65**(1996), 223–253 (1992)
25. F. Glover, Genetic algorithms and scatter search: unsuspected potentials. *Stat. Comput.* **4**, 131–140 (1994)
26. F. Glover, Scatter search and star-paths: beyond the genetic metaphor. *OR Spektrum* **17**, 125–137 (1995)
27. F. Glover, A template for scatter search and path relinking, in *Artificial Evolution*, ed. by J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers. Lecture Notes in Computer Science, vol. 1363 (Springer, Berlin, 1997), pp. 13–54
28. F. Glover, Tabu search—uncharted domains. *Ann. Oper. Res.* **149**(1), 89–98 (2007)
29. F. Glover, H. Greenberg, New approaches for Heuristic search: a bilateral linkage with artificial intelligence. *Eur. J. Oper. Res.* **39**(2), 119–130 (1989)
30. F. Glover, G. Kochenberger, Critical event Tabu search for multidimensional Knapsack problems, in *Meta-Heuristics: Theory and Applications*, ed. by I.H. Osman, J.P. Kelly (Kluwer Academic Publishers, Boston, 1996), pp. 407–427
31. F. Glover, M. Laguna, *Tabu Search* (Kluwer Academic Publishers, Boston, 1997)
32. F. Glover, J.P. Kelly, M. Laguna, New advances and applications of combining simulation and optimization, in *Proceedings of the 1996 Winter Simulation Conference*, Coronado, ed. by J.M. Charnes, D.J. Morrice, D.T. Brunner, J.J. Swain (1996), pp. 144–152
33. F. Glover, G. Kochenberger, B. Alidaee, Adaptive memory Tabu search for binary quadratic programs. *Manag. Sci.* **44**(3), 336–345 (1998)
34. F. Glover, J. Mulvey, D. Bai, M. Tapia, Integrative population analysis for better solutions to large-scale mathematical programs, *Industrial Applications of Combinatorial Optimization*, ed. by G. Yu (Kluwer Academic Publishers, Boston, 1998), pp. 212–237
35. F. Glover, M. Laguna, R. Marti, Fundamentals of scatter search and path relinking. *Control Cybern.* **29**(3), 653–684 (2000)
36. H.J. Greenberg, W.P. Pierskalla, Surrogate mathematical programs. *Oper. Res.* **18**, 924–939 (1970)
37. H.J. Greenberg, W.P. Pierskalla, Quasi-conjugate functions and surrogate duality. *Cahiers du Centre d'Etudes de Recherche Operationnelle* **15**, 437–448 (1973)
38. J.H. Holland, Adaptation in natural and artificial systems (University of Michigan Press, Ann Arbor, 1975)
39. D.S. Johnson, Local optimization and the traveling salesman problem, in *Proceedings of the 17th International Colloquium on Automata, Languages and Programming* (1990), pp. 446–460

40. M.H. Karwan, R.L. Rardin, Surrogate dual multiplier search procedures in integer programming. School of Industrial Systems Engineering, Report series no. J-77-13, Georgia Institute of Technology, 1976
41. M.H. Karwan, R.L. Rardin, Some relationships between lagrangian and surrogate duality in integer programming. *Math. Program.* **17**, 230–334 (1979)
42. J. Kelly, B. Rangaswamy, J. Xu, A scatter search-based learning algorithm for neural network training. *J. Heuristics* **2**, 129–146 (1996)
43. M. Laguna, Optimizing complex systems with OptQuest. Research report, University of Colorado, 1997
44. M. Laguna, T. Feo, H. Elrod, A greedy randomized adaptive search procedure for the 2-partition problem. *Oper. Res.* **42**(4), 677–687 (1994)
45. M. Laguna, F. Glover, Integrating target analysis and Tabu search for improved scheduling systems. *Expert Syst. Appl.* **6**, 287–297 (1993)
46. M. Laguna, R. Martí, GRASP and Path Relinking for 2-Layer straight line crossing minimization. *INFORMS J. Comput.* **11**(1), 44–52 (1999)
47. M. Laguna, R. Martí, V. Campos, Tabu search with path relinking for the linear ordering problem. Research report, University of Colorado, 1997
48. M. Laguna, R. Martí, V. Valls, Arc crossing minimization in hierarchical digraphs with Tabu search. *Comput. Oper. Res.* **24**(12), 1175–1186 (1997)
49. A. Lokketangen, K. Jornsten, S. Storoy, Tabu search within a pivot and complement framework. *Int. Trans. Oper. Res.* **1**(3), 305–316 (1994)
50. A. Lokketangen, F. Glover, Probabilistic move selection in Tabu search for 0/1 mixed integer programming problems, in *Meta-Heuristics: Theory and Applications*, ed. by I.H. Osman, J.P. Kelly (Kluwer Academic Publishers, Boston, 1996), pp. 467–488
51. A. Lokketangen, F. Glover, Surrogate constraint analysis—new heuristics and learning schemes for satisfiability problems, in *Proceedings of the DIMACS workshop on Satisfiability Problems: Theory and Applications*, Providence, ed. by D.-Z. Du, J. Gu, P. Pardalos (1997)
52. H.R. Lourenco, M. Zwijnenburg, Combining the large-step optimization with Tabu search: application to the job shop scheduling problem, in *Meta-Heuristics: Theory and Applications*, ed. by I.H. Osman, J.P. Kelly (Kluwer Academic Publishers, Boston, 1996), pp. 219–236
53. O. Martin, S.W. Otto, E.W. Felten, Large-step Markov chains for the traveling salesman problem. *Complex Syst.* **5**(3), 299–326 (1991)
54. O. Martin, S.W. Otto, E.W. Felten, Large-step Markov chains for TSP incorporating local search heuristics. *Oper. Res. Lett.* **11**(4), 219–224 (1992)
55. Z. Michalewicz, C. Janikow, Genetic algorithms for numerical optimization. *Stat. Comput.* **1**, 75–91 (1991)
56. H. Mühlenbein, D. Schlierkamp-Voosen, The science of breeding and its application to the Breeder genetic algorithm. *Evolut. Computat.* **1**, 335–360 (1994)
57. H. Mühlenbein, H.-M. Voigt, Gene pool recombination in genetic algorithms, *Meta-Heuristics: Theory and Applications*, ed. by I.H. Osman, J.P. Kelly (Kluwer Academic Publishers, Boston, 1996), 53–62
58. H. Mühlenbein, M. Gorges-Schleuter, O. Krämer, Evolution algorithms in combinatorial optimization. *Parallel Comput.* **7**, 65–88 (1988)
59. K. Nonobe, T. Ibaraki, A Tabu search approach for the constraint satisfaction problem as a general problem solver. *Eur. J. Oper. Res.* **106**, 599–623 (1998)
60. K. Nonobe, T. Ibaraki, An improved tabu search method for the weighted constraint satisfaction problem. *INFOR* **39**, 131–151 (2001)
61. A.P. Punnen, F. Glover, Ejection chains with combinatorial leverage for the traveling salesman problem, Graduate School of Business, University of Colorado at Boulder, 1997
62. S. Rana, D. Whitley, Bit representations with a twist, in *Proceedings of the 7th International Conference on Genetic Algorithms*, ed. by T. Baeck (Morgan Kaufman, San Francisco, 1997), pp. 188–196
63. C. Rego, F. Glover, Ejection chain and filter-and-fan methods in combinatorial optimization. *Ann. Oper. Res.* (2009). Springer Science+Business Media, LLC, doi:10.1007/s10479-009-0656-7

64. Y. Rochat, É.D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing. *J. Heuristics* **1**, 147–167 (1995)
65. W.M. Spears, K.A. DeJong, On the virtues of uniform crossover, in *Proceedings of the 4th International Conference on Genetic Algorithms*, La Jolla, CA, 1991
66. É.D. Taillard, A heuristic column generation method for the heterogeneous VRP. Publication CRT-96-03, Centre de recherche sur les transports, Université de Montréal. To appear in *RAIRO-OR*, 1996
67. T. Trafalis, I. Al-Harkan, A continuous scatter search approach for Global optimization. Extended abstract in *Conference in Applied Mathematical Programming and Modeling (APMOD'95)*, London, UK, 1995
68. N.L.J. Ulder, E. Pech, P.J.M. van Laarhoven, H.J. Bandelt, E.H.L. Aarts, Genetic local search algorithm for the traveling salesman problem, in *Parallel Problem Solving from Nature*, ed. by R. Maenner, H.P. Schwefel (Springer, Berlin, 1991), pp. 109–116
69. D. Whitley, V.S. Gordon, K. Mathias, Lamarckian evolution, the Baldwin effect and function optimization, in *Proceedings of the Parallel Problem Solving from Nature*, vol. 3 (Springer, New York, 1994) pp. 6–15
70. A.H. Wright, Genetic algorithms for real parameter optimization, *Foundations of Genetic Algorithms*, ed. by G. Rawlins, (Morgan Kaufmann, Los Altos, CA, 1990) pp. 205–218
71. T. Yamada, R. Nakano, Scheduling by Genetic local search with multi-step crossover, in *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, Berlin (1996), pp. 960–969
72. T. Yamada, C. Reeves, Permutation flowshop scheduling by genetic local search, in *Proceedings of the 2nd IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems (GALESIA '97)*, Glasglow, UK (1997), pp. 232–238
73. S. Zenios, Dynamic financial modeling and optimizing the design of financial products, Presented at the National INFORMS Meeting, Washington, DC, 1996

Variations of Dominating Set Problem

Liying Kang

Contents

1	Introduction	3364
1.1	Graph Theory Terminology and Concepts	3365
1.2	Hypergraph Terminology and Concepts	3366
2	Power Domination in Graphs	3367
2.1	Complexity Results	3367
2.2	An Approximation Algorithm for the PDS Problem in Planar Graphs	3369
2.3	Polynomial Algorithms for the PDS Problem in Some Special Graphs	3371
2.4	Upper Bounds on the Power Domination Number in Graphs	3375
3	Paired-Domination and Related Variations of Graphs	3376
3.1	Upper Bounds on the Paired-Domination Number of Graphs	3376
3.2	Hardness Results and Approximation Algorithms for Paired-Domination in Graphs	3379
3.3	Paired-Domination in Special Graphs	3380
3.4	Paired-Domination Subdivision Number of Graphs	3384
3.5	Distance Paired-Domination Numbers of Graphs	3385
4	Total Domination in Graphs	3387
4.1	Complexity and Algorithmic Results on the Total Domination Problem	3387
4.2	Heuristics	3388
4.3	Bounds on the Total Domination Number of Graphs	3389
5	Conclusion	3390
	Cross-References	3391
	Recommended Reading	3391

Abstract

Domination and its variations in graphs have been extensively studied. The basic problem is as follows: given an undirected graph $G = (V, E)$, determine a minimum-size vertex set $S \subseteq V$ such that each vertex v is contained in S or v is

L. Kang (✉)

Department of Mathematics, Shanghai University, Shanghai, People's Republic of China
e-mail: lykang@shu.edu.cn

a neighbor of at least one vertex in S . In this chapter, we study three variants of domination: power domination, paired-domination, and total domination which were widely studied in recent years. We give a review of these variants of domination from complexity, approximation algorithm, polynomial algorithm, and bound.

1 Introduction

Domination is a central theme in graph theory. Domination and its variations in graphs have been extensively studied [17, 44, 45, 51]. The basic problem is as follows: given an undirected graph $G = (V, E)$, determine a minimum-size vertex set $S \subseteq V$ such that each vertex v is contained in S or v is a neighbor of at least one vertex in S .

Let $G = (V, E)$ be a graph with vertex set V and edge set E . A *dominating set*, abbreviated by DS, of G is a set S of vertices such that every vertex in $V \setminus S$ is adjacent to a vertex in S . The *domination number* of G , denoted by $\gamma(G)$, is the minimum cardinality of a DS. A dominating set of G of cardinality $\gamma(G)$ is called a γ -set of G (similar notation is used for the other domination parameters). An *independent dominating set* S of a graph G is a dominating set that no two vertices of S are connected by any edge of G . The *independent domination number* $i(G)$ is the minimum cardinality of an independent dominating set of G . A *connected dominating set* (CDS) of G is a set S of vertices of G such that S is a dominating set and the subgraph induced by S is connected. The *connected domination number* $\gamma_c(G)$ is the minimum cardinality of a connected dominating set of G . Equivalently, $\gamma_c(G) = |V(G)| - L$, where L is the maximum number of endpoints of a spanning tree of G [68]. A *total dominating set* (TDS) of G with no isolated vertex is a set S of vertices of G such that every vertex is adjacent to a vertex in S (other than itself). The *total domination number* of G , denoted by $\gamma_t(G)$, is the minimum cardinality of a total dominating set of G . A *matching* of G is a set of independent edges in G . The cardinality of a maximum matching in G is denoted by $\beta'(G)$. A *perfect matching* M in G is a matching in G such that every vertex of G is incident to a vertex in M .

In recent years paired-domination and power domination problems were widely studied. A *paired-dominating set* (PD) of G is a set S of vertices of G such that every vertex is adjacent to some vertex in S and the subgraph induced by S contains a perfect matching. The *paired-domination number* $\gamma_{pr}(G)$ is defined to be the minimum cardinality of a paired-dominating set in G . Every graph without isolated vertices has a paired-dominating set since the endvertices of any maximal matching form such a set. Clearly, $\gamma_t(G) \leq \gamma_{pr}(G)$ for any graph G without isolated vertices. Paired-domination was introduced by Haynes and Slater [43] as a model for assigning backups to guard for security purpose.

The power-domination problem was first studied in terms of graphs by Haynes et al. in [42]. Electric power companies need to continually monitor their system's state as defined by a set of state variable. One method of monitoring these variables is to place phase measurement units (PMUs) at selected locations in the system. Because of the high cost of a PMU, it is desirable to minimize the number of PMUs

while maintaining the ability to monitor (observe) the entire system. A system is said to be observed if all of the state variables of the system can be determined from a set of measurements.

Indeed, an electrical power network can be modeled by a graph where the vertices represent the electrical vertices and the edges are associated with the transmission lines joining two electrical vertices. This problem is of somehow different flavor than standard domination-type problems, since putting a phase measurement unit into a vertex of a graph can have global effects. For instance, if an electric power system can be modeled by a path, then a simple measurement unit suffices to monitor the system no matter how long the path.

Let G be a connected graph and S a subset of its vertices. Denote the *set monitored* by S with \mathcal{P}_S , and define it algorithmically as follows:

1. (*Domination*)

$$\mathcal{P}_S \leftarrow S \cup N(S).$$

2. (*Propagation*)

As long as there exists $v \in \mathcal{P}_S$ such that

$$N(v) \cap (V(G) - \mathcal{P}_S) = \{w\}$$

$$\text{set } \mathcal{P}_S \leftarrow \mathcal{P}_S \cup \{w\}.$$

In other words, the set \mathcal{P}_S is obtained from S as follows. First put into \mathcal{P}_S the vertices from the closed neighborhood of S . Then repeatedly add to \mathcal{P}_S vertices w that have a neighbor v in \mathcal{P}_S such that the other neighbors of v are already in \mathcal{P}_S . After no such vertex w exists, the set monitored by S has been constructed.

The set S is called a *power dominating set* (PDS) of G if $\mathcal{P}_S = V(G)$ and the *power domination number* $\gamma_P(G)$ is the minimum cardinality of a power dominating set in G .

A number of definitions are used in this chapter.

1.1 Graph Theory Terminology and Concepts

Let $G = (V, E)$ be a (nonempty) graph. Two vertices x, y of G are *adjacent*, or *neighbors*, if xy is an edge of G . The set of neighbors of a vertex v in G is denoted by $N_G(v)$, or briefly by $N(v)$. The *degree* of v in G , denoted by $d_G(v)$, is the number of edges of G incident with v . In particular, if G is a simple graph, $d_G(v)$ is equal to the number of neighbors of v in G . A vertex of degree 0 is called an *isolated vertex*. We denote by $\delta(G)$ and $\Delta(G)$ the *minimum* and *maximum degrees* of the vertices of G . If all the vertices of G have the same degree k , then G is *k -regular*, or simply *regular*. A regular 3-regular graph is called *cubic*.

A *star* is a graph $K_{1,a}$ composed of a central vertex c and a other vertices only adjacent to c . In the special case $a = 3$, the graph $K_{1,3}$ is called a *claw*. A *subdivided star* is a star whose edges are subdivided once, that is, each edge is replaced by a path of length 2 by adding a vertex of degree 2.

If G does not contain a graph F as an induced subgraph, then G is said to be F -*free*. As usual, K_n is the complete graph on n vertices. The graph $K_4 - e$ (with one edge removed from K_4) is called a *diamond*. In particular, we say a graph is *claw-free* if it is $K_{1,3}$ -free and *diamond-free* if it is $(K_4 - e)$ -free.

A graph is *bipartite* if its vertex set can be partitioned into two subsets X and Y so that every edge has one end in X and one end in Y . A graph which can be drawn in the plane in such a way that edges meet only at points corresponding to their common ends is called a *planar graph*, and such a drawing is called a planar embedding of the graph. A *split graph* is a graph in which the vertices can be partitioned into a clique and an independent set. For terminology and notation not given here, the reader is referred to [13].

A subset S of vertices in a graph G is a *packing* (respectively, an *open packing*) if the closed (respectively, open) neighborhoods of vertices in S are pairwise disjoint. The *packing number* $\rho(G)$ is the maximum cardinality of a packing in G , while the *open packing number* $\rho^o(G)$ is the maximum cardinality of an open packing in G .

A *tree decomposition* of G is a pair $\langle X_i \subseteq V(G) \mid i \in I, T = (I, F) \rangle$, where each X_i is a subset of $V(G)$, called a *bag*, and T is a tree with the elements of I as vertices. The following three properties must hold:

1. $\cup_{i \in I} X_i = V(G)$.
2. Every edge $uv \in E$ has both ends in some X_i .
3. For all $i, j, k \in I$, if j is on the unique path from i to k in T , then we have $X_i \cap X_k \subseteq X_j$.

The *width* of $\langle X_i \subseteq V \mid i \in I, T \rangle$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* of G is defined as the minimum width over all tree decompositions of G .

MinRep Problem ([55])

In the MinRep [55] problem we are given a bipartite graph $G = (A, B, E)$ with a partition of A and B into equal-sized subsets, let q_A and q_B denote the number of sets in the partition of A and B , respectively. Let $A = A_1 \cup A_2 \dots \cup A_{q_A}$ denote the partition of A , and let $B = B_1 \cup B_2 \dots \cup B_{q_B}$ denote the partition of B . This partition naturally defines a superbipartite graph $H = (A, B, E)$. The supervertices of H are A_1, A_2, \dots, A_{q_A} and B_1, B_2, \dots, B_{q_B} . There is a superedge between supervertices A_i and B_j if there exists some $a \in A_i$ and $b \in B_j$ such that ab is an edge of G . We say that a superedge $A_i B_j$ is covered by vertices a, b if $a \in A_i, b \in B_j$, and if there is an edge between a and b in G . Given $S \subseteq A \cup B$, we say that the superedge $A_i B_j$ is covered by S if there exists $a, b \in S$ that covers $A_i B_j$. The goal in the *MinRep problem* is to pick a minimum-size set of vertices, $A' \cup B' \subseteq V(G)$, to cover all superedges in H .

1.2 Hypergraph Terminology and Concepts

Hypergraphs are systems of sets which are conceived as natural extensions of graphs. A hypergraph $H = (V, E)$ is a finite set V of elements, called *vertices*, together with a finite multiset E of arbitrary subsets of V , called *hyperedges*. A *transversal* or *hitting set* in H is a subset of the vertices of H which has

a nonempty intersection with each hyperedge of H . The *transversal number* $\tau(H)$ of H is the minimum cardinality of a transversal in H . A transversal of H of cardinality $\tau(H)$ is called $\tau(H)$ -transversal. A k -uniform hypergraph is a hypergraph in which every hyperedge has size k . Graphs are special hypergraphs.

In this chapter, we will focus on three variants of domination: power domination, paired-domination and total domination which were widely studied in recent years.

2 Power Domination in Graphs

Power domination has been widely studied recently (see [9, 31, 32, 41, 43, 61, 62]). Given a graph G and a positive integer k , the power domination decision problem consists of deciding if the graph has a PDS of size at most k . The power domination decision problem has been proven to be NP-complete [42] even when the input graph is bipartite; Haynes et al. in [42] presented a linear-time algorithm to solve PDS optimally on trees. Kneis et al. [53] generalized this results to a linear-time algorithm that finds an optimal solution for PDS on graphs that have bounded treewidth, relying on earlier results of Courcelle et al. [31]. Guo et al. [41] developed a combinatorial algorithm based on dynamic programming for optimally solving the PDS problem on graphs of treewidth k . The running time of their algorithm is $O(c^{k^2} \cdot n)$, where c is a constant. Even for planar graphs, the dominating set problem (which is to determine a minimum dominating set in a graph G) is NP-hard [39], and the same holds for PDS [41]. Aazami and Stilp [1] gave an $O(\sqrt{n})$ -approximation algorithm for planar graphs and showed that their methods cannot improve on this approximation guarantee. Liao and Lee [61] proved that the PDS problem on split graphs is NP-complete, and they presented a polynomial-time algorithm for solving PDS optimally on interval graphs. Dorfling and Henning [36] computed the power domination number, that is, the size of an optimal power dominating set, for $n \times m$ grids. Zhao et al. [77] established a sharp upper bound on the power domination number of any connected graph G of order $n \geq 3$, and $\gamma_p(G) \leq n/4$ for any connected claw-free cubic graph G of order n and characterized the extremal cubic graphs attaining the bound.

2.1 Complexity Results

The dominating set problem is itself a classical NP-complete problem [39]. Haynes et al. [42] showed the NP-completeness of the PDS problem by giving a reduction from 3-SAT. What makes power dominating set seemingly harder to solve is its nonlocal structure: For dominating set, the placement of a dominating vertex cannot influence other parts in the graph that are away, while this is perfectly possible for power dominating set. A path of arbitrary length, for example, can be measured by a single vertex at one of its ends. Guo et al. [41] gave some first evident for this intuition. They showed that the PDS problem is at least as hard to solve as the DS problem by reducing the DS problem to the PDS problem.

Given a graph G , they constructed an augmented graph G' such that S is an optimal solution for the domination set problem on G if and only if it is an optimal solution for PDS on G' ; the G' is obtained from G by adding a new vertex v' for each vertex v in G and adding the edge vv' .

By reducing the DS problem to the PDS problem, Guo et al. [41] proved the following:

Theorem 1 ([41]) *The power dominating set problem is NP-complete in bipartite, chordal, circle, and planar graphs.*

Guo et al. [41] showed that the PDS problem is NP-complete in split graphs. The reduction is from the NP-complete vertex cover problem.

Theorem 2 ([41]) *The power dominating set problem is NP-complete in split graphs.*

The following table was given in [41], which compared the computational complexity of the PDS problem and the DS problem.

Aazami and Stilp [1] proved a result on the hardness of approximating the PDS problem by a reduction from the MinRep problem. They gave a gap preserving reduction from MinRep to the PDS problem. The hardness result showed that the PDS problem cannot be approximated within a factor of $2^{\log^{1-\epsilon} n}$ unless $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$. This substantially improved the $\Omega(\log n)$ hardness result in [41, 53]; both papers gave a gap preserving reduction from the DS problem to the PDS problem.

Table 1 The computation complexity of the PDS problem and the DS problem [41]

Graph classes	Dominating set	Power dominating set
Bipartite	NP-complete	NP-complete
Chordal	NP-complete	NP-complete
Circle	NP-complete	NP-complete
Comparability	NP-complete	NP-complete
Planar	NP-complete	NP-complete
Split	NP-complete	NP-complete
AT-free	Poly. time	
Cocomparability	Poly. time	
Distance hereditary	Poly. time	
Dually chordal	Linear time	
Interval	Poly. time	Poly. time
k -Polygon ($k \geq 3$)	Poly. time	
Partial k -tree ($k \geq 1$)	Linear time	Linear time
Permutation	Linear time	
Strongly chordal	Poly. time	

Empty entries mean that this has not been studied yet

Theorem 3 ([1]) *The PDS problem cannot be approximated within ratio $2^{\log^{1-\epsilon} n}$ for any fixed $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$.*

Theorem 3 is proved by a reduction from the MinRep problem. The following result is due to Kortsarz [55].

Theorem 4 ([55]) *The MinRep problem cannot be approximated within ratio $2^{\log^{1-\epsilon} n}$, for any fixed $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\text{polylog}(n)})$.*

Aazami and Stilp [1] created an instance $\hat{G} = (\hat{V}, \hat{E})$ of the PDS problem from a given instance $G = (A, B, E)$ of the MinRep problem. The idea is to replace each superedge with a “cover testing gadget.” Start with a copy of each vertex in $A \cup B$ in \hat{G} . Add a new vertex w^* to the graph \hat{G} and connect w^* to all vertices in $A \cup B$. Also add new vertices w_1^*, w_2^*, w_3^* and connect them to w^* (the vertices w_1^*, w_2^*, w_3^* are added to force w^* to be in any optimal solution). For all $i \in \{1, \dots, q_A\}$, $j \in \{1, \dots, q_B\}$, if $A_i B_j$ is a superedge, then do the following: Let E_{ij} denote the set of edges between A_i and B_j in G , and let l_{ij} denote $|E_{ij}|$. Denote the edges in E_{ij} by $e_1, e_2, \dots, e_k, \dots$. Let C_{ij} be a cycle of $3l_{ij}$ vertices. Sequentially label the vertices of C_{ij} as $u_1, v_1, w_1, u_2, v_2, w_2, \dots, u_k, v_k, w_k, \dots$. Make $\lambda = 4$ new copies of the graph C_{ij} . For each edge $e_k = a_k b_k \in E_{ij}$ and for each of the four copies of C_{ij} , add an edge from a_k to u_k and an edge from b_k to v_k . Let $\hat{G} = (\hat{V}, \hat{E})$ be the obtained graph. Let S be a feasible solution for the resulting PDS instance \hat{G} , and suppose $w^* \in S$.

The next lemma shows that the size of an optimal solution in PDS is exactly one more than the size of an optimal solution in MinRep. This will complete the proof of **Theorem 3** by showing that the above reduction is a gap preserving reduction from MinRep to the PDS problem with the same gap (hardness ratio) as the MinRep problem.

Lemma 1 ([1]) *$A^* \cup B^*$ is an optimal solution to the instance $G = (A, B, E)$ of the MinRep problem if and only if $S^* = A^* \cup B^* \cup \{w^*\} \subseteq V(\hat{G})$ is an optimal solution to the instance \hat{G} of the PDS problem.*

2.2 An Approximation Algorithm for the PDS Problem in Planar Graphs

Aazami and Stilp [1] described an $O(k)$ -approximation algorithm for the PDS problem in graphs with treewidth k ; the running time is $O(n^3)$, independent of k . This algorithm gives an $O(\sqrt{n})$ -approximation algorithm for the PDS problem in planar graphs. They showed that the analysis of the algorithm is tight on planar graphs.

Aazami and Stilp [1] introduced the notation of a strong region. For a graph $G = (V, E)$, a subset $R \subseteq V$ is called *strong* if every feasible solution to the

Algorithm 1 (see [1])

```

1: A tree decomposition  $\langle \{X_i | i \in I\}, T \rangle$  of  $G$  is given, where  $T$  is rooted at  $r$ .
2: Let  $I_l$  be the set of  $T$ -vertices at distance  $l$  from the root  $r$ , and let  $d$  be the maximum distance
   from  $r$  in  $T$ .
3:  $S \leftarrow \emptyset$ 
4: for  $i = d$  to 0 do
5:   Let  $I_i = \{r_1, \dots, r_{k_i}\}$  and denote by  $T_{r_j}$  the subtree in  $T$  rooted at  $r_j$ .
6:   Let  $Y_{r_j}$  be the union of bags corresponding to the  $T$ -vertices in  $T_{r_j}$ .
7:   for  $j = 1$  to  $k_i$  do
8:     if  $Y_{r_j}$  is an  $S$ -strong region, then
9:        $S \leftarrow S \cup X_{r_j}$ , where  $X_{r_j}$  is the bag corresponding to  $r_j$ .
10:    end if
11:   end for
12: end for
13: Output  $S_0 = S$ .

```

PDS problem has a vertex of R . The neighborhood of $R \subseteq V$ is defined as $\text{nbr}(R) = \{v \in V | \exists u \in R, v \in E, u \in R, v \notin R\}$, and the exterior of R is defined as $\text{ext}(R) = \text{nbr}(V \setminus R)$; that is, $\text{ext}(R)$ consists of the vertices in V that are adjacent to a vertex in $V \setminus R$.

Given a graph $G = (V, E)$ and a set $S \subseteq V$, the subset $R \subseteq V$ is called an *S -strong region* if $R \not\subseteq \mathcal{P}_{S \cup \text{nbr}(R)}$; otherwise, R is called an *S -weak region*. The region R is called *minimally S -strong* if it is an S -strong region, and for all $r \in R$, $R - r$ is an S -weak region.

It is easy to check from the definition that an S -strong region is also a strong region. Any feasible solution to the PDS problem needs to have at least one vertex from every strong region.

Lemma 2 ([2]) *For a graph $G = (V, E)$ and a subset $R \subseteq V$, the subset $R \subseteq V$ is an S -strong region if and only if for every feasible solution $S \cup S^*$ of G , we have $R \cap (S^* \setminus S) \neq \emptyset$.*

The following algorithm makes one level-by-level and bottom-to-top pass over the tree T of the tree decomposition of G and constructs a solution S for the PDS problem. At each vertex r_j of T , check whether the union of the bags in the subtree rooted at r_j forms an S -strong region; if yes, then the bag X_{r_j} of r_j is added to S ; otherwise, S is not updated.

Analysis of the Algorithm

To show that the solutions S_0 found by the algorithm is feasible, authors first got the following lemmas.

Lemma 3 ([1]) *Suppose Z is an S -weak region such that $\text{ext}(Z) \subseteq S$. Then we have $Z \subseteq \mathcal{P}_S$.*

Lemma 4 ([1]) *Let $Z \subseteq V$ be an S -strong region. Suppose that Y is a subset of V such that $Y \subseteq \mathcal{P}_S$ and $\text{ext}(Y) \subseteq S$. Then $Z \setminus Y$ is an S -strong region.*

For any vertex q of T , Y_q denotes the union of the bags corresponding to the T -vertices in the subtree rooted at q in T . Then $\text{ext}(Y_q) \subseteq X_q$. Using induction on the height of the subtree of T rooted at q and Lemmas 3 and 4, they proved that if Y_q is S^* -strong, then $Y_q \subseteq \mathcal{P}_{S^* \cup X_q}$, where S^* denotes the solution just before the algorithm examines Y_q .

The above statement implies that $V \subseteq \mathcal{P}_{S_0}$ when the algorithm examines the root r of T .

To show that the approximation guarantee is $(k + 1)$, they constructed a set Δ of pairwise disjoint strong regions R_1, R_2, \dots , such that there is a strong region R_j corresponding to each step of the algorithm that adds a nonempty bag X_{q_j} to S . Thus $|S_0| \leq (k + 1)|\Delta|$ since each bag has $\leq k + 1$ vertices, and $\text{Opt}(G) \geq \Delta$ because every feasible solution has size $\geq |\Delta|$ by Lemma 2. Hence $|S_0| \leq (k + 1)\text{opt}(G)$.

Theorem 5 ([1]) *Given a graph $G = (V, E)$ and a tree decomposition of G of width k as input, Algorithm 1 runs in time $O(n \cdot |E|)$ and achieves an approximation guarantee of $k + 1$.*

It is known that planar graphs have treewidth $O(\sqrt{n})$, and such a tree decomposition can be found in $O(n^{\frac{3}{2}})$ time [4]. This fact together with the above theorem proves the following theorem.

Theorem 6 ([1]) *Algorithm 1 achieves an approximation guarantee of $O(\sqrt{n})$ for the planar PDS problem.*

2.3 Polynomial Algorithms for the PDS Problem in Some Special Graphs

2.3.1 Power Domination in Trees

Haynes et al. [42] showed that the PDS problem is NP-complete even when the input graph is bipartite; they presented a linear-time algorithm to solve PDS optimally on trees. Guo et al. [41] presented simplified linear-time algorithm for the PDS problem in trees.

Without loss of generality, Guo et al. [41] assumed that the input is rooted at a degree-one vertex r and the *depth* of a vertex is defined as the length of the path between v and r . The algorithm follows a bottom-up strategy.

Analysis of the Algorithm

Based on an induction on the depth of the vertices u in T , they proved that the output P of Algorithm PDS-Trees is a power dominating set of T . The proof works top-down whereas the algorithm works bottom-up. Next, they proved the optimal of P by showing a more general statement.

Claim. Given a rooted tree $T = (V, E)$ with root r , Algorithm PDS-Trees outputs a power dominating set P such that $|P \cap V_u| \leq |G \cap V_u|$ for any optimal solution Q

PDS-Trees ([41])

Input: Rooted tree T with r denoting the root.

Output: Minimum power dominating set P .

```

1   Sort the inner vertices of  $T$  in a list  $L$  according to a post-order traversal of  $T$ ;
2   while  $L \neq \{r\}$  do
3      $v \leftarrow$  the first vertex in  $L$ ;  $L \leftarrow L \setminus \{v\}$ ;
4     if  $v$  has at least two unobserved children then
5        $P \leftarrow P \cup \{v\}$ ;
6     Exhaustively apply the two observation rules to  $T$  ;
7     end if
8   end while
9   if  $r$  is unobserved then
10   $P \leftarrow P \cup \{r\}$ ;
11 end if
12 return  $P$ .

```

of PDS for T and any tree vertex u , where V_u denotes the vertex set of the subtree of T rooted at u .

2.3.2 Power Domination in Graphs of Bounded Treewidth

Guo et al. [41] gave a linear-time algorithm for graphs of bounded treewidth; assuming that the corresponding tree decomposition is given, use basically the same strategy as the algorithms for DS [4, 5, 72], that is, bottom-up dynamic programming from the leaves to the root. There are two difficulties associated “propagation” that makes PDS “hard” than DS and which cannot be solved by a simple modification of algorithm for DS. Guo et al. [41] defined the orientations of graphs.

An *orientation* of an undirected graph $G = (V, E)$ is a graph $D = (V, E_1 \dot{\cup} E_2)$ such that, for each $\{u, v\} \in E$, there is either a directed edge (u, v) or (v, u) in E_1 or an undirected edge $\{u, v\}$ in E_2 . The indegree of a vertex v in D , denoted by $d^-(v)$, is defined as $|\{(u, v) : (u, v) \in E_1\}|$, and the outdegree of v , denoted by $d^+(v)$, as $|\{(v, u) : (v, u) \in E_1\}|$.

A *valid orientation* of $D = (V, E_1 \dot{\cup} E_2)$ of an undirected graph $G = (V, E)$ is an orientation such that:

1. For any $v \in V$: $(d^-(v) \leq 1)$ and $(d^-(v) = 1 \Rightarrow d^+(v) \leq 1)$.
2. And there is no dependent cycle in D .

The set of vertices with $d^-(v) = 0$ is called *origin* of D .

Valid Orientation with Minimum Origin (VOMO)

Input: An undirected graph $G = (V, E)$ and an integer $k \geq 0$.

Output: Is there a vertex subset $O \subseteq V$ with $|O| \leq k$ such that G has a valid orientation with O as origin?

Guo et al. [41] showed that orientation problem is a reformulation of the PDS problem.

Theorem 7 ([41]) *An undirected graph G has a power dominating set P if G has a valid orientation with P as origin.*

Guo et al. [41] described a dynamic programming procedure solving VOMO on graphs of bounded treewidth. As a consequence, they also arrived at a solution for the PDS problem.

Theorem 8 ([41]) *For a graph of order n with given width- k -tree decomposition, the PDS problem can be solved in $O(c^{k^2} \cdot n)$ time for a constant c .*

2.3.3 Power Domination in Block Graphs

A *block* of G is a maximal connected subgraph of G without a cut-vertex. A *block graph* is a connected graph whose blocks are complete graphs. Xu et al. [76] achieved a linear-time algorithm for block graphs. For block graphs, they first presented the following result.

Lemma 5 ([76]) *If G is a block graph, then there exists a minimum power dominating set in which every vertex is a cut-vertex of G .*

Based on Lemma 5, they gave PDS-BG Algorithm. The algorithm works on a treelike decomposition structure, named *refined cut-tree*, of a block graph. Let G be a block graph with h blocks BK_1, \dots, BK_h and p cut-vertices v_1, \dots, v_p . The cut-tree of G , denoted by $T^B(V^B, E^B)$, is defined as $V^B = \{BK_1, \dots, BK_h, v_1, \dots, v_p\}$ and $E^B = \{(BK_i, v_j) \mid v_j \in BK_i, 1 \leq i \leq h, 1 \leq j \leq p\}$. It is shown in [3] that the cut-tree of a block graph can be constructed in linear time by the depth-first-search method. For any block BK_i of G , define $B_i = \{v \in BK_i \mid v \text{ is not a cut-vertex}\}$, where $1 \leq i \leq h$. They can refine the cut-tree $T^B(V^B, E^B)$ as $V^B = \{B_1, \dots, B_h, v_1, \dots, v_p\}$ and $E^B = \{(B_i, v_j) \mid v_j \in BK_i, 1 \leq i \leq h, 1 \leq j \leq p\}$, and each B_i is called a *block-vertex*. It is shown that the cut-tree of a block graph can be constructed in linear time by the depth-first-search method.

PDS-BG Algorithm is presented actually as a color-marking process, and we give a brief overview on this algorithm. Let G be a connected block graph, and $T^B(V^B, E^B)$ the refined cut-tree of G . For notation convenience, set $T^B = T^B(V^B, E^B)$, and let $V_{v_j} = \{v \in V(G) \mid v \in BK_i, \text{ and } v_i^B \text{ is a vertex of } T_{v_j}^B\}$ for a cut-vertex v_j of G . Suppose $T^B(V^B, E^B)$ is a rooted tree with vertex set $\{v_1, v_2, \dots, v_n\}$ such that $l(v_i) \leq l(v_j)$ for $i > j$, and the root is a cut-vertex v_n (in G). Let H_i be the set of all vertices with level number i , and k be the largest level number. By the definition of $T^B(V^B, E^B)$, it is clear that $H_0 = \{v_n\}$ and H_i contains only either cut-vertices of G for even i or black-vertices for odd i . Clearly, k is odd. By Lemma 5, there exists a minimum power dominating set that contains only cut-vertices for a block graph G . PDS-BG Algorithm considers directly the cut-vertices in the rooted tree T^B . It starts from the largest even level of T^B and works upward to the root of the tree. Initially, all vertices of T^B are marked with white (which means all needed to be observed), and eventually, every white vertex will be marked with black or gray (except for the possibility that some will remain white and can also be observed by the rules of power observation). In the end of the algorithm, all black-vertices form a minimum PDS

of G , and each gray vertex is observed by some black cut-vertex. For a gray cut-vertex v , $\text{bound}(v) = 1$ means that v is bounded, that is, v itself is already observed and there exists unique white vertex which is adjacent to v in $G[V_v]$ (noting that such a white vertex can be observed by rules of power observation in $G[V_v]$); $\text{bound}(v) = 0$ means that there does not exist any white vertex adjacent to v in $G[V_v]$.

Theorem 9 ([76]) *Algorithm PDS-BG computes in linear time a minimum PDS of a given connected block graph G .*

2.3.4 Power Domination in Product Graphs

Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the *Cartesian product* of G_1 and G_2 is the graph $G_1 \square G_2$ whose set of vertices is the Cartesian product of the sets V_1 and V_2 , so $V(G_1 \square G_2) = V_1 \times V_2$. Two vertices of $G_1 \square G_2$, say (v_1, v_2) and (u_1, u_2) , are adjacent if and only if $v_1 = u_1$ and v_2 is adjacent to u_2 in G_2 , or, alternatively, if v_1 is adjacent to u_1 in G_1 and $v_2 = u_2$. For particular pairs of graphs G_1 and G_2 , the Cartesian product provides well-known families of graphs. For example, the $n \times m$ grid can be obtained as the Cartesian product $P_n \square P_m$, where P_n and P_m , respectively, denote the paths with n and m vertices. Analogously, the $n \times m$ cylinder is the Cartesian product $P_n \square C_m$, where P_n denotes, as above, the path on n vertices and C_m stands for the cycle on m vertices. The $n \times m$ torus is the Cartesian product of two cycles, say C_n and C_m , respectively, having n and m vertices.

The *direct product* of G_1 and G_2 is the graph $G_1 \times G_2$, whose set of vertices is $V(G_1) \times V(G_2)$. Two vertices of $G_1 \times G_2$ are adjacent in $G_1 \times G_2$ if and only if they are adjacent in both coordinates. For any integers $m, n \geq 2$, the direct product $P_m \times P_n$ is not connected. If m or n is even, $P_m \times P_n$ consists of two isomorphic connected components; otherwise, it has two nonisomorphic connected components (see [75]).

A fundamental unsolved problem involving domination in graphs is to find the domination number of the $n \times m$ grid graph $P_n \square P_m$. Jacobson and Kinch [50] determined $\gamma(P_n \times P_m)$ for $n = 1, 2, 3, 4$ and all m , while Chang and Clark [19] determined $\gamma(P_n \square P_m)$ for $n = 5, 6$ and all m . For $n \geq 7$ and all m , the domination number of the grid $P_n \square P_m$ has yet to be determined. However, unlike the domination number of a grid graph, Dorfling and Henning [36] showed that the power domination number of an $n \times m$ grid graph can be completely determined.

Algorithm A (Dorfling and Henning's placement)

Input: An $n \times m$ grid with $m \geq n \geq 1$.

Output: A $\gamma_P(G)$ -set S .

1. $k \leftarrow \lfloor n/8 \rfloor$; $j \leftarrow n - 8k$; $S \leftarrow \bigcup_{i=0}^{k-1} \{(8i + 3, 2), (8i + 5, 3)\}$;
 2. **if** $j \in \{1, 2\}$ **then** $S \leftarrow S \cup \{(n, 1)\}$;
if $j = 3$ **then** $S \leftarrow S \cup \{(n - 1, 1)\}$;
if $j = 4$ **then** $S \leftarrow S \cup \{(n - 2, 1), (n - 1, 1)\}$;
if $j \in \{5, 6, 7\}$ **then** $S \leftarrow S \cup \{(n + 3 - j, 2), (n + 5 - j, 3)\}$;
-

Theorem 10 ([36]) *If G is an $n \times m$ grid, $m \geq n \geq 1$, then*

$$\gamma_P(G) = \begin{cases} \lceil \frac{n+1}{4} \rceil & \text{if } n \equiv 4 \pmod{8}, \\ \lceil \frac{n}{4} \rceil & \text{otherwise.} \end{cases}$$

By construction methods, Barrera and Ferrero [10] found upper bounds on the power domination number of some families of Cartesian products of graphs: the cylinders $P_n \square C_m$ for integers $n \geq 2, m \geq 3$ and the tori $C_n \square C_m$ for integers $n, m \geq 3$.

Theorem 11 ([10]) *The power domination number for the graph $P_n \square C_m$ is*

$$\gamma_P(P_n \square C_m) \leq \min\left\{\lceil \frac{m+1}{4} \rceil, \lceil \frac{n+1}{2} \rceil\right\}.$$

Theorem 12 ([10]) *The power domination number for the graph $G = C_n \square C_m$, $n \leq m$, is*

$$\gamma_P(G) \leq \begin{cases} \lceil \frac{n}{2} \rceil & \text{if } n \equiv 2 \pmod{4}, \\ \lceil \frac{n+1}{2} \rceil & \text{otherwise.} \end{cases}$$

Dorbec et al. [33] determined the power domination number for all direct products of paths except for the odd component of the direct product of two odd paths. They first presented a lower bound, then gave a construction that reaches it.

Theorem 13 ([36]) *For two positive integers m and n , let n be even and C a connected component of $P_m \times P_n$. If m is odd or $m \geq n$, then $\gamma_P(C) = \lceil \frac{n}{4} \rceil$.*

Theorem 14 ([36]) *For two positive integers m and n , if $m \leq n$ is odd and C is the even component of $P_m \times P_n$, then*

$$\gamma_P(C) = \max\left\{\lceil \frac{n}{4} \rceil, \lceil \frac{m+n}{6} \rceil\right\}.$$

Theorem 15 ([36]) *For two positive integers m and n , if $m \leq n$ is odd and C is the odd component of $P_m \times P_n$, then*

$$\gamma_P(C) \leq \max\left\{\lceil \frac{n-2}{4} \rceil, \lceil \frac{m+n-2}{6} \rceil\right\}.$$

2.4 Upper Bounds on the Power Domination Number in Graphs

Haynes et al. [43] proved that $\gamma_P(T) \leq n/3$ for a tree T of order n . Zhao et al. [77] further showed that $\gamma_P(G) \leq n/3$ for any connected graph of order n and this bound is tight. Furthermore, they also proved that $\gamma_P(G) \leq n/4$ for any connected

claw-free cubic graph G of order n and characterized extremal graphs attaining the upper bound.

Let Γ be the family of graphs obtained from connected graphs H by adding two new vertices v' and v'' to each vertex v of H and new edges vv' and vv'' , while $v'v''$ may be added or not.

Theorem 16 ([77]) *If $G = (V, E)$ is a connected graph of order $n \geq 3$, then $\gamma_P(G) \leq n/3$ with equality if and only if $G \in \Gamma \cup \{K_{3,3}\}$.*

The main idea of the proof is as follows. Among all these $\gamma_P(G)$ -sets, let S be chosen so that $\omega(G[S])$ is minimum. We claimed each vertex v in S has at least two external private neighbors, that is, $|pn(v, S) - S| \geq 2$. Therefore, G has at least $3|S|$ vertices; this implies that $\gamma_P(G) \leq n/3$.

Let D be a diamond. For each positive integer k , let D_k be the connected claw-free cubic graph formed from k disjoint copies of D by joining pairwise $2k$ vertices of degree two. Let $\Lambda = \{D_k | k \geq 1\}$.

Theorem 17 ([77]) *If $G = (V, E)$ is a connected claw-free cubic graph of order n , then $\gamma_P(G) \leq n/4$ with equality if and only if $G \in \Lambda$.*

The main idea of the proof is the following. Let S be a $\gamma_P(G)$ -set of G such that $G[S]$ has as few edges as possible and under this condition that $|N[S]|$ is as large as possible. They completed their proof by the following claims.

Claim 1 S is independent in G .

Claim 2 $N[x] \cap N[y] = \emptyset$ for any two distinct vertices x and y in S .

As each $N[v]$ has exactly four vertices, **Claim 2** implies that G has at least $4|S|$ vertices and so $\gamma_P(G) \leq n/4$.

3 Paired-Domination and Related Variations of Graphs

The decision problem to determine the paired-domination number of a graph is known to be NP-complete [43]. Hence it is of interest to determine upper bounds on the paired-domination number of a graph.

3.1 Upper Bounds on the Paired-Domination Number of Graphs

3.1.1 Upper Bounds on Paired-Domination Number of Graphs with Minimum Degree One

Haynes and Slater [43] obtained the following upper bound on the paired-domination number of a connected graph in terms of the order of the graph.

Theorem 18 ([43]) *If G is a connected graph of order $n \geq 3$, then $\gamma_{\text{pr}}(G) \leq n - 1$ with equality if and only if G is C_3 , C_5 or a subdivided star.*

Dorbec et al. [33] studied the upper bound on the paired-domination number in star-free graphs. They proved the following:

Theorem 19 ([36]) *For $a > 0$ an integer, if G is a connected $K_{1,a+2}$ -free graph of order $n \geq 2$, $\gamma_{\text{pr}}(G) \leq \frac{2(an+1)}{2a+1}$ and this bound is sharp.*

Considering claw-free or star-free graphs is a good way to avoid the critical cases; nevertheless, just a claw or a star is not really a problem for paired-domination. Dorbec and Gravier [35] considered graphs without the path P_5 on five vertices.

Theorem 20 ([35]) *Let G be a connected graph of order $n \geq 2$. If G is not C_5 and does not contain an induced P_5 , then*

$$\gamma_{\text{pr}}(G) \leq \frac{n}{2} + 1,$$

and this bound is sharp.

Concluding this study, Dorbec and Gravier [34] considered paired-domination in graphs containing no subdivided stars $K_{1,a+2}^*$ with $a \geq 1$. They proved the following result.

Theorem 21 ([34]) *Let G be a connected graph of order $n \geq 3$. If G contains no induced $K_{1,a+2}^*$ for some $a \geq 1$, then*

$$\gamma_{\text{pr}}(G) \leq \frac{2(an+1)}{2a+1}$$

and this bound is sharp.

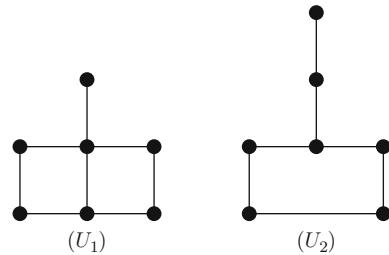
Observe that the bound here is exactly the same as the bound in [Theorem 19](#).

3.1.2 Upper Bounds on Paired-Domination Number of Graphs with Minimum Degree at Least Two

If we restrict the minimum degree to be at least two and the order to be at least six, then the upper bound in [Theorem 18](#) on the paired-domination number can be improved from one less than its order to two-thirds its order.

Theorem 22 ([43]) *If G is a connected graph of order $n \geq 6$ with $\delta(G) \geq 2$, then $\gamma_{\text{pr}}(G) \leq 2n/3$.*

The upper bound in [Theorem 22](#) can be improved slightly if the order of the graph is large.

Fig. 1 The units U_1 and U_2 

Theorem 23 ([46]) *If G is a connected graph of order $n \geq 10$ with $\delta(G) \geq 2$, then $\gamma_{\text{pr}}(G) \leq 2(n - 1)/3$ and there are infinite families of graphs that achieve equality in this bound.*

A characterization of connected graphs with minimum degree at least 2 and order at least 14 that achieve the upper bound in [Theorem 23](#) is given in [46].

Chen et al. [21] proved the following result.

Theorem 24 ([21]) *If G is a cubic graph of order n , then $\gamma_{\text{pr}}(G) \leq 3n/5$.*

Later, Goddard and Henning [40] further provided a slightly simpler proof of [Theorem 24](#) and used the proof to characterize the cubic graphs that achieve equality in the bound of [Theorem 24](#).

Theorem 25 ([40]) *If G is a connected cubic graph of order n , then $\gamma_{\text{pr}}(G) \leq 3n/5$ with equality if and only if G is the Petersen graph.*

To improve this bound, Favaron and Henning studied paired-domination on claw-free cubic graphs in [37]. They proved the following theorem.

Theorem 26 ([37]) *If G is a connected claw-free cubic graph of order n , then $\gamma_{\text{pr}}(G) \leq n/2$.*

Henning [48] gave an upper bound on the paired-domination number in terms of the number of edges in the graphs. Henning [48] defined a *unit* to be a graph that is isomorphic to the graph U_1 or U_2 shown in [Fig. 1](#). The vertex named v in each unit in [Fig. 1](#) was called *link vertex* of the unit. For $i = 1, 2$, a unit isomorphic to the graph U_i was called a *type- i unit*.

For $n_1 + n_2 \geq 2$, let $F = F(n_1, n_2)$ be the graph obtained from the disjoint union of n_1 units of type 1 and n_2 units of type 2 by identifying the $n_1 + n_2$ link vertices, one from each unit, into one new vertex which are called the *identified vertex* of G . Let \mathcal{F} denote the family of all such graph F .

Theorem 27 ([48]) *Let G be a connected graph of size $m \geq 18$ with $\delta(G) \geq 2$. Then, $\gamma_{\text{pr}}(G) \leq 4m/7$ with equality if and only if G is a cycle C_{21} or if $G \in \mathcal{F}$.*

Chen et al. [22] presented some upper bounds on the paired-domination number of a graph with girth at least 6 in terms of its minimum degree, maximum degree, or girth.

Theorem 28 ([22]) *If G is a connected graph of order n with minimum degree $\delta(G) \geq 2$ and girth $g(G) \geq 6$, then $\gamma_{\text{pr}}(G) \leq \frac{2}{3} \left(n - \frac{(\delta(G)-1)(\delta(G)-2)}{2} \right)$.*

Theorem 29 ([22]) *If G is a connected graph of order n with minimum degree $\delta(G) \geq 2$ and girth $g(G) \geq 6$, then $\gamma_{\text{pr}}(G) \leq \frac{2}{3} (n + 1 - \Delta(G))$.*

Theorem 30 ([22]) *If G is a connected graph of order n with minimum degree $\delta(G) \geq 3$, then $\gamma_{\text{pr}}(G) \leq \frac{2n}{3} - \frac{g(G)}{6} + \frac{5}{6}$.*

3.2 Hardness Results and Approximation Algorithms for Paired-Domination in Graphs

It was proved that the paired-domination problem is NP-complete, even restricted to split graphs and bipartite graphs [17].

Chen et al. [23] investigated the approximation hardness of paired-domination in general graphs. By using a reduction from minimum set cover to minimum paired-dominating set, they got the following result.

Theorem 31 ([23]) *Let G be a graph of order n . If there is some $\varepsilon > 0$ such that a polynomial-time algorithm can approximate minimum paired-domination set within $(1 - \varepsilon) \ln n$ in G , then $NP \subseteq DTIME(n^{O(\log \log n)})$.*

Chen et al. [23] showed that the paired-domination problem is APX-complete, that is, there is no PATS, even if the degree of the graph is bounded by three. Given two NP optimization problem F and G and a polynomial-time transformation f from instances of F to instance of G , we say f is an L -reduction if there are positive constants α and β such that for every instance x of F :

1. $\text{opt}_G(f(x)) \leq \alpha \cdot \text{opt}_F(x)$.
2. (For every feasible solution y of $f(x)$ with objective value $m_G(f(x), y) = c_2$, we can in polynomial time find a solution y' of x with $m_F(x, y') = c_1$ such that $|\text{opt}_F(x) - c_1| \leq \beta \cdot |\text{opt}_G(f(x)) - c_2|$.

MIN VERTEX COVER-B

Instance: Graph $G = (V, E)$ of degree bounded by B .

Solution: A vertex cover of G , that is, a subset $V' \subseteq V$ such that for all $uv \in E$ at least one of u and v in V' .

Measure: Cardinality of the vertex cover.

MIN PAIRED-DOM SET-B

Instance: Graph $G = (V, E)$ without isolated vertices of degree bounded by B .

Solution: A paired-dominating set of G

Measure: Cardinality of the paired-dominating set.

Theorem 32 ([6]) *MIN VERTEX COVER-3 is APX-complete.*

By an L -reduction f from MIN VERTEX-COVER-3 to MIN PAIRED-DOM SET-7, Chen et al. [23] proved MIN PAIRED-DOM SET-7 is APX-complete. Then they gave an L -reduction f_1 from MIN PAIRED-DOM SET-7, which is APX-complete, to MIN PAIRED-DOM SET-4 and gave L -reduction f_2 from MIN PAIRED-DOM SET-4 to MIN PAIRED-DOM SET-3.

Theorem 33 ([6]) *MIN PAIRED-DOM SET-3 is APX-complete.*

Chen et al. [23] presented a greedy algorithm for finding a paired-dominating set in general graphs and proved the upper bound on the approximation ratio of this algorithm.

Theorem 34 ([6]) *The minimum paired-dominating set in any graph $G = (V, E)$ without isolated vertices can be approximated with an approximation ratio of $\ln(2\Delta(G)) + 1$.*

Algorithm GREEDY-PAIRED-DOM

Input: A graph $G = (V, E)$ without isolated vertices.

1. PD:= \emptyset ;
 2. $i = 0, S'_i := \emptyset$;
 3. While $(V - (S'_0 \cup \dots \cup S'_{i-1})) \neq \emptyset$ do
 4. $i := i + 1$
 5. Choose two vertices $u, v \in V - PD$ such that
 $uv \in E$ and $|N(u) \cup N(v) - (S'_0 \cup S'_1 \cup \dots \cup S'_{i-1})|$ is maximized;
 6. $S_i = N(u) \cup N(v)$;
 7. $S'_i = S_i - (S'_0 \cup \dots \cup S'_{i-1})$;
 8. PD:=PD $\cup \{u, v\}$;
 9. Output PD.
-

3.3 Paired-Domination in Special Graphs

Since the problem of determining the paired-domination number of an arbitrary graph is NP-hard, it is theoretically important to consider algorithm of the paired-domination number in special graphs.

3.3.1 Paired-Domination of Trees

Qiao et al. [65] first presented a linear-time algorithm computing the paired-domination number for trees. The algorithm first breaks the original tree T into

a collection of components, T_0 , where each component is a path. Then, for each path in T_0 , the minimum paired-dominating set is computed.

Observation 1 ([43]) For any graph G without isolated vertices, $\gamma(G) \leq \gamma_{\text{pr}}(G) \leq 2\beta'(G)$ and $\gamma_{\text{pr}}(G)$ is even, where $\beta'(G)$ denotes the size of a maximum independent set of edges.

Qiao et al. [65] characterized trees with equal domination and paired-domination numbers. To state the characterization, they introduced four types of operations that were used in constructing trees with equal domination and paired-domination numbers.

Type 1 operation: Attach a path P_1 to a vertex of a tree T , which is in a γ_{pr} -set of T .

Type 2 operation: Attach a P_5 to a vertex v of a tree T , where v is in a γ_{pr} -set of T , and for every γ -set X of T , there is no vertex u such that $PN(u, X) = v$ in T .

Type 3 operation: Attach a remote vertex of P_4 to a vertex v of a tree T , where v is a vertex such that for every γ -set X of T , there is no vertex u such that $PN(u, X) = v$ in T .

Type 4 operation: Attach a vertex u_0 of tree T_1 to a vertex of a tree T , where T_1 is a tree with $V(T_1) = \{u_0, u_1, u_2, u_3, u_4\}$ and $E(T_1) = \{u_0u_1, u_1u_2, u_1u_3, u_2u_4\}$.

Define the family F_p as

$F_p = \{T \mid T \text{ is obtained from } P_4 \text{ by a finite sequence of operations of Type 1, Type 2, Type 3, or Type 4}\}$.

Theorem 35 ([65]) For a tree T , $\gamma(T) = \gamma_{\text{pr}}(T)$ if and only if $T \in F_p$.

3.3.2 Paired-Domination on Permutation Graphs

Let $\pi = [\pi_1, \pi_2, \dots, \pi_n]$ be a permutation on the set $V_n = \{1, 2, \dots, n\}$. Then the *permutation graph* $G[\pi] = (V, E)$ is the undirected graph such that $V = V_n$ and $(i, j) \in E$ if and only if

$$(i - j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0,$$

where $\pi^{-1}(i)$ is the position of i in π .

Cheng et al. [27] proposed an efficient $O(mn)$ algorithm for solving the minimum paired-dominating set on permutation graphs. The algorithm is based on a recursive formula by using the dynamic programming method.

For notational convenience, they introduce more notation as follows:

1. For any $1 \leq i, j \leq n$ and $V_i = \{\pi_1, \pi_2, \dots, \pi_i\}$, denote $V_{i,j}$ as the subset of V_i containing all elements smaller than or equal to j , that is, $V_{i,j} = \{\pi_k \in V_i, \pi_k \leq j\}$.
2. For any $i, 1 \leq i \leq n$, denote π_i^* as the minimum number over the suffix $\pi_i, \pi_{i+1}, \dots, \pi_n$, that is, $\pi_i^* = \min\{\pi_i, \pi_{i+1}, \dots, \pi_n\}$, and set $V_i^* = V_i \cup \{\pi_i^*\}$.

Algorithm: Finding an MPDS on a permutation graph.

Input: A permutation $\pi = [\pi_1, \pi_2, \dots, \pi_n]$.

Output: A minimum cardinality paired-dominating set of $G[\pi]$.

Step 1. Initialize $\text{PD}_{0,j} = \emptyset$.

$$\text{PD}_{i,j} = \begin{cases} \emptyset & \text{if } j < \pi_1; \\ \{1, \pi_1\} & \text{otherwise.} \end{cases}$$

for $j = 1, 2, \dots, n$.

Step 2. for $i \leftarrow 2$ to n do

Step 3. $\text{PD}_{\pi_i^*} = \text{Min}\{\text{PD}_{l-1, \pi_i^*} \cup \{\pi_i^*, \pi_l\} : \pi_l \in N(\pi_i^*), \pi_i^* \notin \text{PD}_{l-1, \pi_i^*}, l \leq i\}$

Step 4. for $j \leftarrow 1$ to n do

Step 5.

$$\text{PD}_{\max} = \begin{cases} \text{PD}_{i-1,j} \cup \{\pi_i^*, \max(V_i)\} & \text{if } \pi_i \neq \max(V_i); \\ V_i & \text{otherwise.} \end{cases}$$

Step 6.

$$\text{PD}_{i,j} = \begin{cases} \text{Min}(\{\text{PD}_{\pi_i^*}, \text{PD}_{\max}\}) & \text{if } j \geq \pi_i \text{ and } \max(\text{PD}_{i-1,j}) < \pi_i; \\ \text{Min}(\{\text{PD}_{\pi_i^*}, \text{PD}_{i-1,j}\}) & \text{otherwise.} \end{cases}$$

Step 7. End

Step 8. End

Step 9. Output $\text{PD}_{n,n}$.

3.3.3 Paired-Domination in Strongly Chordal Graphs

A vertex u is *simple* if $N_G[u] = \{u_1, u_2, \dots, u_k\}$, where $u = u_1$ satisfies $N_G[u_i] \subseteq N_G[u_j]$ for $1 \leq i \leq j \leq k$. A graph is *strongly chordal* if every induced subgraph has a simple vertex.

Starting with an end block and working recursive inward, Chen et al. [26] found a vertex ordering v_1, v_2, \dots, v_n in $O(n + m)$ time such that $v_i v_j \in E$ and $v_i v_k \in E$ imply that $v_j v_k \in E$ for $i < j < k \leq n$. They defined two labels on each vertex u , denoted by $(D(u), L(u))$. Based on labels, Chen et al. [26] gave algorithm MPDB to determine a minimum paired-dominating set in block graphs.

Theorem 36 ([26]) *Given a vertex ordering of a block graph G on n vertices and m edges, the algorithm MPDB can produce a minimum paired-dominating set of G in $O(m + n)$ time.*

Chen et al. [26] also employed the labeling technique to give a linear algorithm for finding a minimum paired-dominating set of an interval graph.

Chen et al. [24] presented a linear-time algorithm for the paired-domination problem in strongly chordal graphs, which generalized the algorithms in [26, 65].

$$D(v_i) = \begin{cases} 0 & \text{if } v_i \text{ is not dominated;} \\ 1 & \text{if } v_i \text{ is dominated.} \end{cases}$$

$$L(v_i) = \begin{cases} 0 & \text{if } v_i \text{ is not put into } \text{PD}(G); \\ 1 & \text{if } v_i \text{ is put into } \text{PD}(G), \text{ but it has no paired vertex in } \text{PD}(G); \\ 2 & \text{if } v_i \text{ is put into } \text{PD}(G), \text{ but it has a paired vertex in } \text{PD}(G). \end{cases}$$

Algorithm MPDS. Find a MPES in a strongly chordal graph without isolated vertices.

Input: A strongly chordal graph $G = (V, E)$ with a strong (elimination) ordering v_1, v_2, \dots, v_n ($n \leq 2$). Each vertex v_i has labels $(D(v_i), L(v_i)) = (0, 0)$, and $F(v_i) = v_j$ with $j = \max\{k \mid k \in M_i\}$.

Output: A minimum paired-dominating set PD of G .

Method

```

For  $i = 1$  to  $n$  do
    If  $(D(v_i) = 0 \text{ and } F(v_i) \neq v_i)$  then
         $L(F(v_i)) = 1;$ 
         $D(u) = 1$  for  $u \in N[F(v_i)]$ ;
    else if  $(D(v_i) = 0 \text{ and } F(v_i) = v_i)$  then
         $L(v_i) = 2;$ 
         $L(u) = 2$  for some  $u \in N(v_i)$  such that  $L(u) = 0$ ;
         $D(v_i) = 1;$ 
    else if  $(L(v_i) = 1)$  then
        Let  $C = N(v_i) \cap \{v_j \mid L(v_j) = 1 \text{ and } j > i\}$ ;
        If  $(C = \emptyset \text{ and } L(F(v_i)) = 0)$  then
             $L(v_i) = 2;$ 
             $L(F(v_i)) = 2;$ 
             $D(u) = 1$  for all  $u \in N[F(v_i)]$ ;
        else if  $(C = \emptyset)$  then
             $L(v_i) = 2$ 
             $L(u) = 2$  for some  $u \in N(v_i)$  such that  $L(u) = 0$ ;
        else
            Take  $v_j$  such that  $j = \min\{k \mid v_k \in C\}$ ;
             $L(v_i) = 2;$ 
             $L(v_j) = 2;$ 
        end if
    end if
end if
Output PD =  $\{v \mid L(v) = 2\}$ 
end
```

Theorem 37 ([24]) Let G be a strongly chordal graph on n vertices and m edges without isolated vertices. Algorithm MPDS can produce a minimum paired-dominating set of G in $O(m + n)$ time, if the strong (elimination) ordering of G is given in advance.

3.3.4 Paired-Domination in Inflated Graphs

The inflated graph G_I of the graph G without isolated vertices is obtained as follows: Each vertex x_i of degree $d(x_i)$ of G is replaced by a clique $X_i \cong K_{d(x_i)}$

and each edge (x_i, x_j) of G is replaced by an edge (u, v) in such a way that $u \in X_i, v \in X_j$, and two different edges of G are replaced by nonadjacent edges of G_I . An obvious consequence of the definition is that $|V(G_I)| = \sum_{x_i \in V(G)} d_G(x_i) = 2|E(G)|$, $\Delta(G_I) = \Delta(G)$, and $\delta(G_I) = \delta(G)$.

Kang et al. [52] gave some bounds on paired-domination number and presented a linear algorithm to compute a minimum paired-dominating set for an inflated tree.

Theorem 38 ([52]) *If G is a graph of order n with $\delta(G) \geq 2$, then $\gamma_{\text{pr}}(G_I) \geq n$ with equality if and only if G has a perfect matching.*

Theorem 39 ([52]) *If $\delta(G) \geq 2$, then $\gamma_{\text{pr}}(G_I) \leq 4|E(G)|/[\delta(G) + 1]$ and the bound is tight.*

3.4 Paired-Domination Subdivision Number of Graphs

For a vertex $v \in V(G)$, we denote by $N_2(v)$ the set of vertices of G at distance 2 from v and by $E_2(v)$ the set of edges of G with at least one endvertex in $N(v)$. A *leaf* is a vertex of degree one, and a *support vertex* is a vertex adjacent with a leaf. A support vertex is *strong* if it is adjacent with more than one leaf. We call the *core* of a graph and denote by $C(G)$ the set of vertices of G which are neither leaves nor support vertices. An edge $uv \in E(G)$ is *subdivided* if the edge uv is deleted, but a new vertex x is added, along with two new edges ux and vx . The vertex x is called a *subdivision vertex*. The *paired-domination subdivision number* $\text{sd}_{\gamma_{\text{pr}}}(G)$ of a graph G is the minimum number of edges that must be subdivided (where each edge in G can be subdivided at most once) in order to increase the paired-domination number. Similar definitions exist for the (connected, total) domination subdivision number $\text{sd}_\gamma(G)$. The first of these was introduced for the domination number in Velammal's thesis [74]. Favaron et al. [38] established upper bounds on the paired-domination subdivision number.

Theorem 40 ([38]) *Let G be a connected graph of order $n \geq 3$. If $C(G) = \emptyset$, then $\text{sd}_{\gamma_{\text{pr}}}(G) \leq 2$, and if $C(G) \neq \emptyset$, then $\text{sd}_{\gamma_{\text{pr}}}(G) \leq \min\{|E_2(v)| : v \in C(G)\}$.*

Theorem 41 ([38]) *Let $G = (V, E)$ be a connected graph of order $n \geq 3$ containing a vertex v such that $V = \{v\} \cup N(v) \cup N_2(v)$ and $N_2(v)$ is an independent set, and let F be the set of edges of G with exactly one endvertex in $N(v)$. Then $\text{sd}_{\gamma_{\text{pr}}}(G) \leq |F|$.*

Theorem 42 ([38]) *Let $G = (A \cup B, E)$ be a connected bipartite graph of order $n \geq 3$ containing a vertex v adjacent to every vertex of A and such that every vertex of A has a neighbor in $B \setminus \{v\}$. Then $\text{sd}_{\gamma_{\text{pr}}}(G) \leq n - 1$.*

Theorem 43 ([38]) *For every connected graph G on $n \geq 3$ vertices and m edges, $\text{sd}_{\gamma_{\text{pr}}}(G) = m$ if and only if G is a subdivided star.*

3.5 Distance Paired-Domination Numbers of Graphs

A set $D \subseteq V(G)$ is a k -distance dominating set of G if every vertex in $V(G) - D$ is within distance k of at least one vertex in D . The k -distance domination number $\gamma^k(G)$ of G equals the minimum cardinality among all k -distance dominating sets of G . A set D is a k -distance paired-dominating set of G if D is a k -distance dominating set of G and the induced subgraph $\langle D \rangle$ has a perfect matching. The k -distance paired-domination number $\gamma_p^k(G)$ is the minimum cardinality of a k -distance paired-dominating set of G .

3.5.1 Bounds on the k -Distance Paired-Domination Number of Graphs

Using a polynomial reduction from domination set, Raczek [66] proved that k -distance dominating set problem for bipartite graphs is NP-complete. Haynes and Slater [43] have proved that if G is a connected graph of order $n \geq 3$, then $\gamma_{pr}(G) \leq n - 1$ and characterized the extremal graphs. Raczek [66] generalized their result for k -distance paired-domination number of a graph.

Theorem 44 ([66]) *If T is a tree of order $n \geq k + 2$, then $\gamma_p^k(T) \leq 2\frac{n-1}{k+1}$.*

It is easily seen that $\gamma_p^k(G) \leq \gamma_p^k(T)$ for every spanning tree T of G .

Theorem 45 ([66]) *If G is a connected graph of order $n \geq k + 2$, then $\gamma_p^k(G) \leq 2\frac{n-1}{k+1}$.*

Raczek [66] characterized all trees for which $\gamma_p^k(T) = 2\frac{n-1}{k+1}$. For this purpose, the author defined \mathcal{R}^p to be the family of trees such that:

- (a) Each k -subdivide star $K_{1,t}^k$ belongs to \mathcal{R}^p .
- (b) Each tree T on $k + 2$ vertices belongs to \mathcal{R}^p .

Theorem 46 ([66]) *If T is a tree of order n , then*

$$\gamma_p^k(T) = 2\frac{n-1}{k+1}$$

if and only if T belongs to the family \mathcal{R}^p .

Raczek [66] characterized all connected graphs G for which $\gamma_p^k(G) = 2\frac{n-1}{k+1}$. For this purpose, a family \mathcal{R} of trees is defined as follows:

- (a) Each k -subdivided star $K_{1,t}^k$ belongs to \mathcal{R} .
- (b) Each connected graph G on $k + 2$ vertices belongs to \mathcal{R} .
- (c) The cycle C_{2k+3} belongs to \mathcal{R} .

Theorem 47 ([66]) *If G is a connected graph of order n , then $\gamma_p^k(G) = 2\frac{n-1}{k+1}$ if and only if G belongs to the family \mathcal{R} .*

Raczek [66] also gave a lower bound on the k -distance paired-domination number of tree in terms of order of T and the number of leaves in T and also characterized the extremal trees.

Theorem 48 ([66]) *If T is a tree of order $n \geq 2$, then*

$$(k + 1)\gamma_p^k(T) \geq n + 2k - kn_1(T),$$

where $n_1(T)$ is the number of leaves of T .

3.5.2 Distance Paired-Domination on Interval Graphs and Block Graphs

Chen et al. [25] presented two linear-time algorithms to find a minimum cardinality k -distance paired-dominating set in interval graphs and block graphs.

Let $G = (V, E)$ be an interval graph and its interval representation is I . For every vertex $u_i \in V$, I_i is the corresponding interval, and let a_i (b_i , respectively) denote the left endpoint (right endpoint, respectively) of interval I_i . We order the vertices of G by u_1, u_2, \dots, u_n in increasing order of their left endpoints.

Let $F(u_i) = u_j$ for $2 \leq i \leq n$, where $j = \min\{a | u_a u_i \in E \text{ and } a < i\}$. They define the notation $F^l(u)$ as follows:

$$F^l(u) = \begin{cases} F(u) & \text{if } l = 1; \\ F(F^{l-1}(u)) & \text{if } u \geq 2. \end{cases}$$

Algorithm k -MPDI. Find a minimum k -distance paired-dominating set of an interval graph.

Input: An interval graph $G = (V, E)$ with a vertex ordering u_1, u_2, \dots, u_n ordered by the increasing order of left endpoints, in which each vertex u_i has a label $D(u_i) = 0$.

Output: A minimum k -distance paired-dominating set kPD of G .

Method

```

 $kPD = \emptyset;$ 
For  $i = n$  to 1 do
    If  $(D(u_i) = 0)$  then
        If  $(F^k(u_i) \neq u_i \text{ and } F^{k+1}(u_i) \neq F^k(u_i))$  then
             $kPD = kPD \cup \{F^k(u_i), F^{k+1}(u_i)\};$ 
             $D(u) = 1 \text{ for every vertex } u \in N^k[F^k(u_i)] \cup N^k[F^{k+1}(u_i)];$ 
        else if  $(F^k(u_i) \neq u_i)$  then
             $kPD = kPD \cup \{u_1, u_2\};$ 
             $D(u) = 1 \text{ for every vertex } u \in N^k[u_1];$ 
        else
             $kPD = kPD \cup \{u_1, u_2\};$ 
             $D(u_i) = 1;$ 
        end if
    end if
end for

```

Chen et al. [25] presented the following algorithm for k -distance paired-domination problem in interval graphs.

Theorem 49 ([25]) *Given a vertex ordering ordered by the increasing order of left endpoints, the algorithm k-MPDI can produce a minimum k -distance paired-dominating set of an interval graph G on n vertices and m edges in $O(m + n)$.*

Using the similar labeling method, Chen et al. [25] presented a linear-time algorithm to finding a minimum k -distance paired-dominating set in block graphs.

4 Total Domination in Graphs

For a graph $G = (V, E)$, we denote by H_G the *open neighborhood hypergraph*, abbreviated by ONH, of G ; that is, $H_G = (V, C)$ is the hypergraph with vertex set $V(H_G) = V$ and with hyperedge set $E(H_G) = C$ consisting of the open neighborhood of vertices of V in G . The transversal number of the ONH of a graph is precisely the total domination number of the graph.

Observation 2 *If G is a graph with no isolated vertex and H_G is the ONH of G , then $\gamma_t(G) = \tau(H_G)$.*

It is easily seen that total domination in graphs can be translated to the problem of finding transversals in hypergraphs. The main advantage of considering hypergraphs rather than graphs is that the structure is easier to handle. This idea of using transversals in hypergraphs to obtain results on total domination in graphs first appeared in a paper by Thomassé and Yeo [71]. Subsequent to the Thomassé–Yeo paper, several results on total domination in graphs have been obtained using transversals in hypergraphs that appear very difficult to obtain using graph-theoretic techniques.

4.1 Complexity and Algorithmic Results on the Total Domination Problem

The basic complexity question concerning the decision for the total domination number takes the following form:

Total Domination Set

Instance: A graph $G = (V, E)$ and a positive integer k .

Question: Does G has a TDS of cardinality at most k ?

Table 2 [47] summarized the NP-completeness results for the total domination number with the corresponding citation.

Laskar et al. [60] constructed the first linear algorithm for computing the total domination number of a tree. Bertossi and Gori [12] constructed an $O(n \ln n)$ algorithm for the total domination number of an interval graph of order n . Kratsch and Stewart [56] constructed an $O(n^6)$ algorithm for computing the total domination number of a cocomparability graph of order n .

Table 2 NP-complete results for the total domination number

Graph classes	NP-completeness result	Citation
General graph	NP-complete	[64]
Bipartite graph	NP-complete	[64]
Comparability graph	NP-complete	[64]
Split graph	NP-complete	[59]
Chordal graph	NP-complete	[60]
Line graph	NP-complete	[63]
Line graph of bipartite graph	NP-complete	[63]
Claw-free graph	NP-complete	[63]
Circle graph	NP-complete	[54]
Interval graph	P	[11, 12, 18, 54, 67]
Permutation graph	P	[14, 30, 58]
Strongly chordal graph	P	[16]
Dually chordal graph	P	[58]
Cocomparability graph	P	[57]
Asteroidal triple-free graph	P	[56]
Distance-hereditary graph	P	[20, 58]
k -polygon graph (fixed $k \geq 3$)	P	[58]
Partial k -tree (fixed $k \geq 3$)	P	[8, 70]

4.2 Heuristics

Since total dominating set is NP-complete, it is important to develop efficient approximation algorithms for the TDS problem. Henning and Yeo [47, 49] gave a greedy approach to find a TDS in a graph G .

Let H_G be the ONH of G . Then, each hyperedge of H_G has size at least $\delta(G)$. Let H be obtained from H_G by shrinking all hyperedges of H_G , if necessary, to hyperedges of size $\delta(G)$. Then, H is a $\delta(G)$ -uniform hypergraph with n vertices and n hyperedges.

Heuristic Total Domination

Input: A graph $G = (V, E)$ with minimum degree $\delta(G) \geq 1$ and order n .

Output: A TDS of G .

- Step 1. Construct a hypergraph H where $V(H) = V$ and $E(H_G)$ consists of n hyperedges $N_x, x \in V$, where N_x is a set of $\delta(G)$ neighbors of x in G .
- Step 2. Compute the degrees of the vertices in H .
- Step 3. Initialize a hash table, L , of size n .
- Step 4. For each vertex, x , add it to the list $L(d(x))$ in L .
- Step 5. Set $T = \emptyset$.
- Step 6. If $L(k)$ is empty for all $k > 0$, then stop and output. Otherwise, go to Step 8.
- Step 7. Find the largest k such that $L(k)$ is not empty.

- Step 8. Put a vertex, x , from $L(k)$ to our set T and then delete it from $L(k)$.
 Step 9. For all y in $N(x)$ decrease the degree of y by 1 and move y from $L(i)$ to $L(i - 1)$ where i was the degree of y .
 Step 10. Return to Step 6.

By construction, the resulting set T hits every hyperedge of H and is therefore a transversal of H . It is known in [49] that $|H| \leq (\frac{1+\ln\delta}{\delta})n$ where $\delta = \delta(G)$. Every transversal of H is a transversal of H_G , and every transversal of H_G is a TDS of G . Hence the set T produced by this greedy algorithm is a TDS of G .

Theorem 50 ([49]) *If G is a graph with minimum degree $\delta \geq 2$ and order n , then heuristic total domination produces a TDS T in G satisfying*

$$|T| \leq \left(\frac{1 + \ln \delta}{\delta}\right)n.$$

The complexity of the algorithm is $O(n + \delta n)$.

For sufficiently large δ , the TDS T produced by heuristic total domination is essentially optimal, as can be deduced from the following result due to Thomasse and Yeo [71].

Theorem 51 ([71]) *For any $\epsilon > 0$ and for sufficiently large δ , there exists a δ -uniform hypergraph H with n vertices and n hyperedges satisfying*

$$\tau(H) > \left(\frac{(1 - \epsilon) \ln \delta}{\delta}\right)n.$$

4.3 Bounds on the Total Domination Number of Graphs

Any TDS must have a nonempty intersection with every open neighborhood. Hence if G is a graph with no isolated vertices, then $\gamma_t(G) \geq \rho^o(G)$.

4.3.1 Upper Bounds on the Total Domination Number of Graphs with Minimum Degree One

Cockayne et al. [29] obtained the following upper bound on the total domination number of a connected graph in terms of the graph.

Theorem 52 ([29]) *If G is a connected graph of order $n \geq 3$, then $\gamma_t(G) \leq 2n/3$.*

Brigham et al. [15] characterized the connected graphs of order at least 3 with total domination number exactly two-thirds their order.

Theorem 53 ([15]) *Let G be a connected graph of order $n \geq 3$. Then $\gamma_t(G) = 2n/3$ if and only if G is C_3 , C_6 or $H \circ P_2$ for some connected graph H .*

4.3.2 Upper Bounds on the Total Domination Number of Graphs with Minimum Degree at Least Two

Sun [69] showed that the upper bound in Theorem 52 can be improved if we restrict G to be connected graphs with the minimum degree two.

Theorem 54 ([69]) *If G is a connected graph of order n with $\delta(G) \geq 2$, then $\gamma_t(G) \leq \lfloor \frac{4}{7}(n+1) \rfloor$.*

Chvátal and McDiarmid [28] and Tuza [73] independently established the following results about transversals in hypergraphs (see also Thomassé and Yeo [71] for a short proof of this result).

Theorem 55 ([28, 73]) *Every hypergraph H where all hyperedges have size at least three on n vertices and m hyperedges has a transversal T such that $4|T| \leq m + n$.*

As an immediate consequence of Theorem 55, Henning [47] got that the total domination number of a graph with minimum degree at least three is at most one-half order. Archdeacon et al. [7] found an elegant proof of Theorem 56.

Theorem 56 ([7, 47]) *If G is a graph of order n with minimum degree $\delta(G) \geq 3$, then $\gamma_t(G) \leq n/2$.*

Thomasse and Yeo [71] further proved the following result.

Theorem 57 ([71]) *Every 4-uniform hypergraph on n vertices and m hyperedges has a transversal with no more than $(5n + 4m)/21$ vertices.*

As a consequence of Theorem 57, they have the following result.

Theorem 58 ([71]) *If G is a graph of order n with minimum degree $\delta(G) \geq 4$, then $\gamma_t(G) \leq 3n/7$.*

5 Conclusion

In this chapter, we study three variants of domination: power domination, paired-domination, and total domination which were widely studied in recent years. We give a review of these variants of domination from complexity, approximation algorithm, polynomial algorithm, and bound.

Acknowledgements Research was partially supported by the National Nature Science Foundation of China (no. 10971131), Pujiang Project of Shanghai (no. 09PJ1405000), and Shanghai Leading Academic Discipline Project (no. S30104).

Cross-References

- [A Unified Approach for Domination Problems on Different Network Topologies](#)
 - [Algorithmic Aspects of Domination in Graphs](#)
 - [Connected Dominating Set in Wireless Networks](#)
-

Recommended Reading

1. A. Aazami, K. Stilp, Approximation algorithm and hardness for domination with propagation. *SIAM J. Discret. Math.* **23**, 1382–1399 (2009)
2. A. Aazami, K. Stilp, Approximation algorithm and hardness for domination with propagation. Preprint, 2007, available online at <http://arxiv.org/abs/0710.2139>
3. A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, 1974)
4. J. Alber, H.L. Bodlaender, H. Fernau, T. Kloks, R. Niedermeier, Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica* **33**, 461–493 (2002)
5. J. Alber, H. Fan, M.R. Fellows, H. Fernau, R. Niedermeier, F. Rosamond, U. Stege, A refined search tree technique for dominating set on planar graphs. *J. Comput. Syst. Sci.* **71**, 385–405 (2005)
6. P. Alimonti, V. Kann, Some APX-completeness results for cubic graphs. *Theor. Comput. Sci.* **237**, 123–134 (2000)
7. D. Archdeacon, J. Ellis-Monaghan, D. Fischer, D. Froncek, P.C.B. Lam, S. Seager, B. Wei, R. Yuster, Some remarks on domination. *J. Graph Theory* **46**, 207–210 (2004)
8. S. Arnborg, J. Lagergren, D. Seese, Easy problems for tree-decomposable graphs. *J. Algorithms* **12**, 308–340 (1991)
9. T.L. Baldwin, L. Mili, M.B. Bpisen Jr., R. Adapa, Power system observability with minimal phasor measurement placement. *IEEE Trans. Power Syst.* **8**, 707–715 (1993)
10. R. Barrera, D. Ferrero, Power domination in cylinders, tori, and generalized peterson graphs. *Networks* **58**, 43–49 (2011)
11. A.A. Bertossi, Total domination in interval graphs. *Inf. Process. Lett.* **23**, 131–134 (1986)
12. A.A. Bertossi, A. Gori, Total domination and irredundance in weighted interval graphs. *SIAM J. Discret. Math.* **1**, 317–327 (1988)
13. B. Bollobás, *Random Graphs* (Academic/Harcourt Brace Jovanovich, London, 1985)
14. A. Brandstädt, D. Kratsch, On domination problems on permutation and other graphs. *Theor. Comput. Sci.* **54**, 181–198 (1987)
15. R.C. Brigham, J.R. Carrington, R.P. Vitray, Connected graphs with maximum total domination number. *J. Combin. Comput. Combin. Math.* **34**, 81–96 (2000)
16. G.J. Chang, Labeling algorithms for the domination problems in sun-free chordal graphs. *Discret. Appl. Math.* **22**, 21–34 (1988)
17. G.J. Chang, Algorithmic aspects of domination in graphs, in *Handbook of Combinatorial Optimization*, vol. 3, ed. by D.Z. Du, P.M. Paradalos (Kluwer Academic, Boston, 1998), pp. 339–405
18. M.S. Chang, Efficient algorithms for the problem on interval and circular-arc graphs. *SIAM J. Comput.* **27**, 1671–1694 (1998)

19. T.Y. Chang, W.E. Clark, The domination numbers of the $5 \times n$ and $6 \times n$ grid graphs. *J. Graph Theory* **17**, 81–107 (1993)
20. M.S. Chang, S.C. Wu, G.J. Chang, H.G. Yeh, Domination in distance-hereditary graphs. *Discret. Appl. Math.* **116**, 103–113 (2002)
21. X.G. Chen, L. Sun, H.M. Xing, Paired-domination numbers of cubic graphs (in Chinese). *Acta Math. Sci. Ser. A Chin. Ed.* **27**, 166–170 (2007)
22. X.G. Chen, W.C. Shiu, W.H. Chan, Upper bounds on the paired-domination number. *Appl. Math. Lett.* **21**, 1194–1198 (2008)
23. L. Chen, C.H. Lu, Z.B. Zeng, Hardness results and approximation algorithms for (weighted) paired-domination in graphs. *Theor. Comput. Sci.* **410**, 5063–5071 (2009)
24. L. Chen, C.H. Lu, Z.B. Zeng, A linear-time algorithm for paired-domination problem in strongly chordal graphs. *Inf. Process. Lett.* **110**, 20–23 (2009)
25. L. Chen, C.H. Lu, Z.B. Zeng, Discrete paired-domination problems on subclasses of chordal graphs. *Theor. Comput. Sci.* **410**, 5072–5081 (2009)
26. L. Chen, C.H. Lu, Z.B. Zeng, Labelling algorithm for paired-domination problems in block and interval graphs. *J. Comb. Optim.* **19**, 457–470 (2010)
27. T.C.E. Cheng, L.Y. Kang, E.F. Shan, A polynomial-time algorithm for the paired-domination problem on permutation graphs. *Discret. Appl. Math.* **157**, 262–271 (2009)
28. V. Chvátal, C. McDiarmid, Small transversals in hypergraphs. *Combinatorica* **12**, 19–26 (1992)
29. E.J. Cockayne, R.M. Dawes, S.T. Hedetniemi, Total domination in graphs. *Networks* **10**, 211–219 (1980)
30. D.G. Corneil, L. Stewart, Dominating sets in perfect graphs. *Discret. Math.* **86**, 19–26 (1990)
31. B. Courcelle, J.A. Makowsky, U. Rotics, Linear time solvable optimization problems on graphs of bounded clique width, in *Workshop on Graph-Theoretic Concepts in Computer Science*. Lecture Notes in Computer Science, vol. 1517 (Springer, Berlin, 1998), pp. 1–16
32. P. Dorbec, M. Mollard, S. Klavžar, Špacapan, Power domination in product graphs. *SIAM J. Discret. Math.* **22**, 554–567 (2008)
33. P. Dorbec, S. Gravier, M.A. Henning, Paired-domination in generalized claw-free graphs. *J. Combin. Optim.* **14**, 1–7 (2007)
34. P. Dorbec, S. Gravier, Paired-domination in P_5 -free graphs. *Graphs Combin.* **24**, 303–308 (2008)
35. P. Dorbec, S. Gravier, Paired-domination in subdivided star-free graphs. *Graphs Combin.* **26**, 43–49 (2010)
36. M. Dorfling, M.A. Henning, A note on power domination in grid graphs. *Discret. Appl. Math.* **154**, 1023–1027 (2006)
37. O. Favaron, M.A. Henning, Paired-domination in claw-free cubic graphs. *Graphs Combin.* **20**, 447–456 (2004)
38. O. Favaron, H. Karami, S.M. Sheikholeslami, Paired-domination subdivision number of graphs. *Graphs Combin.* **25**, 503–512 (2009)
39. M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W.H. Freeman, New York, 1979)
40. W. Goddard, M.A. Henning, A characterization of cubic graphs with paired-domination number three-fifths their order. *Graphs Combin.* **25**, 675–692 (2009)
41. J. Guo, R. Niedermeier, D. Raible, Improved algorithm and complexity results for power domination in graphs. *Algorithmica* **52**, 177–202 (2008)
42. T.W. Haynes, S.M. Hedetniemi, M.A. Henning, Domination in graphs applied to electric power networks. *SIAM J. Discret. Math.* **15**, 519–529 (2002)
43. T.W. Haynes, P.J. Slater, Paired-domination in graphs. *Networks* **32**, 199–206 (1998)
44. T.W. Haynes, S.T. Hedetniemi, P.J. Salter (eds.), *Fundamentals of Domination in Graphs* (Marcel Dekker, New York, 1998)
45. T.W. Haynes, S.T. Hedetniemi, P.J. Salter (eds.), *Domination in Graphs: Advanced Topics* (Marcel Dekker, New York, 1998)
46. M.A. Henning, Graphs with large paired-domination number. *J. Comb. Opt.* **13**, 61–78 (2007)

47. M.A. Henning, A survey of selected recent results on total domination in graphs. *Discret. Math.* **309**, 32–63 (2009)
48. M.A. Henning, An upper bound on the paired-domination number in terms of the number of edges in the graph. *Discret. Math.* **310**, 2847–2857 (2010)
49. M.A. Henning, A. Yeo, A transition from total domination in graphs to transversals in hypergraphs. *Quaest. Math.* **30**, 417–436 (2007)
50. M.S. Jacobson, L.F. Kinch, On the domination number of products of graphs: I. *Ars Combin.* **18**, 33–4 (1984)
51. D.S. Johnson, Approximation algorithm for combinatorial problems. *J. Comput. Syst. Sci.* **9**, 256–278 (1974)
52. L.Y. Kang, M.Y. Sohn, T.C.E. Cheng, Paired-domination in inflated graphs. *Theor. Comput. Sci.* **320**, 485–494 (2004)
53. J. Kneis, D. Mölle, P. Rossmanith, Parameterized power domination complexity. *Inf. Process. Lett.* **98**, 145–149 (2006)
54. J.M. Keil, Total domination in interval graphs. *Inf. Process. Lett.* **22**, 171–174 (1986)
55. G. Kortsarz, On the hardness of approximating spanners. *Algorithmica* **30**, 432–450 (2001)
56. D. Kratsch, Domination and total domination on asteroidal triple-free graphs, in *Proceedings of the 5th Twente Workshop on Graphs and Combinatorial Optimization*, Enschede, 1997. *Discret. Appl. Math.* **99**, 111–123 (2000)
57. D. Kratsch, L. Stewart, Domination on cocomparability graphs. *SIAM J. Discret. Math.* **6**, 400–417 (1993)
58. D. Kratsch, L. Stewart, Total domination and transformation. *Inf. Process. Lett.* **63**, 167–170 (1997)
59. R.C. Laskar, J. Pfaff, Domination and irredundance in split graphs. Technical Report 430, Clemson University, Department of Mathematical Sciences, 1983
60. R.C. Laskar, J. Pfaff, S.M. Hedetniemi, S.T. Hedetniemi, On the algorithmic complexity of total domination. *SIAM J. Algebr. Discret. Method* **5**, 420–425 (1984)
61. C.S. Liao, D.T. Lee, Power domination problem in graphs, in *Proceeding of the 11th International Computing and Combinatorics*. Lecture Notes in Computer Science, vol. 3595 (Springer, New York, 2005), pp. 818–828
62. C. Lund, M. Yannakakis, On the hardness of approximating minimization problems. *J. ACM* **41**, 960–981 (1994)
63. A.A. McRae, Generalizing NP-completeness proofs for bipartite and chordal graphs. Ph.D. thesis, Clemson University, 1994
64. J. Pfaff, R.C. Laster, S.T. Hedetniemi, NP-completeness of total and connected domination and irredundance for bipartite graphs. Technical Report 428, Clemson University, Department of Mathematical Sciences, 1983
65. H. Qiao, L.Y. Kang, M. Cardei, D.Z. Du, Paired-domination of trees. *J. Glob. Opt.* **25**, 43–54 (2003)
66. J. Raczek, Distance Paired domination number of graphs. *Discret. Math.* **308**, 2473–2483 (2008)
67. G. Ramalingam, C.P. Rangan, Total domination in interval graphs revised. *Inf. Process. Lett.* **27**, 17–21 (1988)
68. E. Sampathkumar, H.B. Walika, The connected domination number of a graph. *J. Math. Phys. Sci.* **13**, 607–613 (1979)
69. L. Sun, An upper bound for the total domination number. *J. Beijing Inst. Technol.* **4**, 111–114 (1995)
70. J.A. Telle, Complexity of domination-type problems in graphs. *Nordic J. Comput.* **1**, 157–171 (1994)
71. S. Thomassé, A. Yeo, Total domination of graphs and small transversal of hypergraphs. *Combinatorica* **27**, 473–487 (2007)
72. J.A. Telle, A. Proskurowski, Algorithms for vertex partitioning problems on partial k -trees. *SIAM J. Discret. Math.* **10**, 529–550 (1993)
73. Z. Tuza, Covering all cliques of a graph. *Discret. Math.* **86**, 117–126 (1990)

-
- 74. S. Velammal, Studies in graph theory: covering, independence, domination and related topics. Ph.D. thesis, Manonmaniam Sundaranar University, Tirunelveli, 1997
 - 75. P.M. Weichsel, The Kronecker product of graphs. *Proc. Amer. Math. Soc.* **13**, 47–52 (1962)
 - 76. G.J. Xu, L.Y. Kang, E.F. Shan, M. Zhao, Power domination in block graphs. *Theor. Comput. Sci.* **359**, 299–305 (2006)
 - 77. M. Zhao, L.Y. Kang, G.J. Chang, Power domination in graphs. *Discret. Math.* **306**, 1812–1816 (2006)

Index

A

ACO. *See* Ant colony optimization (ACO)
Acyclic coloring, 2097, 2110–2132, 2174
Acyclic subdigraph problem, 2746
Adams–Johnson bound (AJB), 2766
Adaptive memory programming, 3263, 3265–3266, 3297, 3323, 3326, 3337–3338, 3353
Ad hoc assignments, 1112–1113, 1116
Admissible transformation, 2759–2761, 2763–2765
Affinity propagation, 610
Agglomerative method, 2468–2471
AJB. *See* Adams–Johnson bound (AJB)
Alanine torsion angle, 3242–3244
Algebraic methods, 1250, 1271–1286
Algorithm design, 3326
Algorithms, 221–274, 2816, 2818–2819, 2821, 2822, 2824–2826, 2829, 2830, 2832–2836, 2838, 2839, 2842
Almost Any-Fit algorithm, 466, 470
Almost Worst-Fit (AWF) algorithm, 470, 487, 515
Amino acids, 3221–3230, 3232–3234, 3236, 3238, 3239, 3242, 3246, 3247, 3250, 3252, 3254
Analysis, 1755–1812
Anomalous algorithm, 487
Ant colony optimization (ACO), 300–303, 2782, 2789–2791
Antihole, 1571, 1572
Anti-Monge, 2793, 2794
Antiweb, 1571, 1572
Any-Fit constraint, 466, 470, 471, 481, 493, 494, 507, 510, 514
Applications, 3179–3255
Approximability ratio, 2653, 2657, 2658
Approximate dynamic programming, 2989, 2990, 3006, 3011

Approximation, 1126, 1138–1140, 1142, 1143, 1145, 1147, 1148, 1150
Approximation algorithms, 3, 6, 8, 17–20, 23, 32, 726, 727, 735, 749, 779, 1295–1297, 1300, 1303, 1306, 1566–1567, 1573, 1578, 1585, 1586, 1642–1643, 1648–1658, 1661, 1663, 1665, 2751
Arborescence, 998, 1023, 1025, 1026, 1033
Area coverage, 901, 904–913, 917, 918
Assignment constraints, 2747
Assignment problem, 1068–1071, 1074–1076, 1082–1085, 1088–1090, 1094–1099, 1103–1113, 1117
Asymmetric partitionable cycle, 1579
Asymptotic analysis, 2754, 2759, 2788, 2795–2799, 2801, 2802, 2804
Auction, 2256–2258, 2261, 2273–2275, 2287–2289, 2291, 2292, 2295–2298
Autocorrelation, 284–295, 303, 304
AWF. *See* Almost Worst-Fit (AWF) algorithm
Azimuthal equidistant projection, 3190

B

Backward local independence number (BLIN), 1625–1627
Balanced cut, 1645, 1646, 1648–1653
Balanced r -partite graph, 1234
Balanced tree decomposition, 1649–1650
Bases, 1601, 1603, 1604
heuristic, 2991–3011
policy, 2989, 2990
sequences (Golay, base, Turyn-type), 288, 292, 293
Batch updates, 1715–1751
Benders’ decomposition, 2779–2781, 2783
Best-fit decreasing (BFD) algorithm, 480, 481, 483, 484, 487, 488, 491

- Best improvement rule, 2785
 Best-Two-Fit (B2F) algorithm, 482, 484, 498, 519
 Beta-complex, 2685–2731
 B2F. *See* Best-Two-Fit (B2F) algorithm
 BFD. *See* Best-fit decreasing (BFD) algorithm
 Bibranching, 1023, 1025, 1026
 Binary classification, 598
 Binary search method, 1313, 1317, 1318
 Binary search trees, 52–53, 76–85
 Bin covering problems, 462
 Bin packing game, 503, 504
 Bin stretching, 464, 491, 519, 520
 Bioinformatics, 1590
 Biological networks, 655–659
 Biquadratic assignment problem, 2743, 2802–2804
 Birkhoff’s theorem, 2747, 2768, 2771
 BLIN. *See* Backward local independence number (BLIN)
 Blocker, 1003, 1010, 1023–1031, 1033, 1050, 1057, 1058
 Blocking pair, 1003, 1023, 1027, 1030, 1031
 Boolean quadric, 2757
 Boundary recognition, 1416, 1445–1452
 Bounded max coloring, 1877, 1881, 1906–1909
 Bounded space algorithm, 463, 465, 466, 470–473, 489, 490, 492, 512, 513
 Bound-factors, 2855, 2857, 2860–2863, 2871, 2872, 2874, 2875, 2880, 2887, 2888
 Bounds, 2750, 2752–2756, 2759–2783, 2791, 2794, 2799, 2801, 2804, 2805
 Box- $1/d$ -Mengerian, 1003
 Box-half-Mengerian (box-1/2-Mengerian), 1047, 1048, 1050, 1055–1056
 Box-integral, 997–999, 1003, 1004, 1009, 1014, 1019
 Box-Mengerian, 997, 1002, 1005, 1010–1029, 1031, 1047
 box-TDI. *See* Box-totally dual integral (box-TDI)
 Box-totally dual integral (box-TDI), 997–1000, 1002, 1004–1006, 1008–1012, 1016, 1020, 1038, 1040, 1041, 1043–1046, 1055, 1056
 Brain networks, 661–662, 3058, 3059, 3066–3083
 Branch-and-bound, 2779–2780, 3193, 3214–3215, 3222, 3224
 Branch-and-cut, 1573, 1584, 1587, 2779, 2781–2782
 “Buckyball” rule, 3201, 3207, 3208
- C**
 Capacitated set cover, 1614
 Capacitated transportation problem (CTP), 2770
 Capacity analysis, 2524–2527, 2530–2535
 Cardinality, 3181, 3183, 3217, 3224, 3229
 Cardinality constraints, 510–513
 CDC. *See* Continuous data collection (CDC)
 CDG. *See* Combinatorial Delaunay graph (CDG)
 CDM. *See* Combinatorial Delaunay map (CDM)
 CDS. *See* Connected dominating set (CDS)
 CED. *See* Critical edge disruptor (CED)
 Central graph, 1217, 1242
 Channel assignment, 2361, 2365–2375
 Characteristic path length, 3060–3065, 3068, 3071, 3072, 3078–3083
 Checkable codes, 2679
 Chemical networks, 659–661
 Chessboard, 2792, 2793, 2801
 Chordal graphs, 224, 231, 242, 248, 251–266
 Chromatic index, 1237, 1239
 Chromatic number, 1200, 1203–1210, 1215, 1219, 1223, 1226, 1232, 1236, 1238, 1562, 1566, 1576, 1624
 Circle packing, 2549–2581
 Circuit, 1601, 1607
 Circulant, 2792–2794
 Class constrained bin packing problem (CLCP), 513–514
 Class constraints, 518, 520
 Claw, 1561, 1567, 1572
 CLCP. *See* Class constrained bin packing problem (CLCP)
 Clique, 633, 634, 646–655, 657, 659, 662, 663, 666–668, 1560, 1562–1563, 1590, 1591
 inequalities, 2757
 number, 1562, 1575, 1584, 1588
 relaxations, 650–655, 657, 662, 667, 668, 1559–1592
 Clique separator (CS), 1637–1639
 Clustering, 2306, 2308, 2331, 2347, 2348
 assumption, 614
 coefficient, 3060–3065, 3068, 3069, 3071, 3072, 3078–3083
 items, 518
 Clutter, 1001–1003, 1013, 1020–1032, 1049–1052, 1057
 Coccomparability graphs, 224, 231, 268–271
 Cognitive radio, 2253–2298
 Cographic matroid, 1008, 1053, 1055

- Cohesive subgroups, 1560, 1563, 1574, 1575, 1591
Co- k -plex, 1563–1573
 chromatic number, 1566
 coloring, 1565–1567
Collagen, 3232–3236, 3238, 3255
Collection, 1001, 1011–1013, 1016, 1022, 1024, 1032, 1034, 1035, 1041, 1045–1048, 1050–1052, 1056–1058
Color
 classes, 1200, 1202, 1203, 1214, 1224, 1231, 1235–1242
 sequence, 1202
Coloring, 1562, 1565, 1567, 1576, 1577, 1584
Color-set (CS) algorithm, 513, 514
Combinatorial algorithms, 1573, 1584, 1590, 1592
Combinatorial Delaunay graph (CDG), 1429, 1430, 1438–1439
Combinatorial Delaunay map (CDM), 1440
Combinatorial methods, 1250–1271
Combinatorial optimization, 534, 542, 548, 551, 554, 559–589, 934, 978
Community detection, 3091, 3102, 3107, 3110, 3129
Compact routing scheme, 3090, 3120–3129
Competitive, 2255–2257, 2260–2264, 2269–2271, 2273, 2276–2278, 2280–2282, 2287–2288, 2291, 2294–2297
Competitive ratio, 2255, 2256, 2260–2264, 2270–2271, 2273, 2276–2278, 2280–2282, 2294–2295, 2297
Complexity, 97, 113–122, 124, 129–130, 142, 457, 462, 463, 465, 467, 468, 474, 478, 479, 482, 483, 485, 486, 498, 500, 505, 512–514, 518, 2744–2745, 2747–2751, 2766, 2775, 2783, 2785–2786, 2793, 2794
Complexity analysis, 3214–3216
Computational complexity, 727
Computational geometry, 1126, 1128, 1130, 1838, 3182, 3185, 3190, 3204, 3207
Computational lower bound, 726, 739–742
Computer codes, 2791
Concave minimization, 867, 892
Conditional logic, 2854, 2880–2886
Conflicts, 515–517
Conic representation, 2778–2779
Connected dominating set (CDS), 783–829, 1608, 1614, 1616–1622, 1624, 2557, 2558, 2565, 2569, 2576–2581
Connected k -clique, 1579
Connected vertex cover, 1613
Conservative algorithms, 464, 465, 487–488
Consistent biclustering, 604–607
Constant QAP, 2792
Constraint-factors, 2869, 2871, 2873, 2887
Construction method, 2782, 2783, 2786
Continuation S3VM, 617–618
Continuous data collection (CDC), 2504, 2506, 2507, 2512–2516, 2527–2535, 2542, 2543
Continuous optimization, 617, 1068, 2074, 2819, 2844
Convex function, 2899–2902, 2904, 2907, 2929, 2931, 2932, 2941, 2946, 2951–2956, 2962, 2967, 2977
Convex hull, 2851–2855, 2857, 2859–2861, 2867–2872, 2874, 2876, 2878–2880, 2883, 2887, 2890
Convexity, 2305, 2308–2315, 2317–2322, 2329–2335, 2338, 2339, 2341–2343, 2350
Co-2-plex facets, 1569–1571
Cop number, 1534–1550, 1552, 1553
Cops and Robbers game, 1512, 1513, 1544, 1552
Cop-win graph, 1534–1538, 1541, 1543, 1546, 1549–1552
Core, 1224, 1225, 1237
Cost-to-reliability ratio problem, 1351
Co-training, 620
Coupled selection equations, 1067, 1076, 1085–1099, 1105–1107, 1109–1114, 1116–1118
Coverage deficiency function, 1608, 1611, 1620, 1621
Coverage function, 1608, 1609, 1613–1616, 1621
Covering number, w -, 1001, 1033
Crawler, 2457, 2477
c-repacking, 478, 479
Criminal network analysis, 1591
Critical area, 1817, 1818, 1823, 1829, 1848, 1853, 1855, 1858–1860
Critical edge disruptor (CED), 1632–1639, 1665
Critical node detection, 1632, 1664
Critical vertex disruptor (CVD), 1632, 1633, 1639–1645, 1665
Cross-free collection, 998, 1022, 1023
Cross product, 1219, 1221
Cross validation, 596
CS. *See* Clique separator (CS); Color-set (CS) algorithm

- CTP. *See* Capacitated transportation problem (CTP)
- Cubic assignment problem, 2743, 2802, 2803
- Curvature, 1609, 1613, 1615, 1624
- Cut(s)
 ratio, 1658–1661
 surplus of, 1340–1349
- CVD. *See* Critical vertex disruptor (CVD)
- Cyclic triple-exchange, 2784
- Cysteine torsion angle, 3244–3246
- D**
- DAG-width, 1521, 1522, 1533
- Data aggregation, 2568–2569
- Data correcting algorithms, 933, 934, 978, 979
- Data retrieval, 2417–2420
- DBP. *See* Dynamic bin packing (DBP)
- d -degenerate, 1211–1213, 1228
- Decisional influence maximization (DIM), 2459, 2460
- Decomposition, 2759, 2762, 2771, 2773–2775, 2778–2781, 2783
- Decoy experiments, 3248–3252
- Defective coloring, 1240
- Degree, 1993, 2001–2003, 2020, 2022, 2024
- Degree discount heuristics, 2466
- Delaunay tetrahedralization, 3207
- Delaunay triangulation, 3185, 3187, 3188, 3190, 3191, 3207–3209, 3214
- Dendrogram, 2469–2471
- Densest subgraph problem, 1585
- Densest t -subgraph problem, 1584–1590
- Dense subgraphs, 1585, 1991, 2010–2012
- Dependence system, 1608–1624
- Dependency graph, 1236
- Dependent set, 1608, 1624
- Diadic clutters, 1029, 1051
- Diameter, 787, 802, 806
- Dicuts, 1022–1024, 1026, 1027
- Dijoins, 1024, 1026–1029, 1057
- DIM. *See* Decisional influence maximization (DIM)
- Dimension, 2743, 2757, 2758, 2761, 2762, 2771, 2772, 2776, 2782, 2791, 2801, 2803, 2804
- Dimensionality reduction, 608–609
- $1/d$ -integral, 999–1000
- Dirac type, 1226, 1228
- Directed hypergraph, 1041
- Directed pathwidth, 1513, 1519, 1521–1524
- Directed treewidth, 1520–1522, 1533
- Discrete convex function, 2900
- Discrete convex program, 2889
- Discrete optimization, 837, 838, 872, 2849–2891
- Discrete size, 466
- Distance-hereditary graph, 224, 231, 271–274
- Distance- k coloring, 1576, 1584
- Distance- k independence, 1576, 1582
- Distributed robotic systems, 1066, 1111–1118
- Divisible sequence, 509
- Divisive algorithm, 2468, 2469, 2471
- $1/d$ -Mengerian, 1003
- DNF. *See* Dual Next-Fit (DNF)
- Dominating set, 2–7, 10, 13–16, 20, 37, 1563, 1564, 1584, 2557, 2566, 2568, 2569, 2576–2581
- Domination, 221–274, 3363–3390
- D-optimal matrices, 288, 289, 303, 304
- Double-bracket flow, 1084–1085
- Doubly stochastic, 2747, 2768, 2770, 2771, 2778
- $1/d$ -TDI. *See* $1/d$ -totally dual integral ($1/d$ -TDI)
- $1/d$ -totally dual integral ($1/d$ -TDI), 999, 1000, 1003
- Dual approximation, 507
- ϵ -Dual approximation algorithm, 486, 497
- Dual bin packing, 500
- Dual construction, 3181, 3198–3201
- Dual formulation, 3193, 3195, 3198, 3201
- Duality, 1757, 1767, 1770–1774, 1780
- Dual Next-Fit (DNF), 499, 500, 520
- Dual problem, 3185, 3195–3198
- Du, D.-Z., 95, 137, 138, 206, 874, 1620, 2308, 2590
- Dynamical systems, 1066, 1067, 1071–1076, 1079, 1084, 1087, 1088, 1092, 1095, 1098, 1103, 1117
- Dynamic bin packing (DBP), 479, 493, 500–502, 510
- Dynamic programming, 41–88
- Dynamic programming speedup, 42, 43, 63, 68, 77, 87
- E**
- ECC. *See* Equitable coloring conjecture (ECC)
- E Δ CC. *See* Equitable Δ -coloring conjecture (E Δ CC)
- Edges, 1991–2024
 cover, 1007, 1029–1031, 1050
 cover chromatic index, 1237
 labeled graphs, 1913–1945
 search, 1514–1516, 1524, 1525, 1528, 1529, 1531–1534

- Efficient polynomial-time approximation scheme (EPTAS), 725–727, 732–733, 737–742
- Eigenvalue bound, 2767, 2769, 2777
- Elimination bound, 2772, 2774, 2791
- EMST. *See* Euclidean minimal spanning tree (EMST)
- Energy-efficiency, 900, 903–909, 911, 913–915, 917, 924, 1155–1195
- EPTAS. *See* Efficient polynomial-time approximation scheme (EPTAS)
- Equilateral tetrahedron, 3193
- Equitable chromatic number, 1200, 1203, 1204, 1208, 1209, 1220, 1223, 1236
- Equitable chromatic threshold, 1201, 1203, 1210, 1211
- Equitable coloring conjecture (ECC), 1201
- Equitable Δ -coloring conjecture ($E\Delta CC$), 1206–1207, 1242
- Equitable k -partition, 1212, 1213
- Equitable subpartition, 1011–1013, 1016, 1022–1024, 1047, 1048, 1050, 1052
- Equitable total chromatic index, 1239
- Equitably k -choosable, 1223–1225
- Equitably k -colorable, 1200–1206, 1211, 1212, 1214, 1218, 1220, 1226
- Equitably subpartitionable (ESP), 999, 1010–1024, 1026, 1027, 1031, 1047–1052
- Equivalence theorems, 895
- ES. *See* Evolutionary strategies (ES)
- ESMTs. *See* Euclidean Steiner minimal trees (ESMTs)
- ESP. *See* Equitably subpartitionable (ESP)
- Euclidean minimal spanning tree (EMST), 3181, 3183, 3186, 3187
- Euclidean Steiner minimal trees (ESMTs), 3183–3186, 3190, 3191, 3201, 3255
- Evolutionary strategies (ES), 1076, 1078
- Evolutionary trees, 747–779
- Exact exponential algorithms, 1249–1289
- Exchange property, 1603, 1604
- Expectation-maximization (EM) algorithm, 610, 619
- F**
- Face routing, 1433, 1436, 1437, 1440
- Facility location (FL), 1294–1296, 1307
- Factorable program, 2850
- Family, 1001, 1036, 1040, 1042, 1043
- Fano matroid, 1007, 1008, 1019, 1053
- FASP. *See* Feedback arc set problem (FASP)
- Fault tolerance, 1294
- FCP. *See* Fuzzy clustering problems (FCP)
- Feature selection, 597, 606, 621–623
- Feedback arc set problem (FASP), 2746, 2795
- Feedback vertex set (FVS), 1010, 1013, 1015, 1031–1033, 1058
- FFD. *See* First-Fit Decreasing (FFD)
- FFI. *See* First-Fit Increasing (FFI)
- Filter models, 621
- Finance, 632, 644, 1590, 2890
- Financial networks, 664–667
- FIP. *See* Fuzzy integer programming (FIP)
- First-Fit Decreasing (FFD), 480–484, 487, 490, 491, 496, 498, 506, 509, 512, 515–517, 519, 520
- First-Fit Increasing (FFI), 498
- Fixed-parameter tractable, 1562, 1566, 1573, 1578, 1584
- Fixed point, 2112–2114, 2124–2125, 3015–3049
- FL. *See* Facility location (FL)
- Flexible manufacturing, 1111–1116
- Flows, 2002–2010, 2012–2016
- Folkman-Graham inequality, 2554–2556
- Fortified rollout, 3002–3004, 3006, 3011
- FPTAS. *See* Fully polynomial-time approximation scheme (FPTAS)
- Fractional combinatorial optimization
- general case, 1312–1318, 1322, 1326, 1333
 - linear, 1313–1316, 1321–1324, 1326, 1329–1334, 1338, 1340, 1349–1351
 - non-fractional version, 1314, 1332, 1340
 - uniform, 1315, 1321, 1338, 1341
- Fractional knapsack problem, 1350
- Fractional prize-collecting Steiner tree problem, 1352
- Fragile objects, 507
- Friendship ranking, 2495–2497
- Frieze–Yadegar, 2799
- FSTs. *See* Full Steiner trees (FSTs)
- Full Steiner trees (FSTs), 3183
- Fully polynomial-time approximation scheme (FPTAS), 462, 465, 485, 491, 497, 500, 519, 725, 726, 729–735, 739, 743
- Fuzzy approximation algorithms, 1382–1385
- Fuzzy clustering problems (FCP), 609
- Fuzzy combinatorial optimization, 1357–1404
- Fuzzy facility location, 1402
- Fuzzy graph, 1358–1360, 1370, 1385
- Fuzzy integer programming (FIP), 1363–1365
- Fuzzy linear programming, 1360–1363
- Fuzzy matching algorithm, 1370–1380
- Fuzzy matroids, 1380–1382

- Fuzzy multicommodity flows and edge-disjoint paths, 1390–1395
- Fuzzy network problems, 1395–1398
- FVS. *See* Feedback vertex set (FVS)
- G**
- GA. *See* Genetic algorithms (GAs)
- Gabriel graph (GG), 1432, 1435
- Gadget reduction, 2650
- GAs. *See* Genetic algorithms (GAs)
- Generalized set multi-cover, 1614
- Generalized systems, 835–895
- Generalized upper bounding (GUB) constraints, 2861, 2873–2877
- Generation of data sets, 1100–1103
- Genetic algorithms (GAs), 294–295, 297, 1076, 1078, 2782, 2789–2790
- Geodesic minimum spanning tree (GMST), 3188, 3190, 3191
- Geodesic Steiner minimal tree (GSMT), 3188–3191
- Geometric algorithms, 1127
- Geometric data structures, 1128, 1130
- Geometric minimal cut, 1815–1867
- Geometric optimization, 1460, 1463, 1464
- Geometric routing, 1430–1445, 1447
- GG. *See* Gabriel graph (GG)
- Gilbert networks, 1460, 1492, 1493, 1505
- Gilmore–Lawler bound, 2753, 2754, 2761–2766, 2774, 2779, 2783, 2791, 2801
- Girth, 1210, 1234, 1240
- Globally rigid, 1425–1430
- Global optimization, 2854, 2886–2887
- GMST. *See* Geodesic minimum spanning tree (GMST)
- Golomb’s sequence, 460–461, 472, 477
- GPP. *See* Graph partitioning problem (GPP)
- Gradient-constrained networks, 1459–1507
- Gradient flow, 1090–1091
- Graham, R.L., 20, 457, 487, 495, 496, 1261, 1716, 1720, 1721, 1723, 2196, 2197, 2199, 2209, 2296, 2308, 2550–2552, 3135, 3137, 3161–3163, 3202
- Graph-based clustering, 650–655
- Graph-based semi-supervised methods, 619
- Graphic matroid, 1008
- Graph partitioning problem (GPP), 2746, 2785
- Graphs, 785–790, 793, 794, 796–806, 808–817, 819, 821–823, 826–829, 1559–1592, 1991–1993, 1995–2015, 2017–2024, 2096–2174
- algorithm, 3173, 3174
- coloring, 648, 649, 666
- packing, 508
- packing problem, 2747
- searching, 1511–1553
- theory, 633, 642–655, 663
- GRASP. *See* Greedy randomized adaptive search (GRASP)
- Greedy algorithm, 2900, 2908
- Greedy embedding, 1431, 1442–1444
- Greedy heuristic, 2993
- Greedy randomized adaptive search (GRASP), 2782, 2786–2789, 2791, 2804
- Greedy routing, 1442–1444
- Grids, 3161–3171
- Group testing (GT), 93–142
- Growing partition, 2587, 2590, 2613–2617, 2620
- GSMT. *See* Geodesic Steiner minimal tree (GSMT)
- GT. *See* Group testing (GT)
- GUB constraints. *See* Generalized upper bounding (GUB) constraints
- H**
- Hadamard matrices, 284, 288–290, 295, 304
- Hahn–Grant bound (HGB), 2766, 2767
- Hajnal and Szemerédi Theorem, 1201, 1212, 1226–1228, 1233–1235
- Half-Mengerian (1/2-Mengerian), 1053–1055
- Hard clustering problems, 609
- Hard margin loss SVM, 602
- Hardness of approximation, 3, 7–18
- Harmonic-Fit algorithm (HF_k), 472, 474, 475, 489, 492, 515
- Hausdorff Voronoi diagram, 1817–1819, 1821, 1829–1835, 1841, 1843–1844, 1858–1867
- Heaviest (unweighted) t -subgraph problem, 1585
- Heider’s rule, 2785
- Helical axis, 3204
- Hereditary, 1566, 1576–1578
- Hermitian matrices, 2772, 2773
- Heuristic algorithm, 3202, 3206, 3208, 3209
- Heuristics, 2759, 2774, 2781–2791, 2795, 2804, 2805, 2817, 2818, 2829, 2834, 2840, 3181, 3182, 3185–3191, 3202, 3206–3217, 3219, 3224
- HGB. *See* Hahn–Grant bound (HGB)
- Hierarchical clustering, 608
- Hierarchical clustering-based algorithm, 2468, 2469

- HIV-1 protease, 3222, 3233, 3239–3242
Hole, 1571–1572
Homotopy method, 2105, 3017, 3025–3026, 3043–3049
Hyperedges, 1001–1004, 1007, 1011, 1012, 1014, 1019, 1024, 1029, 1030, 1047, 1048, 1051, 1052, 1057
Hypergraph, 998, 1001–1005, 1007–1015, 1017, 1019, 1023–1027, 1030–1033, 1041, 1046–1048, 1050–1052, 1241–1242
Hypo-Mycielskian, 1239
- I**
- IC. *See* Independent cascade (IC)
Ideal clutter/hypergraph, 1057
IFFD. *See* Iterated First-Fit Decreasing (IFFD) algorithm
ILFD. *See* Iterated Lowest-Fit Decreasing (ILFD) algorithm
ILIN. *See* Inward local independence number (ILIN)
IM. *See* Influence maximization (IM)
In-approximability, 1642–1645, 1665, 2678
Incremental algorithm, 2908–2909, 2918–2921, 2929, 2931, 2938, 2939, 2959–2962, 2965
Independence number, 1562
Independence system, 1601–1608
Independent cascade (IC), 2458–2460, 2463, 2465–2467
Independent set, 633, 646–650, 662, 666, 667, 1200, 1202, 1204, 1207, 1214–1216, 1226, 1227, 1563, 1565–1567, 1569–1571, 1582–1584, 1601–1608
stable set, 1562
vertex packing, 1562
Indexing, 2420–2440
Inductive learning, 614
Inductivity, 1625
Industry and government applications, 3337, 3339
Influence graph, 2458, 2459, 2461, 2462, 2465, 2467
Influence maximization (IM), 2456–2459, 2463, 2465, 2485, 2499, 3090, 3091, 3110–3120, 3129
Influence measure, 2457
Influence spread, 2458, 2459, 2465–2468
Influence spread computation (ISC), 2459–2462
Inner product, 2748–2750, 2771, 2775
- Integer labeling, 2148–2153, 2156–2157, 2161, 2166–2174, 3021–3023, 3027, 3030–3034, 3043–3044
Integer point, 2096, 2105, 2123–2125, 2139–2140, 3017, 3019, 3029–3040, 3042–3045, 3048–3049
Integer programming, 547, 1757, 1774, 1778–1779, 1784, 1787–1789, 1791–1792, 1799, 1808, 1958, 1975, 1983, 2097, 2098, 2100, 2102, 2109, 2110, 2112, 2116–2118, 2121, 2122, 2124–2136, 2138–2141, 2143, 2146–2150, 2156, 2157, 2161, 2165–2173, 3017, 3029–3049
Integer programming formulation, 1568, 1579–1581, 1586
Integral polyhedron, 1009
Interdiction, 1949–1984
Intersection family, 1215
Interval graph, 224, 231, 245–251, 268, 1217–1218, 1224
Inward local independence number (ILIN), 1626, 1627
ISC. *See* Influence spread computation (ISC)
Item divisibility, 491, 498, 509
Item fragmentation, 505–507
Iterated First-Fit Decreasing (IFFD) algorithm, 491, 498, 500
Iterated Lowest-Fit Decreasing (ILFD) algorithm, 500
Iterative peel-branch-and-cut algorithm, 1573
- J**
- Job-shop scheduling, 1667–1708
- K**
- Kalmanson matrices, 2793, 2794
Kaufman–Broeckx, 2753
 k -clans, 651
 k -cliques, 1574–1585
 k -club, 1574–1585, 1588, 1591
 k -club number, 1575
 k -colorable, 1200–1202, 1204–1206, 1211, 1212, 1214, 1218, 1220, 1223, 1226
Kelly-width, 1522, 1523, 1533
Kernel trick, 599, 600, 603, 618
Kernighan–Lin, 2785, 2786
Kernighan–Lin (K–L) algorithm, 2468–2469
 k -fold cross validation, 596
 k -means, 309–311
Kneser graphs, 1214–1216
Known objective function value, 2774

- Kohonen networks, 1083–1084
 Koopmans–Beckmann problem, 2744, 2750, 2758, 2761, 2767, 2768, 2773, 2775, 2792
 k -plex, 1563–1574, 1576, 1585, 1590, 1591
 Kronecker product, 2749, 2755, 2765, 2803
- L**
 LAD. *See* Logical analysis of data (LAD)
 Lagrange multipliers, 3197
 Lagrangian, 3196, 3197
 Laminar family, 1036
 Laplace, 2778
 Large, 2746, 2748, 2752, 2754, 2757, 2759, 2764, 2770, 2778, 2780–2783, 2785, 2788–2790, 2792, 2795, 2796, 2800, 2803, 2804
 Larger item on the bottom (LIB) constraints, 514–515
 Largest processing time (LPT) algorithm, 467, 496, 513
 Latent variable models, 609–612
 Lawler, E.L., 2743–2745, 2749, 2752, 2761, 2764, 2766, 2803
 Learning and intelligent optimization (LION), 2818, 2819, 2843–2844
 Leave-one-out cross validation (jackknife), 596
 Left higher graded, 2794
 Level-2 reformulation, 2755–2756, 2779, 2780
 Level-3 reformulation, 2756
 LIB. *See* Larger item on the bottom (LIB) constraints
 Lift-and-project, 2853, 2862
 Lifting, 2861, 2880–2882, 2884
 Limited enumeration, 2782, 2783
 Linear algorithm, 1330, 1331, 1334, 1338
 Linear arrangement problem, 2744–2746, 2795
 Linearization, 2748, 2751–2756, 2779–2782, 2799, 2803
 Linear ordering, 2757
 Linear programming, 1296, 1755–1812
 Linear threshold (LT), 2458–2467
 Linear-time algorithms, 465, 478, 483–484, 491
 Link prediction, 3090–3092, 3097, 3099, 3129
 LION. *See* Learning and intelligent optimization (LION)
 List coloring, 2098, 2104, 2106, 2108, 2109, 2116–2118, 2128, 2130, 2153, 2155
 Live-edge paths, 2461, 2464, 2465
 $L(p, q)$ -labeling, 2148–2173
 Localization, 1416–1430, 1438, 1448, 1452
 Localization ambiguity, 1417, 1425–1430
 Local minimum phenomenon, 1431, 1432, 1437, 1440
 Local search, 2751, 2782–2788
 Location graph, 2491–2495
 Logical analysis of data (LAD), 608
 Logical lifting, 2880–2881, 2883
 Long code, 2626, 2646–2650, 2652, 2662, 2664, 2666–2673, 2678
 Look ahead, 478, 479, 502
 Lower bounds (on APR), 466, 476–477, 498
 Lower rank, 1603
 LPT. *See* Largest processing time (LPT) algorithm
 LT. *See* Linear threshold (LT)
- M**
 Majorization order, 1202
 Manifold assumption, 614
 Manifold-cluster assumption, 614
 Marginal value, 1600
 Matchings, 1991–2024
 Matrix, 2744–2751, 2755, 2760–2805
 Matroid, 998, 1005, 1007, 1008, 1012, 1018–1022, 1038, 1046, 1053, 1055, 1056, 1603, 1605, 1607, 1613–1616, 1627
 Max coloring, 1871–1909
 Max edge coloring, 1876, 1877, 1880–1881, 1903–1906, 1908–1909
 Maximal independent set (MIS), 2558, 2559, 2561, 2562, 2565, 2566, 2569, 2571, 2576–2580
 Maximum clique, 1643
 Maximum edge subgraph problem, 1584, 1585
 Maximum flow problem, 1913–1945
 Maximum independent set, 2587–2590, 2593–2595, 2605–2608, 2614–2615, 2620
 Maximum influence arborescence (MIA), 2466, 2467
 Maximum influence out-arborescence, 2467
 Maximum influence path (MIP), 2467
 Maximum-mean cycle, 1312, 1313, 1338–1340
 Maximum mean-weight cut, 1312, 1313, 1340–1349
 Maximum profit-to-time ratio cycle, 1336–1338
 Maximum-ratio paths in acyclic graphs, 1315–1317
 Maximum-ratio spanning-tree problem, 1332, 1333
 Maximum-surplus cut, 1341, 1343

- Maximum-Weight Directed Hamiltonian Cycle, 1607
Maximum-Weight Hamiltonian Cycle, 1607
Maximum-Weight Stable Set of Unit-Disk Graphs, 1608, 1627
Maxwell's theorem, 3218, 3220, 3225, 3228, 3229, 3234, 3242, 3254, 3255
m-bounded chromatic number, 1205, 1206, 1208
m-bounded coloring, 1205, 1206
MCDS. *See* Minimum connected dominating sets (MCDS)
Mean-weight surplus of a cut, 1341
Mechanism design, 2253–2298
MECs. *See* Minimum energy configurations (MECs)
Megiddo's parametric search, 1312, 1313, 1317, 1328–1338, 1340, 1342, 1349–1352
parallel algorithms for, 1332, 1340
Melzak circle, 3191–3192, 3195, 3196, 3198, 3200, 3201
Mengerian hypergraphs, 1001–1005, 1011–1022, 1024–1027, 1029–1033, 1046, 1052, 1058
Message-passing method, 610
Metaheuristics, 540, 3263, 3268
Metropolis algorithm, 1076–1077
Meyniel's conjecture, 1535, 1538–1540, 1542, 1543, 1547, 1551–1553
MFF. *See* Modified First-Fit (MFF)
MFFD. *See* Modified First-Fit Decreasing (MFFD) algorithm
MF_k. *See* Most-*k*-Fit (MF_k) algorithm
MHF. *See* Modified Harmonic-Fit (MHF)
MHR. *See* Rejective Modified Harmonic (MHR) algorithm
MIA. *See* Maximum influence arborescence (MIA)
Microarray design, 566
MIL. *See* Multiple instance learning (MIL)
Minimal dependent set, 1608
Minimax optimization, 1950
Minimum connected dominating sets (MCDS), 608, 1609, 1616, 1618–1621, 2587–2590, 2597, 2617–2620
Minimum-cost dependent set, 1608
Minimum-cost flow problem, 1341
Minimum-Cost Submodular Cover, 1608–1616, 1624
Minimum dominating set, 2587, 2588, 2590, 2593, 2594, 2596, 2600, 2616–2617, 2620
Minimum energy configurations (MECs), 3218–3220, 3242, 3246, 3249, 3254
Minimum-label spanning tree, 1614
Minimum node-weighted Steiner tree, 1615, 1621, 1624
Minimum-power spanning tree, 1613
Minimum routing cost connected dominating set (MOC-CDS), 2579, 2580
Minimum Steiner tree, 1615
Minimum sum radii cover, 2589, 2609–2613, 2620
Minimum unit ball cover, 2587, 2591
Minimum vertex coloring, 34, 1624–1625
Minimum-weight Connected Dominating Set (MWCDS), 1608, 1609, 1621
Minor, 997, 1005, 1006, 1013, 1018–1021, 1023, 1049, 1053–1058
MIP. *See* Maximum influence path (MIP); Mixed-integer programming (MIP)
MIS. *See* Maximal independent set (MIS)
Mixed-integer nonlinear program, 2850–2862, 2867, 2868, 2879, 2880, 2887–2890
Mixed-integer programming (MIP), 2860, 2861, 2879, 2886–2889
Mixed search, 1514–1516, 1526
MOC-CDS. *See* Minimum routing cost connected dominating set (MOC-CDS)
Modified First-Fit (MFF), 501, 508
Modified First-Fit Decreasing (MFFD) algorithm, 482, 484
Modified Harmonic-Fit (MHF), 475
Monge, 2792–2794, 2801
Monotone searching, 68, 69, 73
Monotonic algorithm, 487
Most-*k*-Fit (MF_k) algorithm, 482, 496, 497
Multichannel wireless networks, 2359–2386
Multifit algorithm, 496
Multi-interface assignment, 2360–2365
Multilateration, 1417–1420, 1423
Multilayer friendship model, 2484–2487, 2499
Multilayer partition, 2587, 2589, 2604–2608, 2620
Multilinear programs, 2854, 2862–2867
Multiple choice constraints, 2873–2877
Multiple independent random walks (MultipleRW), 2476
Multiple instance learning (MIL), 603
Multistep lookahead, 3006–3007, 3011
Mutual exclusion, 466
MWCDS. *See* Minimum-weight Connected Dominating Set (MWCDS)
Myosin, 3232, 3236–3238

N

- Nash equilibrium (NE), 502–504
NE. See Nash equilibrium (NE)
 Nearest-neighbor-interchange distance, 749
 Nearly equitable r -coloring, 1231, 1232
 Neighborhood, 2783–2789
Network
 community, 2457
 efficiency, 3060–3066, 3082
 optimization, 1463–1464
 vulnerability, 1631–1665
 Network-based data mining, 612, 631–668
 Neural networks, 1067, 1079–1084, 1108, 1109
Newton method
 for linear fractional combinatorial optimization, 1321–1323
 for uniform fractional combinatorial optimization, 1315, 1321
Next-Fit (N_F) algorithm, 467, 472, 473, 478, 485, 489, 492, 499–500, 506, 514, 518–520
Next-Fit Decreasing (NFD) algorithm, 480, 484, 491, 509
NF. See Next-Fit (NF) algorithm
NFD. See Next-Fit Decreasing (NFD) algorithm
Nodes, 1991, 1993, 1995, 1997, 1999, 2001–2002, 2006–2008, 2010–2012, 2014–2017, 2020 cover in hypergraphs, 1001–1004, 1013, 1019, 1029, 1030, 1057 search, 1514–1518, 1524, 1527–1530, 1532, 1534
Nonblocking, 1756, 1757, 1760–1761, 1765–1811
Nonconvex, 2850, 2862, 2887
Nonhereditary, 1576–1579, 1584, 1588
Nonlinear integer programming, 2899, 2981
Nonlinear programming, 2899
Non unique probe selection problem, 567, 576–588
NP-complete, 1585–1586
NP-completeness, 224, 251–261
NP-hard, 1562, 1566, 1567, 1578, 1579, 1584, 1585, 1587
NP-hardness, 724, 749, 761, 763–764, 766–767
NP-hard problems, 1250, 1286, 1288
NP optimization, 2626, 2677
Number, 1560–1566, 1568, 1570, 1576, 1585, 1587, 1590

O

- OFFD.** *See* Ordered First-Fit Decreasing (OFFD) algorithm
Off-line algorithms, 457, 458, 463–465, 478–488, 498, 499, 504, 509, 510, 512
Oligonucleotide microarrays, 562, 564
On-demand data broadcast, 2393, 2394, 2406–2417
One-shot scheduling problem, 2574
Online, 2191–2245, 2253–2298 algorithm, 457, 458, 460, 464, 466–480, 483, 489–491, 493–495, 498, 499, 502, 504, 505, 507, 510–519 allocation, 2257, 2262, 2264–2271 coloring, 1876, 1879, 1880, 1884, 1886, 1898, 1899, 1902, 1903 optimization, 43, 86
Open faults, 1817, 1845, 1848, 1851–1856
Optimal partitions, 2301–2354
Optimal path queries, 1126, 1131, 1132
Optimization, 673–717, 1713–1751, 2391–2450, 3134, 3137, 3138, 3263, 3266, 3268, 3275, 3280, 3307, 3322, 3323, 3326–3328, 3331, 3332, 3336, 3339, 3340, 3345–3346, 3348, 3355–3359
Ordered First-Fit Decreasing (OFFD) algorithm, 517
Ore degree, 127, 1226, 1233
Ore-type, 1226–1228
Outerplanar graphs, 1208–1209, 1211, 1218, 1225, 1236, 1238
Output-sensitive algorithm, 1817–1818

P

- Pack**, 1225–1228
Packing number, 1001, 1017
Packing, w -, 1001, 1013, 1030, 1032, 1033
Paired-domination, 3364, 3367, 3376–3387, 3390
Pair-exchange, 2784, 2785, 2787, 2788
Pairwise connectivity, 1632–1644, 1646, 1648–1650, 1652, 1653, 1655, 1659, 1660, 1662, 1663, 1665
Parallel repetition, 2626, 2638–2643, 2667, 2679
Parametric problem, 1314, 1316, 1330–1334, 1337, 1338, 1341, 1351, 1352
Pardalos, P.M., 1573
Pareto-optimal frontier, 1352

- Partial k -trees, 1212, 1213
Partial orders, 517–518
Particle swarm optimization (PSO), 297–301
Partitional clustering, 608
Partition method, 2586–2590, 2617, 2620
Partitions, 2301–2354
Past-sequence-dependent delivery time, 147, 148, 168
Path relinking, 3297–3298, 3301, 3302, 3313–3317, 3323, 3326, 3331–3334, 3339
Path topology, 3200, 3202, 3203
Pathwidth, 1513, 1515, 1516, 1526, 1527, 1532, 1534
Pattern formation, 1086–1096, 1105–1111, 1117
PCP theorem, 2626, 2628, 2631–2633, 2642, 2644, 2646, 2648, 2655, 2662–2667
PDB structure, 3232–3233, 3251, 3252
Penalization, 837–852, 880
Penalty, 2255, 2258–2260, 2263, 2264, 2266, 2270, 2274, 2296, 2297
Penalty method, 1066–1068, 1072–1076, 1081, 1083, 1085, 1089, 1091–1093, 1104–1109, 1117
Periodic and aperiodic, 284–295, 303, 304
Periodicity, 3204
Permutation, 2743, 2744, 2747–2749, 2755, 2756, 2759, 2760, 2762, 2765, 2767, 2768, 2770–2774, 2776, 2777, 2780, 2782–2786, 2793–2795, 2800, 2801, 2803, 2804
Permutation graph, 224, 231, 266–268, 270, 271
Persistency, 2854, 2887
Perturbed amino acids, 3227–3230
Petri net, 1667–1708
Physical interference model, 2573, 2574
Pipeline, 2504, 2509, 2510, 2519, 2527–2538, 2540, 2542, 2543
Planar graph, 2097–2099, 2106–2112, 2116–2124, 2126–2131, 2137, 2146–2157, 2161–2172
Plane sweep algorithm, 1819, 1825–1828, 1831–1835, 1837, 1841, 1843, 1858, 1863, 1865, 1867
PLS complete, 2785, 2786
Point coverage, 901, 910, 914–924
Polyhedra approach, 2313–2316, 2333–2341
Polyhedral results, 1561, 1568–1572, 1582–1584, 1586–1587
Polymatroid function, 1600, 1605, 1608
Polynomial programming problem, 2850, 2852, 2859, 2862, 2869, 2886
Polynomial solvability, 2743, 2744, 2751, 2755, 2775, 2780–2781, 2785, 2786, 2791–2795, 2801, 2805
Polynomial space algorithms, 1259, 1262, 1265, 1284, 1285, 1287, 1288
Polynomial-time algorithm, 2900–2902, 2910–2913, 2921–2925, 2929, 2930, 2938, 2941, 2948–2949, 2951, 2953, 2956, 2962, 2968, 2970, 2977, 2979, 2980, 2982
Polynomial-time approximation scheme (PTAS), 462, 465, 485, 491, 497, 500, 519, 723–743, 2586–2590, 2593, 2595–2598, 2600, 2604, 2608, 2613, 2614, 2617, 2618, 2620
Polytope, 2747, 2756–2758, 2781, 3016, 3017, 3029, 3031, 3035, 3049
Pooling design, 95–113, 129, 137, 142
 ℓ -Port, matroid, 1018–1022
PoS. *See* Price of stability (PoS)
Positive semidefinite splitting, 2776, 2777
Power domination, 3364, 3365, 3367–3376, 3390
Power graph, 1575–1576, 1584
Power spectral density (PSD), 290–294
Preemption, 2255, 2258, 2259, 2263–2271, 2274–2277, 2280, 2296, 2297
Prefix algorithm, 498–499
Presorting, 479–483, 490, 499–500
Price of stability (PoS), 503
Probabilistically checkable proofs, 2627–2630, 2638, 2653
Probabilistic analysis, 457, 463
Probabilistic verification, 2625–2679
Probe selection and design, 562–567
Product, 2743, 2744, 2748–2750, 2753, 2755, 2765, 2770, 2771, 2775, 2792, 2801, 2803
Proper function, 1600, 1608, 1609, 1613
Protein interaction networks, 1591
Protein structures, 3221, 3222, 3241, 3248, 3252
Protocol interference model, 2504, 2508, 2510–2515, 2537, 2542, 2568
Pseudo-approximation, 1633, 1648, 1650–1654, 1656, 1657, 1659–1664
Pseudo-randomness, 2673, 2678
Pseudo random numbers, 1100–1103
PSO. *See* Particle swarm optimization (PSO)
PTAS. *See* Polynomial-time approximation scheme (PTAS)
Pursuit-and-evasion problem, 1512

Q

QAPs. *See* Quartic assignment problems (QAPs)
 QBAP. *See* Quadratic bottleneck assignment problem (QBAP)
 Quadrangle inequality, 43, 76–85, 87
 Quadratic assignment, 2741–2805
 Quadratic assignment problem, 2854, 2862–2867
 Quadratic bottleneck assignment problem (QBAP), 2743, 2800–2802
 Quadratic optimization, 533–555
 Quadratic programming, 543, 2775, 2777–2779
 Quadratic semi-assignment problem (SQAP), 2743, 2804–2805
 Quartic assignment problems (QAPs), 2741–2805
 Quasi-clique, 1590, 1591
 γ -Quasi-clique, 1584–1590
 γ -Quasi-clique number, 1587
 Quasi equitably subpartitionable (Quasi-ESP), 1047–1052
 Quasi-ESP. *See* Quasi equitably subpartitionable (Quasi-ESP)
 Quasi hereditary, 1588
 Quasi-triangulation, 2689, 2713, 2715–2717, 2731

R

Ramp loss SVM, 602
 Randomization, 463, 477, 483–484
 Random walker (RW) algorithm, 2457, 2468, 2471–2473, 2475–2477
 Rank inequalities, 1570, 1571
 Ranking, 597, 620–621
 Rank quotient, 1603, 1604, 1607, 1608
 RBI. *See* Reactive business intelligence (RBI)
 r -decomposition, 1231, 1233
 Reactive business intelligence (RBI), 2815–2844
 Recursive bisection, 1659
 Reduction method, 2754, 2763–2766, 2768–2770
 Reduction to extremal rays, 2794
 Refined First-Fit (RFF) algorithm, 473–474, 504, 515
 Refined First-Fit Decreasing (RFFD) algorithm, 482, 490, 491, 498, 503
 Refined Harmonic-Fit (RHF), 474, 491
 Reformulation-linearization technique (RLT), 2849–2891
 Rejection (bin packing with), 504–505

Rejective Modified Harmonic (MHR) algorithm, 505

Relative neighborhood graph (RNG), 1432–1433, 1435

Relaxation, 837–852, 880, 886

Repack, 465, 473, 478–479, 490, 493, 499–502, 519

Replicator equations, 1068, 1085

r -equitable, 1229, 1230, 1232, 1233

Reservation techniques, 471–472, 474

Resource allocation problem, 2897–2982

Resource augmentation, 491–494, 512

Resources sharing, 1670, 1673–1682

Restricted size, 462, 466, 509

Revised Wharehouse (RW) algorithm, 479

RFF. *See* Refined First-Fit (RFF) algorithm

RFFD. *See* Refined First-Fit Decreasing (RFFD) algorithm

RHF. *See* Refined Harmonic-Fit (RHF)

Ribbon decomposition, 3207–3209

Right lower graded, 2794

Rigid, 1425–1430

r -irreducible, 1230

RLT. *See* Reformulation-linearization technique (RLT)

RMSD. *See* Root-mean-square-difference (RMSD)

RNG. *See* Relative neighborhood graph (RNG)

Robustness, 1112, 1113, 1115, 1116, 1118

Rollout policy, 2990

Root-mean-square-difference (RMSD), 3250, 3251

Rotation distance, 749, 773–779

Routing, 785–787, 805–807, 809, 812–815, 819, 829

Routing and location problems, 929–990

r -reducible, 1230

r -robust k -club, 1582

R-sausage, 3202–3208, 3211, 3216, 3217, 3221, 3225, 3227, 3233, 3255

RW. *See* Revised Wharehouse (RW) algorithm

S

SA. *See* Simulated annealing (SA)

“Sausage” conjecture, 3202

Sausage network, 3216

Scale-free graphs, 1573

Scale-free networks, 6, 8

Scheduling, 145–168, 2191–2245, 2394–2409, 2411–2419, 2435, 2437, 2444, 2449, 2504, 2508–2514, 2519–2538, 2540, 2542, 2543

Schroeder’s conjecture, 1543, 1552

- Schur-convexity, 2308, 2310–2313, 2331
Scientific collaboration networks, 1591
Search number, 1514–1518, 1521–1529,
 1532–1534
Secondary structure, 3221, 3222, 3230–3254
Seed set, 2458–2462, 2464–2468
Selective learning, 603
Selfish bin packing, 502–5004
Self-organization, 1111
 control, 1111
 maps, 1083–1084
Self-training, 614–615
Semidefinite cuts, 2854, 2886–2887, 2891
Semidefinite relaxation, 2750, 2772,
 2775–2778
Semi-online, 2191–2245
Semi-on-line algorithm (SOL), 478–480,
 493, 515
Semi-supervised learning, 596, 614–620
Semi-truthful, 2256, 2274, 2275
Sensor networks, 899–925
Separable convex function, 2899–2901, 2904,
 2907, 2925, 2929–2930, 2951, 2967,
 2969, 2977, 2982
Sequential consistency, 2996–3007, 3011
Sequential improvement, 3001–3002, 3011
Series-parallel graph, 1055
Set packing, 507
Set partitioning problem, 2866, 2872
Shifted submodularity, 1608, 1616
Shifting strategy, 2587–2589, 2592, 2600,
 2604, 2613
Shortest paths, 1126–1134, 1136, 1138,
 1140, 1142–1144, 1146,
 1148–1150, 1991, 2002, 2005, 2012,
 2015–2019
Signal-to-interference-plus-noise ratio (SINR),
 2573, 2574
Signed graph, 1020
Silk, 3232, 3235–3238
Similarity measures, 637–641
Simplicial method, 2096, 2097, 2099, 2100,
 2114, 2119, 2125, 2128, 2136, 2142,
 2147, 2154, 2168, 3015–3049
Simplicial subdivision, 2099, 2102,
 2111, 3018
Simpson line, 3191–3192, 3196, 3200, 3201
Simulated annealing (SA), 295–297, 1067,
 1077–1078, 1104, 1106–1111,
 1117, 2788, 2791, 2799, 2804, 3138,
 3139, 3141, 3142
Sink placement, 2565–2568
SINR. *See* Signal-to-interference-plus-noise
 ratio (SINR)
Skewed variable neighbourhood search
 (SVNS), 1938–1945
Small, 2752, 2755, 2759, 2764, 2768, 2770,
 2777, 2781, 2786, 2789, 2792, 2804,
 2805
Smallest-degree-last ordering, 1625–1627
Small world networks, 3057–3083
SNA. *See* Social network analysis (SNA)
Snapshot data collection (SDC), 2504,
 2506–2510, 2512–2514, 2516,
 2519–2524, 2527, 2537–2539,
 2542, 2543
Social graph, 2485–2488, 2498
Social network analysis (SNA), 1560, 1563,
 1574, 1575, 1590, 1591
Social networks, 2–5, 29, 33–36, 650–668,
 1560, 1590, 3090–3092, 3094, 3101,
 3102, 3110, 3120–3129
SOL. *See* Semi-on-line algorithm (SOL)
Sortability, 2327–2329, 2347, 2353
Source-sink connected graph, 1025, 1026
Spanner, 1434–1438
Spanning trees, 2016–2019, 2021, 2022, 2024
Sparsest cut, 1659–1661
Spectrum, 2254–2298
Split graph, 1207, 1208, 1239
SQAP. *See* Quadratic semi-assignment
 problem (SQAP)
Square product, 1219–1221, 1239
SS. *See* Subset Sum (SS) algorithm
Stable matching, 996, 1029, 1034
Stackelberg games, 1950, 1976, 1978, 1981
Steiner points, 1460, 1463–1464, 1470–1474,
 1476–1491, 1493–1507
Steiner problem, 3180, 3181, 3190, 3195,
 3202, 3217–3218, 3220, 3234, 3255
Steiner ratio, 3180, 3181, 3190, 3195, 3202,
 3217, 3218, 3220, 3234, 3255
Steiner trees, 1464, 1469–1471, 1473–1491,
 1497, 1498
Stock-market graphs, 1590
Strategic oscillation, 3296–3298, 3302, 3306,
 3308–3313, 3315, 3321, 3322, 3324,
 3325, 3339, 3350, 3352–3353
Strategyproof, 2256, 2261, 2262, 2283,
 2285–2286, 2289
Strategyproof mechanism, 2261–2262
Stretch factor, 1434–1436, 1438, 1441, 1442,
 1444
String sorting algorithm, 289–294
Strong connector, 1023, 1025
Strongly chordal graph, 224, 231, 242, 248,
 261–266
Strongly polynomial algorithm, 1313

- Structurally equivalent, 2470, 2471
 Structurally similar, 2470, 2471
 Submodular cover, 1608, 1613
 Submodular function, 1039, 1043, 1607, 1613,
 1621, 2463–2465, 2913, 2914, 2930,
 2954, 2968, 2982
 Subset interconnection design, 614
 Subset sum (SS) algorithm, 504
 Subtree transfer distance, 749–752, 762–773,
 779
 Sum, 2745, 2750, 2760, 2761, 2769, 2790,
 2792, 2796–2798, 2800, 2801, 2804
 Sum-of-ratios problem, 1352
 Supervised learning, 596–608
 Support vector machines (SVMs), 597–604
 Support vector regression (SVR), 598,
 600, 601
 SVMs. *See* Support vector machines (SVMs)
 SVNS. *See* Skewed variable neighbourhood
 search (SVNS)
 SVR. *See* Support vector regression (SVR)
 Switching networks, 1755–1812
 Sylvester’s sequence, 460–461, 477, 492, 512
 Symmetric, 2743, 2745, 2748–2751, 2758,
 2767–2769, 2772, 2773,
 2778–2795, 2801
 Symmetric quadratic assignment, 2758,
 2769, 2791
- T**
 Tabu search (TS), 2782, 2787–2791, 2804
 Tag graph, 2488–2491, 2498
 t -cluster problem, 1585
 TCPs. *See* Ternary complementary pairs
 (TCPs)
 T-cut, 1021
 TDI. *See* Totally dual integral (TDI)
 Telecommunication, 1590
 Telecommunication networks, 662–663
 Ternary complementary pairs (TCPs), 287, 288
 Test set, 596, 607, 614
 Text analytics, 1590
 Three-index assignment problem, 1068, 1070,
 1071, 1076, 1088, 1094–1096, 1098,
 1099, 1103–1106, 1109–1111, 1117
 Tiny Modified-Fit algorithm, 493
 T-join, 1021
 Toeplitz, 2793–2794
 Toll pricing, 676, 692–700
 Torsion energy, 3242–3246
 Total domination, 3364, 3367, 3387–3390
 Totally balanced hypergraph, 1052
 Totally balanced matrix, 1051
 Totally dual integral (TDI), 997–999, 1005,
 1008–1010, 1029, 1030, 1034–1036,
 1044, 1053, 1055, 1056
 Trace, 2750, 2767, 2772, 2775
 Traffic assignment, 676–693
 Training set, 596, 598, 599, 606, 614,
 615, 620
 Tramp-steamer problem, 1337
 Transductive learning, 614
 Transitive closure, 1026, 1027
 Transportation modeling, 693, 713
 Travelling salesman, 2756, 2757, 2786
 Travelling salesman problem (TSP), 2745,
 2751, 2788, 2789, 2793, 2800, 2854,
 2880–2886
 Tree, 223, 224, 231–245, 251, 255, 256, 272
 Tree decomposition, 1213
 Treewidth, 1213–1214, 1513, 1515–1520,
 1522, 1526, 1527, 1534
 Triangulation, 1420, 1423, 1435, 1439,
 1442–1444, 1446, 2105, 2106,
 3016–3020, 3023, 3025, 3032–3034,
 3043–3045
 Trilateration, 1417–1418, 1425, 1427–1428,
 1430
 TS. *See* Tabu search (TS)
 TSP. *See* Travelling salesman problem (TSP)
 Two-commodity cut, 1021
 Two-commodity path, 1021
 Two-index assignment problem, 1068, 1074,
 1075, 1082–1085, 1088–1090,
 1094–1096, 1098, 1099, 1103, 1105,
 1107–1109, 1112, 1117
 Two-prover games, 2679
- U**
 UBG. *See* Unit ball graph (UBG)
 Uniform matroid, 1007, 1019, 1053,
 1605, 1607
 Uniform node sampling, 2477
 Unique games conjecture, 2667–2674
 Unit ball graph (UBG), 2589, 2593–2598,
 2600, 2602, 2604, 2613–2615,
 2617, 2618
 Unsupervised learning, 596, 608–614
 Upper rank, 1603
- V**
 Valid inequalities, 2851–2854, 2861–2862,
 2881, 2885, 2887, 2889, 2890
 Variable dimension method, 2114–2116, 2128,
 2137, 2147, 2165, 3016, 3017

- Variable-size bin packing, 488–491, 498, 512, 518
Variable upper bounding constraints, 2865, 2873, 2877–2879
Variational inequalities, 835–895
Vector labeling, 2150, 3025
Vector variational inequalities, 891
Vehicle routing, 675, 676, 700–717
Vertex coloring, 1624, 2097, 2099–2116, 2129, 2143, 2144
Vertex cover, 1007, 1016, 1023, 1049, 1562, 1565, 1566, 1573
 β -Vertex disruptor, 1648, 1654–1658, 1662–1664
Vertex-distinguishing coloring, 2132–2148
Vertex separation, 1513, 1516, 1523, 1525
Viral marketing, 2
Virtual backbone, 785–787, 797, 805, 816, 819–829
Virtual coordinate, 1431, 1438–1442
VLSI layout, 1844–1846
Voronoi cell, 1429, 1430, 1439–1440, 1442–1443
Voronoi diagram of spheres, 2710–2711, 2713, 3185–3191, 3207
Voronoi edges, 3186, 3187, 3207
Voronoi points, 3186
Voronoi polyhedra, 3207
- Weakly-connected dominating set, 1614
Web, 1571, 1572, 1590
Web mining, 1590
Weighing matrices, 288, 294, 295
Weighted cascade (WC), 2466
Weighted Connected Vertex Cover (WCVC), 1613
Weighting functions, 468–470, 474, 478–481, 490, 492, 493, 505
WF. *See* Worst-Fit (WF)
W[1]-hard, 1566
Wiener index, 2792
Wireless communication, 902
Wireless data broadcast, 2392, 2394–2395, 2409, 2420, 2421, 2427, 2449
Wireless networks, 784, 785, 788, 805, 806, 808, 820–829, 1155–1195, 2253–2298, 2557, 2569, 2581
Wireless sensor networks (WSNs), 2504–2517, 2519, 2525, 2527, 2528, 2531, 2535–2538, 2540–2544
Work functions, 43, 54–58, 86
Worst-Fit (WF), 467, 470, 487, 494, 515, 520
Wrapper models, 621
WSNs. *See* Wireless sensor networks (WSNs)

W

- WC. *See* Weighted cascade (WC)
WCVC. *See* Weighted Connected Vertex Cover (WCVC)

Z

- Zassenhaus-Groemer-Oler inequality, 2555, 2556, 2573
Zero-one optimization, 551